



# Liberty ID-WSF Discovery Service Specification

Version: 2.0-errata-v1.0

## **Editors:**

Conor Cahill, Intel Corporation

Jeff Hodges, NeuStar, Inc.

## **Contributors:**

Robert Aarts, Nokia Corporation

John Beatty, Sun Microsystems, Inc.

Carolina Canales-Valenzuela, Ericsson

Darryl Champagne, IEEE-ISTO

Gary Ellison, Sun Microsystems, Inc.

Jukka Kainulainen, Nokia Corporation

John Kemp, Nokia Corporation

Rob Lockhart, IEEE-ISTO

Paul Madsen, NTT, formerly Entrust, Inc.

Jonathan Sergent, Sun Microsystems, Inc.

Greg Whitehead, Hewlett-Packard

Emily Xu, Sun Microsystems, Inc.

## **Abstract:**

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

**Filename:** liberty-idwsf-disco-svc-2.0-errata-v1.0.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 © 2007 2FA Technology; Adobe Systems; Agencia Catalana De Certificacio; America Online, Inc.;  
16 American Express Company; Amssoft Systems Pvt Ltd.; Avatier Corporation; BIPAC; BMC Software, Inc.; Bank of  
17 America Corporation; Beta Systems Software AG; British Telecommunications plc; Computer Associates  
18 International, Inc.; Credentica; DataPower Technology, Inc.; Deutsche Telekom AG, T-Com; Diamelle Technologies,  
19 Inc.; Diversinet Corp.; Drummond Group Inc.; Enosis Group LLC; Entrust, Inc.; Epok, Inc.; Ericsson; Falkin  
20 Systems LLC; Fidelity Investments; Forum Systems, Inc.; France Télécom; French Government Agence pour le  
21 développement de l'administration électronique (ADAE); Fugen Solutions, Inc; Fulvens Ltd.; GSA Office of  
22 Governmentwide Policy; Gamefederation; Gemalto; General Motors; GeoFederation; Giesecke & Devrient GmbH;  
23 Hewlett-Packard Company; Hochhauser & Co., LLC; IBM Corporation; Intel Corporation; Intuit Inc.; Kantega;  
24 Kayak Interactive; Livo Technologies; Luminance Consulting Services; MasterCard International; MedCommons  
25 Inc.; Mobile Telephone Networks (Pty) Ltd; NEC Corporation; NTT DoCoMo, Inc.; Netegrity, Inc.; Neustar, Inc.;  
26 New Zealand Government State Services Commission; Nippon Telegraph and Telephone Corporation; Nokia  
27 Corporation; Novell, Inc.; OpenNetwork; Oracle Corporation; Ping Identity Corporation; RSA Security Inc.;  
28 Reactivity Inc.; Royal Mail Group plc; SAP AG; Senforce; Sharp Laboratories of America; Sigaba; SmartTrust; Sony  
29 Corporation; Sun Microsystems, Inc.; Supremacy Financial Corporation; Symlabs, Inc.; Telecom Italia S.p.A.;  
30 Telefónica Móviles, S.A.; Telenor R&D; Thales e-Security; Trusted Network Technologies; UNINETT AS; UTI;  
31 VeriSign, Inc.; Vodafone Group Plc.; Wave Systems Corp. All rights reserved.

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

## Contents

38		
39	1. Introduction	6
40	1.1. Conceptual Model and Terminology	6
41	1.2. Scope	7
42	1.3. Notation and Conventions	7
43	1.3.1. XML Namespaces	7
44	2. Discovery Service Information Model	9
45	2.1. Overview of Discovery Service Information Model	9
46	2.2. Versioning in ID-WSF	10
47	2.3. ID-WSF Endpoint References (EPRs)	10
48	2.3.1. EPR Profiling Attributes	11
49	2.3.2. EPR Profiling Elements	12
50	2.3.3. ID-WSF Web Services Addressing EPR Profile	16
51	2.3.4. Effective Web Services Addressing EPR	20
52	2.3.5. Example Liberty ID-WSF EPRs	21
53	2.4. Service Metadata	23
54	2.4.1. Service Metadata element	23
55	2.4.2. Minting ID-WSF EPRs based upon Service Metadata	26
56	2.4.3. Service Metadata Example	26
57	3. Discovery Service	30
58	3.1. Service URIs	30
59	3.2. Status Codes	31
60	3.3. Operation: <i>DiscoveryQuery</i>	32
61	3.3.1. wsa:Action values for <i>DiscoveryQuery</i> Messages	32
62	3.3.2. <Query> Message	32
63	3.3.3. <i>QueryResponse</i>	36
64	3.3.4. <i>DiscoveryQuery</i> Processing Rules	38
65	3.4. Operation: <i>MDAssociationAdd</i>	38
66	3.4.1. wsa:Action values for <i>MDAssociationAdd</i> Messages	38
67	3.4.2. <i>SvcMDAssociationAdd</i> Message	39
68	3.4.3. <i>SvcMDAssociationAddResponse</i> Message	39
69	3.4.4. <i>MDAssociationAdd</i> Processing Rules	40
70	3.5. Operation: <i>MDAssociationQuery</i>	40
71	3.5.1. wsa:Action values for <i>MDAssociationQuery</i> Messages	41
72	3.5.2. <i>SvcMDAssociationQuery</i> Message	41
73	3.5.3. <i>SvcMDAssociationQueryResponse</i> Message	41
74	3.5.4. <i>MDAssociationQuery</i> Processing Rules	42
75	3.6. Operation: <i>MDAssociationDelete</i>	42
76	3.6.1. wsa:Action values for <i>MDAssociationDelete</i> Messages	42
77	3.6.2. <i>SvcMDAssociationDelete</i> Message	42
78	3.6.3. <i>SvcMDAssociationDeleteResponse</i> Message	43
79	3.6.4. <i>MDAssociationDelete</i> Processing Rules	43
80	3.7. Operation: <i>MetadataRegister</i>	44
81	3.7.1. wsa:Action values for <i>MetadataRegister</i> Messages	44
82	3.7.2. <i>SvcMDRegister</i> Message	44
83	3.7.3. <i>SvcMDRegisterResponse</i> Message	45
84	3.7.4. <i>MetadataRegister</i> Processing Rules	46
85	3.8. Operation: <i>MetadataQuery</i>	46
86	3.8.1. wsa:Action values for <i>MetadataQuery</i> Messages	47
87	3.8.2. <i>SvcMDQuery</i> Message	47
88	3.8.3. <i>SvcMDQueryResponse</i> Message	47
89	3.8.4. <i>MetadataQuery</i> Processing Rules	48
90	3.9. Operation: <i>MetadataReplace</i>	48

91	3.9.1. wsa:Action values for MetadataReplace Messages	49
92	3.9.2. SvcMDReplace Message	49
93	3.9.3. SvcMDReplaceResponse Message	49
94	3.9.4. Metadata Replace Processing Rules	50
95	3.10. Operation: <i>MetadataDelete</i>	51
96	3.10.1. wsa:Action values for MetadataDelete Messages	51
97	3.10.2. SvcMDDelete Message	51
98	3.10.3. SvcMDDeleteResponse Message	51
99	3.10.4. Metadata Delete Processing Rules	52
100	3.11. Option Value for Response Authentication	53
101	3.12. Including Keys in the SvcMDRegisterResponse Message	53
102	3.13. Discovery Service Example Messages (NON-Normative)	54
103	3.13.1. Query People Service	54
104	3.13.2. Query provider	56
105	3.13.3. Query (empty)	57
106	3.13.4. Query People Service and Provider	59
107	3.13.5. SvcMDQuery (empty)	60
108	3.13.6. SvcMDQuery w/Bad SvcMDID	60
109	3.13.7. SvcMDRegister w/single SvcMD	61
110	3.13.8. SvcMDQuery w/Good SvcMDID	61
111	3.13.9. SvcMDDelete w/Good SvcMDID	62
112	3.13.10. SvcMDDelete w/Already Deleted SvcMDID	62
113	3.13.11. SvcMDRegister w/Complex SvcMD	63
114	3.13.12. SvcMDQuery w/Good SvcMDID for Complex SvcMD	64
115	3.13.13. SvcMDReplace of existing SvcMD	65
116	3.13.14. SvcMDQuery of Replaced SvcMDID	66
117	3.13.15. SvcMDDelete of replaced SvcMD	67
118	3.13.16. SvcMDRegister w/multiple SvcMDs	67
119	3.13.17. Query Calendar Service	69
120	3.13.18. SvcMDAssociationAdd the Calendar Service SvcMD	69
121	3.13.19. SvcMDAssociationQuery w/SvcMDID	70
122	3.13.20. SvcMDAssociationQuery w/o SvcMDID	70
123	3.13.21. Query Calendar Service (again)	70
124	3.13.22. Query Calendar Service w/Action	71
125	3.13.23. Query Calendar Service w/resultsType=all	72
126	3.13.24. Query for all Calendar Service EPRs	74
127	3.13.25. Query for one Calendar Service EPRs	75
128	3.13.26. Query specific version of Calendar Service	76
129	3.13.27. SvcMDReplace Calendar Service	78
130	3.13.28. Query old Calendar Service	78
131	3.13.29. Query for all Calendar Service EPRs	79
132	3.13.30. SvcMDAssociationAdd the ATM Service SvcMD	80
133	3.13.31. Query ATM Service w/resultsType=best	80
134	3.13.32. Query ATM Service w/Withdraw Action	82
135	3.13.33. Query ATM Service w/Option	83
136	3.13.34. Query ATM Service w/unknown Option	84
137	3.13.35. Query ATM Service w/good and bad Option	85
138	4. Discovery Service ID-WSF EPR conveyed via a Security Token	87
139	4.1. EPR Generation Rules	87
140	4.2. SAML 2.0 Security Tokens	87
141	4.3. SAML 1.x (Liberty ID-FF) Security Tokens	88
142	5. ID-WSF 1.x Resource Offering conveyed in an EPR	90
143	6. Acknowledgments	92
144	References	93

---

145	A. Discovery Service Version 2.0 XSD .....	95
146	B. Discovery Service WSDL .....	102

## 1. Introduction

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

### 1.1. Conceptual Model and Terminology

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

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

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

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

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

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

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

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

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

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

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

## 193 1.2. Scope

194 This specification:

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

## 201 1.3. Notation and Conventions

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

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

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

### 209 1.3.1. XML Namespaces

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

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

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

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

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

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

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

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

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

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

## 2. Discovery Service Information Model

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

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

### 2.1. Overview of Discovery Service Information Model

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

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

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

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

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

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

277 • The above Principal's-perspective model is now enabled.

278 There are various permutations of this general WSP-perspective service instance registration model. For example,  
279 the same administrative entity may be deploying the both the Discovery Service and the other services and so may  
280 employ alternative means, e.g., bulk configuration, to effect service metadata registration with their Discovery  
281 Service.

## 282 2.2. Versioning in ID-WSF

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

286 The specific areas of versioning include:

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

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

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

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

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

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

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

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

## 315 2.3. ID-WSF Endpoint References (EPRs)

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

```

317
318 <wsa:EndpointReference ...>
319   <wsa:Address>...some URI here...</wsa:Address>
320
321   <wsa:ReferenceParameters>.....</wsa:ReferenceParameters>
322
323   <wsa:Metadata>...some metadata here... </wsa:Metadata>
324 </wsa:EndpointReference>
325

```

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

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

**Note**

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

```

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

```

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

### 2.3.1. EPR Profiling Attributes

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

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

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

```
<!-- wsu:Id attribute for EPR - for xml document references to EPR -->
<xs:attribute ref="wsu:Id" use="optional" />
```

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

#### 2.3.1.2. reqRef — request reference

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

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

**Figure 2. reqRef — Schema Fragment**

#### 2.3.1.3. notOnOrAfter

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

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

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

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

**Figure 3. notOnOrAfter — Schema Fragment**

### 2.3.2. EPR Profiling Elements

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

### 416 **2.3.2.1. Abstract**

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

```
419  
420 <!-- Abstract: natural-language description of service -->  
421  
422 <xs:element name="Abstract" type="xs:string"/>  
423  
424
```

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

### 426 **2.3.2.2. Provider ID**

427 The <ProviderID> element ([Figure 5](#)) contains the URI of the provider of this service instance.

```
428  
429 <!-- Provider ID -->  
430  
431 <xs:element name="ProviderID" type="xs:anyURI"/>  
432  
433
```

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

### 435 **2.3.2.3. Service Type**

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

```
439  
440 <!-- Service Type -->  
441  
442 <xs:element name="ServiceType" type="xs:anyURI"/>  
443  
444
```

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

446 Some examples of possible ServiceType URIs:

```
447 urn:liberty:disco:2006-08  
448 urn:liberty:id-sis-pp:2003-08  
449 http://myservices.com/gaming/1.0
```

### 450 **2.3.2.4. securityContext**

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

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

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

- 459 • `urn:liberty:disco:tokenref:ObtainFromIDP`

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

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

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

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

```
470  
471 <!-- Security Context Container -->  
472  
473 <xs:element name="SecurityContext">  
474   <xs:complexType>  
475     <xs:sequence>  
476       <xs:element ref="SecurityMechID"  
477         minOccurs="1"  
478         maxOccurs="unbounded" />  
479  
480       <xs:element ref="sec:Token"  
481         minOccurs="0"  
482         maxOccurs="unbounded" />  
483     </xs:sequence>  
484   </xs:complexType>  
485 </xs:element>  
486  
487
```

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

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

### 491 **2.3.2.5. SecurityMechID**

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

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

```
497
498 <!-- Security Mechanism ID -->
499
500 <xs:element name="SecurityMechID" type="xs:anyURI" />
501
502
```

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

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

```
505 urn:liberty:security:2006-08:ClientTLS:SAMLV2
```

```
506 urn:liberty:security:2003-08:ClientTLS:SAML
```

```
507 urn:liberty:security:2006-08:TLS:SAMLV2
```

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

### 510 2.3.2.6. Framework

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

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

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

```
516
517 <!-- Framework Description -->
518
519 <xs:element ref="sb:Framework" maxOccurs="unbounded" />
520
521
```

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

### 523 2.3.2.7. Action

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

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

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

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

```
533
534 <!-- Action(s) - the interfaces available at this service -->
535
536 <xs:element name="Action" type="xs:anyURI" />
537
```

538 **Figure 10. <Action> — Schema Fragment**539 **2.3.2.8. Options**

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

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

```
547
548 <!-- Options -->
549
550 <xs:element name="Options" type="OptionsType"/>
551
552 <xs:element name="Option" type="xs:anyURI" />
553
554 <xs:complexType name="OptionsType">
555   <xs:sequence>
556     <xs:element ref="Option" minOccurs="0" maxOccurs="unbounded"/>
557   </xs:sequence>
558 </xs:complexType>
559
560
```

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

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

567 **2.3.3. ID-WSF Web Services Addressing EPR Profile**

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

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

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

576 **Note**

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

```
580  
581 <xs:element name="EndpointReference" type="tns:EndpointReferenceType" />  
582  
583 <xs:complexType name="EndpointReferenceType">  
584   <xs:sequence>  
585     <xs:element name="Address"  
586       type="tns:AttributedURIType" />  
587  
588     <xs:element name="ReferenceParameters"  
589       type="tns:ReferenceParametersType"  
590       minOccurs="0" />  
591  
592     <xs:element ref="tns:Metadata"  
593       minOccurs="0" />  
594  
595     <xs:any namespace="##other"  
596       processContents="lax"  
597       minOccurs="0"  
598       maxOccurs="unbounded" />  
599   </xs:sequence>  
600  
601   <xs:anyAttribute namespace="##other" processContents="lax" />  
602 </xs:complexType>  
603  
604  
605
```

606 **Figure 12. <wsa:EndpointReference> — Schema Fragment**607 **2.3.3.1. ID-WSF EPR Minting Rules**

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

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

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

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

### 649 2.3.3.2. ID-WSF EPR Specifics

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

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

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

- 657 • The <wsdl:binding> contains a <wsdlsoap:binding> element. This specifies that the SOAP binding for  
658 WSDL is being used.
- 659 • The style attribute of the <wsdlsoap:binding> element is "document".
- 660 • The transport attribute of the <wsdlsoap:binding> element is *http://schemas.xmlsoap.org/soap/http*.

- 661 • The abstract WSDL corresponding to the `<ServiceType>` MUST contain a single `<portType>` element. The  
662 `<wsdl:binding>` element provides bindings for the operations specified in this `<wsdl:portType>`. Each  
663 operation binding includes an input element and an output element, each containing a single `<wsdlsoap:body>`  
664 element. The `use` attribute of the `<wsdlsoap:body>` elements is "literal".
- 665 • The `location` attribute of `<wsdlsoap:address>` is equal to `<wsa:Address>`.
- 666 • All other optional elements and attributes are not specified and thus default to the SOAP binding of WSDL.

### 667 2.3.3.3. Security Mechanism Specifics

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

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

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

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

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

### 685 2.3.3.4. Action Specifics

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

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

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

### 692 2.3.3.5. Identity Invocation Context specifics

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

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

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

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

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

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

710 **Note**

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

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

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

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

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

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

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

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

### 737 2.3.4. Effective Web Services Addressing EPR

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

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

- 742 • While the elements within the <Metadata> element appear to be ordered, they can all appear in any order and can  
743 have other elements appear between the listed elements. This is why they are contained within a multi-occurrence  
744 <xs:choice>.
- 745 • Four attributes have been added to the EPR element itself: the notOnOrAfter timestamp and three different IDs  
746 (for use in different circumstances).
- 747 • Seven sub-elements were added to the <Metadata> element.
- 748 • The <Metadata> sub-element <disco:ProviderID> MUST appear exactly once in an ID-WSF EPR, even  
749 though the schema below does not enforce that requirement (because of a limitation in XML Schema – or perhaps  
750 in the author’s understanding of XML schema).
- 751 The <Metadata> sub-elements: <sbf:Framework>, and <disco:ServiceType> MUST appear at least once  
752 in an ID-WSF EPR, even though the schema below does not enforce that requirement (because of a limitation in  
753 XML Schema – or perhaps in the author’s understanding of XML schema).

```

754
755 ...
756 <xs:element name="EndpointReference" type="tns:EndpointReferenceType"/>
757 <xs:complexType name="EndpointReferenceType" mixed="false">
758   <xs:sequence>
759     <xs:element name="Address" type="tns:AttributedURIType"/>
760     <xs:element name="ReferenceParameters"
761       type="tns:ReferenceParametersType" minOccurs="0"/>
762     <xs:element ref="tns:Metadata" minOccurs="0"/>
763     <xs:any namespace="##other" processContents="lax"
764       minOccurs="0" maxOccurs="unbounded"/>
765   </xs:sequence>
766   <xs:attribute name="notOnOrAfter" type="xs:dateTime" use="optional" />
767   <xs:attribute ref="wsu:Id" use="optional" />
768   <xs:attribute name="reqRef" type="xs:string" use="optional" />
769   <xs:anyAttribute namespace="##other" processContents="lax"/>
770 </xs:complexType>
771
772 <xs:complexType name="ReferenceParametersType" mixed="false">
773   <xs:sequence>
774     <xs:any namespace="##any" processContents="lax"
775       minOccurs="0" maxOccurs="unbounded" />
776   </xs:sequence>
777   <xs:anyAttribute namespace="##other" processContents="lax"/>
778 </xs:complexType>
779
780 <xs:element name="Metadata" type="tns:MetadataType"/>
781 <xs:complexType name="MetadataType" mixed="false">
782   <xs:sequence>
783     <xs:choice minOccurs="0" maxOccurs="unbounded">
784       <xs:element ref="disco:Abstract" minOccurs="0" />
785       <xs:element ref="sbf:Framework" maxOccurs="unbounded"/>
786       <xs:element ref="disco:ProviderID" />
787       <xs:element ref="disco:ServiceType" maxOccurs="unbounded"/>
788       <xs:element ref="disco:SecurityContext" minOccurs="0" maxOccurs="unbounded" />
789       <xs:element ref="disco:Options" minOccurs="0" maxOccurs="unbounded" />
790       <xs:element ref="disco:Action" minOccurs="0" maxOccurs="unbounded" />
791       <xs:any namespace="##other" processContents="lax"
792         minOccurs="0" maxOccurs="unbounded" />
793     </xs:choice>
794   </xs:sequence>
795   <xs:anyAttribute namespace="##other" processContents="lax"/>
796 </xs:complexType>
797

```

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

799 **2.3.5. Example Liberty ID-WSF EPRs**

```
800
801 <wsa:EndpointReference
802     notOnOrAfter="2005-08-15T23:18:56Z"
803     ...>
804   <wsa:Address>
805     http://profile-provider.com/profiles/someFoobarProfileAddr
806   </wsa:Address>
807
808   <wsa:Metadata>
809     <ds:Abstract>
810       This is a personal profile containing common name information.
811     </ds:Abstract>
812
813     <ds:ProviderID>http://profile-provider.com/</ds:ProviderID>
814
815     <ds:ServiceType>urn:liberty:id-sis-pp:2003-08</ds:ServiceType>
816
817     <sbf:Framework version="2.0" />
818
819     <ds:SecurityContext>
820       <ds:SecurityMechID>
821         urn:liberty:security:2006-08:ClientTLS:SAMLV2
822       </ds:SecurityMechID>
823
824       <ds:SecurityMechID>
825         urn:liberty:security:2005-02:ClientTLS:SAML
826       </ds:SecurityMechID>
827
828       <sec:Token wsu:id="_10"
829         usage="urn:liberty:security:tokenusage:2006-08:SecurityToken">
830         <!-- some security token goes here -->
831       </sec:Token>
832     </ds:SecurityContext>
833
834     <ds:SecurityContext >
835       <ds:SecurityMechID>
836         urn:liberty:security:2005-02:ClientTLS:X509
837       </ds:SecurityMechID>
838
839       <sec:Token wsu:id="_20"
840         usage="urn:liberty:security:tokenusage:2006-08:InvocationIdentity">
841         <!-- Identity Token goes here -->
842       </sec:Token>
843     </ds:SecurityContext>
844
845     <ds:Options>
846       <ds:Option>urn:liberty:id-sis-pp</ds:Option>
847       <ds:Option>urn:liberty:id-sis-pp:cn</ds:Option>
848       <ds:Option>urn:liberty:id-sis-pp:can</ds:Option>
849       <ds:Option>urn:liberty:id-sis-pp:can:cn</ds:Option>
850     </ds:Options>
851   </wsa:Metadata>
852 </wsa:EndpointReference>
853
854 <wsa:EndpointReference
855     notOnOrAfter="2005-08-15T23:18:56Z"
856     ...>
857   <wsa:Address>
858     http://profile-provider.com/profiles/anotherFoobarProfileEndpointAddr
859   </wsa:Address>
860
861   <wsa:Metadata>
862     <ds:Abstract>
863       This is a personal profile containing common name information.
```

```
864     </ds:Abstract>
865
866     <ds:ProviderID>http://profile-provider.com/</ds:ProviderID>
867
868     <ds:ServiceType>urn:liberty:id-sis-pp:2003-08</ds:ServiceType>
869
870     <sb:Framework version="2.0" />
871
872     <ds:SecurityContext>
873       <ds:SecurityMechID>
874         urn:liberty:security:2006-08:TLS:SAMLV2
875       </ds:SecurityMechID>
876
877       <sec:Token ref="_10" usage="urn:liberty:security:tokenusage:2006-08:SecurityToken" />
878     </ds:SecurityContext>
879
880     <ds:Options>
881       <ds:Option>urn:liberty:id-sis-pp</ds:Option>
882       <ds:Option>urn:liberty:id-sis-pp:cn</ds:Option>
883       <ds:Option>urn:liberty:id-sis-pp:can</ds:Option>
884       <ds:Option>urn:liberty:id-sis-pp:can:cn</ds:Option>
885     </ds:Options>
886   </wsa:Metadata>
887 </wsa:EndpointReference>
888
```

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

## 891 2.4. Service Metadata

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

### 894 2.4.1. Service Metadata element

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

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

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

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

```

907
908 <!-- Service Metadata (SvcMD) - metadata about service instance -->
909
910 <xs:element name="SvcMD" type="SvcMetadataType"/>
911 <xs:complexType name="SvcMetadataType">
912   <xs:sequence>
913     <xs:element ref="Abstract"           />
914     <xs:element ref="ProviderID"        />
915     <xs:element ref="ServiceContext"    maxOccurs="unbounded" />
916   </xs:sequence>
917   <xs:attribute name="svcMDID" type="xs:string" use="optional" />
918 </xs:complexType>
919
920 <!-- ServiceContext - describes service type/option/endpoint context -->
921 <xs:element name="ServiceContext" type="ServiceContextType"/>
922 <xs:complexType name="ServiceContextType">
923   <xs:sequence>
924     <xs:element ref="ServiceType"      maxOccurs="unbounded" />
925     <xs:element ref="Options"          minOccurs="0"
926                                     maxOccurs="unbounded" />
927     <xs:element ref="EndpointContext"  maxOccurs="unbounded" />
928   </xs:sequence>
929 </xs:complexType>
930
931 <!-- EndpointContext - describes endpoints used to access service -->
932 <xs:element name="EndpointContext" type="EndpointContextType"/>
933 <xs:complexType name="EndpointContextType">
934   <xs:sequence>
935     <xs:element ref="Address"          maxOccurs="unbounded" />
936     <xs:element ref="sbf:Framework"   maxOccurs="unbounded" />
937     <xs:element ref="SecurityMechID"   maxOccurs="unbounded" />
938     <xs:element ref="Action"           minOccurs="0"
939                                     maxOccurs="unbounded" />
940   </xs:sequence>
941 </xs:complexType>
942
943 <!-- SvcMD ID element used to refer to Service Metadata elements -->
944 <xs:element name="SvcMDID" type="xs:string" />
945

```

Figure 13. Service Metadata — Schema Fragment

#### 2.4.1.1. svcMDID

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

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

#### 2.4.1.2. Abstract

A text description of the service.

#### 2.4.1.3. ProviderID

The URI of the provider of this service instance.

#### 2.4.1.4. ServiceContext

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

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

#### 961 **2.4.1.4.1. ServiceType**

962 The URI of which defines the type of service.

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

#### 966 **2.4.1.4.2. Option**

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

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

#### 969 **2.4.1.4.3. EndpointContext**

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

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

974 The sub-elements include:

##### 975 **2.4.1.4.3.1. Address**

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

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

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

##### 982 **2.4.1.4.3.2. Framework**

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

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

##### 987 **2.4.1.4.3.3. SecurityMechID**

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

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

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

##### 994 **2.4.1.4.3.4. Action**

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

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

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

## 999 **2.4.2. Minting ID-WSF EPRs based upon Service Metadata**

1000 Service Metadata is stored in the Discovery Service in order to guide the minting of ID-WSF EPRs by the Discovery  
1001 Service in response to queries from WSCs.

1002 One can visualize that the entire set of elements within a single Service Metadata can result in a large number of  
1003 possible EPRs based upon the possible combinations of those elements.

1004 The Discovery Service MUST mint ID-WSF EPRs as if the following process took place (there is NOT a normative  
1005 requirement to implement this exact process, just a requirement that the results generated by whatever process is used  
1006 by the DS MUST result in the set of data that would result from this process).

1007 1. Eliminate portions of the Service Metadata that do not conform to the search requirements (such as unsupported  
1008 (by the WSC) security mechanisms or framework versions, or undesired service types).

1009 2. If an <EndpointContext> element had all occurrences of a given sub-element (such as <Framework>)  
1010 eliminated, eliminate the context.

1011 3. For each remaining <Address>) element within a remaining <EndpointContext> element, an EPR SHOULD  
1012 be minted.

1013 4. For each EPR, assign one <Address>) element to the <wsa:Address>) element in the EPR and use the rest of  
1014 the <EndpointContext> that contained this address to build the necessary <Metadata> sub-elements for the  
1015 ID-WSF EPR (. <SecurityContext>(s), <Action>(s), <Framework>(s), etc.).

1016 5. Fill out the rest of the ID-WSF EPR using the service wide elements ( <Abstract>, <ProviderID>,  
1017 <ServiceType>(s), etc.).

1018 6. If necessary, generate any security and/or identity tokens and place them into the appropriate  
1019 <SecurityContext> element(s).

1020 The set of EPRs generated by this process may be further restricted by the request parameters on the DiscoveryQuery  
1021 operation, see [Section 3.3](#)).

## 1022 **2.4.3. Service Metadata Example**

1023 Some examples to help show how service metadata works.

### 1024 **2.4.3.1. A simple service**

1025 This is an example of a simple service that has a single endpoint, supports a single framework version (2.0), and only  
1026 supports a single security mechanism.

```
1027
1028 <ds:svcMD svcMDID="1234">
1029   <ds:Abstract>This is a simple service metadata definition</ds:abstract>
1030   <ds:ProviderID>http://simpler.providers.com</ds:ProviderID>
1031   <ds:ServiceContext>
1032     <ds:ServiceType>urn:liberty:pp:2003-08</ds:ServiceType>
1033     <ds:EndpointContext>
1034       <ds:Address>https://simple.providers.com/PP</ds:Address>
1035       <sb:Framework version="2.0" />
1036       <ds:SecurityMechID>
1037         urn:liberty:security:2003-08:TLS:Bearer
1038       </ds:SecurityMechID>
1039     </ds:EndpointContext>
1040   </ds:ServiceContext>
1041 </ds:SvcMD>
1042
```

1043 **Figure 14. Service Metadata example: A simple service**

### 1044 2.4.3.2. A complex service

1045 This is an example of a service metadata definition with a number of complex attributes including:

- 1046 • Multiple service versions **and** multiple framework versions on the same endpoint.
- 1047 • There are two service contexts, one for one version of the service and one for a different version of the service.  
1048 So, for example, the 2003-08 version of the service is only available at the URL *https://old.providers.com/PP* and  
1049 only for framework version *1.1*
- 1050 • Multiple interfaces on different endpoints with different security mechanisms
- 1051 • There are multiple, redundant, addresses for the TLS endpoint for the *2007-11* version of the service.

```

1052
1053 <!-- Service Metadata Example: A complex service -->
1054 <ds:SvcMD svcMDID="4567">
1055   <ds:Abstract>This is a complex service metadata definition</ds:abstract>
1056   <ds:ProviderID>http://complex.providers.com</ds:ProviderID>
1057   <ds:ServiceContext>>
1058     <ds:ServiceType>urn:liberty:pp:2003-08</ds:ServiceType>
1059     <ds:EndpointContext>
1060       <ds:Address>https://old.providers.com/PP</ds:Address>
1061       <sb:Framework version="1.1" />
1062       <ds:SecurityMechID>
1063         urn:liberty:security:2003-08:TLS:Bearer
1064       </ds:SecurityMechID>
1065     </ds:EndpointContext>
1066   </ds:ServiceContext>>
1067   <ds:ServiceContext>>
1068     <ds:ServiceType>urn:liberty:pp:2007-11</ds:ServiceType>
1069     <ds:EndpointContext>
1070       <ds:Address>https://complex.providers.com/PP</ds:Address>
1071       <ds:Address>https://backup.complex.providers.com/PP</ds:Address>
1072       <sb:Framework version="2.0" />
1073       <ds:SecurityMechID>
1074         urn:liberty:security:2003-08:TLS:Bearer
1075       </ds:SecurityMechID>
1076     </ds:EndpointContext>
1077     <ds:EndpointContext>
1078       <ds:Address>http://complex.providers.com/PP</ds:Address>
1079       <sb:Framework version="2.0" />
1080       <ds:SecurityMechID>
1081         urn:liberty:security:2003-08:null:SAMLV2
1082       </ds:SecurityMechID>
1083     </ds:EndpointContext>
1084   </ds:ServiceContext>>
1085 </ds:SvcMD>
1086

```

Figure 15. Service Metadata example: A complex service

### 2.4.3.3. Another complex service

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

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

```
1092
1093 <!-- Service Metadata Example: A simple service -->
1094 <ds:SvcMD svcMDID="8901">
1095   <ds:Abstract>Another example complex service</ds:abstract>
1096   <ds:ProviderID>http://split.providers.com</ds:ProviderID>
1097   <ds:ServiceContext>>
1098     <ds:ServiceType>urn:liberty:pp:2003-08</ds:ServiceType>
1099     <ds:EndpointContext>
1100       <ds:Address>https://cluster1.split.providers.com/PP</ds:Address>
1101       <sb:Framework version="2.0" />
1102       <ds:SecurityMechID>
1103         urn:liberty:security:2003-08:TLS:Bearer
1104       </ds:SecurityMechID>
1105       <ds:Action>urn:liberty:pp:2003-08:Query</ds:Action>
1106     </ds:EndpointContext>
1107     <ds:EndpointContext>
1108       <ds:Address>https://writer.split.providers.com/PP</ds:Address>
1109       <sb:Framework version="2.0" />
1110       <ds:SecurityMechID>
1111         urn:liberty:security:2003-08:TLS:Bearer
1112       </ds:SecurityMechID>
1113       <ds:Action>urn:liberty:pp:2003-08:Modify</ds:Action>
1114     </ds:EndpointContext>
1115   </ds:ServiceContext>>
1116 </ds:SvcMD>
1117
```

1118

**Figure 16. Service Metadata example: Another complex service**

### 3. Discovery Service

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

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

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

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

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

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

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

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

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

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

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

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

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

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

1161 tokens necessary to access the service. To avoid extra network round-trips, arrangements are made here so that  
 1162 security tokens may be provided as part of the Discovery Service lookup response.

### 1163 3.1. Service URIs

1164 **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>

### 1165 3.2. Status Codes

1166 The following status code strings are defined:

- 1167 • *OK*: message processing succeeded
- 1168 • *Failed*: general failure code
- 1169 • *Forbidden*: the request was denied based on policy
- 1170 • *Duplicate*: the request was denied because it would result in duplicate data in the service
- 1171 • *LogicalDuplicate*: the request was denied because it would result in logically duplicate data in the service
- 1172 • *NoResults*: the query had no matching results
- 1173 • *NotFound*: the specified item(s) were not found

1174 These strings are expected to appear in the "code" attribute of <Status> elements used in SOAP-bound Discovery  
1175 Service protocol messages [LibertySOAPBinding]. Specific uses for the status codes are defined in the processing  
1176 rules for individual messages. The "ref" attribute on the <Status> element is not used in this specification, so it MUST  
1177 NOT appear on Status elements in Discovery Service protocol messages. The contents of the comment attribute are  
1178 not defined by this specification, but it may be used for additional descriptive text intended for human consumption  
1179 (for example, to carry information that will aid debugging).

### 1180 3.3. Operation: *DiscoveryQuery*

1181 The *DiscoveryQuery* WSDL operation enables a requester to obtain an enumeration of ID-WSF EPRs (see [Section 2:](#)  
1182 [Discovery Service Information Model](#)) — the requester sends a <Query> message and receives a <QueryResponse>  
1183 message in return. Also, because a provider hosting a Discovery Service may also be playing other roles on behalf  
1184 of Principals (such as a *Policy Decision Point* or an *Authentication Authority*), the *DiscoveryQuery* operation can also  
1185 function as a security token service, providing the requester with an efficient means of obtaining security tokens that  
1186 may be necessary to invoke service instances described in the <QueryResponse>.

#### 1187 3.3.1. wsa:Action values for DiscoveryQuery Messages

1188 <Query> messages MUST include a <wsa:Action> SOAP header with the value of urn:liberty:disco:2006-08:Query.

1189 <QueryResponse> messages MUST include a <wsa:Action> SOAP header with the value of  
1190 urn:liberty:disco:2006-08:QueryResponse.

#### 1191 3.3.2. <Query> Message

1192 A <Query> request is an attempt to retrieve ID-WSF EPRs suitable for use in the same identity context that was used  
1193 to make the request. In particular, the Target Identity (See [LibertySOAPBinding]), if applicable, is used to restrict the  
1194 results to just those for the specified principal. The Invocation Identity will be verified against an access control list to  
1195 ensure that they have access to the requested results.

1196 A <Query> request message is empty in the minimal case. Such a request indicates the requester is requesting all  
1197 available ID-WSF EPRs, regardless of security mechanisms or service types. The result set is dependant upon the  
1198 local access control policies of the discovery service instance.

1199 Alternatively, a request can be qualified with a set of <RequestedService> elements, which enables the requester to  
1200 specify that all ID-WSF EPRs returned must be offered via one or more service instances complying with the specified  
1201 search criteria. For each <RequestedService> specified, the requester specifies the search criteria for the DS to use  
1202 in determining if there is a matching instance. The search criteria includes zero or more of any of the following:

1203 • <ServiceType> the requested type of service. Service Type URIs are defined by the individual service  
1204 specifications and contain both service class and service versioning information.

1205 Multiple <ServiceType>s MAY be specified in a single <RequestedService>s element in order to allow the  
1206 WSC to specify what the WSC considers to be different versions of the same service.

1207 When multiple entries are listed, the order of such entries is an indication of the preference as to which  
1208 <ServiceType> the WSC would prefer to see in the results, with the first being the most preferred. This  
1209 typically only impacts a request where the WSC indicates that they only want a subset of the results returned (see  
1210 resultsType below).

1211 When a request results in multiple ID-WSF EPRs in a response, the preference order specified by the WSC on the  
1212 request MAY have no impact on the order of results returned by the Discovery Services. The Discovery Service  
1213 is free to return the results of the request in whatever order it chooses.

1214 If not specified, then any service instance would be considered a match for this criteria.

1215 A service instance must support at least one of the specified service types in order to be considered a match for  
1216 this criteria.

- 1217 • `<ProviderID>` the requested provider ID(s). This is used when the WSC wants to communicate with a particular  
1218 WSP. Frequently such requests are made without specifying a `<ServiceType>` element in the request, but doing  
1219 so is not prohibited.
- 1220 If not specified, then any service instance would be considered a match for this criteria.
- 1221 A service instance must contain at least one of the specified providerIDs in order to be considered a match for this  
1222 criteria.
- 1223 The order of the `<ProviderID>` elements is an indication as to the preference of the requester with the first such  
1224 element being the most desired (declining preference order). The Discovery Services is free to return the results  
1225 of the request in whatever order it chooses.
- 1226 • `<Options>` — an optional multi-occurrence element defining options SETs desired for the service.
- 1227 If not specified, any service instance will be considered a match for this criteria.
- 1228 An option SET is defined within each `<Options>` element and contains a list of the desired options. The service  
1229 instance MUST support ALL of the options within the option SET in order to be considered a match for this  
1230 request.
- 1231 If more than one `<Options>` element is specified (thus defining multiple option SETs), service instances that  
1232 match ANY of the SETS are considered a match for this request. As noted above, to match a SET, you have to  
1233 match ALL of the entries within the SET.
- 1234 Service instance EPRs registered without an `<Options>` element are always considered a match from the point of  
1235 view of any possible `<Options>` search criteria.
- 1236 The order of the `<options>` elements is an indication as to the preference of the requester with the first such  
1237 element being the most desired (declining preference order). The Discovery Services is free to return the results  
1238 of the request in whatever order it chooses.
- 1239 • `<SecurityMechID>` - an optional multi-occurrence element specifying the security mechanism identifier(s) (see  
1240 [[LibertySecMech](#)]) that the WSC is willing to use to invoke the WSP. If not specified, any security mechanism  
1241 registered for a service will be considered a match for this criteria.
- 1242 A service instance MUST support at least one of the requested security mechanisms in order to be considered a  
1243 match for this request.
- 1244 The order of the `<SecurityMechID>` elements is an indication as to the preference of the requester with the first  
1245 such element being the most desired (declining preference order). The Discovery Services is free to return the  
1246 results of the request in whatever order it chooses.
- 1247 • `<Framework>` — an optional multi-occurrence element specifying the framework description(s) supported by the  
1248 WSC.
- 1249 If not specified, the Discovery Service SHOULD use the value of the framework description used in the ID-WSF  
1250 framework layer for the current request (e.g., if the call to the Discovery Service was made using an ID-WSF  
1251 version 2.0 message the request SHOULD be treated as if a `<disco:Framework>` element was present and  
1252 contained the value specified in the `<sbf:Framework>` SOAP header.
- 1253 Multiple `<disco:Framework>` elements MAY be specified, indicating that the WSC has the capability to support  
1254 ANY of the specified versions. The order of elements in such a case indicates the WSC's preference with the most  
1255 preferred coming first.
- 1256 Note that while both the `<disco:Framework>` and the `<sbf:Framework>` elements are of the same type  
1257 (`<sbf:FrameworkType>`), the elements themselves are in different namespaces. The element within the  
1258 `<RequestedService>` is in the Discover Service Namespace, while the element within any ID-WSF EPRs and  
1259 the SOAP header block on an ID-WSF message are in the SOAP Bindings namespace.

1260 • `<Action>` — an optional multi-occurrence element specifying the `wsa:Action` value(s) for the interfaces of the  
1261 service that the WSC intends to make use of.

1262 If not specified, the Discovery Service SHOULD treat this request as a request for all of the interfaces at the  
1263 requested specified instance.

1264 Unlike the other sub-elements of the `<RequestedService>` element, if multiple `<Action>` elements are  
1265 specified it indicates that the WSC intends to invoke **all** of the specified interfaces and the Discovery Service  
1266 SHOULD return the set of EPRs that are necessary to reach the complete set of specified interfaces.

1267 Services registered without an `<Action>` element (which is the norm) are treated as exposing **all** of the interfaces  
1268 defined for that type of service.

1269 The Discovery Service will return the set of EPRs for service instances that intersect with the search criteria specified  
1270 in the `<RequestedService>` element. This may result in a single EPR in the response or it may result in a multitude  
1271 of EPRs, depending upon the search criteria and the service instance definitions available (see [Section 2.3.3: ID-WSF](#)  
1272 [Web Services Addressing EPR Profile](#) ).

1273 The result set of EPRs generated in response to a particular `<RequestedService>` element can be further controlled  
1274 using the following attributes:

1275 • `reqID` — an optional attribute identifying this `<RequestedService>` request. Typically only used when  
1276 multiple `<RequestedService>` elements are included in a single Discovery Service `<Query>`.

1277 If present the value of this attribute will be placed into the `reqRef` attribute in any EPRs that result from this  
1278 `<RequestedService>` element (see [Section 2.3.1.2](#) above).

1279 The value of `reqID` SHOULD be different for all `<RequestedService>` elements in a given `<Query>`.

1280 • `resultsType` — an optional attribute describing the results desired by the requestor. This value may be set to:

1281 • **best** — the Discovery Service SHOULD return the smallest set of EPRs while still meeting the minimum  
1282 requirements of the request. This will typically be a single EPR.

1283 • **all** — the Discovery Service SHOULD return all of the matching entries for the given search criteria.

1284 This would typically be used when the client wants to choose which EPRs within the DS database it should  
1285 use.

1286 This option should be used with caution as it can cause the DS to perform substantial work in order to mint all  
1287 of the matching EPRs and the necessary security tokens for those EPRs.

1288 • **only-one** — a restricted version of **best** which further restricts the resulting output to **exactly one** EPR, even if  
1289 the minimum requirements of the request would require multiple EPRs. A client would typically specify this  
1290 option if it was going to ignore anything other than the first EPR returned.

1291 If `resultsType` is not specified the Discovery Service may make its own determination (under local policy) as to  
1292 which set of results to return.

1293 Requestors SHOULD include at least one <ServiceType> or <ProviderID> element, and MAY include any number  
 1294 of both of them.

1295 Requestors SHOULD construct a Query to be as qualified as possible, as the Discovery Service instance may have to  
 1296 perform significant work for each item in the result set, especially if security tokens will be generated.

```

1297
1298 <!-- Query Message Element & Type -->
1299
1300 <xs:element name="Query" type="QueryType"/>
1301
1302 <xs:complexType name="QueryType">
1303   <xs:sequence>
1304     <xs:element name="RequestedService"
1305       type="RequestedServiceType"
1306       minOccurs="0"
1307       maxOccurs="unbounded" />
1308   </xs:sequence>
1309
1310   <xs:anyAttribute namespace="##other" processContents="lax" />
1311 </xs:complexType>
1312
1313 <xs:complexType name="RequestedServiceType">
1314   <xs:sequence>
1315     <xs:element ref="ServiceType" minOccurs="0" maxOccurs="unbounded" />
1316
1317     <xs:element ref="ProviderID" minOccurs="0" maxOccurs="unbounded" />
1318
1319     <xs:element ref="Options" minOccurs="0" maxOccurs="unbounded" />
1320
1321     <xs:element ref="SecurityMechID" minOccurs="0" maxOccurs="unbounded" />
1322
1323     <xs:element ref="Framework" minOccurs="0" maxOccurs="unbounded" />
1324
1325     <xs:element ref="Action" minOccurs="0" maxOccurs="unbounded" />
1326
1327     <xs:any namespace="##other"
1328       processContents="lax"
1329       minOccurs="0"
1330       maxOccurs="unbounded" />
1331   </xs:sequence>
1332
1333   <xs:attribute name="reqID" type="xs:string" use="optional" />
1334   <xs:attribute name="resultsType" type="xs:string" use="optional" />
1335 </xs:complexType>
1336
1337
1338
1339

```

1340 **Figure 17. Query Message — Schema Fragment**

```
1341
1342 <soap:Envelope
1343   xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
1344
1345   <soap:Header>
1346     ...
1347   </soap:Header>
1348
1349   <soap:Body>
1350     <Query xmlns="&DS2Namespace;">
1351       <RequestedService>
1352         <ServiceType>urn:liberty:id-sis-pp:2003-08</ds:ServiceType>
1353
1354         <SecurityMechID>urn:liberty:security:2006-08:ClientTLS:SAMLV2</SecurityMechID>
1355         <SecurityMechID>urn:liberty:security:2005-02:ClientTLS:SAML</SecurityMechID>
1356         <SecurityMechID>urn:liberty:security:2006-08:TLS:SAMLV2</SecurityMechID>
1357         <Framework version="2.0" />
1358
1359       </RequestedService>
1360     </Query>
1361   </soap:Body>
1362 </soap:Envelope>
1363
```

1364 **Example 5. SOAP message containing a Query**

### 1365 3.3.3. QueryResponse

1366 A <QueryResponse> message conveys the results of the query as a set of ID-WSF EPRs, i.e., profiled  
1367 <wsa:EndpointReference> elements (see [Section 2.3.3: ID-WSF Web Services Addressing EPR Profile](#)  
1368 ).

1369 As specified in [Section 2.3.3](#), security tokens, appropriate for subsequent invocation(s) of the service instances  
1370 represented by the returned ID-WSF EPRs, MAY be provided within the ID-WSF EPRs in the response.

1371 A status code is also included in the response.

```
1372
1373 <!-- QueryResponse Message Element & Type -->
1374
1375 <xs:element name="QueryResponse" type="QueryResponseType"/>
1376
1377 <xs:complexType name="QueryResponseType">
1378   <xs:sequence>
1379     <xs:element ref="lu:Status"/>
1380
1381     <xs:element ref="wsa:EndpointReference"
1382       minOccurs="0"
1383       maxOccurs="unbounded"/>
1384   </xs:sequence>
1385   <xs:anyAttribute namespace="##other" processContents="lax"/>
1386 </xs:complexType>
1387
1388
```

1389 **Figure 18. <QueryResponse> — Schema Fragment**

1390 An example SOAP message containing a <QueryResponse> message is illustrated in [Example 6](#). This example  
1391 includes a security token embedded in the returned ID-WSF EPR. Parts of the security token have been omitted due  
1392 to size.

```

1393
1394 <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
1395
1396   <soap:Header>
1397     ...
1398   </soap:Header>
1399
1400   <soap:Body>
1401     <QueryResponse xmlns="&DS2Namespace;">
1402       <Status code="OK"/>
1403
1404       <wsa:EndpointReference
1405         notOnOrAfter="2005-08-15T23:18:56Z" >
1406         <wsa:Address>http://example.com/pip/bob</wsa:Address>
1407
1408         <wsa:Metadata>
1409           <ds:Abstract>
1410             Bob's personal profile
1411           </ds:Abstract>
1412
1413           <ds:ProviderID>http://example.com/</ds:ProviderID>
1414
1415           <ds:ServiceType>urn:liberty:id-sis-pp:2003-08</ds:ServiceType>
1416
1417           <ds:FrameworkVersion="2.0" />
1418
1419           <ds:SecurityContext>
1420             <ds:SecurityMechID>urn:liberty:security:2006-08:ClientTLS:SAMLV2</ds:SecurityMechID>
1421             <ds:SecurityMechID>urn:liberty:security:2005-02:ClientTLS:SAML</ds:SecurityMechID>
1422
1423             <sec:Token usage="urn:liberty:security:tokenusage:2006-08:SecurityToken" >
1424               <saml2:Assertion xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
1425                 ID="sxJu9g/vvLG9sAN9bKp/8q0NKU="
1426                 Issuer="idp.example.com"
1427                 IssueInstant="2003-09-09T16:58:33.173Z">
1428                 <ds:Signature>...</ds:Signature>
1429                 <saml2:Subject>
1430                   <saml2:NameID Format="urn:oasis:names:tc:SAML:2.0:nameid-format:entity">
1431                     http://serviceprovider.com/
1432                   </saml2:NameID>
1433
1434                   <saml2:SubjectConfirmation Method="urn:oasis:names:tc:SAML:2.0:cm:holder-of-key">
1435                     <saml2:SubjectConfirmationData xsi:type="saml:KeyInfoConfirmationDataType">
1436                       <ds:KeyInfo>
1437                         <ds:KeyName>
1438                           CN=serviceprovider.com,
1439                           OU=Services R US,O=Service Nation,...
1440                         </ds:KeyName>
1441                       </ds:KeyInfo>
1442                     </saml2:SubjectConfirmationData>
1443                   </saml2:SubjectConfirmation>
1444                 </saml2:Subject>
1445
1446                 <saml2:AuthnStatement AuthnInstant="2003-09-09T16:57:30.000Z"
1447                   SessionIndex="..."
1448                   SessionNotOnOrAfter="..."
1449                 >
1450                   <saml2:AuthnContext>
1451                     ...
1452                   </saml2:AuthnContext>
1453                 </saml2:AuthnStatement>
1454
1455               </saml2:Assertion>
1456             </sec:Token>
1457           </ds:SecurityContext>
1458         </wsa:Metadata>
1459       </wsa:EndpointReference>

```

```
1460     </QueryResponse>
1461 </soap:Body>
1462 </soap:Envelope>
1463
```

1464 **Example 6. SOAP-bound <QueryResponse> Message with Embedded Security Token**

### 1465 3.3.4. DiscoveryQuery Processing Rules

1466 The discovery Service returns entries based on the requester's search criteria (interpreted as described above in  
1467 [Section 3.3.2: <Query> Message](#) ), the policies of the discovery resource, and the contents of the discovery resource.

1468 For each <RequestedService> element in a <Query> message, the matching rules **MUST** applied independently  
1469 (as if the other <RequestedService> elements were not present (potentially returning equivalent EPRs in response  
1470 to different <RequestedService> elements).

1471 When building the results for a given <RequestedService> element, the Discovery Service **SHOULD** return the  
1472 data in as few EPRs as possible (within the constraints of the EPR) especially with respect to data originating from the  
1473 same SvcMD.

1474 The Discovery Service **SHOULD**, when possible, provide the security tokens necessary for the security mechanism(s)  
1475 identified in the ID-WSF EPRs in the response. If the Discovery Service is not able to generate the necessary security  
1476 token, it should indicate so by including an empty <sec:Token> element with the `ref` attribute set to the value:

```
1477     urn:liberty:disco:tokenref:ObtainFromIDP
```

1478 The Discovery Service **SHOULD** mint new EPRs such that they carry the same identity context that was used to invoke  
1479 the Discovery Service in the invocation context for the targeted WSP.

1480 When minting EPRs in response to a request, the Discovery Service:

- 1481 • **MUST** include in the EPR the matching (or all, if none were specified on the request) <Option> values contained  
1482 within the <ServiceContext> of the <SvcMD> that matched the request criteria.
- 1483 • **SHOULD NOT** include any data from the <SvcMD> that was not specified in the request **and** is not necessary to  
1484 create a useful EPR. For example, if the request specified a single <SecurityMechID> value, the resulting EPRs  
1485 should not include other <SecurityMechID> elements, even if they are present in the <SvcMD> and otherwise  
1486 could be specified in the same EPR.

1487 The Discovery Service **MAY** order <wsa:EndpointReference> elements as it sees fit. If the Discovery Service is  
1488 rank ordering the entries, it **MUST** use descending rank order. This enables the requester to assume that if the results  
1489 were ordered, the first result is the most relevant.

1490 The following rules specify the status code in the response:

- 1491 • If request processing succeeded, the top-level status code **MUST** be *OK*. Otherwise, the top-level status code  
1492 **MUST** be *Failed*.
  - 1493 • If the top-level status code is *Failed*, the response **MAY** also contain *Forbidden* or *NoResults* as a second-level  
1494 status code.
- 1495 The service may not wish to reveal the reason for failure, in which case no second-level status code will appear.

## 3.4. Operation: *MDAssociationAdd*

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

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

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

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

### 3.4.2. *SvcMDAssociationAdd* Message

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

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

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

```
1510 <!-- SvcMDAssociationAdd operation -->
1511
1512 <xs:element name="SvcMDAssociationAdd" type="SvcMDAssociationAddType" />
1513
1514 <xs:complexType name="SvcMDAssociationAddType">
1515   <xs:sequence>
1516     <xs:element ref="SvcMDID" maxOccurs="unbounded" />
1517   </xs:sequence>
1518   <xs:anyAttribute namespace="##other" processContents="lax" />
1519 </xs:complexType>
```

Figure 19. *<SvcMDAssociationAdd>* — Schema Fragment

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

```
1527 <ds:SvcMDAssociationAdd>
1528   <ds:SvcMDID>2323872</ds:SvcMDID>
1529 </ds:SvcMDAssociationAdd>
```

Example 7. *<SvcMDAssociationAdd>* Message

### 3.4.3. *SvcMDAssociationAddResponse* Message

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

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

```

1536
1537 <!-- Response for SvcMDAssociationAdd operation -->
1538
1539 <xs:element name="SvcMDAssociationAddResponse"
1540           type="SvcMDAssociationAddResponseType"/>
1541
1542 <xs:complexType name="SvcMDAssociationAddResponseType">
1543   <xs:sequence>
1544     <xs:element ref="lu:Status" />
1545   </xs:sequence>
1546   <xs:anyAttribute namespace="##other" processContents="lax"/>
1547 </xs:complexType>
1548

```

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

```

1550
1551 <ds:SvcMDAssociationAddResponse>
1552   <lu:Status code="OK" />
1553 </ds:SvcMDAssociationAddResponse>
1554

```

**Example 8.** <SvcMDAssociationAddResponse> Message

### 3.4.4. MDAssociation Add Processing Rules

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

## 3.5. Operation: *MDAssociationQuery*

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

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

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

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

### 3.5.2. *SvcMDAssociationQuery* Message

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

```

1593 <!-- SvcMDAssociationQuery operation -->
1594 <xs:element name="SvcMDAssociationQuery" type="SvcMDAssociationQueryType" />
1595
1596 <xs:complexType name="SvcMDAssociationQueryType">
1597   <xs:sequence>
1598     <xs:element ref="SvcMDID" minOccurs="0" maxOccurs="unbounded" />
1599   </xs:sequence>
1600   <xs:anyAttribute namespace="##other" processContents="lax" />
1601 </xs:complexType>
1602
1603
1604

```

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

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

```

1608 <ds:SvcMDAssociationQuery />
1609
1610

```

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

### 3.5.3. *SvcMDAssociationQueryResponse* Message

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

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

```
1618
1619 <!-- Response for SvcMDAssociationQuery operation -->
1620
1621 <xs:element name="SvcMDAssociationQueryResponse"
1622     type="SvcMDAssociationQueryResponseType" />
1623
1624 <xs:complexType name="SvcMDAssociationQueryResponseType">
1625     <xs:sequence>
1626         <xs:element ref="lu:Status" />
1627         <xs:element ref="SvcMDID" minOccurs="0" maxOccurs="unbounded" />
1628     </xs:sequence>
1629     <xs:anyAttribute namespace="##other" processContents="lax" />
1630 </xs:complexType>
1631
```

1632 **Figure 22. <SvcMDAssociationQueryResponse> — Schema Fragment**

```
1633
1634 <ds:SvcMDAssociationQueryResponse>
1635     <lu:Status code="OK" />
1636     <ds:SvcMDID>2323872</ds:SvcMDID>
1637 </ds:SvcMDAssociationQueryResponse>
1638
```

1639 **Example 10. <SvcMDAssociationQueryResponse> Message**

### 1640 3.5.4. MDAssociation Query Processing Rules

- 1641 • The Discovery Service **MUST** limit the operation to only those associations added by the WSP to the current  
1642 principal's resource (a WSP **MUST NOT** be able to query associations added at the same Discovery Service by  
1643 other WSPs or associations added to a different principal). There **MUST NOT** be any indication on the response  
1644 as to whether or not other such elements exist.
- 1645 • If request processing succeeded, the top-level status code **MUST** be *OK*. Otherwise, the top-level status code  
1646 **MUST** be *Failed*.
- 1647 • If the top-level status code is *Failed*, the response **MAY** also contain *Forbidden* or *NotFound* as a second-level  
1648 status code. The Discovery Service instance may not wish to reveal the reason for failure, in which case no  
1649 second-level status code will appear.

## 1650 3.6. Operation: *MDAssociationDelete*

1651 The *MDAssociationDelete* operation is used by the WSP to delete a previously added association of the principal to  
1652 the specified metadata.

### 1653 3.6.1. wsa:Action values for MDAssociationDelete Messages

1654 <SvcMDAssociationDelete> messages **MUST** include a <wsa:Action> SOAP header with the value of  
1655 "urn:liberty:disco:2006-08:SvcMDAssociationDelete."

1656 <SvcMDAssociationDeleteResponse> messages **MUST** include a <wsa:Action> SOAP header with the value  
1657 of "urn:liberty:disco:2006-08:SvcMDAssociationDeleteResponse."

### 1658 3.6.2. SvcMDAssociationDelete Message

1659 The <SvcMDAssociationDelete> is called with one or more <SvcMDID> elements to delete associations to these  
1660 service metadata descriptions.

1661 Note that the service metadata description is not impacted by this call. Only the principal's association with the  
1662 metadata is impacted.

```
1663
1664 <!-- SvcMDAssociationDelete operation -->
1665
1666 <xs:element name="SvcMDAssociationDelete" type="SvcMDAssociationDeleteType"/>
1667
1668 <xs:complexType name="SvcMDAssociationDeleteType">
1669   <xs:sequence>
1670     <xs:element ref="SvcMDID" maxOccurs="unbounded" />
1671   </xs:sequence>
1672   <xs:anyAttribute namespace="##other" processContents="lax"/>
1673 </xs:complexType>
1674
```

1675 **Figure 23. <SvcMDAssociationDelete> — Schema Fragment**

1676 An example message body containing a <SvcMDAssociationDelete> message follows. This request deletes a  
1677 single association for the current principal (note that the identity of the principal is carried in the invocation context  
1678 and not in the body of the message).

```
1679
1680 <ds:SvcMDAssociationDelete>
1681   <ds:SvcMDID>2323872</ds:SvcMDID>
1682 </ds:SvcMDAssociationDelete>
1683
```

1684 **Example 11. <SvcMDAssociationDelete> Message**

### 1685 3.6.3. SvcMDAssociationDeleteResponse Message

1686 This response to the <SvcMDAssociationDelete> request contains the following elements and attributes.

- 1687 • <lu:Status>: Contains status code; see processing rules.

```
1688
1689 <!-- Response for SvcMDAssociationDelete operation -->
1690
1691 <xs:element name="SvcMDAssociationDeleteResponse"
1692   type="SvcMDAssociationDeleteResponseType"/>
1693
1694 <xs:complexType name="SvcMDAssociationDeleteResponseType">
1695   <xs:sequence>
1696     <xs:element ref="lu:Status" />
1697   </xs:sequence>
1698   <xs:anyAttribute namespace="##other" processContents="lax"/>
1699 </xs:complexType>
1700
```

1701 **Figure 24. <SvcMDAssociationDeleteResponse> — Schema Fragment**

```
1702
1703 <ds:SvcMDAssociationDeleteResponse>
1704   <lu:Status code="OK" />
1705 </ds:SvcMDAssociationDeleteResponse>
1706
```

1707 **Example 12. <SvcMDAssociationDeleteResponse> Message**

### 1708 3.6.4. MDAssociation Delete Processing Rules

- 1709 • Once deleted, the association **MUST NOT** be subsequently used by the DS to mint ID-WSF EPRs in response to  
1710 queries relative to this principal. However, WSPs should be prepared to receive requests from WSCs from clients  
1711 who previously obtained ID-WSF EPRs minted from the associaton which haven't expired.
- 1712 • The Discovery Service **MUST** limit the operation to only those associations added by the WSP to the current  
1713 principal's resource (a WSP **MUST NOT** be able to delete associations added at the same Discovery Service by  
1714 other WSPs or associations added to a different principal). There **MUST NOT** be any indication on the response  
1715 as to whether or not other such elements exist.
- 1716 • The Discovery Service **MUST** treat attempts to delete non-existent associations as a successful no-op. This applies  
1717 whether or not there are other existing associations being deleted in the same request (so a request to delete a single  
1718 association that doesn't exist will succeed, even though the Discovery Service does not have to actually delete the  
1719 record).
- 1720 • This operation **MUST** be atomic and if successful, all portions of the request **MUST** have succeeded. If any  
1721 portion of the request fails, the entire request must fail.
- 1722 • If request processing succeeded, the top-level status code **MUST** be *OK*. Otherwise, the top-level status code  
1723 **MUST** be *Failed*.
- 1724 • If the top-level status code is *Failed*, the response **MAY** also contain *Forbidden* as a second-level status code. The  
1725 Discovery Service instance may not wish to reveal the reason for failure, in which case no second-level status code  
1726 will appear.

## 1727 **3.7. Operation: *MetadataRegister***

1728 The *MetadataRegister* operation is used to register a new service metadata description with the Discovery Service.

### 1729 **3.7.1. *wsa:Action* values for *MetadataRegister* Messages**

1730 <SvcMDRegister> messages **MUST** include a <wsa:Action> SOAP header with the value of  
1731 "urn:liberty:disco:2006-08:SvcMDRegister."

1732 <SvcMDRegisterResponse> messages **MUST** include a <wsa:Action> SOAP header with the value of  
1733 "urn:liberty:disco:2006-08:SvcMDRegisterResponse."

### 1734 **3.7.2. *SvcMDRegister* Message**

1735 The <SvcMDRegister> is called with one or more service metadata descriptions to be registered at the Discovery  
1736 Service on behalf of the WSP.

```

1737 <!-- Register operation for Service Metadata -->
1738
1739 <xs:element name="SvcMDRegister" type="SvcMDRegisterType"/>
1740
1741 <xs:complexType name="SvcMDRegisterType">
1742   <xs:sequence>
1743     <xs:element ref="SvcMD" maxOccurs="unbounded" />
1744   </xs:sequence>
1745   <xs:anyAttribute namespace="##other" processContents="lax"/>
1746 </xs:complexType>
1747
1748
1749

```

1750 **Figure 25. <SvcMDRegister> — Schema Fragment**

1751 An example message body containing a <SvcMDRegister> message follows. This request registers a new service  
1752 metadata description. Note that the WSP has not set the svcMDID attribute on the <SvcMD> element – this will be  
1753 assigned by the DS and returned in the response to the WSP.

```
1754 <ds:SvcMDRegister>
1755   <ds:SvcMD>
1756     <ds:Abstract>Profile Service</ds:abstract>
1757     <ds:ProviderID>http://profile.com</ds:ProviderID>
1758     <ds:ServiceContext>
1759       <ds:ServiceType>urn:liberty:pp:2003-08</ds:ServiceType>
1760       <ds:EndpointContext>
1761         <ds:Address>https://profile.com/</ds:Address>
1762         <sb:Framework version="2.0" />
1763         <ds:SecurityMechID>
1764           urn:liberty:security:2003-08:TLS:Bearer
1765         </ds:SecurityMechID>
1766       </ds:EndpointContext>
1767     </ds:ServiceContext>
1768   </ds:SvcMD>
1769 </ds:SvcMDRegister>
1770
1771
```

#### 1772 Example 13. <SvcMDRegister> Message

### 1773 3.7.3. SvcMDRegisterResponse Message

1774 This response to the <SvcMDRegister> request contains the following elements and attributes.

- 1775 • <lu:Status>: Contains status code; see processing rules.
- 1776 • One or more <SvcMDID> if the call was successful (status code is OK). One SvcMDID is returned for each service  
1777 metadata element registered.
- 1778 • <Keys>: Contains the key descriptors for the keys used by the Discovery Service to sign security tokens (see  
1779 [Section 3.12](#) for a description of when and why this may be necessary).

```
1780 <!-- Response for SvcMDRegister operation -->
1781
1782 <xs:element name="SvcMDRegisterResponse"
1783   type="SvcMDRegisterResponseType" />
1784
1785 <xs:complexType name="SvcMDRegisterResponseType">
1786   <xs:sequence>
1787     <xs:element ref="lu:Status" />
1788     <xs:element ref="SvcMDID" minOccurs="0" maxOccurs="unbounded" />
1789     <xs:element ref="Keys" minOccurs="0" maxOccurs="unbounded" />
1790   </xs:sequence>
1791   <xs:anyAttribute namespace="##other" processContents="lax" />
1792 </xs:complexType>
1793
1794
1795
1796
1797
```

1798 Figure 26. <SvcMDRegisterResponse> — Schema Fragment

```
1799
1800 <ds:SvcMDRegisterResp>
1801   <lu:Status code="OK" />
1802   <ds:SvcMDID>2323872</ds:SvcMDID>
1803 </ds:SvcMDRegister>
1804
```

1805 **Example 14. <SvcMDRegisterResponse> Message**

### 1806 3.7.4. Metadata Register Processing Rules

- 1807 • This operation **MUST** be processed in the context of the WSP, (as opposed to the context of the principal) so that  
1808 the WSP can maintain a single set of service metadata across all principals at the same Discovery Service.

1809 Even if this operation is invoked with an invocation identity of a principal, the Discovery Service **MUST** use  
1810 the Sender's identity (the WSP) when processing this call. The Discovery Service **MAY** refuse to process the  
1811 operation if the identity of the Sender cannot be established to the Discovery Service's satisfaction.

- 1812 • The transaction unit for this operation is the entire set of <SvcMD> elements; they either all succeed or all fail. The  
1813 Discovery Service **MUST** enforce this atomicity.

- 1814 • For each <SvcMD> element, the Discovery Service instance **MAY** store the metadata provided such that it can be  
1815 used (subject to policy) to mint ID-WSF EPRs in response to future *DiscoveryQuery* operations should that service  
1816 metadata be associated with a principal's resource at the Discovery Service.

1817 If the Discovery Service instance does not store the metadata, it **MUST** return a *Failed* status code for the operation,  
1818 and therefore not register any of the other entries provided.

1819 If the Discovery Service does store the metadata, it **MUST** assign a permanent identifier for the metadata usable  
1820 by the WSP to subsequently reference the metadata. This identifier **MUST** be unique across all metadata objects  
1821 stored by a WSP and **MAY** be unique across all metadata objects stored by all WSPs at that Discovery Service.  
1822 This identifier is provided to the WSP in the response and can be subsequently used by the WSP to associate this  
1823 metadata with a principal or to manage the metadata using one of the other metadata operations.

- 1824 • A WSP **MAY** register multiple service metadata descriptions that for all intents and purposes, appear to be fully  
1825 equal. The Discovery Service **MUST NOT** generate an error solely because it thinks the descriptions are equal.  
1826 The Discovery Service **MUST** treat these records as independent registrations and assign the associated unique  
1827 SvcMDID values.

- 1828 • The Discovery Service **MAY** have some policy driven limit on the number of service metadata descriptions that it  
1829 will allow a WSP to register. If a WSP attempts to register a new service metadata description that would exceed  
1830 such a limit, the DS **SHOULD** include a secondary-level status code of *LimitExceeded*.

1831 A WSP should exercise care to only register new service metadata descriptions when an existing, registered,  
1832 description that meets the WSP's needs is not available.

- 1833 • The Discovery Service **SHOULD** validate that a given SvcMD only contains entries for a single logical service  
1834 (i.e., allow for different versions, but **not** allow differences in the basic service type). For example, a single  
1835 SvcMD **SHOULD** not contain data for both a contact book service and a calendar service.

1836 The Discovery Service **MAY**, subject to local policies, perform additional validations on the content of a SvcMD.

1837 If any validation on the SvcMD fails, the Discovery Service **SHOULD** reject the registration request and **MAY**  
1838 include a secondary-level status code of *Invalid*.

- 1839 • If request processing succeeded, the top-level status code **MUST** be *OK*. Otherwise, the top-level status code  
1840 **MUST** be *Failed*.

1841 • If the top-level status code is *Failed*, the response MAY also contain *Forbidden*, *Invalid*, or *OverLimit* as a second-  
1842 level status code. The Discovery Service instance may not wish to reveal the reason for failure, in which case no  
1843 second-level status code will appear.

## 1844 3.8. Operation: *MetadataQuery*

1845 The *MetadataQuery* operation is used to query the Discovery Service for existing, registered, service metadata  
1846 descriptions.

### 1847 3.8.1. *wsa:Action* values for *MetadataQuery* Messages

1848 <SvcMDQuery> messages MUST set the value of the <wsa:Action> header to "urn:liberty:disco:2006-08:SvcMDQuery."

1849 <SvcMDQueryResponse> messages MUST include a <wsa:Action> SOAP header with the value of  
1850 "urn:liberty:disco:2006-08:SvcMDQueryResponse."

### 1851 3.8.2. *SvcMDQuery* Message

1852 The <SvcMDQuery> is called with zero or more <SvcMDID> elements to retrieve the specified list of service metadata  
1853 descriptions. If no <SvcMDID>s are specified, ALL of the metadata stored at the Discovery service by the invoking  
1854 WSP will be returned.

```
1855 <!-- Query operation on Service Metadata -->
1856
1857 <xs:element name="SvcMDQuery" type="SvcMDQueryType"/>
1858
1859 <xs:complexType name="SvcMDQueryType">
1860 <xs:sequence>
1861 <xs:element ref="SvcMDID"
1862 minOccurs="0"
1863 maxOccurs="unbounded"/>
1864 </xs:sequence>
1865 <xs:anyAttribute namespace="##other" processContents="lax"/>
1866 </xs:complexType>
1867
1868
1869
```

1870 **Figure 27. <SvcMDQuery> — Schema Fragment**

1871 An example message body containing a <SvcMDQuery> message follows. This request queries for a specific service  
1872 metadata description by providing the ID of the desired metadata in the <SvcMDID> element.

```
1873 <ds: SvcMDQuery>
1874 <ds: SvcMDID>2323872</ds: SvcMDID>
1875 </ds: SvcMDQuery>
1876
1877
```

1878 **Example 15. <SvcMDQuery> Message**

### 1879 3.8.3. *SvcMDQueryResponse* Message

1880 This response to the <SvcMDQuery> request contains the following elements and attributes.

- 1881 • <lu:Status>: Contains status code; see processing rules.
- 1882 • One or more <SvcMD> elements if the call was successful (status code is OK).

```
1883
1884 <!-- Response for Query operation on Service Metadata -->
1885
1886 <xs:element name="SvcMDQueryResponse" type="SvcMDQueryResponseType" />
1887
1888 <xs:complexType name="SvcMDQueryResponseType">
1889   <xs:sequence>
1890     <xs:element ref="lu:Status" />
1891     <xs:element ref="SvcMD" minOccurs="0" maxOccurs="unbounded" />
1892   </xs:sequence>
1893   <xs:anyAttribute namespace="##other" processContents="lax" />
1894 </xs:complexType>
1895
1896
```

1897 **Figure 28. <SvcMDQueryResponse> — Schema Fragment**

```
1898
1899 <ds:SvcMDQueryResponse>
1900   <lu:Status code="OK" />
1901   <ds:SvcMD svcMDID="2323872" >
1902     <ds:Abstract>Profile Service</ds:Abstract>
1903     <ds:ProviderID>http://profile.com</ds:ProviderID>
1904     <ds:ServiceContext>
1905       <ds:ServiceType>urn:liberty:pp:2003-08</ds:ServiceType>
1906       <ds:EndpointContext>
1907         <ds:Address>https://profile.com/</ds:Address>
1908         <sb:Framework version="2.0" />
1909         <ds:SecurityMechID>
1910           urn:liberty:security:2003-08:TLS:Bearer
1911         </ds:SecurityMechID>
1912       </ds:EndpointContext>
1913     </ds:ServiceContext>
1914   </ds:SvcMD>
1915 </ds:SvcMDQueryResponse>
1916
```

1917 **Example 16. <SvcMDQueryResponse> Message**1918 

### 3.8.4. Metadata Query Processing Rules

- 1919 • This operation **MUST** be processed in the context of the WSP, (as opposed to the context of the principal) so that  
1920 the WSP can maintain a single set of service metadata across all principals at the same Discovery Service.  
1921 Even if this operation is invoked with an invocation identity of a principal, the Discovery Service **MUST** use  
1922 the Sender's identity (the WSP) when processing this call. The Discovery Service **MAY** refuse to process the  
1923 operation if the identity of the Sender cannot be established to the Discovery Service's satisfaction.
- 1924 • The Discovery Service **MUST** limit the results to only those metadata elements stored by the WSP (a WSP **MUST**  
1925 **NOT** be able to retrieve metadata elements stored at the same Discovery Service by other WSPs). There **MUST**  
1926 **NOT** be any indication on the response as to whether or not other such elements exist.
- 1927 • The Discovery Service **SHOULD** treat a request that matches a subset of the `svcMDID` values specified in the  
1928 request as a successful request returning the entries that were found and nothing for the missing entries. The  
1929 WSC will be able to distinguish which entries were found by examining the `svcMDID` attribute on the `<svcMD>`  
1930 element(s) in the response.
- 1931 • If request processing succeeded **AND** results are returned, the top-level status code **MUST** be *OK*. Otherwise, the  
1932 top-level status code **MUST** be *Failed*.

1933 • If the top-level status code is *Failed*, the response MAY also contain *Forbidden* or *NoResults* as a second-level  
1934 status code. The Discovery Service instance may not wish to reveal the reason for failure, in which case no  
1935 second-level status code will appear.

### 1936 3.9. Operation: *MetadataReplace*

1937 The *MetadataReplace* operation is used by a WSP to replace previously stored metadata in the Discovery Service.  
1938 This is how the WSP updates their metadata without having to reassociate with a principal.

#### 1939 3.9.1. *wsa:Action* values for *MetadataReplace* Messages

1940 <SvcMDReplace> messages MUST include a <wsa:Action> SOAP header with the value of  
1941 "urn:liberty:disco:2006-08:SvcMDReplace."

1942 <SvcMDReplaceResponse> messages MUST include a <wsa:Action> SOAP header with the value of  
1943 "urn:liberty:disco:2006-08:SvcMDReplaceResponse."

#### 1944 3.9.2. *SvcMDReplace* Message

1945 The <SvcMDReplace> is called with one or more replacement <SvcMD> elements each of which must include the  
1946 svcMDID attribute set to the ID of the respective metadata element they are to replace.

```
1947 <!-- Replace operation on Service Metadata -->
1948
1949
1950 <xs:element name="SvcMDReplace" type="SvcMDReplaceType" />
1951
1952 <xs:complexType name="SvcMDReplaceType">
1953   <xs:sequence>
1954     <xs:element ref="SvcMD" maxOccurs="unbounded" />
1955   </xs:sequence>
1956   <xs:anyAttribute namespace="##other" processContents="lax" />
1957 </xs:complexType>
1958
1959
```

1960 **Figure 29. <SvcMDReplace> — Schema Fragment**

1961 An example message body containing a <SvcMDReplace> message follows. This request replaces an existing  
1962 metadata element to update the endpoint for the service.

```
1963 <ds:SvcMDReplace>
1964   <ds:SvcMD svcMDID="2323872" >
1965     <ds:Abstract>Profile Service</ds:abstract>
1966     <ds:ProviderID>http://profile.com</ds:ProviderID>
1967     <ds:ServiceContext>
1968       <ds:ServiceType>urn:liberty:pp:2003-08</ds:ServiceType>
1969       <ds:EndpointContext>
1970         <ds:Address>https://newaddr.com/</ds:Address>
1971         <sb:Framework version="2.0" />
1972         <ds:SecurityMechID>
1973           urn:liberty:security:2003-08:TLS:Bearer
1974         </ds:SecurityMechID>
1975       </ds:EndpointContext>
1976     </ds:ServiceContext>
1977   </ds:SvcMD>
1978 </ds:SvcMDReplace>
1979
1980
```

1981 **Example 17. <SvcMDReplace> Message**

1982 **3.9.3. SvcMDReplaceResponse Message**

1983 This response to the <SvcMDReplace> request contains the following elements and attributes.

- 1984 • <lu:Status>: Contains status code; see processing rules.

```

1985 <!-- Response for SvcMDReplace operation -->
1986
1987 <xs:element name="SvcMDReplaceResponse" type="SvcMDReplaceResponseType"/>
1988
1989 <xs:complexType name="SvcMDReplaceResponseType">
1990   <xs:sequence>
1991     <xs:element ref="lu:Status" />
1992   </xs:sequence>
1993   <xs:anyAttribute namespace="##other" processContents="lax"/>
1994 </xs:complexType>
1995
1996
1997

```

1998 **Figure 30. <SvcMDReplaceResponse> — Schema Fragment**

```

1999 <ds:SvcMDReplaceResponse>
2000   <lu:Status code="OK"/>
2001 </ds:SvcMDReplaceResponse>
2002
2003

```

2004 **Example 18. <SvcMDReplaceResponse> Message**

2005 **3.9.4. Metadata Replace Processing Rules**

- 2006 • This operation **MUST** be processed in the context of the WSP, (as opposed to the context of the principal) so that  
2007 the WSP can maintain a single set of service metadata across all principals at the same Discovery Service.

2008 Even if this operation is invoked with an invocation identity of a principal, the Discovery Service **MUST** use  
2009 the Sender's identity (the WSP) when processing this call. The Discovery Service **MAY** refuse to process the  
2010 operation if the identity of the Sender cannot be established to the Discovery Service's satisfaction.

- 2011 • The Discovery Service **MUST** limit the operation to only those metadata elements stored by the WSP (a WSP  
2012 **MUST NOT** be able to replace metadata elements stored at the same Discovery Service by other WSPs). There  
2013 **MUST NOT** be any indication on the response as to whether or not other such elements exist.

- 2014 • The Discovery Service **SHOULD** validate that the replacement SvcMD contains the same logical service as the  
2015 original SvcMD. By "logical service" we mean to allow for different versions, but **not** allow differences in the  
2016 basic service type. For example, a calendar service SvcMD **SHOULD** not be allowed to replace a contact book  
2017 service SvcMD.

2018 The Discovery Service **SHOULD** also validate that the replacement SvcMD only contains entries for a single  
2019 logical service (as described above). For example, a single SvcMD **SHOULD** not contain data for both a contact  
2020 book service and a calendar service.

2021 The Discovery Service **MAY**, subject to local policies, perform additional validations on the content of a SvcMD.

2022 If any validation on the SvcMD fails, the Discovery Service **SHOULD** reject the replacement request and **MAY**  
2023 include a secondary-level status code of *Invalid*.

- 2024 • The transaction unit for this operation is the entire set of <SvcMD> elements; they either all succeed or all fail. The  
2025 Discovery Service MUST enforce this atomicity.
  
- 2026 • Once replaced, the previous service metadata element MUST NOT be subsequently used by the DS to mint ID-  
2027 WSF EPRs. However, WSPs should be prepared to receive requests from WSCs from clients who previously  
2028 obtained ID-WSF EPRs minted from the prior metadata which haven't expired.
  
- 2029 • If request processing succeeded, the top-level status code MUST be *OK*. Otherwise, the top-level status code  
2030 MUST be *Failed*.
  
- 2031 • If the top-level status code is *Failed*, the response MAY also contain *Forbidden*, *Invalid*, or *NotFound* as a second-  
2032 level status code. The Discovery Service instance may not wish to reveal the reason for failure, in which case no  
2033 second-level status code will appear.

### 2034 **3.10. Operation: *MetadataDelete***

2035 The *MetadataDelete* operation is used by the WSP to delete previously registered metadata elements in the Discovery  
2036 Service.

#### 2037 **3.10.1. wsa:Action values for MetadataDelete Messages**

2038 <SvcMDDelete> messages MUST include a <wsa:Action> SOAP header with the value of  
2039 "urn:liberty:disco:2006-08:SvcMDDelete."

2040 <SvcMDDeleteResponse> messages MUST include a <wsa:Action> SOAP header with the value of  
2041 "urn:liberty:disco:2006-08:SvcMDDeleteResponse."

#### 2042 **3.10.2. SvcMDDelete Message**

2043 The <SvcMDDelete> is called with one or more <SvcMDID> elements to delete the specified list of service metadata  
2044 descriptions.

```
2045 <!-- Delete operation on Service Metadata -->
2046
2047
2048 <xs:element name="SvcMDDelete" type="SvcMDDeleteType"/>
2049
2050 <xs:complexType name="SvcMDDeleteType">
2051   <xs:sequence>
2052     <xs:element ref="SvcMDID" maxOccurs="unbounded" />
2053   </xs:sequence>
2054   <xs:anyAttribute namespace="##other" processContents="lax"/>
2055 </xs:complexType>
2056
2057
```

2058 **Figure 31. <SvcMDDelete> — Schema Fragment**

2059 An example message body containing a <SvcMDDelete> message follows. This request deletes a single service  
2060 metadata description.

```
2061 <ds: SvcMDDelete>
2062   <ds: SvcMDID>2323872</ds: SvcMDID>
2063 </ds: SvcMDDelete>
2064
2065
```

2066 **Example 19. <SvcMDDelete> Message**

### 3.10.3. SvcMDeleteResponse Message

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

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

```
<!-- Response for delete operation on Service Metadata -->
<xs:element name="SvcMDeleteResponse" type="SvcMDeleteResponseType" />
<xs:complexType name="SvcMDeleteResponseType">
  <xs:sequence>
    <xs:element ref="lu:Status" />
  </xs:sequence>
  <xs:anyAttribute namespace="##other" processContents="lax" />
</xs:complexType>
```

Figure 32. <SvcMDeleteResponse> — Schema Fragment

```
<ds:SvcMDeleteResponse>
  <lu:Status code="OK" />
</ds:SvcMDeleteResponse>
```

Example 20. <SvcMDeleteResponse> Message

### 3.10.4. Metadata Delete Processing Rules

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

- 2110 • If request processing succeeded, the top-level status code MUST be *OK*. Otherwise, the top-level status code  
2111 MUST be *Failed*.
- 2112 • If the top-level status code is *Failed*, the response MAY also contain *Forbidden* as a second-level status code. The  
2113 Discovery Service instance may not wish to reveal the reason for failure, in which case no second-level status code  
2114 will appear.

### 2115 3.11. option Value for Response Authentication

2116 The ID-WSF EPR `<SecurityContext>` element provides a way for services to indicate to clients what mecha-  
2117 nisms are necessary for the client to authenticate itself to the service via the `<SecurityMechID>` element. The  
2118 `<SecurityMechID>` values defined by [LibertySecMech] also indicate whether the service uses peer entity authenti-  
2119 cation (for example, server-side SSL/TLS). However, a web service client may need to know whether the service will  
2120 use message authentication (that is, whether the service will sign the response message) and may not be willing to use  
2121 a service which does not sign its responses.

2122 To avoid situations where a client requests data and then discovers it does not trust it because it is not signed, an  
2123 `<Option>` value is defined:

2124 `urn:liberty:disco:2006-08:options:security-response-x509`

2125 If a service instance always authenticates its response messages according to the "X.509 v3 Certificate Message  
2126 Authentication" mechanism in [LibertySecMech], registrations of ID-WSF EPRs describing the service instance  
2127 SHOULD include this option value. Otherwise, its registered ID-WSF EPRs MUST NOT include this option value.  
2128 Clients MAY include this option value in `<Query>` messages in order to locate only services which always authenticate  
2129 their response messages. A service MAY authenticate its response messages even if this option value was not included  
2130 in its description at the Discovery Service instance.

2131 In case the service also supports a previous version of the security mechanism specification [LibertySecMech11],  
2132 it should be able to register two different endpoints at the Discovery Service, each of them with different Options  
2133 values—one according to [LibertySecMech], the other one according to [LibertySecMech11]. This information  
2134 will aid the client in determining which version of the WSS-SMS specification ([wss-sms-draft] and/or [wss-sms]) is  
2135 supported by the service, and the service will act accordingly, depending on the ID-WSF EPR used by the client. Note  
2136 that this behavior only applies to the case when the client's request does not use message authentication mechanisms.

2137 Otherwise, it should be possible for the service to determine the version of the WSS-SMS specification supported by  
2138 the client by simply analyzing the `<wsse:Security>` header present in the request.

2139 In general, it is recommended that services do not sign their responses unless they positively know that clients are able  
2140 to perform message authentication and are aware of the version of the WSS-SMS spec used by that client.

### 2141 3.12. Including Keys in the `svcMDRegisterResponse` Message

2142 The Discovery Service instance may need to generate signed security tokens in `<QueryResponse>` messages for the  
2143 ID-WSF EPRs in question (which are later included in a message to a WSP). The WSP which receives the signed  
2144 security tokens from a client needs to be able to verify the Discovery service instance's signature on the security tokens.  
2145 Typically the metadata (see [SAMLMeta2]) for the Discovery service instance is sufficient for such information. In  
2146 certain situations, such as when the Discovery service instance is hosted on a LUAD (see [LibertyClientProfiles]), it  
2147 may not be feasible to assign the LUAD a `ProviderID` with which to obtain metadata. However, the key material  
2148 still needs to be made available to service instances which register ID-WSF EPRs with the Discovery Service which  
2149 include security mechanisms requiring such tokens.

2150 The Discovery Service instance may include a `<Keys>` element in the `<SvcMDRegisterResponse>` in order to  
2151 provide such keys.

2152 The Discovery Service instance SHOULD ONLY include the <Keys> element in <SvcMDRegisterResponse>  
2153 messages if it has no <ProviderID> and the <SvcMDRegister> message included Service Metadata that relies  
2154 upon signed security tokens for one or more of its security mechanisms.

```
2155
2156 <!-- Keys Element - For use in ModifyResponse -->
2157
2158 <xs:element name="Keys" type="KeysType" />
2159
2160 <xs:complexType name="KeysType">
2161   <xs:sequence>
2162     <xs:element ref="md:KeyDescriptor"
2163       minOccurs="1"
2164       maxOccurs="unbounded" />
2165   </xs:sequence>
2166 </xs:complexType>
2167
2168
```

2169 **Figure 33. <Keys> — Schema Fragment**

2170 The <Keys> element appears as a child of the <SvcMDRegisterResponse> element. It contains one or more  
2171 <KeyDescriptor> elements.

### 2172 **3.13. Discovery Service Example Messages (NON-Normative)**

2173 This section walks through a series of messages to show examples of inputs and outputs. The information in this  
2174 section is **NOT** normative with respect to the specification (in cases where the other normative section(s) of the  
2175 specification conflict with this section, the normative section(s) will prevail).

2176 Notes about this sequence:

- 2177 • It is an actual network capture of a real session between two independent liberty implementations.
- 2178 • The messages are from a **test sequence** which exercises the features of the Discovery Service. We do **not** expect  
2179 that any real world situation would result in this sequence of messages (or anything similar to this sequence of  
2180 messages).
- 2181 • All of these messages were invoked using the same SAML Assertion to establish the invocation context with the  
2182 principal as the subject and the WSP in the subject confirmation. Note that some of the processing rules for the  
2183 Discovery Service interfaces (those that manage the service metadata) require that such an invocation context be  
2184 interpreted as a WSP invoker invocation context.
- 2185 • The initial state is that the principal has a single service associated with their identity (an instance of the ID-WSF  
2186 People Service).
- 2187 • The messages **are** part of a sequence that was invoked in this order and one request (such as a new registration)  
2188 can and usually does impact the results of subsequent requests.
- 2189 • The first request and response messages are shown as a full SOAP-bound ID-\* message (with all of the SOAP  
2190 headers). The remaining request and response messages are shown with just the ID-\* message component (i.e.,  
2191 only the contents within the <S:Body> element are shown).
- 2192 • The messages have been formatted for easier reading (pretty-printing) and data has been ellided (such as the  
2193 contents of SAML assertions) for brevity.
- 2194 • Many of the service types used in these example messages do **not** exist as real ID-\* services and are only used  
2195 here as testing input to exercise the Discovery Service.

### 2196 3.13.1. Query People Service

#### 2197 Query for the People Service

```

2198 <S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/"
2199     xmlns:wsse="http://.../oasis-200401-wss-wssecurity-secext-1.0.xsd"
2200     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
2201     xmlns:sb2="urn:liberty:sb:2006-08"
2202     xmlns:wsm="http://.../oasis-200401-wss-wssecurity-utility-1.0.xsd"
2203     xmlns:wsa="http://www.w3.org/2005/08/addressing">
2204   <S:Header>
2205     <wsa:MessageID S:mustUnderstand="true"
2206       S:actor="http://schemas.xmlsoap.org/soap/actor/next" id="msgHdrID">
2207       uuid:asdqwer-238asf-44353608-000b8c14
2208     </wsa:MessageID>
2209     <wsa:To S:mustUnderstand="true"
2210       S:actor="http://schemas.xmlsoap.org/soap/actor/next" id="wsaToID">
2211       https://s-ds.liberty-iop.org:8681/DISCO-S
2212     </wsa:To>
2213     <wsa:Action S:mustUnderstand="true"
2214       S:actor="http://schemas.xmlsoap.org/soap/actor/next" id="wsaActionID">
2215       urn:liberty:disco:2006-08:Query
2216     </wsa:Action>
2217     <wsse:Security S:mustUnderstand="true"
2218       S:actor="http://schemas.xmlsoap.org/soap/actor/next">
2219       <sa:Assertion
2220         xmlns:sa="urn:oasis:names:tc:SAML:2.0:assertion"
2221         ID="CRED6q6HqDRRCAPsq3L8d_sh"
2222         IssueInstant="2006-04-06T15:39:12Z"
2223         Version="2.0">
2224         ... assertion data was here ...
2225       </sa:Assertion>
2226       <wsu:Timestamp wsu:Id="WsuTimestampID">
2227         <wsu:Created>2006-04-06T15:38:48Z</wsu:Created>
2228       </wsu:Timestamp>
2229     </wsse:Security>
2230   </S:Header>
2231   <S:Body wsu:id="msgBodyID">
2232     <disco:Query xmlns:disco="urn:liberty:disco:2006-08" id="discReq">
2233       <disco:RequestedService>
2234         <disco:ServiceType>urn:liberty:ps:2006-08</disco:ServiceType>
2235         <disco:SecurityMechID>urn:liberty:security:2006-08:TLS:SAMLV2</disco:SecurityMechID>
2236         <disco:SecurityMechID>urn:liberty:security:2005-02:TLS:Bearer</disco:SecurityMechID>
2237       </disco:RequestedService>
2238     </disco:Query>
2239   </S:Body>
2240 </S:Envelope>

```

2241 Things to note about this query:

- 2242 • it is a query of a service type (in this case, a particular version of the ID-WSF People Service)
- 2243 • the client has stated that they can support 2 specific security mechanisms (TLS:SAMLV2 and TLS:Bearer) for
- 2244 communicating with the specified service

2245 The response from the Discovery Service:

```

2246 <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
2247     xmlns:lib="urn:liberty:iff:2003-08">
2248   <soap:Header>
2249     <wsa:MessageID xmlns:wsa="http://www.w3.org/2005/08/addressing" id="MID">
2250       uuid:IKvqaaaE0dZ5Hhg1IQZ1
2251     </wsa:MessageID>
2252     <wsa:RelatesTo xmlns:wsa="http://www.w3.org/2005/08/addressing">
2253       uuid:asdqwer-238asf-44353608-000b8c14
2254     </wsa:RelatesTo>

```

```

2255 <wsa:Action xmlns:wsa="http://www.w3.org/2005/08/addressing">
2256   urn:liberty:disco:2006-08:QueryResponse
2257 </wsa:Action>
2258 <wsa:ReplyTo xmlns:wsa="http://www.w3.org/2005/08/addressing">
2259   <wsa:Address>
2260     http://www.w3.org/2005/03/addressing/role/anonymous
2261   </wsa:Address>
2262 </wsa:ReplyTo>
2263 <wsse:Security xmlns:wsse="http://.../oasis-200401-wss-wssecurity-secext-1.0.xsd">
2264   <wsu:Timestamp xmlns:wsu="http://.../oasis-200401-wss-wssecurity-utility-1.0.xsd">
2265     <wsu:Created>2006-04-06T15:39:13Z</wsu:Created>
2266   </wsu:Timestamp>
2267 </wsse:Security>
2268 <sb2:Sender xmlns:sb2="urn:liberty:sb:2006-08"
2269   xmlns:wsu="http://.../oasis-200401-wss-wssecurity-utility-1.0.xsd"
2270   providerID="https://s-ds.liberty-iop.org:8681/idp.xml"
2271   soap:actor="http://schemas.xmlsoap.org/soap/actor/next"
2272   wsu:Id="PRV" />
2273 <sb2:Framework
2274   xmlns:sb2="urn:liberty:sb:2006-08"
2275   version="2.0"
2276   soap:actor="http://schemas.xmlsoap.org/soap/actor/next" />
2277 </soap:Header>
2278 <soap:Body>
2279   <disco:QueryResponse xmlns:disco="urn:liberty:disco:2006-08">
2280     <lu>Status xmlns:lu="urn:liberty:util:2006-08" code="OK" />
2281     <wsa:EndpointReference
2282       xmlns:wsa="http://www.w3.org/2005/08/addressing"
2283       xmlns:wsu="http://.../oasis-200401-wss-wssecurity-utility-1.0.xsd"
2284       notOnOrAfter="2006-04-06T17:39:14Z"
2285       wsu:Id="EPRIDIo61485WDEA701qHi4Ey">
2286     <wsa:Address>https://s-wsp.liberty-iop.org:8743/PS-PSBEARER</wsa:Address>
2287     <wsa:Metadata>
2288       <disco:Abstract>SYmfiam urn:liberty:ps:2006-01 service</disco:Abstract>
2289       <sbf:Framework xmlns:sbf="urn:liberty:sb" version="2.0" />
2290       <disco:ProviderID>https://s-wsp.liberty-iop.org:8743/sp.xml</disco:ProviderID>
2291       <disco:ServiceType>urn:liberty:ps:2006-01</disco:ServiceType>
2292       <disco:SecurityContext>
2293         <disco:SecurityMechID>urn:liberty:security:2005-02:TLS:Bearer</disco:SecurityMechID>
2294         <sec:Token xmlns:sec="urn:liberty:security:2006-08"
2295           usage="urn:liberty:security:tokenusage:2006-08:SecurityToken">
2296           <sa:Assertion xmlns:sa="urn:oasis:names:tc:SAML:2.0:assertion"
2297             ID="CREDI4cINqS4SV1HPm_xOoGh"
2298             IssueInstant="2006-04-06T15:39:14Z"
2299             Version="2.0">
2300             ... assertion data was here ...
2301           </sa:Assertion>
2302         </sec:Token>
2303       </disco:SecurityContext>
2304     </wsa:Metadata>
2305   </wsa:EndpointReference>
2306 </disco:QueryResponse>
2307 </soap:Body>
2308 </soap:Envelope>

```

2309 Things to note about this response:

- 2310 • the query was successful (status code is OK).
- 2311 • there is a single ID-WSF EPR in the response for the service type specified in the request
- 2312 • the details within the assertion were edited out to save space

### 2313 3.13.2. Query provider

2314 Query for the same service using the ProviderID of the WSP.

```
2315 <disco:Query xmlns:disco="urn:liberty:disco:2006-08" id="discReq">
2316   <disco:RequestedService>
2317     <disco:ProviderID>https://s-wsp.liberty-iop.org:8743/sp.xml</disco:ProviderID>
2318     <disco:SecurityMechID>urn:liberty:security:2006-08:TLS:SAMLV2</disco:SecurityMechID>
2319     <disco:SecurityMechID>urn:liberty:security:2005-02:TLS:Bearer</disco:SecurityMechID>
2320   </disco:RequestedService>
2321 </disco:Query>
```

2322 Things to note about this query:

- 2323 • it is a query of a ProviderID, so all services provided by that ProviderID would be returned (and depending upon  
2324 the invocation context of the ID-WSF framework, this may be all services provided by that provider for a particular  
2325 principal or just all services).
- 2326 • the client has stated that they can support 2 specific security mechanisms (TLS:SAMLV2 and TLS:Bearer) for  
2327 communicating with the specified service

2328 Server Response:

```
2329 <disco:QueryResponse xmlns:disco="urn:liberty:disco:2006-08">
2330   <lu:Status xmlns:lu="urn:liberty:util:2006-08" code="OK"/>
2331   <wsa:EndpointReference
2332     xmlns:wsa="http://www.w3.org/2005/08/addressing"
2333     xmlns:wsu="http://.../oasis-200401-wss-wssecurity-utility-1.0.xsd"
2334     notOnOrAfter="2006-04-06T17:39:18Z"
2335     wsu:Id="EPRIDyGxcYpQ8Gace5y8lDm7Z">
2336     <wsa:Address>https://s-wsp.liberty-iop.org:8743/PS-PSBEARER</wsa:Address>
2337     <wsa:Metadata>
2338       <disco:Abstract>SYMfiam urn:liberty:ps:2006-01 service</disco:Abstract>
2339       <sbf:Framework xmlns:sbf="urn:liberty:sb" version="2.0"/>
2340       <disco:ProviderID>https://s-wsp.liberty-iop.org:8743/sp.xml</disco:ProviderID>
2341       <disco:ServiceType>urn:liberty:ps:2006-01</disco:ServiceType>
2342       <disco:SecurityContext>
2343         <disco:SecurityMechID>urn:liberty:security:2005-02:TLS:Bearer</disco:SecurityMechID>
2344         <sec:Token xmlns:sec="urn:liberty:security:2006-08"
2345           usage="urn:liberty:security:tokenusage:2006-08:SecurityToken">
2346           <sa:Assertion
2347             xmlns:sa="urn:oasis:names:tc:SAML:2.0:assertion"
2348             ID="CREDOEV7U7-mJk02pEVJAn--"
2349             IssueInstant="2006-04-06T15:39:18Z"
2350             Version="2.0">
2351             ... assertion data was here ...
2352           </sa:Assertion>
2353         </sec:Token>
2354       </disco:SecurityContext>
2355     </wsa:Metadata>
2356   </wsa:EndpointReference>
2357 </disco:QueryResponse>
```

2358 Things to note about this response:

- 2359 • the query was successful (status code is OK).
- 2360 • there is a single ID-WSF EPR in the response denoting the single service offered by the provider specified in the  
2361 request for this user.
- 2362 • The ID-WSF EPR is **almost** identical to the ID-WSF EPR returned in the previous request. The differences are  
2363 only in the timestamps and the element IDs.

### 3.13.3. Query (empty)

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

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

Things to note about this query:

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

Server Response:

```
<disco:QueryResponse xmlns:disco="urn:liberty:disco:2006-08">
  <lu:Status xmlns:lu="urn:liberty:util:2006-08" code="OK"/>
  <wsa:EndpointReference
    xmlns:wsa="http://www.w3.org/2005/08/addressing"
    xmlns:wsu="http://.../oasis-200401-wss-wssecurity-utility-1.0.xsd"
    notOnOrAfter="2006-04-06T17:39:21Z" wsu:Id="EPRIDt3MOElDHPL0EBD8whEvw">
    <wsa:Address>https://s-ds.liberty-iop.org:8681/DISCO-S</wsa:Address>
    <wsa:Metadata>
      <disco:Abstract>SYMfiam Discovery Service</disco:Abstract>
      <sbf:Framework xmlns:sbf="urn:liberty:sb" version="2.0"/>
      <disco:ProviderID>https://s-ds.liberty-iop.org:8681/idp.xml</disco:ProviderID>
      <disco:ServiceType>urn:liberty:disco:2006-08</disco:ServiceType>
      <disco:SecurityContext>
        <disco:SecurityMechID>urn:liberty:security:2005-02:TLS:Bearer</disco:SecurityMechID>
        <sec:Token xmlns:sec="urn:liberty:security:2006-08"
          usage="urn:liberty:security:tokenusage:2006-08:SecurityToken">
          <sa:Assertion
            xmlns:sa="urn:oasis:names:tc:SAML:2.0:assertion"
            ID="CREDr2tN6rxYICAXxD6CtenC"
            IssueInstant="2006-04-06T15:39:21Z" Version="2.0">
            ... assertion data was here ...
          </sa:Assertion>
        </sec:Token>
      </disco:SecurityContext>
    </wsa:Metadata>
  </wsa:EndpointReference>
  <wsa:EndpointReference
    xmlns:wsa="http://www.w3.org/2005/08/addressing"
    xmlns:wsu="http://.../oasis-200401-wss-wssecurity-utility-1.0.xsd"
    notOnOrAfter="2006-04-06T17:39:21Z" wsu:Id="EPRIDQBqOfWHDp3GKFSqgnZyj">
    <wsa:Address>https://s-wsp.liberty-iop.org:8743/PS-PSBEARER</wsa:Address>
    <wsa:Metadata>
      <disco:Abstract>SYMfiam urn:liberty:ps:2006-01 service</disco:Abstract>
      <sbf:Framework xmlns:sbf="urn:liberty:sb" version="2.0"/>
      <disco:ProviderID>https://s-wsp.liberty-iop.org:8743/sp.xml</disco:ProviderID>
      <disco:ServiceType>urn:liberty:ps:2006-01</disco:ServiceType>
      <disco:SecurityContext>
        <disco:SecurityMechID>urn:liberty:security:2005-02:TLS:Bearer</disco:SecurityMechID>
        <sec:Token xmlns:sec="urn:liberty:security:2006-08"
          usage="urn:liberty:security:tokenusage:2006-08:SecurityToken">
          <sa:Assertion
            xmlns:sa="urn:oasis:names:tc:SAML:2.0:assertion"
            ID="CREDAsw9RdingIu3m4HtUxX5"
            IssueInstant="2006-04-06T15:39:22Z" Version="2.0">
            ... assertion data was here ...
          </sa:Assertion>
        </sec:Token>
      </disco:SecurityContext>
    </wsa:Metadata>
  </wsa:EndpointReference>
</disco:QueryResponse>
```

Things to note about this response:

- 2422 • the query was successful (status code is OK).
- 2423 • two ID-WSF EPRs were returned, one for the Discovery Service and one for the People Service.
- 2424 • Discovery Service instances will usually expose ID-WSF EPRs which point to themselves in this way in order to  
2425 expose alternative invocation methods and/or allow the client to obtain newer credentials.

### 2426 3.13.4. Query People Service and Provider

2427 A discovery query specifying both the Provider ID and the Service Type (so that only services of that type by that  
2428 provider are returned)

```
2429 <disco:Query xmlns:disco="urn:liberty:disco:2006-08" id="discReq">
2430   <disco:RequestedService>
2431     <disco:ServiceType>urn:liberty:ps:2006-01</disco:ServiceType>
2432     <disco:ProviderID>https://s-wsp.l liberty-iop.org:8743/sp.xml</disco:ProviderID>
2433     <disco:SecurityMechID>urn:liberty:security:2006-08:TLS:SAMLV2</disco:SecurityMechID>
2434     <disco:SecurityMechID>urn:liberty:security:2005-02:TLS:Bearer</disco:SecurityMechID>
2435   </disco:RequestedService>
2436 </disco:Query>
```

2437 Things to note about this query:

- 2438 • both the <disco:ServiceType> element and the <disco:ProviderID> element are included.
- 2439 • The provider and service type are the same ones we've been using so we should get the same response we  
2440 had on earlier requests.

2441 Server Response:

```
2442 <disco:QueryResponse xmlns:disco="urn:liberty:disco:2006-08">
2443   <lu:Status xmlns:lu="urn:liberty:util:2006-08" code="OK"/>
2444   <wsa:EndpointReference
2445     xmlns:wsa="http://www.w3.org/2005/08/addressing"
2446     xmlns:wsu="http://.../oasis-200401-wss-wssecurity-utility-1.0.xsd"
2447     notOnOrAfter="2006-04-06T17:39:26Z" wsu:Id="EPRIDwrdfxWpBRhCXsEMW1cjo">
2448     <wsa:Address>https://s-wsp.liberty-iop.org:8743/PS-PSBEARER</wsa:Address>
2449     <wsa:Metadata>
2450       <disco:Abstract>SYMfiam urn:liberty:ps:2006-01 service</disco:Abstract>
2451       <sbfi:Framework xmlns:sbfi="urn:liberty:sbfi" version="2.0"/>
2452       <disco:ProviderID>https://s-wsp.liberty-iop.org:8743/sp.xml</disco:ProviderID>
2453       <disco:ServiceType>urn:liberty:ps:2006-01</disco:ServiceType>
2454       <disco:SecurityContext>
2455         <disco:SecurityMechID>urn:liberty:security:2005-02:TLS:Bearer</disco:SecurityMechID>
2456         <sec:Token xmlns:sec="urn:liberty:security:2006-08"
2457           usage="urn:liberty:security:tokenusage:2006-08:SecurityToken">
2458           <sa:Assertion
2459             xmlns:sa="urn:oasis:names:tc:SAML:2.0:assertion"
2460             ID="CREDZbnmfBjqRINmQeWvyxCG"
2461             IssueInstant="2006-04-06T15:39:26Z" Version="2.0">
2462             ... assertion data was here ...
2463           </sa:Assertion>
2464         </sec:Token>
2465       </disco:SecurityContext>
2466     </wsa:Metadata>
2467   </wsa:EndpointReference>
2468 </disco:QueryResponse>
```

2469 Things to note about this response:

- 2470 • the query was successful (status code is OK).

- 2471 • The ID-WSF EPR is **almost** exactly the same as the ID-WSF EPR returned in the first 2 examples above.  
2472 The differences are only in the timestamps and the element IDs. This is because both EPRs are for the same  
2473 principal accessing the same service (we just discovered the EPR through a different query).

### 2474 3.13.5. SvcMDQuery (empty)

2475 A query for all SvcMD stored at the DS on behalf of the invoking provider.

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

2477 Things to note about this query:

- 2478 • it's empty – meaning that all registered SvcMDs are requested.
- 2479 • the query is executed in the context of the provider invoking the query (identified in the invocation context specified  
2480 in the ID-WSF framework headers)

2481 Server Response:

```
2482 <disco:SvcMDQueryResponse xmlns:disco="urn:liberty:disco:2006-08">
2483 <lu:Status xmlns:lu="urn:liberty:util:2006-08" code="OK" />
2484 <disco:SvcMD svcMDID="SVCMDIDy_7yFABwuFPkgYIjKCjv">
2485 <disco:Abstract>SYMfiam urn:liberty:ps:2006-01 service</disco:Abstract>
2486 <disco:ProviderID>https://s-wsp.liberty-iop.org:8743/sp.xml</disco:ProviderID>
2487 <disco:ServiceContext>
2488 <disco:ServiceType>urn:liberty:ps:2006-01</disco:ServiceType>
2489 <disco:EndpointContext>
2490 <disco:Address>https://s-wsp.liberty-iop.org:8743/PS-PSBEARER</disco:Address>
2491 <sbf:Framework xmlns:sbf="urn:liberty:sb" version="2.0" />
2492 <disco:SecurityMechID>urn:liberty:security:2005-02:TLS:Bearer</disco:SecurityMechID>
2493 </disco:EndpointContext>
2494 </disco:ServiceContext>
2495 </disco:SvcMD>
2496 </disco:SvcMDQueryResponse>
```

2497 Things to note about this response:

- 2498 • the query was successful (status code is OK).
- 2499 • one SvcMD was returned for the ID-WSF People Service (which is the only SvcMD registered by the invoking  
2500 provider).

### 2501 3.13.6. SvcMDQuery w/Bad SvcMDID

2502 A service metadata query using an invalid SvcMDID.

```
2503 <disco:SvcMDQuery xmlns:disco="urn:liberty:disco:2006-08">
2504 <disco:SvcMDID>123</disco:SvcMDID>
2505 </disco:SvcMDQuery>
```

2506 Things to note about this query:

- 2507 • 123 is an invalid SvcMDID

2508 **Server Response:**

```
2509 <disco:SvcMDQueryResponse xmlns:disco="urn:liberty:disco:2006-08">
2510   <lu:Status xmlns:lu="urn:liberty:util:2006-08" code="Failed">
2511     <lu:Status code="NoResults"/>
2512   </lu:Status>
2513 </disco:SvcMDQueryResponse>
```

2514 **Things to note about this response:**

- 2515 • the query failed (status code is Failed).
- 2516 • The optional sub-status is included which indicates that there were *NoResults* for the query.

2517 **3.13.7. SvcMDRegister w/single SvcMD**

2518 **Registration of a single service metadata instance in the DS.**

```
2519 <disco:SvcMDRegister xmlns:disco="urn:liberty:disco:2006-08">
2520   <disco:SvcMD>
2521     <disco:Abstract>TestDisco Test Payment Service</disco:Abstract>
2522     <disco:ProviderID>https://s-wsp.liberty-iop.org:8743/sp.xml</disco:ProviderID>
2523     <disco:ServiceContext>
2524       <disco:ServiceType>urn:x-test:pmt:2007-11</disco:ServiceType>
2525       <disco:EndpointContext>
2526         <disco:Address>https://payment.testing.com</disco:Address>
2527         <sbef:Framework xmlns:sbef="urn:liberty:sb" version="2.0"/>
2528         <disco:SecurityMechID>urn:liberty:security:2005-02:TLS:SAML2</disco:SecurityMechID>
2529         <disco:SecurityMechID>urn:liberty:security:2005-02:TLS:Bearer</disco:SecurityMechID>
2530         <disco:SecurityMechID>urn:liberty:security:2005-02:TLS:null</disco:SecurityMechID>
2531       </disco:EndpointContext>
2532     </disco:ServiceContext>
2533   </disco:SvcMD>
2534 </disco:SvcMDRegister>
```

2535 **Things to note about this request:**

- 2536 • a single SvcMD is registered
- 2537 • no `svcMDID` attribute is specified on the SvcMD during registration (it will be assigned by the DS if the request is  
2538 successful).
- 2539 • the service being registered is a test payment service provided by the same provider we've seen earlier, supports a  
2540 single service and framework version and exposes three different security mechanisms.

2541 **Server Response:**

```
2542 <disco:SvcMDRegisterResponse xmlns:disco="urn:liberty:disco:2006-08">
2543   <lu:Status xmlns:lu="urn:liberty:util:2006-08" code="OK"/>
2544   <disco:SvcMDID>SVCMDIDg9WP0thd_HvPd427KY9M</disco:SvcMDID>
2545 </disco:SvcMDRegisterResponse>
```

2546 **Things to note about this response:**

- 2547 • the registration was successful (status code is OK).
- 2548 • The SvcMD was assigned the svcMDID *SVCMDIDg9WP0thd\_HvPd427KY9M*.

### 3.13.8. SvcMDQuery w/Good SvcMDID

2549

2550 Query the SvcMD that we just registered using the SvcMDID that was returned in the response.

```
2551 <disco:SvcMDQuery xmlns:disco="urn:liberty:disco:2006-08">
2552   <disco:SvcMDID>SVCMDIDg9WP0thd_HvPd427KY9M</disco:SvcMDID>
2553 </disco:SvcMDQuery>
```

2554 Things to note about this query:

- 2555 • the SvcMDID is the ID that was returned in the response to the <SvcMDRegister> we executed in the previous  
2556 step.

2557 Server Response:

```
2558 <disco:SvcMDQueryResponse xmlns:disco="urn:liberty:disco:2006-08">
2559   <lu:Status xmlns:lu="urn:liberty:util:2006-08" code="OK"/>
2560   <disco:SvcMD svcMDID="SVCMDIDg9WP0thd_HvPd427KY9M">
2561     <disco:Abstract>TestDisco Test Payment Service</disco:Abstract>
2562     <disco:ProviderID>https://s-wsp.liberty-iop.org:8743/sp.xml</disco:ProviderID>
2563     <disco:ServiceContext>
2564       <disco:ServiceType>urn:x-test:pmt:2007-11</disco:ServiceType>
2565       <disco:EndpointContext>
2566         <disco:Address>https://payment.testing.com</disco:Address>
2567         <sbf:Framework xmlns:sbf="urn:liberty:sb" version="2.0"/>
2568         <disco:SecurityMechID>urn:liberty:security:2005-02:TLS:SAML2</disco:SecurityMechID>
2569         <disco:SecurityMechID>urn:liberty:security:2005-02:TLS:Bearer</disco:SecurityMechID>
2570         <disco:SecurityMechID>urn:liberty:security:2005-02:TLS:null</disco:SecurityMechID>
2571       </disco:EndpointContext>
2572     </disco:ServiceContext>
2573   </disco:SvcMD>
2574 </disco:SvcMDQueryResponse>
```

2575 Things to note about this response:

- 2576 • the query was successful (status code is OK).
- 2577 • The SvcMD has the svcMDID attribute assigned by the Discovery Service.
- 2578 • The remaining data is identical to that registered by the provider in the previous request.

### 3.13.9. SvcMDDelete w/Good SvcMDID

2579

2580 Delete the Service Metadata that we just registered and queried.

```
2581 <disco:SvcMDDelete xmlns:disco="urn:liberty:disco:2006-08">
2582   <disco:SvcMDID>SVCMDIDg9WP0thd_HvPd427KY9M</disco:SvcMDID>
2583 </disco:SvcMDDelete>
```

2584 Things to note about this query:

- 2585 • the SvcMDID that we obtained in the registration above is used

2586 Server Response:

```
2587 <disco:SvcMDDeleteResponse xmlns:disco="urn:liberty:disco:2006-08">
2588   <lu:Status xmlns:lu="urn:liberty:util:2006-08" code="OK"/>
2589 </disco:SvcMDDeleteResponse>
```

2590 Things to note about this response:

- 2591 • the delete was successful (status code is OK).

### 2592 **3.13.10. SvcMDelete w/Already Deleted SvcMDID**

2593 Delete the same service metadata that we just deleted.

```
2594 <disco:SvcMDelete xmlns:disco="urn:liberty:disco:2006-08">
2595   <disco:SvcMDID>SVCMDIDg9WP0thd_HvPd427KY9M</disco:SvcMDID>
2596 </disco:SvcMDelete>
```

2597 Things to note about this query:

- 2598 • the SvcMDID that we already deleted is specified

2599 Server Response:

```
2600 <disco:SvcMDeleteResponse xmlns:disco="urn:liberty:disco:2006-08">
2601   <lu:Status xmlns:lu="urn:liberty:util:2006-08" code="OK"/>
2602 </disco:SvcMDeleteResponse>
```

2603 Things to note about this response:

- 2604 • the delete was successful (status code is OK), even though the SvcMD was already deleted as the delete of a  
2605 non-existent SvcMD is defined by this specification to be successful.

### 2606 **3.13.11. SvcMDRegister w/Complex SvcMD**

2607 A basic registration of a SvcMD with the Discovery Service. This is a little bit different in that the SvcMD is rather  
2608 complex (which is good for testing the Query interface).

```
2609 <disco:SvcMDRegister xmlns:disco="urn:liberty:disco:2006-08">
2610   <disco:SvcMD>
2611     <disco:Abstract>TestDisco Test Calendar Service</disco:Abstract>
2612     <disco:ProviderID>https://s-wsp.liberty-iop.org:8743/sp.xml</disco:ProviderID>
2613     <disco:ServiceContext>
2614       <disco:ServiceType>urn:x-test:cal:2006-01</disco:ServiceType>
2615       <disco:ServiceType>urn:x-test:cal:2006-09</disco:ServiceType>
2616       <disco:EndpointContext>
2617         <disco:Address>https://old-calendars.testing.com</disco:Address>
2618         <sbf:Framework xmlns:sbf="urn:liberty:sb" version="2.1"/>
2619         <disco:SecurityMechID>urn:liberty:security:2003-08:TLS:Bearer</disco:SecurityMechID>
2620         <disco:SecurityMechID>urn:liberty:security:2003-08:TLS:null</disco:SecurityMechID>
2621       </disco:EndpointContext>
2622     </disco:EndpointContext>
2623     <disco:Address>https://old2-calendars.testing.com</disco:Address>
2624     <sbf:Framework xmlns:sbf="urn:liberty:sb" version="2.0"/>
2625     <disco:SecurityMechID>urn:liberty:security:2005-02:TLS:Bearer</disco:SecurityMechID>
2626     <disco:SecurityMechID>urn:liberty:security:2005-02:TLS:null</disco:SecurityMechID>
2627   </disco:EndpointContext>
2628 </disco:EndpointContext>
2629     <disco:Address>http://old-calendars.testing.com</disco:Address>
2630     <sbf:Framework xmlns:sbf="urn:liberty:sb" version="2.0"/>
2631     <disco:SecurityMechID>urn:liberty:security:2005-02:null:Bearer</disco:SecurityMechID>
2632     <disco:SecurityMechID>urn:liberty:security:2005-02:null:SAML2</disco:SecurityMechID>
2633   </disco:EndpointContext>
2634 </disco:ServiceContext>
2635 </disco:ServiceContext>
2636 <disco:ServiceType>urn:x-test:cal:2008-03</disco:ServiceType>
2637 <disco:EndpointContext>
2638   <disco:Address>https://calendars.testing.com</disco:Address>
2639   <disco:Address>https://calendars.testing.backup.com</disco:Address>
2640   <sbf:Framework xmlns:sbf="urn:liberty:sb" version="2.0"/>
2641   <disco:SecurityMechID>urn:liberty:security:2005-02:TLS:SAML2</disco:SecurityMechID>
2642   <disco:SecurityMechID>urn:liberty:security:2005-02:TLS:Bearer</disco:SecurityMechID>
2643 </disco:EndpointContext>
2644 </disco:EndpointContext>
```

```
2645     <disco:Address>http://calendars.testing.com</disco:Address>
2646     <sbf:Framework xmlns:sbf="urn:liberty:sb" version="2.0"/>
2647     <disco:SecurityMechID>urn:liberty:security:2005-02:null:SAML2</disco:SecurityMechID>
2648   </disco:EndpointContext>
2649 </disco:ServiceContext>
2650 </disco:SvcMD>
2651 </disco:SvcMDRegister>
```

2652 Things to note about this query:

- 2653 • it's a rather complex SvcMD, but registration is still the same.

2654 Server Response:

```
2655 <disco:SvcMDRegisterResponse xmlns:disco="urn:liberty:disco:2006-08">
2656   <lu:Status xmlns:lu="urn:liberty:util:2006-08" code="OK"/>
2657   <disco:SvcMDID>SVCMDIDmifyjzIKO6tNd8evymnL</disco:SvcMDID>
2658 </disco:SvcMDRegisterResponse>
```

2659 Things to note about this response:

- 2660 • the registration was successful (status code is OK).
- 2661 • The SvcMD was assigned the svcMDID *SVCMDIDmifyjzIKO6tNd8evymnL*.
- 2662 • Now the fun begins as this SvcMD let's us exercise many of the interesting portions of the Discovery Query interface.

### 2664 3.13.12. SvcMDQuery w/Good SvcMDID for Complex SvcMD

2665 Query the SvcMD that we just registered using the SvcMDID that was returned in the response to make sure it was  
2666 registered correctly.

```
2667 <disco:SvcMDQuery xmlns:disco="urn:liberty:disco:2006-08">
2668   <disco:SvcMDID>SVCMDIDmifyjzIKO6tNd8evymnL</disco:SvcMDID>
2669 </disco:SvcMDQuery>
```

2670 Things to note about this query:

- 2671 • the SvcMDID is the ID that was returned in the response to the <SvcMDRegister> we executed in the previous  
2672 step.

2673 Server Response:

```

2674 <disco:SvcMDQueryResponse xmlns:disco="urn:liberty:disco:2006-08">
2675   <lu:Status xmlns:lu="urn:liberty:util:2006-08" code="OK"/>
2676   <disco:SvcMD svcMDID="SVCMDIDmifyjzIKO6tNd8evymnL">
2677     <disco:Abstract>TestDisco Test Calendar Service</disco:Abstract>
2678     <disco:ProviderID>https://s-wsp.liberty-iop.org:8743/sp.xml</disco:ProviderID>
2679     <disco:ServiceContext>
2680       <disco:ServiceType>urn:x-test:cal:2006-01</disco:ServiceType>
2681       <disco:ServiceType>urn:x-test:cal:2006-09</disco:ServiceType>
2682       <disco:EndpointContext>
2683         <disco:Address>https://old-calendars.testing.com</disco:Address>
2684         <sbf:Framework xmlns:sbf="urn:liberty:sb" version="2.1"/>
2685         <disco:SecurityMechID>urn:liberty:security:2003-08:TLS:Bearer</disco:SecurityMechID>
2686         <disco:SecurityMechID>urn:liberty:security:2003-08:TLS:null</disco:SecurityMechID>
2687       </disco:EndpointContext>
2688       <disco:EndpointContext>
2689         <disco:Address>https://old2-calendars.testing.com</disco:Address>
2690         <sbf:Framework xmlns:sbf="urn:liberty:sb" version="2.0"/>
2691         <disco:SecurityMechID>urn:liberty:security:2005-02:TLS:Bearer</disco:SecurityMechID>
2692         <disco:SecurityMechID>urn:liberty:security:2005-02:TLS:null</disco:SecurityMechID>
2693       </disco:EndpointContext>
2694       <disco:EndpointContext>
2695         <disco:Address>http://old-calendars.testing.com</disco:Address>
2696         <sbf:Framework xmlns:sbf="urn:liberty:sb" version="2.0"/>
2697         <disco:SecurityMechID>urn:liberty:security:2005-02:null:Bearer</disco:SecurityMechID>
2698         <disco:SecurityMechID>urn:liberty:security:2005-02:null:SAML2</disco:SecurityMechID>
2699       </disco:EndpointContext>
2700     </disco:ServiceContext>
2701     <disco:ServiceContext>
2702       <disco:ServiceType>urn:x-test:cal:2008-03</disco:ServiceType>
2703       <disco:EndpointContext>
2704         <disco:Address>https://calendars.testing.com</disco:Address>
2705         <disco:Address>https://calendars.testing.backup.com</disco:Address>
2706         <sbf:Framework xmlns:sbf="urn:liberty:sb" version="2.0"/>
2707         <disco:SecurityMechID>urn:liberty:security:2005-02:TLS:SAML2</disco:SecurityMechID>
2708         <disco:SecurityMechID>urn:liberty:security:2005-02:TLS:Bearer</disco:SecurityMechID>
2709       </disco:EndpointContext>
2710       <disco:EndpointContext>
2711         <disco:Address>http://calendars.testing.com</disco:Address>
2712         <sbf:Framework xmlns:sbf="urn:liberty:sb" version="2.0"/>
2713         <disco:SecurityMechID>urn:liberty:security:2005-02:null:SAML2</disco:SecurityMechID>
2714       </disco:EndpointContext>
2715     </disco:ServiceContext>
2716   </disco:SvcMD>
2717 </disco:SvcMDQueryResponse>

```

2718 Things to note about this response:

- 2719 • the query was successful (status code is OK).
- 2720 • The SvcMD has the svcMDID attribute assigned by the Discovery Service.
- 2721 • The remaining data is identical to that registered by the provider in the previous request. Key here is that the
- 2722 ordering and grouping of elements has been maintained.

2723 **3.13.13. SvcMDReplace of existing SvcMD**

2724 Replace the complex SvcMD that we just registered with a simpler version

```

2725 <disco:SvcMDReplace xmlns:disco="urn:liberty:disco:2006-08">
2726   <disco:SvcMD svcMDID="SVCMDIDmifyjzIKO6tNd8evymnL">
2727     <disco:Abstract>TestDisco Test Payment Service</disco:Abstract>
2728     <disco:ProviderID>https://s-wsp.liberty-iop.org:8743/sp.xml</disco:ProviderID>
2729     <disco:ServiceContext>

```

```
2730 <disco:ServiceType>urn:x-test:pmt:2007-11</disco:ServiceType>
2731 <disco:EndpointContext>
2732 <disco:Address>https://payment.testing.com</disco:Address>
2733 <sbf:Framework xmlns:sbf="urn:liberty:sb" version="2.0"/>
2734 <disco:SecurityMechID>urn:liberty:security:2005-02:TLS:SAML2</disco:SecurityMechID>
2735 <disco:SecurityMechID>urn:liberty:security:2005-02:TLS:Bearer</disco:SecurityMechID>
2736 <disco:SecurityMechID>urn:liberty:security:2005-02:TLS:null</disco:SecurityMechID>
2737 </disco:EndpointContext>
2738 </disco:ServiceContext>
2739 </disco:SvcMD>
2740 </disco:SvcMDReplace>
```

2741 Things to note about this request:

- 2742 • the svcMDID attribute has the value obtained during the previous registration so that registration should be  
2743 replaced.

2744 Server Response:

```
2745 <disco:SvcMDReplaceResponse xmlns:disco="urn:liberty:disco:2006-08">
2746 <lu:Status xmlns:lu="urn:liberty:util:2006-08" code="OK"/>
2747 </disco:SvcMDReplaceResponse>
```

2748 Things to note about this response:

- 2749 • the replacement was successful (status code is OK).
- 2750 • no other data is returned. The SvcMDID does not change when the SvcMD is replaced.

### 2751 3.13.14. SvcMDQuery of Replaced SvcMDID

2752 Query the SvcMD that we just replaced.

```
2753 <disco:SvcMDQuery xmlns:disco="urn:liberty:disco:2006-08">
2754 <disco:SvcMDID>SVCMDIDmifyjzIKO6tNd8evymnL</disco:SvcMDID>
2755 </disco:SvcMDQuery>
```

2756 Things to note about this query:

- 2757 • We use the same svcMDID that we had for the previous SvcMD as the value doesn't change when we do a  
2758 replacement.

2759 Server Response:

```
2760 <disco:SvcMDQueryResponse xmlns:disco="urn:liberty:disco:2006-08">
2761 <lu:Status xmlns:lu="urn:liberty:util:2006-08" code="OK"/>
2762 <disco:SvcMD svcMDID="SVCMDIDmifyjzIKO6tNd8evymnL">
2763 <disco:Abstract>TestDisco Test Payment Service</disco:Abstract>
2764 <disco:ProviderID>https://s-wsp.liberty-iop.org:8743/sp.xml</disco:ProviderID>
2765 <disco:ServiceContext>
2766 <disco:ServiceType>urn:x-test:pmt:2007-11</disco:ServiceType>
2767 <disco:EndpointContext>
2768 <disco:Address>https://payment.testing.com</disco:Address>
2769 <sbf:Framework xmlns:sbf="urn:liberty:sb" version="2.0"/>
2770 <disco:SecurityMechID>urn:liberty:security:2005-02:TLS:SAML2</disco:SecurityMechID>
2771 <disco:SecurityMechID>urn:liberty:security:2005-02:TLS:Bearer</disco:SecurityMechID>
2772 <disco:SecurityMechID>urn:liberty:security:2005-02:TLS:null</disco:SecurityMechID>
2773 </disco:EndpointContext>
2774 </disco:ServiceContext>
2775 </disco:SvcMD>
2776 </disco:SvcMDQueryResponse>
```

2777 Things to note about this response:

2778 • the query was successful (status code is OK).

2779 • the new SvcMD is returned.

### 2780 3.13.15. SvcMDDelete of replaced SvcMD

2781 Delete the Service Metadata that we just replaced (to clean up after ourselves).

```
2782 <disco: SvcMDDelete xmlns: disco="urn: liberty: disco: 2006-08">
2783   <disco: SvcMDID>SVCMDIDmifyjzIKO6tNd8evymnL</disco: SvcMDID>
2784 </disco: SvcMDDelete>
```

2785 Things to note about this query:

2786 • the SvcMDID that we used in the replacement above is used

2787 Server Response:

```
2788 <disco: SvcMDDeleteResponse xmlns: disco="urn: liberty: disco: 2006-08">
2789   <lu: Status xmlns: lu="urn: liberty: util: 2006-08" code="OK"/>
2790 </disco: SvcMDDeleteResponse>
```

2791 Things to note about this response:

2792 • the delete was successful (status code is OK).

### 2793 3.13.16. SvcMDRegister w/multiple SvcMDs

2794 A registration of 3 different SvcMD elements of varying levels of complexity

```
2795 <disco: SvcMDRegister xmlns: disco="urn: liberty: disco: 2006-08">
2796   <disco: SvcMD>
2797     <disco: Abstract>TestDisco Test Calendar Service</disco: Abstract>
2798     <disco: ProviderID>https://s-wsp.liberty-iop.org:8743/sp.xml</disco: ProviderID>
2799     <disco: ServiceContext>
2800       <disco: ServiceType>urn:x-test:cal:2006-01</disco: ServiceType>
2801       <disco: ServiceType>urn:x-test:cal:2006-09</disco: ServiceType>
2802       <disco: EndpointContext>
2803         <disco: Address>https://1-calendars.testing.com</disco: Address>
2804         <sbf: Framework xmlns: sbf="urn: liberty: sb" version="2.1"/>
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>https://2-calendars.testing.com</disco: Address>
2810         <sbf: Framework xmlns: sbf="urn: liberty: sb" version="2.0"/>
2811         <disco: SecurityMechID>urn: liberty: security: 2005-02: TLS: Bearer</disco: SecurityMechID>
2812         <disco: SecurityMechID>urn: liberty: security: 2005-02: TLS: null</disco: SecurityMechID>
2813       </disco: EndpointContext>
2814       <disco: EndpointContext>
2815         <disco: Address>http://3-calendars.testing.com</disco: Address>
2816         <sbf: Framework xmlns: sbf="urn: liberty: sb" version="2.0"/>
2817         <disco: SecurityMechID>urn: liberty: security: 2005-02: null: Bearer</disco: SecurityMechID>
2818         <disco: SecurityMechID>urn: liberty: security: 2006-08: null: SAMLV2</disco: SecurityMechID>
2819       </disco: EndpointContext>
2820     </disco: ServiceContext>
2821   </disco: SvcMD>
2822   <disco: SvcMD>
2823     <disco: ServiceType>urn:x-test:cal:2008-03</disco: ServiceType>
2824     <disco: EndpointContext>
2825       <disco: Address>https://4-calendars.testing.com</disco: Address>
2826       <disco: Address>https://5-calendars.testing.backup.com</disco: Address>
2827       <sbf: Framework xmlns: sbf="urn: liberty: sb" version="2.0"/>
2828       <disco: SecurityMechID>urn: liberty: security: 2006-08: TLS: SAMLV2</disco: SecurityMechID>
2829       <disco: SecurityMechID>urn: liberty: security: 2005-02: TLS: Bearer</disco: SecurityMechID>
```

```

2829     </disco:EndpointContext>
2830     <disco:EndpointContext>
2831         <disco:Address>http://6-calendars.testing.com</disco:Address>
2832         <sbf:Framework xmlns:sbf="urn:liberty:sb" version="2.0"/>
2833         <disco:SecurityMechID>urn:liberty:security:2006-08:null:SAMLV2</disco:SecurityMechID>
2834     </disco:EndpointContext>
2835 </disco:ServiceContext>
2836 </disco:SvcMD>
2837 <disco:SvcMD>
2838     <disco:Abstract>TestDisco Test Payment Service</disco:Abstract>
2839     <disco:ProviderID>https://s-wsp.liberty-iop.org:8743/sp.xml</disco:ProviderID>
2840     <disco:ServiceContext>
2841         <disco:ServiceType>urn:x-test:pmt:2007-11</disco:ServiceType>
2842         <disco:EndpointContext>
2843             <disco:Address>https://payment.testing.com</disco:Address>
2844             <sbf:Framework xmlns:sbf="urn:liberty:sb" version="2.0"/>
2845             <disco:SecurityMechID>urn:liberty:security:2006-08:TLS:SAMLV2</disco:SecurityMechID>
2846             <disco:SecurityMechID>urn:liberty:security:2005-02:TLS:Bearer</disco:SecurityMechID>
2847             <disco:SecurityMechID>urn:liberty:security:2005-02:TLS:null</disco:SecurityMechID>
2848         </disco:EndpointContext>
2849     </disco:ServiceContext>
2850 </disco:SvcMD>
2851 <disco:SvcMD>
2852     <disco:Abstract>TestDisco Test ATM Service</disco:Abstract>
2853     <disco:ProviderID>https://s-wsp.liberty-iop.org:8743/sp.xml</disco:ProviderID>
2854     <disco:ServiceContext>
2855         <disco:ServiceType>urn:x-test:atm:2003-03</disco:ServiceType>
2856         <disco:Options>
2857             <disco:Option>urn:x-test:atm:options:testopt1</disco:Option>
2858             <disco:Option>urn:x-test:atm:options:testopt2</disco:Option>
2859             <disco:Option>urn:x-test:atm:options:testopt3</disco:Option>
2860         </disco:Options>
2861         <disco:EndpointContext>
2862             <disco:Address>https://test2.atm.CA.US.testing.com</disco:Address>
2863             <sbf:Framework xmlns:sbf="urn:liberty:sb" version="2.0"/>
2864             <disco:SecurityMechID>urn:liberty:security:2006-08:TLS:SAMLV2</disco:SecurityMechID>
2865             <disco:Action>urn:x-test:atm:2007-11:GetBalance</disco:Action>
2866         </disco:EndpointContext>
2867         <disco:EndpointContext>
2868             <disco:Address>https://readers.atm.CA.US.testing.com</disco:Address>
2869             <disco:Address>https://readers.atm.NY.US.testing.com</disco:Address>
2870             <sbf:Framework xmlns:sbf="urn:liberty:sb" version="2.0"/>
2871             <disco:SecurityMechID>urn:liberty:security:2006-08:TLS:SAMLV2</disco:SecurityMechID>
2872             <disco:Action>urn:x-test:atm:2007-11:GetBalance</disco:Action>
2873             <disco:Action>urn:x-test:atm:2007-11:ListAccounts</disco:Action>
2874         </disco:EndpointContext>
2875         <disco:EndpointContext>
2876             <disco:Address>https://writers.atm.testing.com</disco:Address>
2877             <sbf:Framework xmlns:sbf="urn:liberty:sb" version="2.0"/>
2878             <disco:SecurityMechID>urn:liberty:security:2006-08:TLS:SAMLV2</disco:SecurityMechID>
2879             <disco:Action>urn:x-test:atm:2007-11:Withdraw</disco:Action>
2880             <disco:Action>urn:x-test:atm:2007-11:Transfer</disco:Action>
2881         </disco:EndpointContext>
2882     </disco:ServiceContext>
2883 </disco:SvcMD>
2884 </disco:SvcMDRegister>

```

2885 Things to note about this registration:

- 2886 • there SvcMDs use many of the features of the data structures to define various contexts in which the service can  
2887 be reached and which actions and/or options are available at said services.
- 2888 • We do **NOT** represent that the SvcMDs registered here are typical or normal. They were explicitly created to  
2889 examine/test some of the intricacies of handling queries against the SvcMD data structure.

2890 Server Response:

```
2891 <disco:SvcMDRegisterResponse xmlns:disco="urn:liberty:disco:2006-08">
2892   <lu:Status xmlns:lu="urn:liberty:util:2006-08" code="OK"/>
2893   <disco:SvcMDID>SVCMDIDMFPMblwcrSiwnu8D3BGO</disco:SvcMDID>
2894   <disco:SvcMDID>SVCMDIDeZQGkXuw750_uh3Q9OLO</disco:SvcMDID>
2895   <disco:SvcMDID>SVCMDIDrpG8SJpeUdSmla_ZSFUN</disco:SvcMDID>
2896 </disco:SvcMDRegisterResponse >
```

2897 Things to note about this response:

- 2898 • the registration was successful (status code is OK).
- 2899 • the svcMDID for each of the added elements is returned in sequence. The first SvcMDID to the first <SvcMD> in  
2900 the request, the second to the second, and so forth.

### 2901 3.13.17. Query Calendar Service

2902 Query for the test calendar service for this user.

```
2903 <disco:Query xmlns:disco="urn:liberty:disco:2006-08" id="discReq">
2904   <disco:RequestedService>
2905     <disco:ServiceType>urn:x-test:cal:2008-03</disco:ServiceType>
2906     <disco:SecurityMechID>urn:liberty:security:2006-08:null:SAMLV2</disco:SecurityMechID>
2907   </disco:RequestedService>
2908 </disco:Query>
```

2909 Things to note about this query:

- 2910 • the service type on this query is one of those specified in one of the SvcMDs that were just registered in the  
2911 previous call.

2912 Server Response:

```
2913 <disco:QueryResponse xmlns:disco="urn:liberty:disco:2006-08">
2914   <lu:Status xmlns:lu="urn:liberty:util:2006-08" code="Failed">
2915     <lu:Status code="NoResults"/>
2916   </lu:Status>
2917 </disco:QueryResponse>
```

2918 Things to note about this response:

- 2919 • the query failed (status code was Failed). This is because while the SvcMD has been *registered*, it has **not** been  
2920 *associated* with the user.

### 2921 3.13.18. SvcMDAssociationAdd the Calendar Service SvcMD

2922 Associate a single SvcMD with the current principal.

```
2923 <disco:SvcMDAssociationAdd xmlns:disco="urn:liberty:disco:2006-08">
2924   <disco:SvcMDID>SVCMDIDMFPMblwcrSiwnu8D3BGO</disco:SvcMDID>
2925 </disco:SvcMDAssociationAdd>
```

2926 Things to note about this query:

- 2927 • the SvcMDID specified is the SvcMDID assigned to the test calendar service registered above (the first SvcMD in  
2928 the multi-SvcMD registration).

2929 Server Response:

```
2930 <disco:SvcMDAssociationAddResponse xmlns:disco="urn:liberty:disco:2006-08">
2931   <lu:Status xmlns:lu="urn:liberty:util:2006-08" code="OK"/>
2932 </disco:SvcMDAssociationAddResponse>
```

2933 Things to note about this response:

- 2934 • the query was successful (status code is OK).
- 2935 • this service is now associated with the principal and available to subsequent Discovery Service queries.

2936 **3.13.19. SvcMDAssociationQuery w/SvcMDID**

2937 Query the SvcMD Associations to see if that SvcMD is now associated with the principal

```
2938 <disco:SvcMDAssociationQuery xmlns:disco="urn:liberty:disco:2006-08">
2939   <disco:SvcMDID>SVCMDIDMfPMblwcrSiwnu8D3BGO</disco:SvcMDID>
2940 </disco:SvcMDAssociationQuery>
```

2941 Things to note about this query:

- 2942 • the SvcMDID provided is the SvcMDID that was just associated with the user in the previous request

2943 Server Response:

```
2944 <disco:SvcMDAssociationQueryResponse xmlns:disco="urn:liberty:disco:2006-08">
2945   <lu:Status xmlns:lu="urn:liberty:util:2006-08" code="OK"/>
2946   <disco:SvcMDID>SVCMDIDMfPMblwcrSiwnu8D3BGO</disco:SvcMDID>
2947 </disco:SvcMDAssociationQueryResponse>
```

2948 Things to note about this response:

- 2949 • the query was successful (status code is OK).
- 2950 • the matching SvcMDIDs were returned (since you can query more than one and not all may have matched on a
- 2951 successful query)

2952 **3.13.20. SvcMDAssociationQuery w/o SvcMDID**

2953 Query all SvcMD Associations for the current principal.

```
2954 <disco:SvcMDAssociationQuery xmlns:disco="urn:liberty:disco:2006-08"/>
```

2955 Things to note about this query:

- 2956 • No SvcMDIDs are specified which causes all SvcMD associations that were created by the invoking provider to
- 2957 be listed.

2958 Server Response:

```
2959 <disco:SvcMDAssociationQueryResponse xmlns:disco="urn:liberty:disco:2006-08">
2960   <lu:Status xmlns:lu="urn:liberty:util:2006-08" code="OK"/>
2961   <disco:SvcMDID>SVCMDIDy_7yFABwuFPkgYIjKcJv</disco:SvcMDID>
2962   <disco:SvcMDID>SVCMDIDMfPMblwcrSiwnu8D3BGO</disco:SvcMDID>
2963 </disco:SvcMDAssociationQueryResponse>
```

2964 Things to note about this response:

- 2965 • the query was successful (status code is OK).

- 2966 • two SvcMDIDs were returned. One for the service recently associated and one for the ID-WSF People Service  
2967 that had been previously associated. (This call, as well as the previous registration and association calls, was made  
2968 in the context of that provider so both should be visible).

### 2969 3.13.21. Query Calendar Service (again)

2970 Query for the test calendar service for this user (which previously failed, but now the SvcMD has been associated with  
2971 this principal).

```
2972 <disco:Query xmlns:disco="urn:liberty:disco:2006-08" id="discReq">
2973   <disco:RequestedService>
2974     <disco:ServiceType>urn:x-test:cal:2008-03</disco:ServiceType>
2975     <disco:SecurityMechID>urn:liberty:security:2006-08:null:SAMLV2</disco:SecurityMechID>
2976   </disco:RequestedService>
2977 </disco:Query>
```

2978 Things to note about this query:

- 2979 • again we're looking for the test calendar service (which has now been associated with the principal).
- 2980 • This query was constructed to be resolvable only by the data within the 2nd <ServiceContext> element (it's the  
2981 only one with the ServiceType "urn:x-test:cal:2008-03") and within there, the 2nd <EndpointContext> element  
2982 (it's the only one with the SecurithMechID ...:null:SAMLV2).

2983 Server Response:

```
2984 <disco:QueryResponse xmlns:disco="urn:liberty:disco:2006-08">
2985   <lu:Status xmlns:lu="urn:liberty:util:2006-08" code="OK"/>
2986   <wsa:EndpointReference
2987     xmlns:wsa="http://www.w3.org/2005/08/addressing"
2988     xmlns:wsu="http://.../oasis-200401-wss-wssecurity-utility-1.0.xsd"
2989     notOnOrAfter="2006-04-06T17:40:28Z"
2990     wsu:Id="EPRID3jLhZ6fsjx3xFNF22Hyx">
2991     <wsa:Address>http://6-calendars.testing.com</wsa:Address>
2992     <wsa:Metadata>
2993       <disco:Abstract>TestDisco Test Calendar Service</disco:Abstract>
2994       <sbf:Framework xmlns:sbf="urn:liberty:sb" version="2.0"/>
2995       <disco:ProviderID>https://s-wsp.liberty-iop.org:8743/sp.xml</disco:ProviderID>
2996       <disco:ServiceType>urn:x-test:cal:2008-03</disco:ServiceType>
2997       <disco:SecurityContext>
2998         <disco:SecurityMechID>urn:liberty:security:2006-08:null:SAMLV2</disco:SecurityMechID>
2999         <sec:Token xmlns:sec="urn:liberty:security:2006-08"
3000           usage="urn:liberty:security:tokenusage:2006-08:SecurityToken">
3001           <sa:Assertion xmlns:sa="urn:oasis:names:tc:SAML:2.0:assertion"
3002             ID="CRED-HgB2SRBPptovTK7ckof"
3003             IssueInstant="2006-04-06T15:40:28Z"
3004             Version="2.0">
3005             ... assertion data was here ...
3006           </sa:Assertion>
3007         </sec:Token>
3008       </disco:SecurityContext>
3009     </wsa:Metadata>
3010   </wsa:EndpointReference>
3011 </disco:QueryResponse>
```

3012 Things to note about this response:

- 3013 • the query was successful (status code is OK).
- 3014 • the ID-WSF EPR was generated from the expected SvcMD elements (the unique address *http://6-*  
3015 *calendars.testing.com* shows this).

### 3.13.22. Query Calendar Service w/Action

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

```
3018 <disco:Query xmlns:disco="urn:liberty:disco:2006-08" id="discReq">
3019   <disco:RequestedService>
3020     <disco:ServiceType>urn:x-test:cal:2008-03</disco:ServiceType>
3021     <disco:SecurityMechID>urn:liberty:security:2006-08:null:SAMLV2</disco:SecurityMechID>
3022     <disco:Action>urn:x-test:cal:2008-03:GetMeeting</disco:Action>
3023   </disco:RequestedService>
3024 </disco:Query>
```

Things to note about this query:

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

Server Response:

```
3028 <disco:QueryResponse xmlns:disco="urn:liberty:disco:2006-08">
3029   <lu:Status xmlns:lu="urn:liberty:util:2006-08" code="OK"/>
3030   <wsa:EndpointReference xmlns:wsa="http://www.w3.org/2005/08/addressing"
3031     xmlns:wsu="http://.../oasis-200401-wss-wssecurity-utility-1.0.xsd"
3032     notOnOrAfter="2006-04-06T17:41:01Z"
3033     wsu:Id="EPRIDq9XRiWZpP_tftRcutOP">
3034     <wsa:Address>http://6-calendars.testing.com</wsa:Address>
3035     <wsa:Metadata>
3036       <disco:Abstract>TestDisco Test Calendar Service</disco:Abstract>
3037       <sbf:Framework xmlns:sbf="urn:liberty:sb" version="2.0"/>
3038       <disco:ProviderID>https://s-wsp.liberty-iop.org:8743/sp.xml</disco:ProviderID>
3039       <disco:ServiceType>urn:x-test:cal:2008-03</disco:ServiceType>
3040       <disco:SecurityContext>
3041         <disco:SecurityMechID>urn:liberty:security:2006-08:null:SAMLV2</disco:SecurityMechID>
3042         <sec:Token xmlns:sec="urn:liberty:security:2006-08"
3043           usage="urn:liberty:security:tokenusage:2006-08:SecurityToken">
3044           <sa:Assertion xmlns:sa="urn:oasis:names:tc:SAML:2.0:assertion"
3045             ID="CREDst8YWUPHbqUdMKjnV_I7"
3046             IssueInstant="2006-04-06T15:41:01Z"
3047             Version="2.0">
3048             ... assertion data was here ...
3049           </sa:Assertion>
3050         </sec:Token>
3051       </disco:SecurityContext>
3052     </wsa:Metadata>
3053   </wsa:EndpointReference>
3054 </disco:QueryResponse>
```

Things to note about this response:

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

### 3.13.23. Query Calendar Service w/resultsType=all

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

```

3064 <disco:Query xmlns:disco="urn:liberty:disco:2006-08" id="discReq">
3065   <disco:RequestedService resultsType="all">
3066     <disco:ServiceType>urn:x-test:cal:2008-03</disco:ServiceType>
3067     <disco:SecurityMechID>urn:liberty:security:2006-08:TLS:SAMLV2</disco:SecurityMechID>
3068   </disco:RequestedService>
3069 </disco:Query>

```

Things to note about this query:

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

Server Response:

```

3077 <disco:QueryResponse xmlns:disco="urn:liberty:disco:2006-08">
3078   <lu:Status xmlns:lu="urn:liberty:util:2006-08" code="OK"/>
3079   <wsa:EndpointReference
3080     xmlns:wsa="http://www.w3.org/2005/08/addressing"
3081     xmlns:wsu="http://.../oasis-200401-wss-wssecurity-utility-1.0.xsd"
3082     notOnOrAfter="2006-04-06T17:41:05Z" wsu:Id="EPRIDgNpGPKjwRsZVIC63VOMt">
3083     <wsa:Address>https://4-calendars.testing.com</wsa:Address>
3084     <wsa:Metadata>
3085       <disco:Abstract>TestDisco Test Calendar Service</disco:Abstract>
3086       <sbf:Framework xmlns:sbf="urn:liberty:sbf" version="2.0"/>
3087       <disco:ProviderID>https://s-wsp.liberty-iop.org:8743/sp.xml</disco:ProviderID>
3088       <disco:ServiceType>urn:x-test:cal:2008-03</disco:ServiceType>
3089       <disco:SecurityContext>
3090         <disco:SecurityMechID>urn:liberty:security:2006-08:TLS:SAMLV2</disco:SecurityMechID>
3091         <sec:Token xmlns:sec="urn:liberty:security:2006-08"
3092           usage="urn:liberty:security:tokenusage:2006-08:SecurityToken">
3093           <sa:Assertion xmlns:sa="urn:oasis:names:tc:SAML:2.0:assertion"
3094             ID="CREDYA0WkksXWLutGxWWsF2m"
3095             IssueInstant="2006-04-06T15:41:06Z" Version="2.0">
3096             ... assertion data was here ...
3097           </sa:Assertion>
3098         </sec:Token>
3099       </disco:SecurityContext>
3100     </wsa:Metadata>
3101   </wsa:EndpointReference>
3102   <wsa:EndpointReference xmlns:wsa="http://www.w3.org/2005/08/addressing"
3103     xmlns:wsu="http://.../oasis-200401-wss-wssecurity-utility-1.0.xsd"
3104     notOnOrAfter="2006-04-06T17:41:06Z" wsu:Id="EPRIDeB1MvrPuq-TphWEqQ7G0">
3105     <wsa:Address>https://5-calendars.testing.backup.com</wsa:Address>
3106     <wsa:Metadata>
3107       <disco:Abstract>TestDisco Test Calendar Service</disco:Abstract>
3108       <sbf:Framework xmlns:sbf="urn:liberty:sbf" version="2.0"/>
3109       <disco:ProviderID>https://s-wsp.liberty-iop.org:8743/sp.xml</disco:ProviderID>
3110       <disco:ServiceType>urn:x-test:cal:2008-03</disco:ServiceType>
3111       <disco:SecurityContext>
3112         <disco:SecurityMechID>urn:liberty:security:2006-08:TLS:SAMLV2</disco:SecurityMechID>
3113         <sec:Token xmlns:sec="urn:liberty:security:2006-08"
3114           usage="urn:liberty:security:tokenusage:2006-08:SecurityToken">
3115           <sa:Assertion xmlns:sa="urn:oasis:names:tc:SAML:2.0:assertion"
3116             ID="CRED816B4EvhFszmG01Ab5F7"
3117             IssueInstant="2006-04-06T15:41:06Z" Version="2.0">
3118             ... assertion data was here ...
3119           </sa:Assertion>

```

```

3120     </sec:Token>
3121     </disco:SecurityContext>
3122     </wsa:Metadata>
3123     </wsa:EndpointReference>
3124 </disco:QueryResponse>

```

3125 Things to note about this response:

- 3126 • the query was successful (status code is OK).
- 3127 • as expected 2 ID-WSF EPRs were returned because the two endpoints which matched the search criteria cannot
- 3128 be placed into the same EPR.

### 3129 3.13.24. Query for all Calendar Service EPRs

3130 A query of all of the data available for the calendar service

```

3131 <disco:Query xmlns:disco="urn:liberty:disco:2006-08" id="discReq">
3132   <disco:RequestedService resultsType="all">
3133     <disco:ServiceType>urn:x-test:cal:2008-03</disco:ServiceType>
3134     <disco:ServiceType>urn:x-test:cal:2006-01</disco:ServiceType>
3135     <disco:ServiceType>urn:x-test:cal:2006-09</disco:ServiceType>
3136     <disco:SecurityMechID>urn:liberty:security:2006-08:TLS:SAMLV2</disco:SecurityMechID>
3137     <disco:SecurityMechID>urn:liberty:security:2005-02:TLS:Bearer</disco:SecurityMechID>
3138     <disco:SecurityMechID>urn:liberty:security:2005-02:TLS:null</disco:SecurityMechID>
3139     <disco:SecurityMechID>urn:liberty:security:2006-08:null:SAMLV2</disco:SecurityMechID>
3140     <disco:SecurityMechID>urn:liberty:security:2005-02:null:Bearer</disco:SecurityMechID>
3141   </disco:RequestedService>
3142 </disco:Query>

```

3143 Things to note about this query:

- 3144 • All of the service types and all of the SecurityMechIDs in the Calendar Service SvcMD are specified.
- 3145 • The resultsType attribute is set to "all" indicating that the Discovery Service should return all possible results.

3146 Server Response:

```

3147 <disco:QueryResponse xmlns:disco="urn:liberty:disco:2006-08">
3148   <lu:Status xmlns:lu="urn:liberty:util:2006-08" code="OK"/>
3149   <wsa:EndpointReference xmlns:wsa="http://www.w3.org/2005/08/addressing"
3150     xmlns:wsu="http://.../oasis-200401-wss-wssecurity-utility-1.0.xsd"
3151     notOnOrAfter="2006-04-06T17:41:09Z" wsu:Id="EPRIDC08vEmUOfarryqg3zo8j">
3152     <wsa:Address>https://1-calendars.testing.com</wsa:Address>
3153     <wsa:Metadata>
3154       <disco:Abstract>TestDisco Test Calendar Service</disco:Abstract>
3155       <sbef:Framework xmlns:sbef="urn:liberty:sb" version="2.1"/>
3156       <disco:ProviderID>https://s-wsp.liberty-iop.org:8743/sp.xml</disco:ProviderID>
3157       <disco:ServiceType>urn:x-test:cal:2006-01</disco:ServiceType>
3158       <disco:ServiceType>urn:x-test:cal:2006-09</disco:ServiceType>
3159       <disco:SecurityContext>
3160         <disco:SecurityMechID>urn:liberty:security:2005-02:TLS:Bearer</disco:SecurityMechID>
3161         <sec:Token xmlns:sec="urn:liberty:security:2006-08"
3162           usage="urn:liberty:security:tokenusage:2006-08:SecurityToken">
3163           <sa:Assertion xmlns:sa="urn:oasis:names:tc:SAML:2.0:assertion"
3164             ID="CRED1QVGIZghandZqdE90QRa"
3165             IssueInstant="2006-04-06T15:41:09Z" Version="2.0">
3166             ... assertion data was here ...
3167           </sa:Assertion>
3168         </sec:Token>
3169       </disco:SecurityContext>
3170     </disco:SecurityContext>
3171     <disco:SecurityMechID>urn:liberty:security:2005-02:TLS:null</disco:SecurityMechID>
3172   </disco:SecurityContext>

```

```

3173     </wsa:Metadata>
3174 </wsa:EndpointReference>
3175 <wsa:EndpointReference xmlns:wsa="http://www.w3.org/2005/08/addressing"
3176     xmlns:wsu="http://.../oasis-200401-wss-wssecurity-utility-1.0.xsd"
3177     notOnOrAfter="2006-04-06T17:41:09Z" wsu:Id="EPRIDqEcv9d0T43OTQLdhB116">
3178   <wsa:Address>https://2-calendars.testing.com</wsa:Address>
3179   <wsa:Metadata>
3180     ... Metatdata was here ...
3181   </wsa:Metadata>
3182 </wsa:EndpointReference>
3183 <wsa:EndpointReference xmlns:wsa="http://www.w3.org/2005/08/addressing"
3184     xmlns:wsu="http://.../oasis-200401-wss-wssecurity-utility-1.0.xsd"
3185     notOnOrAfter="2006-04-06T17:41:09Z" wsu:Id="EPRIDRP8-wMwgf4A_7DsHS">
3186   <wsa:Address>http://3-calendars.testing.com</wsa:Address>
3187   <wsa:Metadata>
3188     ... Metatdata was here ...
3189   </wsa:Metadata>
3190 </wsa:EndpointReference>
3191 <wsa:EndpointReference xmlns:wsa="http://www.w3.org/2005/08/addressing"
3192     xmlns:wsu="http://.../oasis-200401-wss-wssecurity-utility-1.0.xsd"
3193     notOnOrAfter="2006-04-06T17:41:09Z" wsu:Id="EPRIDkL15ahJdrutDV6gMPLyr">
3194   <wsa:Address>https://4-calendars.testing.com</wsa:Address>
3195   <wsa:Metadata>
3196     ... Metatdata was here ...
3197   </wsa:Metadata>
3198 </wsa:EndpointReference>
3199 <wsa:EndpointReference xmlns:wsa="http://www.w3.org/2005/08/addressing"
3200     xmlns:wsu="http://.../oasis-200401-wss-wssecurity-utility-1.0.xsd"
3201     notOnOrAfter="2006-04-06T17:41:09Z" wsu:Id="EPRID0TmCLBCVoYZV7_R8jgky">
3202   <wsa:Address>https://5-calendars.testing.backup.com</wsa:Address>
3203   <wsa:Metadata>
3204     ... Metatdata was here ...
3205   </wsa:Metadata>
3206 </wsa:EndpointReference>
3207 <wsa:EndpointReference xmlns:wsa="http://www.w3.org/2005/08/addressing"
3208     xmlns:wsu="http://.../oasis-200401-wss-wssecurity-utility-1.0.xsd"
3209     notOnOrAfter="2006-04-06T17:41:09Z" wsu:Id="EPRID9nsZ7Ufg0G5UMMymJLRp">
3210   <wsa:Address>http://6-calendars.testing.com</wsa:Address>
3211   <wsa:Metadata>
3212     ... Metatdata was here ...
3213   </wsa:Metadata>
3214 </wsa:EndpointReference>
3215 </disco:QueryResponse>

```

3216 Things to note about this response:

- 3217 • the query was successful (status code is OK).
- 3218 • Much of the data in this response was elided in order to not waste alot of paper. The first EPR in the response is  
3219 shown fairly completely.
- 3220 • There are 6 EPRs (the minimum way to represent all of the data in the SvcMD that matched the requested  
3221 parameters).
- 3222 • The first 3 EPRs have two service types (for the 2006-01 and the 2006-09 versions of the calendar service) as a  
3223 single ID-WSF EPR may have multiple service types if they are all related to the same logical service.

### 3224 3.13.25. Query for one Calendar Service EPRs

3225 A query for the first EPR out of all those within the Calendar service (to show the impact of the `resultsType` setting  
3226 of *only-one*).

```

3227 <disco:Query xmlns:disco="urn:liberty:disco:2006-08" id="discReq">
3228   <disco:RequestedService resultsType="only-one">

```

```

3229 <disco:ServiceType>urn:x-test:cal:2008-03</disco:ServiceType>
3230 <disco:ServiceType>urn:x-test:cal:2006-01</disco:ServiceType>
3231 <disco:ServiceType>urn:x-test:cal:2006-09</disco:ServiceType>
3232 <disco:SecurityMechID>urn:liberty:security:2006-08:TLS:SAMLV2</disco:SecurityMechID>
3233 <disco:SecurityMechID>urn:liberty:security:2005-02:TLS:Bearer</disco:SecurityMechID>
3234 <disco:SecurityMechID>urn:liberty:security:2005-02:TLS:null</disco:SecurityMechID>
3235 <disco:SecurityMechID>urn:liberty:security:2006-08:null:SAMLV2</disco:SecurityMechID>
3236 <disco:SecurityMechID>urn:liberty:security:2005-02:null:Bearer</disco:SecurityMechID>
3237 </disco:RequestedService>
3238 </disco:Query>

```

3239 Things to note about this query:

- 3240 • All of the service types and all of the SecurityMechIDs in the Calendar Service SvcMD are specified.
- 3241 • The `resultsType` attribute is set to "*only-one*" indicating that the Discovery Service should only return the first
- 3242 matching ID-WSF EPR.

3243 Server Response:

```

3244 <disco:QueryResponse xmlns:disco="urn:liberty:disco:2006-08">
3245 <lu:Status xmlns:lu="urn:liberty:util:2006-08" code="OK"/>
3246 <wsa:EndpointReference xmlns:wsa="http://www.w3.org/2005/08/addressing"
3247   xmlns:wsu="http://.../oasis-200401-wss-wssecurity-utility-1.0.xsd"
3248   notOnOrAfter="2006-04-06T17:41:14Z" wsu:Id="EPRIDVh71zxFCQu4yWBOKLPzH">
3249 <wsa:Address>https://1-calendars.testing.com</wsa:Address>
3250 <wsa:Metadata>
3251 <disco:Abstract>TestDisco Test Calendar Service</disco:Abstract>
3252 <sbf:Framework xmlns:sbf="urn:liberty:sbf" version="2.1"/>
3253 <disco:ProviderID>https://s-wsp.liberty-iop.org:8743/sp.xml</disco:ProviderID>
3254 <disco:ServiceType>urn:x-test:cal:2006-01</disco:ServiceType>
3255 <disco:ServiceType>urn:x-test:cal:2006-09</disco:ServiceType>
3256 <disco:SecurityContext>
3257 <disco:SecurityMechID>urn:liberty:security:2005-02:TLS:Bearer</disco:SecurityMechID>
3258 <sec:Token xmlns:sec="urn:liberty:security:2006-08"
3259   usage="urn:liberty:security:tokenusage:2006-08:SecurityToken">
3260 <sa:Assertion xmlns:sa="urn:oasis:names:tc:SAML:2.0:assertion"
3261   ID="CREDYjkZsJiWfQwKf5gaJoze"
3262   IssueInstant="2006-04-06T15:41:14Z" Version="2.0">
3263   ... assertion data was here ...
3264 </sa:Assertion>
3265 </sec:Token>
3266 </disco:SecurityContext>
3267 <disco:SecurityContext>
3268 <disco:SecurityMechID>urn:liberty:security:2005-02:TLS:null</disco:SecurityMechID>
3269 </disco:SecurityContext>
3270 </wsa:Metadata>
3271 </wsa:EndpointReference>
3272 </disco:QueryResponse>

```

3273 Things to note about this response:

- 3274 • the query was successful (status code is OK).
- 3275 • Only one ID-WSF EPR was returned (because of the query parameters) (as compared to the 6 returned in the
- 3276 previous example with the same query other than the `resultsType` attribute).
- 3277 • The one ID-WSF EPR that is returned is the *\*first\** EPR that would have otherwise been returned.

### 3.13.26. Query specific version of Calendar Service

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

```

3280 <disco:Query xmlns:disco="urn:liberty:disco:2006-08" id="discReq">
3281   <disco:RequestedService>
3282     <disco:ServiceType>urn:x-test:cal:2006-09</disco:ServiceType>
3283     <disco:SecurityMechID>urn:liberty:security:2005-02:TLS:Bearer</disco:SecurityMechID>
3284   </disco:RequestedService>
3285 </disco:Query>

```

Things to note about this query:

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

Server Response:

```

3291 <disco:QueryResponse xmlns:disco="urn:liberty:disco:2006-08">
3292   <lu:Status xmlns:lu="urn:liberty:util:2006-08" code="OK"/>
3293   <wsa:EndpointReference xmlns:wsa="http://www.w3.org/2005/08/addressing"
3294     xmlns:wsu="http://.../oasis-200401-wss-wssecurity-utility-1.0.xsd"
3295     notOnOrAfter="2006-04-06T17:41:19Z" wsu:Id="EPRIDVsvRlCWheodUvbDFelh">
3296     <wsa:Address>https://1-calendars.testing.com</wsa:Address>
3297     <wsa:Metadata>
3298       <disco:Abstract>TestDisco Test Calendar Service</disco:Abstract>
3299       <sbf:Framework xmlns:sbf="urn:liberty:sb" version="2.1"/>
3300       <disco:ProviderID>https://s-wsp.liberty-iop.org:8743/sp.xml</disco:ProviderID>
3301       <disco:ServiceType>urn:x-test:cal:2006-09</disco:ServiceType>
3302       <disco:SecurityContext>
3303         <disco:SecurityMechID>urn:liberty:security:2005-02:TLS:Bearer</disco:SecurityMechID>
3304         <sec:Token xmlns:sec="urn:liberty:security:2006-08"
3305           usage="urn:liberty:security:tokenusage:2006-08:SecurityToken">
3306           <sa:Assertion xmlns:sa="urn:oasis:names:tc:SAML:2.0:assertion"
3307             ID="CREDFy-6w5S8iOVearNxJur_"
3308             IssueInstant="2006-04-06T15:41:19Z" Version="2.0">
3309             ... assertion data was here ...
3310           </sa:Assertion>
3311         </sec:Token>
3312       </disco:SecurityContext>
3313     </wsa:Metadata>
3314   </wsa:EndpointReference>
3315   <wsa:EndpointReference xmlns:wsa="http://www.w3.org/2005/08/addressing"
3316     xmlns:wsu="http://.../oasis-200401-wss-wssecurity-utility-1.0.xsd"
3317     notOnOrAfter="2006-04-06T17:41:19Z" wsu:Id="EPRIDDKWLXiXVXpSJJi3oY3ge">
3318     <wsa:Address>https://2-calendars.testing.com</wsa:Address>
3319     <wsa:Metadata>
3320       <disco:Abstract>TestDisco Test Calendar Service</disco:Abstract>
3321       <sbf:Framework xmlns:sbf="urn:liberty:sb" version="2.0"/>
3322       <disco:ProviderID>https://s-wsp.liberty-iop.org:8743/sp.xml</disco:ProviderID>
3323       <disco:ServiceType>urn:x-test:cal:2006-09</disco:ServiceType>
3324       <disco:SecurityContext>
3325         <disco:SecurityMechID>urn:liberty:security:2005-02:TLS:Bearer</disco:SecurityMechID>
3326         <sec:Token xmlns:sec="urn:liberty:security:2006-08"
3327           usage="urn:liberty:security:tokenusage:2006-08:SecurityToken">
3328           <sa:Assertion xmlns:sa="urn:oasis:names:tc:SAML:2.0:assertion"
3329             ID="CREDW_y92CoMn7x57YOWzbnB"
3330             IssueInstant="2006-04-06T15:41:20Z" Version="2.0">
3331             ... assertion data was here ...
3332           </sa:Assertion>
3333         </sec:Token>
3334       </disco:SecurityContext>
3335     </wsa:Metadata>
3336   </wsa:EndpointReference>
3337 </disco:QueryResponse>

```

3338 Things to note about this response:

- 3339 • the query was successful (status code is OK).
- 3340 • Two ID-WSF EPRs were returned by the Discovery Service because the matching data had two different endpoints  
3341 which must be represented in separate EPRs.
- 3342 • Because there was no `resultsType` specified on the request, the DS could have returned just the first ID-WSF  
3343 EPR if they chose to as that ID-WSF EPR meets the basic requirements of the request. In this particular instance  
3344 the Discovery Service acted as if "all" had been specified, but that should not be depended upon.
- 3345 If you need a particular `resultsType` behavior, you need to specify it on the request.
- 3346 • The Discovery Service could have used the same assertion for the two ID-WSF EPRs if appropriate. In such a  
3347 case, the one of the ID-WSF EPRs would have had a `<sec:Token>` element with a `ref` attribute containing a  
3348 pointer to the `<sec:Token>` in the other ID-WSF EPR.

### 3349 3.13.27. SvcMDReplace Calendar Service

3350 Replace the SvcMD for the Calendar service with a much simpler SvcMD.

```
3351 <disco:SvcMDReplace xmlns:disco="urn:liberty:disco:2006-08">
3352   <disco:SvcMD svcMDID="SVCMDIDMfPMblwcRSiwnu8D3BGO">
3353     <disco:Abstract>TestDisco Test Calendar Service</disco:Abstract>
3354     <disco:ProviderID>https://s-wsp.liberty-iop.org:8743/sp.xml</disco:ProviderID>
3355     <disco:ServiceContext>
3356       <disco:ServiceType>urn:x-test:cal:2008-03</disco:ServiceType>
3357       <disco:EndpointContext>
3358         <disco:Address>https://calendar.testing.com</disco:Address>
3359         <sbf:Framework xmlns:sbf="urn:liberty:sb" version="2.0"/>
3360         <disco:SecurityMechID>urn:liberty:security:2006-08:TLS:SAMLV2</disco:SecurityMechID>
3361       </disco:EndpointContext>
3362     </disco:ServiceContext>
3363   </disco:SvcMD>
3364 </disco:SvcMDReplace>
```

3365 Things to note about this query:

- 3366 • the SvcMDID is the SvcMDID for the complex Calendar Service SvcMD that we have been querying over the past  
3367 few requests.
- 3368 • the new SvcMD is much simpler.

3369 Server Response:

```
3370 <disco:SvcMDReplaceResponse xmlns:disco="urn:liberty:disco:2006-08">
3371   <lu:Status xmlns:lu="urn:liberty:util:2006-08" code="OK"/>
3372 </disco:SvcMDReplaceResponse>
```

3373 Things to note about this response:

- 3374 • the replacement was successful (status code is OK).

### 3375 **3.13.28. Query old Calendar Service**

3376 Query the Calendar Service data that was replaced.

```
3377 <disco:Query xmlns:disco="urn:liberty:disco:2006-08" id="discReq">
3378   <disco:RequestedService>
3379     <disco:ServiceType>urn:x-test:cal:2008-03</disco:ServiceType>
3380     <disco:SecurityMechID>urn:liberty:security:2006-08:null:SAMLV2</disco:SecurityMechID>
3381   </disco:RequestedService>
3382 </disco:Query>
```

3383 Things to note about this query:

- 3384 • this is the same query we ran earlier that obtained results.

3385 Server Response:

```
3386 <disco:QueryResponse xmlns:disco="urn:liberty:disco:2006-08">
3387   <lu:Status xmlns:lu="urn:liberty:util:2006-08" code="Failed">
3388     <lu:Status code="NoResults"/>
3389   </lu:Status>
3390 </disco:QueryResponse>
```

3391 Things to note about this response:

- 3392 • the query failed (status code is Failed)
- 3393 • the sub-status is NoResults - indicating that no matching data was found
- 3394 • The new (replacement) SvcMD for the Calendar Service has taken effect for the principal without the need for it  
3395 to be associated with the principal (as it is already associated).

### 3396 **3.13.29. Query for all Calendar Service EPRs**

3397 A query of all of the data available for the calendar service (same query we ran a few requests ago).

```
3398 <disco:Query xmlns:disco="urn:liberty:disco:2006-08" id="discReq">
3399   <disco:RequestedService resultsType="all">
3400     <disco:ServiceType>urn:x-test:cal:2008-03</disco:ServiceType>
3401     <disco:ServiceType>urn:x-test:cal:2006-01</disco:ServiceType>
3402     <disco:ServiceType>urn:x-test:cal:2006-09</disco:ServiceType>
3403     <disco:SecurityMechID>urn:liberty:security:2006-08:TLS:SAMLV2</disco:SecurityMechID>
3404     <disco:SecurityMechID>urn:liberty:security:2005-02:TLS:Bearer</disco:SecurityMechID>
3405     <disco:SecurityMechID>urn:liberty:security:2005-02:TLS:null</disco:SecurityMechID>
3406     <disco:SecurityMechID>urn:liberty:security:2006-08:null:SAMLV2</disco:SecurityMechID>
3407     <disco:SecurityMechID>urn:liberty:security:2005-02:null:Bearer</disco:SecurityMechID>
3408   </disco:RequestedService>
3409 </disco:Query>
```

3410 Things to note about this query:

- 3411 • All of the service types and all of the SecurityMechIDs in the Calendar Service SvcMD are specified.
- 3412 • The resultsType attribute is set to "all" indicating that the Discovery Service should return all possible results.

3413 Server Response:

```
3414 <disco:QueryResponse xmlns:disco="urn:liberty:disco:2006-08">
3415   <lu:Status xmlns:lu="urn:liberty:util:2006-08" code="OK"/>
3416   <wsa:EndpointReference xmlns:wsa="http://www.w3.org/2005/08/addressing"
3417     xmlns:wsu="http://.../oasis-200401-wss-wssecurity-utility-1.0.xsd"
3418     notOnOrAfter="2006-04-06T17:41:32Z" wsu:Id="EPRID23j-JGWAXusqiV80ARzC">
3419     <wsa:Address>https://calendar.testing.com</wsa:Address>
3420     <wsa:Metadata>
3421       <disco:Abstract>TestDisco Test Calendar Service</disco:Abstract>
3422       <sbef:Framework xmlns:sbef="urn:liberty:sb" version="2.0"/>
3423       <disco:ProviderID>https://s-wsp.liberty-iop.org:8743/sp.xml</disco:ProviderID>
3424       <disco:ServiceType>urn:x-test:cal:2008-03</disco:ServiceType>
3425       <disco:SecurityContext>
3426         <disco:SecurityMechID>urn:liberty:security:2006-08:TLS:SAMLV2</disco:SecurityMechID>
3427         <sec:Token xmlns:sec="urn:liberty:security:2006-08"
3428           usage="urn:liberty:security:tokenusage:2006-08:SecurityToken">
3429           <sa:Assertion xmlns:sa="urn:oasis:names:tc:SAML:2.0:assertion"
3430             ID="CREDxHDqbb4F1TV7jSYxlmxq"
3431             IssueInstant="2006-04-06T15:41:32Z" Version="2.0">
3432             ... assertion data was here ...
3433           </sa:Assertion>
3434         </sec:Token>
3435       </disco:SecurityContext>
3436     </wsa:Metadata>
3437   </wsa:EndpointReference>
3438 </disco:QueryResponse>
```

3439 Things to note about this response:

- 3440 • the query was successful (status code is OK).
- 3441 • Only the data from the replacement SvcMD is represented in the one ID-WSF EPR in the results.

3442 **3.13.30. SvcMDAssociationAdd the ATM Service SvcMD**

3443 Associate the ATM Service SvcMD with the current principal.

```
3444 <disco:SvcMDAssociationAdd xmlns:disco="urn:liberty:disco:2006-08">
3445   <disco:SvcMDID>SvcMDIDrpG8SJpeUdSmla_ZSFUN</disco:SvcMDID>
3446 </disco:SvcMDAssociationAdd>
```

3447 Things to note about this query:

- 3448 • 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).

3450 Server Response:

```
3451 <disco:SvcMDAssociationAddResponse xmlns:disco="urn:liberty:disco:2006-08">
3452   <lu:Status xmlns:lu="urn:liberty:util:2006-08" code="OK"/>
3453 </disco:SvcMDAssociationAddResponse>
```

3454 Things to note about this response:

- 3455 • the association was successful (status code is OK).
- 3456 • the ATM Service SvcMD is now associated with the principal and available to subsequent Discovery Service queries.

### 3.13.31. Query ATM Service w/resultsType=best

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

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

Things to note about this query:

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

Server Response:

```
<disco:QueryResponse xmlns:disco="urn:liberty:disco:2006-08">
  <lu:Status xmlns:lu="urn:liberty:util:2006-08" code="OK"/>
  <wsa:EndpointReference xmlns:wsa="http://www.w3.org/2005/08/addressing"
    xmlns:wsu="http://.../oasis-200401-wss-wssecurity-utility-1.0.xsd"
    notOnOrAfter="2006-04-06T17:41:43Z" wsu:Id="EPRIDfmT9HOSbmMs3jZ1_qSuY">
    <wsa:Address>https://test2.atm.CA.US.testing.com</wsa:Address>
    <wsa:Metadata>
      <disco:Abstract>TestDisco Test ATM Service</disco:Abstract>
      <sbf:Framework xmlns:sbf="urn:liberty:sb" version="2.0"/>
      <disco:ProviderID>https://s-wsp.liberty-iop.org:8743/sp.xml</disco:ProviderID>
      <disco:ServiceType>urn:x-test:atm:2003-03</disco:ServiceType>
      <disco:SecurityContext>
        <disco:SecurityMechID urn:liberty:security:2006-08:TLS:SAMLV2/>
        <sec:Token xmlns:sec="urn:liberty:security:2006-08"
          usage="urn:liberty:security:tokenusage:2006-08:SecurityToken">
          <sa:Assertion xmlns:sa="urn:oasis:names:tc:SAML:2.0:assertion"
            ID="CREDS8x8pCKTOzaQMI14fkel"
            IssueInstant="2006-04-06T15:41:44Z" Version="2.0">
            ... assertion data was here ...
          </sa:Assertion>
        </sec:Token>
      </disco:SecurityContext>
      <disco:Options>
        <disco:Option urn:x-test:atm:options:testopt1/>
        <disco:Option urn:x-test:atm:options:testopt2/>
        <disco:Option urn:x-test:atm:options:testopt3/>
      </disco:Options>
      <disco:Action urn:x-test:atm:2007-11:GetBalance/>
    </wsa:Metadata>
  </wsa:EndpointReference>
  <wsa:EndpointReference xmlns:wsa="http://www.w3.org/2005/08/addressing"
    xmlns:wsu="http://.../oasis-200401-wss-wssecurity-utility-1.0.xsd"
    notOnOrAfter="2006-04-06T17:41:44Z" wsu:Id="EPRIDdHYSEKB8_w5bPicTPe8o">
    <wsa:Address>https://readers.atm.CA.US.testing.com</wsa:Address>
    <wsa:Metadata>
      <disco:Abstract>TestDisco Test ATM Service</disco:Abstract>
      <sbf:Framework xmlns:sbf="urn:liberty:sb" version="2.0"/>
      <disco:ProviderID>https://s-wsp.liberty-iop.org:8743/sp.xml</disco:ProviderID>
      <disco:ServiceType>urn:x-test:atm:2003-03</disco:ServiceType>
      <disco:SecurityContext>
        <disco:SecurityMechID urn:liberty:security:2006-08:TLS:SAMLV2/>
        <sec:Token xmlns:sec="urn:liberty:security:2006-08"
          usage="urn:liberty:security:tokenusage:2006-08:SecurityToken">
          <sa:Assertion xmlns:sa="urn:oasis:names:tc:SAML:2.0:assertion"
            ID="CRED4nQ1FHP3vzuoqRhTMZrf"
            IssueInstant="2006-04-06T15:41:44Z" Version="2.0">

```

```

3517     ... assertion data was here ...
3518     </sa:Assertion>
3519     </sec:Token>
3520 </disco:SecurityContext>
3521 <disco:Options>
3522     <disco:Option>urn:x-test:atm:options:testopt1</disco:Option>
3523     <disco:Option>urn:x-test:atm:options:testopt2</disco:Option>
3524     <disco:Option>urn:x-test:atm:options:testopt3</disco:Option>
3525 </disco:Options>
3526 <disco:Action>urn:x-test:atm:2007-11:GetBalance</disco:Action>
3527 <disco:Action>urn:x-test:atm:2007-11:ListAccounts</disco:Action>
3528 </wsa:Metadata>
3529 </wsa:EndpointReference>
3530 <wsa:EndpointReference xmlns:wsa="http://www.w3.org/2005/08/addressing"
3531     xmlns:wsu="http://.../oasis-200401-wss-wssecurity-utility-1.0.xsd"
3532     notOnOrAfter="2006-04-06T17:41:44Z" wsu:Id="EPRIDggifVjR-zSAxkokPyfCo">
3533 <wsa:Address>https://writers.atm.testing.com</wsa:Address>
3534 <wsa:Metadata>
3535     <disco:Abstract>TestDisco Test ATM Service</disco:Abstract>
3536     <sbf:Framework xmlns:sbf="urn:liberty:sb" version="2.0"/>
3537     <disco:ProviderID>https://s-wsp.liberty-iop.org:8743/sp.xml</disco:ProviderID>
3538     <disco:ServiceType>urn:x-test:atm:2003-03</disco:ServiceType>
3539     <disco:SecurityContext>
3540     <disco:SecurityMechID>urn:liberty:security:2006-08:TLS:SAMLV2</disco:SecurityMechID>
3541     <sec:Token xmlns:sec="urn:liberty:security:2006-08"
3542         usage="urn:liberty:security:tokenusage:2006-08:SecurityToken">
3543     <sa:Assertion xmlns:sa="urn:oasis:names:tc:SAML:2.0:assertion"
3544         ID="CREDZNT1blH6VxPNrpwawtF6"
3545         IssueInstant="2006-04-06T15:41:44Z" Version="2.0">
3546     ... assertion data was here ...
3547     </sa:Assertion>
3548     </sec:Token>
3549 </disco:SecurityContext>
3550 <disco:Options>
3551     <disco:Option>urn:x-test:atm:options:testopt1</disco:Option>
3552     <disco:Option>urn:x-test:atm:options:testopt2</disco:Option>
3553     <disco:Option>urn:x-test:atm:options:testopt3</disco:Option>
3554 </disco:Options>
3555 <disco:Action>urn:x-test:atm:2007-11:Withdraw</disco:Action>
3556 <disco:Action>urn:x-test:atm:2007-11:Transfer</disco:Action>
3557 </wsa:Metadata>
3558 </wsa:EndpointReference>
3559 </disco:QueryResponse>

```

3560 Things to note about this response:

- 3561 • the query was successful (status code is OK).
- 3562 • It took three ID-WSF EPRs to represent the set of actions at the ATM Service.
- 3563 • One might think that because the "...GetBalance" action is on both the 1st and 2nd ID-WSF EPR and it is the only  
3564 action on the 1st ID-WSF EPR, the results could have excluded that ID-WSF EPR and the client would still get to  
3565 all of the resources at the ATM Service.
- 3566 However, the WSP placed the 1st "...GetBalance" action in the first <EndpointContext> and therefore gives it a  
3567 higher priority in the results.
- 3568 • The Discovery service could have left off the "...GetBalance" action on the 2nd ID-WSF EPR but that wouldn't  
3569 have been much of a savings and so it was included. Clients should NOT depend upon this type of behavior.
- 3570 • Even though Options were not specified on the request, they are specified in the SvcMD and so are included in the  
3571 ID-WSF EPRs generated from that SvcMD.

### 3572 3.13.32. Query ATM Service w/Withdraw Action

3573 Query for the ATM Service where "...Withdraw" action is available.

```
3574 <disco:Query xmlns:disco="urn:liberty:disco:2006-08" id="discReq">
3575   <disco:RequestedService resultsType="all">
3576     <disco:ServiceType>urn:x-test:atm:2003-03</disco:ServiceType>
3577     <disco:SecurityMechID>urn:liberty:security:2006-08:TLS:SAMLV2</disco:SecurityMechID>
3578     <disco:Action>urn:x-test:atm:2007-11:Withdraw</disco:Action>
3579   </disco:RequestedService>
3580 </disco:Query>
```

3581 Things to note about this query:

- 3582 • The <disco:Action> element is specified with the "urn:x-test:atm:2007-11:Withdraw") action value. So the  
3583 client only intends to use this operation at the ATM Service.

3584 Server Response:

```
3585 <disco:QueryResponse xmlns:disco="urn:liberty:disco:2006-08">
3586   <lu:Status xmlns:lu="urn:liberty:util:2006-08" code="OK"/>
3587   <wsa:EndpointReference xmlns:wsa="http://www.w3.org/2005/08/addressing"
3588     xmlns:wsu="http://.../oasis-200401-wss-wssecurity-utility-1.0.xsd"
3589     notOnOrAfter="2006-04-06T17:41:52Z" wsu:Id="EPRIDkueZCk5N4IpSmX-0BD-9">
3590     <wsa:Address>https://writers.atm.testing.com</wsa:Address>
3591     <wsa:Metadata>
3592       <disco:Abstract>TestDisco Test ATM Service</disco:Abstract>
3593       <sbef:Framework xmlns:sbef="urn:liberty:sb" version="2.0"/>
3594       <disco:ProviderID>https://s-wsp.liberty-iop.org:8743/sp.xml</disco:ProviderID>
3595       <disco:ServiceType>urn:x-test:atm:2003-03</disco:ServiceType>
3596       <disco:SecurityContext>
3597         <disco:SecurityMechID>urn:liberty:security:2006-08:TLS:SAMLV2</disco:SecurityMechID>
3598         <sec:Token xmlns:sec="urn:liberty:security:2006-08"
3599           usage="urn:liberty:security:tokenusage:2006-08:SecurityToken">
3600           <sa:Assertion xmlns:sa="urn:oasis:names:tc:SAML:2.0:assertion"
3601             ID="CREDYfLT1S7tcZ8LkuU-zlrs"
3602             IssueInstant="2006-04-06T15:41:52Z" Version="2.0">
3603             ... assertion data was here ...
3604           </sa:Assertion>
3605         </sec:Token>
3606       </disco:SecurityContext>
3607       <disco:Options>
3608         <disco:Option>urn:x-test:atm:options:testopt1</disco:Option>
3609         <disco:Option>urn:x-test:atm:options:testopt2</disco:Option>
3610         <disco:Option>urn:x-test:atm:options:testopt3</disco:Option>
3611       </disco:Options>
3612       <disco:Action>urn:x-test:atm:2007-11:Withdraw</disco:Action>
3613       <disco:Action>urn:x-test:atm:2007-11:Transfer</disco:Action>
3614     </wsa:Metadata>
3615   </wsa:EndpointReference>
3616 </disco:QueryResponse>
```

3617 Things to note about this response:

- 3618 • the query was successful (status code is OK).
- 3619 • Only one ID-WSF EPR was returned which contained the endpoint where the "...Withdraw" action is available.
- 3620 • The Discovery service could have left off the "...Transfer" action but that wouldn't have been much of a savings  
3621 and so it was included. Clients should NOT depend upon this type of behavior.
- 3622 • Even though Options were not specified on the request, they are specified in the SvcMD and so are included in the  
3623 ID-WSF EPRs generated from that SvcMD.

### 3.13.33. Query ATM Service w/Option

Query for the ATM service specifying an option

```
3626 <disco:Query xmlns:disco="urn:liberty:disco:2006-08" id="discReq">
3627   <disco:RequestedService resultsType="only-one">
3628     <disco:ServiceType>urn:x-test:atm:2003-03</disco:ServiceType>
3629     <disco:Options>
3630       <disco:Option>urn:x-test:atm:options:testopt1</disco:Option>
3631     </disco:Options>
3632     <disco:SecurityMechID>urn:liberty:security:2006-08:TLS:SAMLV2</disco:SecurityMechID>
3633   </disco:RequestedService>
3634 </disco:Query>
```

Things to note about this query:

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

Server Response:

```
3640 <disco:QueryResponse xmlns:disco="urn:liberty:disco:2006-08">
3641   <lu:Status xmlns:lu="urn:liberty:util:2006-08" code="OK"/>
3642   <wsa:EndpointReference xmlns:wsa="http://www.w3.org/2005/08/addressing"
3643     xmlns:wsu="http://.../oasis-200401-wss-wssecurity-utility-1.0.xsd"
3644     notOnOrAfter="2006-04-06T17:42:19Z" wsu:Id="EPRIDzKyaJ_h5Qb2jLjLjq59E">
3645     <wsa:Address>https://test2.atm.CA.US.testing.com</wsa:Address>
3646     <wsa:Metadata>
3647       <disco:Abstract>TestDisco Test ATM Service</disco:Abstract>
3648       <sbf:Framework xmlns:sbf="urn:liberty:sb" version="2.0"/>
3649       <disco:ProviderID>https://s-wsp.liberty-iop.org:8743/sp.xml</disco:ProviderID>
3650       <disco:ServiceType>urn:x-test:atm:2003-03</disco:ServiceType>
3651       <disco:SecurityContext>
3652         <disco:SecurityMechID>urn:liberty:security:2006-08:TLS:SAMLV2</disco:SecurityMechID>
3653         <sec:Token xmlns:sec="urn:liberty:security:2006-08"
3654           usage="urn:liberty:security:tokenusage:2006-08:SecurityToken">
3655           <sa:Assertion xmlns:sa="urn:oasis:names:tc:SAML:2.0:assertion"
3656             ID="CRED0FAqvGgoeySpLTZHbJa7"
3657             IssueInstant="2006-04-06T15:42:19Z" Version="2.0">
3658             ... assertion data was here ...
3659           </sa:Assertion>
3660         </sec:Token>
3661       </disco:SecurityContext>
3662       <disco:Options>
3663         <disco:Option>urn:x-test:atm:options:testopt1</disco:Option>
3664         <disco:Option>urn:x-test:atm:options:testopt2</disco:Option>
3665         <disco:Option>urn:x-test:atm:options:testopt3</disco:Option>
3666       </disco:Options>
3667       <disco:Action>urn:x-test:atm:2007-11:GetBalance</disco:Action>
3668     </wsa:EndpointReference>
3669   </wsa:EndpointReference>
3670 </disco:QueryResponse>
```

Things to note about this response:

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

- 3676 • Even though only one option was specified on the request, all of the options listed in the SvcMD are included in  
3677 the response.

### 3678 3.13.34. Query ATM Service w/unknown Option

3679 Query for the ATM service with an option that doesn't exist in the SvcMD.

```
3680 <disco:Query xmlns:disco="urn:liberty:disco:2006-08" id="discReq">  
3681   <disco:RequestedService resultsType="all">  
3682     <disco:ServiceType>urn:x-test:atm:2003-03</disco:ServiceType>  
3683     <disco:Options>  
3684       <disco:Option>urn:x-test:atm:options:testopt8</disco:Option>  
3685     </disco:Options>  
3686     <disco:SecurityMechID>urn:liberty:security:2006-08:TLS:SAMLV2</disco:SecurityMechID>  
3687   </disco:RequestedService>  
3688 </disco:Query>
```

3689 Things to note about this query:

- 3690 • The option specified is **not** in the ATM Service SvcMD.

3691 Server Response:

```
3692 <disco:QueryResponse xmlns:disco="urn:liberty:disco:2006-08">  
3693   <lu:Status xmlns:lu="urn:liberty:util:2006-08" code="Failed">  
3694     <lu:Status code="NoResults" />  
3695   </lu:Status>  
3696 </disco:QueryResponse>
```

3697 Things to note about this response:

- 3698 • the query failed (status code is Failed).  
3699 • The sub-status was "NoResults" indicating no data matched the requested parameters

### 3700 3.13.35. Query ATM Service w/good and bad Option

3701 Query for the ATM service with an option that does exist and an option that doesn't exist in the SvcMD.

```
3702 <disco:Query xmlns:disco="urn:liberty:disco:2006-08" id="discReq">  
3703   <disco:RequestedService resultsType="all">  
3704     <disco:ServiceType>urn:x-test:atm:2003-03</disco:ServiceType>  
3705     <disco:Options>  
3706       <disco:Option>urn:x-test:atm:options:testopt2</disco:Option>  
3707       <disco:Option>urn:x-test:atm:options:testopt8</disco:Option>  
3708     </disco:Options>  
3709     <disco:SecurityMechID>urn:liberty:security:2006-08:TLS:SAMLV2</disco:SecurityMechID>  
3710   </disco:RequestedService>  
3711 </disco:Query>
```

3712 Things to note about this query:

- 3713 • both specified options have to be available for this request to be considered matched (if, on the other hand, the  
3714 request had the two <disco:Option> elements in separate <disco:Options> containers, a SvcMD could  
3715 match either one).

3716 **Server Response:**

```
3717 <disco:QueryResponse xmlns:disco="urn:liberty:disco:2006-08">
3718   <lu:Status xmlns:lu="urn:liberty:util:2006-08" code="Failed">
3719     <lu:Status code="NoResults" />
3720   </lu:Status>
3721 </disco:QueryResponse>
```

3722 **Things to note about this response:**

- 3723 • the query failed (status code is Failed).
- 3724 • The sub-status was "NoResults" indicating no data matched the requested parameters

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

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

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

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

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

### 4.1. EPR Generation Rules

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

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

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

For example, with a SAML Assertion, this includes:

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

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

## 4.2. SAML 2.0 Security Tokens

3765

3766 In a SAML 2.0 Assertion, the Discovery Service ID-WSF EPR SHOULD be conveyed as an XML element within the  
3767 <saml2:AttributeStatement> element in a <saml2:Assertion>.

3768 The <saml2:AttributeStatement> SHOULD be constructed according to the following rules:

3769 • The Name attribute of the <saml2:Attribute> element MUST be:

3770 *urn:liberty:disco:2006-08:DiscoveryEPR*

3771 • The NameFormat attribute of the <saml2:Attribute> element MUST be:

3772 *urn:oasis:names:tc:SAML:2.0:attrname-format:uri*

3773 • One or more <saml2:AttributeValue> elements MUST be included which each containing a single  
3774 <wsa:EndpointReference> element identifying a Discovery Service instance(s). These Discovery Ser-  
3775 vice instances SHOULD offer identity services for the Principal identified in the Subject element inside the  
3776 <saml2:Assertion>.

3777 An example <saml2:AttributeStatement> that might be found in a SAMLv2 <saml2:Assertion> follows.  
3778 The example includes a <sec:Token> element which has a reference to the surrounding assertion.

3779

```
3780 <AttributeStatement xmlns="urn:oasis:names:tc:SAML:2.0:assertion">
3781   <Attribute Name="urn:liberty:disco:2006-08:DiscoveryEPR"
3782     NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri">
3783     <AttributeValue>
3784       <wsa:EndpointReference>
3785         <wsa:Address>https://example.com/disco/</wsa:Address>
3786
3787         <wsa:Metadata>
3788           <Abstract>
3789             The Principal's Discovery Service Resource
3790           </Abstract>
3791
3792           <ServiceType>urn:liberty:disco:2006-08</ServiceType>
3793
3794           <ProviderID>http://example.com/</ProviderID>
3795
3796           <SecurityContext>
3797             <SecurityMechID>urn:liberty:security:2005-02:TLS:bearer</SecurityMechID>
3798             <sec:Token ref="..." usage="urn:liberty:security:tokenusage:2006-08:SecurityToken">
3799             </SecurityContext>
3800           </wsa:Metadata>
3801         </wsa:EndpointReference>
3802       </AttributeValue>
3803     </Attribute>
3804   </AttributeStatement>
3805
```

3806 **Example 21. <AttributeStatement> that might be found in a SAMLv2 AuthnResponse**

3807 In all cases, this <AttributeStatement> MUST carry an ID-WSF EPR for the Liberty Discovery Service. Any  
3808 other ID-WSF EPRs are to be discovered by contacting the Discovery Service.

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

3809

3810 In a SAML 1.x Assertion, the Discovery Service ID-WSF EPR SHOULD be conveyed as an XML element within the  
3811 <saml:AttributeStatement> element in a <saml:Assertion>.

3812 The <saml:AttributeStatement> SHOULD be constructed according to the following rules:

- 3813 • For the <saml:Attribute> element:
- 3814 • The AttributeName attribute MUST be "DiscoveryEPR."
- 3815 • The AttributeNamespace attribute MUST be "urn:liberty:disco:2006-08."
- 3816 • The <Subject> element of the <saml:AttributeStatement> element MUST carry the identity of the  
3817 principal whose Discovery Service is referenced by this EPR and SHOULD be the same identity in the subject of  
3818 the other statements in the <saml:Assertion>.
- 3819 • One or more <saml:AttributeValue> elements MUST be included which each containing a single  
3820 <wsa:EndpointReference> element identifying a Discovery Service instance(s). These Discovery Ser-  
3821 vice instances SHOULD offer identity services for the Principal identified in the Subject element inside this  
3822 <saml:AttributeStatment>.

3823 An example <saml:AttributeStatement> that might be found in a SAML 1.1 <saml:Assertion> follows. The  
3824 example includes a <sec:Token> element which has a reference to the surrounding assertion.

```

3825 <AttributeStatement xmlns="urn:oasis:names:tc:SAML:1.0:assertion">
3826   <Subject>
3827     <NameIdentifier Format="urn:liberty:iff:nameid:federated">
3828       d0CQF8elJTDLmzEO
3829     </NameIdentifier>
3830   </Subject>
3831   <Attribute AttributeName="DiscoveryEPR"
3832     AttributeNamespace="urn:liberty:disco:2006-08">
3833     <AttributeValue>
3834       <wsa:EndpointReference>
3835         <wsa:Address>https://example.com/disco/</wsa:Address>
3836       </wsa:EndpointReference>
3837       <wsa:Metadata>
3838         <Abstract>
3839           The Principal's Discovery Service Resource
3840         </Abstract>
3841       </wsa:Metadata>
3842       <ServiceType>urn:liberty:disco:2006-08</ServiceType>
3843       <ProviderID>http://example.com/</ProviderID>
3844       <SecurityContext>
3845         <SecurityMechID>urn:liberty:security:2005-02:TLS:bearer</SecurityMechID>
3846         <sec:Token ref="..." usage="urn:liberty:security:tokenusage:2006-08:SecurityToken">
3847         </sec:Token>
3848       </SecurityContext>
3849     </wsa:Metadata>
3850   </wsa:EndpointReference>
3851 </AttributeValue>
3852 </Attribute>
3853 </AttributeStatement>
3854
3855
3856

```

3857 **Example 22. <AttributeStatement> that might be found in a SAML 1.1 AuthnResponse**

3858 In all cases, this <AttributeStatement> MUST only carry an ID-WSF EPR for the Liberty Discovery Service.  
3859 Any other ID-WSF EPRs are to be discovered by contacting the Discovery Service.

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

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

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

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

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

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

```
<ResourceOffering>
  <ResourceID> 123 </ResourceID>
  <ServiceInstance>
    <ServiceType>urn:liberty:idsis-pp:2003-08</ServiceType>
    <ProviderID>http://pp.services.aol.com</ProviderID>
    <Description CredentialRef="1">
      <SecurityMechID>urn:liberty:security:2006-08:TLS:Bearer</SecurityMechID>
      <Endpoint>https://ep1.pp.service.aol.com</Endpoint>
    </Description>
    <Description>
      <SecurityMechID>urn:liberty:security:2006-08:Client-TLS:Null</SecurityMechID>
      <Endpoint>https://ep1.pp.service.aol.com</Endpoint>
    </Description>
  </ServiceInstance>
</ResourceOffering>
<Credentials>
  <saml:Assertion AssertionID="1" ...>
    Assertion data goes here
  </saml:Assertion>
</Credentials>
```

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

```

3906
3907 <wsa:EndpointReference>
3908   <wsa:Address>https://ep1.pp.services.aol.com</wsa:Address >
3909   <wsa:Metadata>
3910     <ds1:ResourceID>123</ds1:ResourceID>
3911     <ds2:ProviderID>http://pp.services.aol.com</ds2:ProviderID>
3912     <ds2:ServiceType>urn:liberty:idsis-pp:2003-08</ds2:ServiceType>
3913     <ds2:FrameworkVersion="1.1" />
3914     <ds2:SecurityContext>
3915       <ds2:SecurityMechID>urn:liberty:security:2006-08:TLS:Bearer</ds2:SecurityMechID>
3916       <sec:Token usage="urn:liberty:security:tokenusage:2006-08:SecurityToken">
3917         <saml1:Assertion AssertionID="1" ... >
3918           ... assertion data goes here ...
3919         </saml1:Assertion>
3920       </sec:Token>
3921     </ds2:SecurityContext>
3922     <ds2:SecurityContext>
3923       <ds2:SecurityMechID>
3924         urn:liberty:security:2006-08:Client-TLS:Null
3925       </ds2:SecurityMechID>
3926     </ds2:SecurityContext>
3927   </wsa:Metadata>
3928 </wsa:EndpointReference>
3929

```

3930 And subsequently, the invocation of the ID-WSF 1.x service would look to be something along the lines of (assuming  
3931 that the WSC chose to use the "...:TLS:bearer" Security Mechanism):

```

3932
3933 <?xml version="1.0" encoding="utf-8" ?>
3934 <S:Envelope...
3935   <S:Header>
3936     <sb:Correlation S:mustUnderstand="1"
3937       messageID=uuid:958312848-29348938-232342121
3938       timestamp="2003-06-06T18:29:18Z" />
3939     <wsse:Security>
3940       <saml1:Assertion AssertionID="1" ... >
3941         ... assertion data goes here ...
3942       </saml1:Assertion>
3943     </wsse:Security>
3944   </S:Header>
3945   <S:Body>
3946     <pp:Query>
3947       <pp:ResourceID>123</pp:ResourceID>
3948       <pp:QueryItem>
3949         ... query data goes here ...
3950       </pp:QueryItem>
3951     </pp:Query>
3952   </S:Body>
3953 </S:Envelope>
3954

```

---

## 6. Acknowledgments

3955

3956 Many people have made contributions to this specification as it has evolved over time. The original specification was  
3957 written by John Beatty with subsequent versions "inked" by Jonathan Sergent and later, Jeff Hodges and now myself.

3958 The changes made in this latest release of the specification are due in a large part to the work of Jeff Hodges, Robert  
3959 Aarts, John Kemp, Gary Ellison and Greg Whitehead. Many others, including those that are listed as contributors  
3960 on the cover page, have also played a part in this and earlier releases of the specification. Many thanks to all who  
3961 participated (and apologies if I have forgotten to mention your name).

# References

## Normative

3962

3963

3964 [LibertyGlossary] Hodges, Jeff, eds. "Liberty Technical Glossary," Version v2.0, Liberty Alliance Project (30 July,  
3965 2006). <http://www.projectliberty.org/specs>

3966 [LibertySecMech] Hirsch, Frederick, eds. "Liberty ID-WSF Security Mechanisms Core," Version 2.0-errata-v1.0,  
3967 Liberty Alliance Project (21 April, 2007). <http://www.projectliberty.org/specs>

3968 [LibertySecMech11] Ellison, Gary, eds. "Liberty ID-WSF Security Mechanisms," Version 1.1, Liberty Alliance  
3969 Project (18 April 2004). <http://www.projectliberty.org/specs/>

3970 [LibertySecMech20SAML] Hirsch, Frederick, eds. "ID-WSF 2.0 SecMech SAML Profile," Version 2.0-errata-v1.0,  
3971 Liberty Alliance Project (08 November, 2006). <http://www.projectliberty.org/specs>

3972 [LibertyAuthn] Hodges, Jeff, Aarts, Robert, Madsen, Paul, Cantor, Scott, eds. "Liberty ID-WSF Authentication,  
3973 Single Sign-On, and Identity Mapping Services Specification," Version 2.0-errata-v1.0, Liberty Alliance  
3974 Project (28 November, 2006). <http://www.projectliberty.org/specs>

3975 [LibertySOAPBinding] Hodges, Jeff, Kemp, John, Aarts, Robert, Whitehead, Greg, Madsen, Paul, eds. "Liberty  
3976 ID-WSF SOAP Binding Specification," Version 2.0-errata-v1.0, Liberty Alliance Project (21 April, 2007).  
3977 <http://www.projectliberty.org/specs>

3978 [LibertyIDWSF20SCR] Whitehead, Greg, eds. Version 1.0 errata v1.0, Liberty Alliance Project (21 April, 2007).  
3979 <http://www.projectliberty.org/specs>

3980 [LibertyIDWSFv20Errata] Champagne, Darryl, Lockhart, Rob, Tiffany, Eric, eds. "Liberty ID-WSF 2.0 Errata,"  
3981 Version 1.0, Liberty Alliance Project (13 April, 2007). <http://www.projectliberty.org/specs>

3982 [RFC2119] S. Bradner "Key words for use in RFCs to Indicate Requirement Levels," RFC 2119, The Internet  
3983 Engineering Task Force (March 1997). <http://www.ietf.org/rfc/rfc2119.txt>

3984 [RFC3986] Berners-Lee, T., Fielding, R., Masinter, L., eds. (January 2005). "Uniform Resource Identifier  
3985 (URI): Generic Syntax," RFC 3986 (Obsoletes RFC2732, RFC2396, RFC1808) (Updates RFC1738) (Also  
3986 STD0066) (Status: STANDARD), The Internet Engineering Task Force <http://www.ietf.org/rfc/rfc3986.txt>

3987 [SAMLCore2] Cantor, Scott, Kemp, John, Philpott, Rob, Maler, Eve, eds. (15 March 2005). "Assertions  
3988 and Protocol for the OASIS Security Assertion Markup Language (SAML) V2.0," SAML V2.0, OA-  
3989 SIS Standard, Organization for the Advancement of Structured Information Standards [http://docs.oasis-  
3990 open.org/security/saml/v2.0/saml-core-2.0-os.pdf](http://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf)

3991 [SAMLMeta2] Cantor, Scott, Moreh, Jahan, Philpott, Rob, Maler, Eve, eds. (15 March 2005). "Metadata for the OA-  
3992 SIS Security Assertion Markup Language (SAML) V2.0," SAML V2.0, OASIS Standard, Organization for  
3993 the Advancement of Structured Information Standards [http://docs.oasis-open.org/security/saml/v2.0/saml-  
3994 metadata-2.0-os.pdf](http://docs.oasis-open.org/security/saml/v2.0/saml-metadata-2.0-os.pdf)

3995 [Schema1-2] Thompson, Henry S., Beech, David, Maloney, Murray, Mendelsohn, Noah, eds. (28 October  
3996 2004). "XML Schema Part 1: Structures Second Edition," Recommendation, World Wide Web Consortium  
3997 <http://www.w3.org/TR/xmlschema-1/>

3998 [WSAv1.0] "Web Services Addressing (WS-Addressing) 1.0," Gudgin, Martin, Hadley, Marc, Rogers, Tony, eds.  
3999 World Wide Web Consortium W3C Recommendation (9 May 2006). [http://www.w3.org/TR/2006/REC-ws-  
4000 addr-core-20060509/](http://www.w3.org/TR/2006/REC-ws-addr-core-20060509/)

- 4001 [WSAv1.0-SOAP] "WS-Addressing 1.0 SOAP Binding," Gudgin, Martin, Hadley, Marc, eds. World Wide Web Con-  
4002 sultium W3C Recommendation (9 May 2006). <http://www.w3.org/TR/2006/REC-ws-addr-soap-20060509/>
- 4003 [WSDLv1.1] "Web Services Description Language (WSDL) 1.1," Christensen, Erik, Curbera, Francisco, Mered-  
4004 ith, Greg, Weerawarana, Sanjiva, eds. World Wide Web Consortium W3C Note (15 March 2001).  
4005 <http://www.w3.org/TR/2001/NOTE-wsdl-20010315>
- 4006 [wss-sms] Hallam-Baker, Phillip, Kaler, Chris, Monzillo, Ronald, Nadalin, Anthony, eds. (January, 2004). "Web  
4007 Services Security: SOAP Message Security," OASIS Standard V1.0 [OASIS 200401], Organization for  
4008 the Advancement of Structured Information Standards [http://docs.oasis-open.org/wss/2004/01/oasis-200401-  
wss-soap-message-security-1.0.pdf](http://docs.oasis-open.org/wss/2004/01/oasis-200401-<br/>4009 wss-soap-message-security-1.0.pdf)
- 4010 [wss-sms-draft] Hallam-Baker, Phillip, Kaler, Chris, Monzillo, Ronald, Nadalin, Anthony, eds. (June 30,  
4011 2003). "Web Services Security: SOAP Message Security," Draft WSS-SOAPMessageSecurity-14-  
4012 06-2003, Organization for the Advancement of Structured Information Standards [http://www.oasis-  
open.org/committees/download.php/2757/WSS-SOAPMessageSecurity-14-063003-merged.pdf](http://www.oasis-<br/>4013 open.org/committees/download.php/2757/WSS-SOAPMessageSecurity-14-063003-merged.pdf)
- 4014 [xmlenc-core] Eastlake, Donald, Reagle, Joseph, eds. (10 December 2002). "XML Encryption Syntax and Process-  
4015 ing," W3C Recommendation, World Wide Web Consortium <http://www.w3.org/TR/xmlenc-core/>

## 4016 Informative

- 4017 [LibertyPAOS] Aarts, Robert, Kemp, John, eds. "Liberty Reverse HTTP Binding for SOAP Specification," Version  
4018 2.0, Liberty Alliance Project (30 July, 2006). <http://www.projectliberty.org/specs>
- 4019 [LibertyClientProfiles] Aarts, Robert, Kainulainen, Jukka, Kemp, John, eds. "Liberty ID-WSF Profiles for Liberty-  
4020 Enabled User Agents and Devices," Version 2.0-errata-v1.0, Liberty Alliance Project (22 January, 2007).  
4021 <http://www.projectliberty.org/specs>

## A. Discovery Service Version 2.0 XSD

4022

4023

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace="urn:liberty:disco:2006-08"
  xmlns:md="urn:oasis:names:tc:SAML:2.0:metadata"
  xmlns:sb="urn:liberty:sb:2006-08"
  xmlns:sbf="urn:liberty:sb"
  xmlns:sec="urn:liberty:security:2006-08"
  xmlns:lu="urn:liberty:util:2006-08"
  xmlns:wsa="http://www.w3.org/2005/08/addressing"
  xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd"
  xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
  xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns="urn:liberty:disco:2006-08"
  elementFormDefault="qualified"
  attributeFormDefault="unqualified"
>
  <xs:import namespace="urn:liberty:util:2006-08"
    schemaLocation="liberty-idwsf-utility-v2.0.xsd"/>
  <xs:import namespace="urn:liberty:sb:2006-08"
    schemaLocation="liberty-idwsf-soap-binding-v2.0.xsd"/>
  <xs:import namespace="urn:liberty:sb"
    schemaLocation="liberty-idwsf-soap-binding.xsd"/>
  <xs:import namespace="http://www.w3.org/2005/08/addressing"
    schemaLocation="ws-addr-1.0.xsd"/>
  <xs:import namespace="urn:oasis:names:tc:SAML:2.0:metadata"
    schemaLocation="saml-schema-metadata-2.0.xsd"/>
  <xs:import namespace="urn:liberty:security:2006-08"
    schemaLocation="liberty-idwsf-security-mechanisms-v2.0.xsd"/>
  <xs:import
    namespace="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd"
    schemaLocation="wss-secext-1.0.xsd"/>
  <xs:import
    namespace="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
    schemaLocation="wss-util-1.0.xsd"/>
  <!-- **** Discovery Service Data Elements & Types **** -->
  <!-- The data elements and types in this section are used to
    embellish WS-Addressing Endpoint References (EPRs).
    They are placed in the /wsa:EndpointReference/Metadata
    element. Specific usage and cardinalities are stipulated
    in the Discovery Service v2.0 Specification. -->
  <!-- Abstract: natural-language description of service -->
  <xs:element name="Abstract" type="xs:string"/>
  <!-- Provider ID -->
  <xs:element name="ProviderID" type="xs:anyURI"/>
  <!-- Service Type -->
  <xs:element name="ServiceType" type="xs:anyURI"/>
```

```

4088
4089 <!-- Framework Description -->
4090
4091 <xs:element name="Framework" type="sbf:FrameworkType" />
4092
4093 <!-- EPR Expiration Timestamp -->
4094
4095 <xs:attribute name="notOnOrAfter" type="xs:dateTime" />
4096
4097 <!-- Security Context Container -->
4098
4099 <xs:element name="SecurityContext">
4100   <xs:complexType>
4101     <xs:sequence>
4102       <xs:element ref="SecurityMechID"
4103         minOccurs="1"
4104         maxOccurs="unbounded" />
4105
4106       <xs:element ref="sec:Token"
4107         minOccurs="0"
4108         maxOccurs="unbounded" />
4109     </xs:sequence>
4110   </xs:complexType>
4111 </xs:element>
4112
4113 <!-- Security Mechanism ID -->
4114
4115 <xs:element name="SecurityMechID" type="xs:anyURI" />
4116
4117 <!-- Options -->
4118
4119 <xs:element name="Options" type="OptionsType" />
4120
4121 <xs:element name="Option" type="xs:anyURI" />
4122
4123 <xs:complexType name="OptionsType">
4124   <xs:sequence>
4125     <xs:element ref="Option" minOccurs="0" maxOccurs="unbounded" />
4126   </xs:sequence>
4127 </xs:complexType>
4128
4129 <!-- Address -->
4130
4131 <xs:element name="Address" type="xs:anyURI" />
4132
4133 <!-- Action(s) - the interfaces available at this service -->
4134
4135 <xs:element name="Action" type="xs:anyURI" />
4136 <!-- Keys Element - For use in ModifyResponse -->
4137
4138 <xs:element name="Keys" type="KeysType" />
4139
4140 <xs:complexType name="KeysType">
4141   <xs:sequence>
4142     <xs:element ref="md:KeyDescriptor"
4143       minOccurs="1"
4144       maxOccurs="unbounded" />
4145   </xs:sequence>
4146 </xs:complexType>
4147
4148 <!-- Service Metadata (SvcMD) - metadata about service instance -->
4149
4150 <xs:element name="SvcMD" type="SvcMetadataType" />
4151 <xs:complexType name="SvcMetadataType">
4152   <xs:sequence>
4153     <xs:element ref="Abstract" />
4154     <xs:element ref="ProviderID" />

```

```

4155     <xs:element ref="ServiceContext" maxOccurs="unbounded" />
4156 </xs:sequence>
4157     <xs:attribute name="svcMDID" type="xs:string" use="optional" />
4158 </xs:complexType>
4159
4160 <!-- ServiceContext - describes service type/option/endpoint context -->
4161 <xs:element name="ServiceContext" type="ServiceContextType" />
4162 <xs:complexType name="ServiceContextType">
4163     <xs:sequence>
4164         <xs:element ref="ServiceType" maxOccurs="unbounded" />
4165         <xs:element ref="Options" minOccurs="0"
4166             maxOccurs="unbounded" />
4167         <xs:element ref="EndpointContext" maxOccurs="unbounded" />
4168     </xs:sequence>
4169 </xs:complexType>
4170
4171 <!-- EndpointContext - describes endpoints used to access service -->
4172 <xs:element name="EndpointContext" type="EndpointContextType" />
4173 <xs:complexType name="EndpointContextType">
4174     <xs:sequence>
4175         <xs:element ref="Address" maxOccurs="unbounded" />
4176         <xs:element ref="sbf:Framework" maxOccurs="unbounded" />
4177         <xs:element ref="SecurityMechID" maxOccurs="unbounded" />
4178         <xs:element ref="Action" minOccurs="0"
4179             maxOccurs="unbounded" />
4180     </xs:sequence>
4181 </xs:complexType>
4182
4183 <!-- SvcMD ID element used to refer to Service Metadata elements -->
4184 <xs:element name="SvcMDID" type="xs:string" />
4185
4186 <!-- **** Discovery Service Protocol Messages Elements & Types **** -->
4187
4188 <!-- Query Message Element & Type -->
4189
4190 <xs:element name="Query" type="QueryType" />
4191
4192 <xs:complexType name="QueryType">
4193     <xs:sequence>
4194         <xs:element name="RequestedService"
4195             type="RequestedServiceType"
4196             minOccurs="0"
4197             maxOccurs="unbounded" />
4198     </xs:sequence>
4199
4200     <xs:anyAttribute namespace="##other" processContents="lax" />
4201 </xs:complexType>
4202
4203 <xs:complexType name="RequestedServiceType">
4204     <xs:sequence>
4205         <xs:element ref="ServiceType" minOccurs="0" maxOccurs="unbounded" />
4206
4207         <xs:element ref="ProviderID" minOccurs="0" maxOccurs="unbounded" />
4208
4209         <xs:element ref="Options" minOccurs="0" maxOccurs="unbounded" />
4210
4211         <xs:element ref="SecurityMechID" minOccurs="0" maxOccurs="unbounded" />
4212
4213         <xs:element ref="Framework" minOccurs="0" maxOccurs="unbounded" />
4214
4215         <xs:element ref="Action" minOccurs="0" maxOccurs="unbounded" />
4216
4217         <xs:any namespace="##other"
4218             processContents="lax"
4219             minOccurs="0"
4220             maxOccurs="unbounded" />
4221

```

```

4222     </xs:sequence>
4223
4224     <xs:attribute name="reqID" type="xs:string" use="optional" />
4225     <xs:attribute name="resultsType" type="xs:string" use="optional" />
4226
4227 </xs:complexType>
4228
4229 <!-- QueryResponse Message Element & Type -->
4230
4231 <xs:element name="QueryResponse" type="QueryResponseType"/>
4232
4233 <xs:complexType name="QueryResponseType">
4234     <xs:sequence>
4235         <xs:element ref="lu:Status"/>
4236
4237         <xs:element ref="wsa:EndpointReference"
4238             minOccurs="0"
4239             maxOccurs="unbounded"/>
4240     </xs:sequence>
4241     <xs:anyAttribute namespace="##other" processContents="lax"/>
4242 </xs:complexType>
4243
4244
4245 <!-- -->
4246 <!-- DS Interfaces for SvcMD Associations -->
4247 <!-- -->
4248 <!-- These interfaces support the adding, deleting,-->
4249 <!-- querying SvcMD Associations for a principal. -->
4250 <!-- -->
4251
4252 <!-- SvcMDAssociationAdd operation -->
4253
4254 <xs:element name="SvcMDAssociationAdd" type="SvcMDAssociationAddType"/>
4255
4256 <xs:complexType name="SvcMDAssociationAddType">
4257     <xs:sequence>
4258         <xs:element ref="SvcMDID" maxOccurs="unbounded" />
4259     </xs:sequence>
4260     <xs:anyAttribute namespace="##other" processContents="lax"/>
4261 </xs:complexType>
4262
4263 <!-- Response for SvcMDAssociationAdd operation -->
4264
4265 <xs:element name="SvcMDAssociationAddResponse"
4266     type="SvcMDAssociationAddResponseType"/>
4267
4268 <xs:complexType name="SvcMDAssociationAddResponseType">
4269     <xs:sequence>
4270         <xs:element ref="lu:Status" />
4271     </xs:sequence>
4272     <xs:anyAttribute namespace="##other" processContents="lax"/>
4273 </xs:complexType>
4274 <!-- SvcMDAssociationDelete operation -->
4275
4276 <xs:element name="SvcMDAssociationDelete" type="SvcMDAssociationDeleteType"/>
4277
4278 <xs:complexType name="SvcMDAssociationDeleteType">
4279     <xs:sequence>
4280         <xs:element ref="SvcMDID" maxOccurs="unbounded" />
4281     </xs:sequence>
4282     <xs:anyAttribute namespace="##other" processContents="lax"/>
4283 </xs:complexType>
4284 <!-- Response for SvcMDAssociationDelete operation -->
4285
4286 <xs:element name="SvcMDAssociationDeleteResponse"
4287     type="SvcMDAssociationDeleteResponseType"/>
4288

```

```

4289 <xs:complexType name="SvcMDAssociationDeleteResponseType">
4290 <xs:sequence>
4291 <xs:element ref="lu:Status" />
4292 </xs:sequence>
4293 <xs:anyAttribute namespace="##other" processContents="lax"/>
4294 </xs:complexType>
4295 <!-- SvcMDAssociationQuery operation -->
4296
4297 <xs:element name="SvcMDAssociationQuery" type="SvcMDAssociationQueryType"/>
4298
4299 <xs:complexType name="SvcMDAssociationQueryType">
4300 <xs:sequence>
4301 <xs:element ref="SvcMDID" minOccurs="0" maxOccurs="unbounded" />
4302 </xs:sequence>
4303 <xs:anyAttribute namespace="##other" processContents="lax"/>
4304 </xs:complexType>
4305 <!-- Response for SvcMDAssociationQuery operation -->
4306
4307 <xs:element name="SvcMDAssociationQueryResponse"
4308 <!-- type="SvcMDAssociationQueryResponseType"/>
4309
4310 <xs:complexType name="SvcMDAssociationQueryResponseType">
4311 <xs:sequence>
4312 <xs:element ref="lu:Status" />
4313 <xs:element ref="SvcMDID" minOccurs="0" maxOccurs="unbounded" />
4314 </xs:sequence>
4315 <xs:anyAttribute namespace="##other" processContents="lax"/>
4316 </xs:complexType>
4317
4318 <!-- -->
4319 <!-- DS Interfaces for Service Metadata Management -->
4320 <!-- -->
4321 <!-- These interfaces document a create, replace, -->
4322 <!-- delete, and query interface for the service -->
4323 <!-- metadata which is later associated with a -->
4324 <!-- principal. -->
4325 <!-- -->
4326
4327 <!-- Register operation for Service Metadata -->
4328
4329 <xs:element name="SvcMDRegister" type="SvcMDRegisterType"/>
4330
4331 <xs:complexType name="SvcMDRegisterType">
4332 <xs:sequence>
4333 <xs:element ref="SvcMD" maxOccurs="unbounded" />
4334 </xs:sequence>
4335 <xs:anyAttribute namespace="##other" processContents="lax"/>
4336 </xs:complexType>
4337
4338 <!-- Response for SvcMDRegister operation -->
4339
4340 <xs:element name="SvcMDRegisterResponse"
4341 <!-- type="SvcMDRegisterResponseType"/>
4342
4343 <xs:complexType name="SvcMDRegisterResponseType">
4344 <xs:sequence>
4345
4346 <xs:element ref="lu:Status" />
4347 <xs:element ref="SvcMDID" minOccurs="0" maxOccurs="unbounded" />
4348 <xs:element ref="Keys" minOccurs="0" maxOccurs="unbounded" />
4349
4350 </xs:sequence>
4351 <xs:anyAttribute namespace="##other" processContents="lax"/>
4352 </xs:complexType>
4353
4354 <!-- Delete operation on Service Metadata -->
4355

```

```

4356 <xs:element name="SvcMDDelete" type="SvcMDDeleteType"/>
4357
4358 <xs:complexType name="SvcMDDeleteType">
4359   <xs:sequence>
4360     <xs:element ref="SvcMDID" maxOccurs="unbounded" />
4361   </xs:sequence>
4362   <xs:anyAttribute namespace="##other" processContents="lax"/>
4363 </xs:complexType>
4364
4365 <!-- Response for delete operation on Service Metadata -->
4366
4367 <xs:element name="SvcMDDeleteResponse" type="SvcMDDeleteResponseType"/>
4368
4369 <xs:complexType name="SvcMDDeleteResponseType">
4370   <xs:sequence>
4371     <xs:element ref="lu:Status" />
4372   </xs:sequence>
4373   <xs:anyAttribute namespace="##other" processContents="lax"/>
4374 </xs:complexType>
4375
4376 <!-- Query operation on Service Metadata -->
4377
4378 <xs:element name="SvcMDQuery" type="SvcMDQueryType"/>
4379
4380 <xs:complexType name="SvcMDQueryType">
4381   <xs:sequence>
4382     <xs:element ref="SvcMDID"
4383       minOccurs="0"
4384       maxOccurs="unbounded"/>
4385   </xs:sequence>
4386   <xs:anyAttribute namespace="##other" processContents="lax"/>
4387 </xs:complexType>
4388
4389 <!-- Response for Query operation on Service Metadata -->
4390
4391 <xs:element name="SvcMDQueryResponse" type="SvcMDQueryResponseType"/>
4392
4393 <xs:complexType name="SvcMDQueryResponseType">
4394   <xs:sequence>
4395     <xs:element ref="lu:Status" />
4396     <xs:element ref="SvcMD" minOccurs="0" maxOccurs="unbounded" />
4397   </xs:sequence>
4398   <xs:anyAttribute namespace="##other" processContents="lax"/>
4399 </xs:complexType>
4400
4401 <!-- Replace operation on Service Metadata -->
4402
4403 <xs:element name="SvcMDReplace" type="SvcMDReplaceType"/>
4404
4405 <xs:complexType name="SvcMDReplaceType">
4406   <xs:sequence>
4407     <xs:element ref="SvcMD" maxOccurs="unbounded" />
4408   </xs:sequence>
4409   <xs:anyAttribute namespace="##other" processContents="lax"/>
4410 </xs:complexType>
4411
4412 <!-- Response for SvcMDReplace operation -->
4413
4414 <xs:element name="SvcMDReplaceResponse" type="SvcMDReplaceResponseType"/>
4415
4416 <xs:complexType name="SvcMDReplaceResponseType">
4417   <xs:sequence>
4418     <xs:element ref="lu:Status" />
4419   </xs:sequence>
4420   <xs:anyAttribute namespace="##other" processContents="lax"/>
4421 </xs:complexType>
4422

```

4423 </xs: schema>  
4424  
4425

## B. Discovery Service WSDL

4426

```
4427 <?xml version="1.0"?>
4428 <definitions name="disco-svc"
4429   targetNamespace="urn:liberty:disco:2006-08"
4430   xmlns:tns="urn:liberty:disco:2006-08"
4431   xmlns="http://schemas.xmlsoap.org/wsdl/"
4432   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
4433   xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
4434   xmlns:sb="urn:liberty:sb:2006-08"
4435   xmlns:wsaw="http://www.w3.org/2006/02/addressing/wsdl"
4436   xmlns:disco="urn:liberty:disco:2006-08"
4437   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4438   xsi:schemaLocation="http://schemas.xmlsoap.org/wsdl/
4439     http://schemas.xmlsoap.org/wsdl/
4440     http://www.w3.org/2006/02/addressing/wsdl
4441     http://www.w3.org/2006/02/addressing/wsdl/ws-addr-wsdl.xsd">
4442   <types>
4443     <xsd:schema>
4444       <xsd:import namespace="urn:liberty:disco:2006-08"
4445         schemaLocation="liberty-idwsf-disco-svc-v2.0.xsd"/>
4446     </xsd:schema>
4447   </types>
4448   <message name="Query">
4449     <part name="body" element="disco:Query"/>
4450   </message>
4451   <message name="QueryResponse">
4452     <part name="body" element="disco:QueryResponse"/>
4453   </message>
4454   <message name="SvcMDAssociationAdd">
4455     <part name="body" element="disco:SvcMDAssociationAdd"/>
4456   </message>
4457   <message name="SvcMDAssociationAddResponse">
4458     <part name="body" element="disco:SvcMDAssociationAddResponse"/>
4459   </message>
4460   <message name="SvcMDAssociationQuery">
4461     <part name="body" element="disco:SvcMDAssociationQuery"/>
4462   </message>
4463   <message name="SvcMDAssociationQueryResponse">
4464     <part name="body" element="disco:SvcMDAssociationQueryResponse"/>
4465   </message>
4466   <message name="SvcMDAssociationDelete">
4467     <part name="body" element="disco:SvcMDAssociationDelete"/>
4468   </message>
4469   <message name="SvcMDAssociationDeleteResponse">
4470     <part name="body" element="disco:SvcMDAssociationDeleteResponse"/>
4471   </message>
4472   <message name="SvcMDRegister">
4473     <part name="body" element="disco:SvcMDRegister"/>
4474   </message>
4475   <message name="SvcMDRegisterResponse">
4476     <part name="body" element="disco:SvcMDRegisterResponse"/>
4477   </message>
4478   <message name="SvcMDQuery">
4479     <part name="body" element="disco:SvcMDQuery"/>
4480   </message>
4481   <message name="SvcMDQueryResponse">
4482     <part name="body" element="disco:SvcMDQueryResponse"/>
4483   </message>
4484   <message name="SvcMDQueryResponse">
4485     <part name="body" element="disco:SvcMDQueryResponse"/>
4486   </message>
```

```
4492
4493 <message name="SvcMDReplace">
4494   <part name="body" element="disco:SvcMDReplace" />
4495 </message>
4496 <message name="SvcMDReplaceResponse">
4497   <part name="body" element="disco:SvcMDReplaceResponse" />
4498 </message>
4499
4500 <message name="SvcMDDelete">
4501   <part name="body" element="disco:SvcMDDelete" />
4502 </message>
4503 <message name="SvcMDDeleteResponse">
4504   <part name="body" element="disco:SvcMDDeleteResponse" />
4505 </message>
4506
4507
4508 <portType name="DiscoveryPort">
4509
4510   <operation name="DiscoveryQuery">
4511     <input message="tns:Query"
4512       wsaw:Action="urn:liberty:disco:2006-08:Query" />
4513     <output message="tns:QueryResponse"
4514       wsaw:Action="urn:liberty:disco:2006-08:QueryResponse" />
4515   </operation>
4516
4517   <operation name="MDAssociationAdd">
4518     <input message="tns:SvcMDAssociationAdd"
4519       wsaw:Action="urn:liberty:disco:2006-08:SvcMDAssociationAdd" />
4520     <output message="tns:SvcMDAssociationAddResponse"
4521       wsaw:Action="urn:liberty:disco:2006-08:SvcMDAssociationAddResponse" />
4522   </operation>
4523
4524   <operation name="MDAssociationQuery">
4525     <input message="tns:SvcMDAssociationQuery"
4526       wsaw:Action="urn:liberty:disco:2006-08:SvcMDAssociationQuery" />
4527     <output message="tns:SvcMDAssociationQueryResponse"
4528       wsaw:Action="urn:liberty:disco:2006-08:SvcMDAssociationQueryResponse" />
4529   </operation>
4530
4531   <operation name="MDAssociationDelete">
4532     <input message="tns:SvcMDAssociationDelete"
4533       wsaw:Action="urn:liberty:disco:2006-08:SvcMDAssociationDelete" />
4534     <output message="tns:SvcMDAssociationDeleteResponse"
4535       wsaw:Action="urn:liberty:disco:2006-08:SvcMDAssociationDeleteResponse" />
4536   </operation>
4537
4538   <operation name="MetadataRegister">
4539     <input message="tns:SvcMDRegister"
4540       wsaw:Action="urn:liberty:disco:2006-08:SvcMDRegister" />
4541     <output message="tns:SvcMDRegisterResponse"
4542       wsaw:Action="urn:liberty:disco:2006-08:SvcMDRegisterResponse" />
4543   </operation>
4544
4545   <operation name="MetadataQuery">
4546     <input message="tns:SvcMDQuery"
4547       wsaw:Action="urn:liberty:disco:2006-08:SvcMDQuery" />
4548     <output message="tns:SvcMDQueryResponse"
4549       wsaw:Action="urn:liberty:disco:2006-08:SvcMDQueryResponse" />
4550   </operation>
4551
4552   <operation name="MetadataReplace">
4553     <input message="tns:SvcMDReplace"
4554       wsaw:Action="urn:liberty:disco:2006-08:SvcMDReplace" />
4555     <output message="tns:SvcMDReplaceResponse"
4556       wsaw:Action="urn:liberty:disco:2006-08:SvcMDReplaceResponse" />
4557   </operation>
4558
```

```

4559     <operation name="MetadataDelete">
4560         <input message="tns:SvcMDDelete"
4561             wsaw:Action="urn:liberty:disco:2006-08:SvcMDDelete" />
4562         <output message="tns:SvcMDDeleteResponse"
4563             wsaw:Action="urn:liberty:disco:2006-08:SvcMDDeleteResponse" />
4564     </operation>
4565
4566 </portType>
4567
4568 <!--
4569 An example of a binding and service that can be used with this
4570 abstract service description is provided below.
4571 -->
4572 -->
4573
4574 <binding name="DiscoveryBinding" type="tns:DiscoveryPort">
4575
4576     <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
4577
4578     <operation name="DiscoveryQuery">
4579         <input> <soap:body use="literal"/> </input>
4580         <output> <soap:body use="literal"/> </output>
4581     </operation>
4582
4583     <operation name="MDAssociationAdd">
4584         <input> <soap:body use="literal"/> </input>
4585         <output> <soap:body use="literal"/> </output>
4586     </operation>
4587
4588
4589     <operation name="MDAssociationQuery">
4590         <input> <soap:body use="literal"/> </input>
4591         <output> <soap:body use="literal"/> </output>
4592     </operation>
4593
4594     <operation name="MDAssociationDelete">
4595         <input> <soap:body use="literal"/> </input>
4596         <output> <soap:body use="literal"/> </output>
4597     </operation>
4598
4599     <operation name="MetadataRegister">
4600         <input> <soap:body use="literal"/> </input>
4601         <output> <soap:body use="literal"/> </output>
4602     </operation>
4603
4604     <operation name="MetadataQuery">
4605         <input> <soap:body use="literal"/> </input>
4606         <output> <soap:body use="literal"/> </output>
4607     </operation>
4608
4609     <operation name="MetadataReplace">
4610         <input> <soap:body use="literal"/> </input>
4611         <output> <soap:body use="literal"/> </output>
4612     </operation>
4613
4614     <operation name="MetadataDelete">
4615         <input> <soap:body use="literal"/> </input>
4616         <output> <soap:body use="literal"/> </output>
4617     </operation>
4618
4619 </binding>
4620
4621 <service name="DiscoveryService">
4622
4623     <port name="DiscoveryPort" binding="tns:DiscoveryBinding">
4624
4625

```

```
4626     <!-- Modify with the REAL SOAP endpoint -->
4627
4628     <soap:address location="http://example.com/discovery" />
4629
4630 </port>
4631
4632 </service>
4633
4634 </definitions>
4635
4636
```