



Liberty ID-WSF Provisioned Module Manager Service Specification

Version: 1.0

Editors:

Conor P. Cahill, Intel Corporation

Contributors:

George Fletcher, AOL LLC

Hubert Le Van Gong, Sun

Ben Lin, Intel Corporation

Paul Madsen, NTT

Hiroyoshi Takiguchi, NTT

Greg Whitehead, Hewlett-Packard

Janice Yang, , Intel Corporation

Abstract:

This specification defines the interfaces for the Provisioning Module Manager (PMM).

Filename: liberty-idwsf-pmm-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 Adobe Systems; Agencia Catalana De Certificacio; America Online, Inc.; American Express
16 Company; Amsoft Systems Pvt Ltd.; Avatier Corporation; BIPAC; BMC Software, Inc.; Bank of America
17 Corporation; Beta Systems Software AG; British Telecommunications plc; Computer Associates International, Inc.;
18 Credentica; Dan Combs; Danish National IT and Telecom Agency; DataPower Technology, Inc.; Deutsche Telekom
19 AG, T-Com; Diamelle Technologies, Inc.; Diversinet Corp.; Drummond Group Inc.; Enosis Group LLC; Entrust,
20 Inc.; Epok, Inc.; Ericsson; Falkin Systems LLC; Fidelity Investments; Forum Systems, Inc.; France Télécom; French
21 Government Agence pour le développement de l'administration électronique (ADAE); Fugen Solutions, Inc; Fulvens
22 Ltd.; GSA Office of Governmentwide Policy; Gamefederation; Gemalto; General Motors; GeoFederation; Giesecke
23 & Devrient GmbH; Guy Huntington; Hewlett-Packard Company; Hochhauser & Co., LLC; IBM Corporation; Intel
24 Corporation; Intuit Inc.; Kantega; Kayak Interactive; Livo Technologies; Luminance Consulting Services; Mark
25 Wahl; Mary Ruddy; MasterCard International; MedCommons Inc.; Mobile Telephone Networks (Pty) Ltd; Mortgage
26 Bankers Association (MBA); NEC Corporation; NTT DoCoMo, Inc.; Netegrity, Inc.; NHK (Japan Broadcasting
27 Corporation) Science & Technical Research Laboratories; Neustar, Inc.; New Zealand Government State Services
28 Commission; Nippon Telegraph and Telephone Corporation; Nokia Corporation; Novell, Inc.; OpenNetwork; Oracle
29 Corporation; Ping Identity Corporation; Postsecondary Electronic Standards Council (PESC); RSA Security Inc.;
30 Reach; Reactivity Inc.; Rob Marano; Royal Mail Group plc; SAP AG; SanDisk Corporation; Senforce; Sharp
31 Laboratories of America; Sigaba; SmartTrust; Sony Corporation; Sun Microsystems, Inc.; Supremacy Financial
32 Corporation; Symlabs, Inc.; Telecom Italia S.p.A.; Telefónica Móviles, S.A.; Telenor R&D; Thales e-Security;
33 Trusted Network Technologies; UNINETT AS; UTI; VeriSign, Inc.; Vodafone Group Plc.; Wave Systems Corp.
34 Wells Fargo; All rights reserved.

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

41 Contents

42	1. Introduction	4
43	1.1. Notation and Conventions	4
44	1.1.1. XML Namespaces	4
45	2. Overview	5
46	2.1. Provisioning Components	5
47	3. Provisioned Module Manager Service (PMM)	7
48	3.1. Service URIs	7
49	3.2. Status Codes	7
50	3.3. Data Definitions	8
51	3.4. Request and Response Abstract Types	8
52	3.4.1. Complex Type RequestAbstractType	8
53	3.4.2. Complex Type ResponseAbstractType	8
54	3.5. Operation: <i>Provision</i>	8
55	3.5.1. wsa:Action values for Provision Messages	9
56	3.5.2. Provision Message	9
57	3.5.3. ProvisionResponse Message	10
58	3.5.4. Provision Processing Rules	11
59	3.6. Operation: <i>PMActivate</i>	12
60	3.6.1. wsa:Action values for PMActivate Messages	12
61	3.6.2. PMActivate Message	12
62	3.6.3. PMActivateResponse Message	13
63	3.6.4. PMActivate Processing Rules	14
64	3.7. Operation: <i>PMDeactivate</i>	15
65	3.7.1. wsa:Action values for PMDeactivate Messages	15
66	3.7.2. PMDeactivate Message	15
67	3.7.3. PMDeactivateResponse Message	16
68	3.7.4. PMDeactivate Processing Rules	17
69	3.8. Operation: <i>PMDelete</i>	18
70	3.8.1. wsa:Action values for PMDelete Messages	18
71	3.8.2. PMDelete Message	18
72	3.8.3. PMDeleteResponse Message	19
73	3.8.4. PMDelete Processing Rules	20
74	3.9. Operation: <i>PMUpdate</i>	20
75	3.9.1. wsa:Action values for PMUpdate Messages	20
76	3.9.2. PMUpdate Message	20
77	3.9.3. PMUpdateResponse Message	22
78	3.9.4. PMUpdate Processing Rules	23
79	3.10. Operation: <i>PMGetStatus</i>	24
80	3.10.1. wsa:Action values for PMGetStatus Messages	24
81	3.10.2. PMGetStatus Message	24
82	3.10.3. PMGetStatusResponse Message	25
83	3.10.4. PMGetStatus Processing Rules	26
84	3.11. Operation: <i>PMSetStatus</i>	26
85	3.11.1. wsa:Action values for PMSetStatus Messages	26
86	3.11.2. PMSetStatus Message	27
87	3.11.3. PMSetStatusResponse Message	27
88	3.11.4. PMSetStatus Processing Rules	28
89	4. Provisioned Module Manager Service Schema	29
90	5. Provisioned Module Manager Service WSDL	34
91	References	37

92 1. Introduction

93 The Provisioned Module Manager (PMM) is a component used to instantiate and manage Provisioned Modules.
94 This specification documents the interfaces exposed by the PMM. The Provisioning Service Specification (see
95 [\[LibertyPROV\]](#)) provides an overview of the provisioning process and describes how the PMM fits into this process.

96 1.1. Notation and Conventions

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

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

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

104 1.1.1. XML Namespaces

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

106 • The prefix *pmm*: represents the PMM namespace. This namespace is the default for instance fragments, type
107 names, and element names in this document. In schema listings, and in examples of Discovery Service messages
108 and fragments thereof, this is the default namespace *when* no prefix is shown:

109 *urn:liberty:pmm:2007-09*

110 • The prefix *prov*: represents the Liberty ID-WSF Provisioning Service namespace [\[LibertyPROV\]](#):

111 *urn:liberty:prov:2007-09*

112 • The prefix *saml2*: stands for the SAMLv2 assertion namespace [\[SAMLCore2\]](#):

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

114 • The prefix *samlp2*: stands for the SAMLv2 protocol namespace [\[SAMLCore2\]](#):

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

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

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

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

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

120 2. Overview

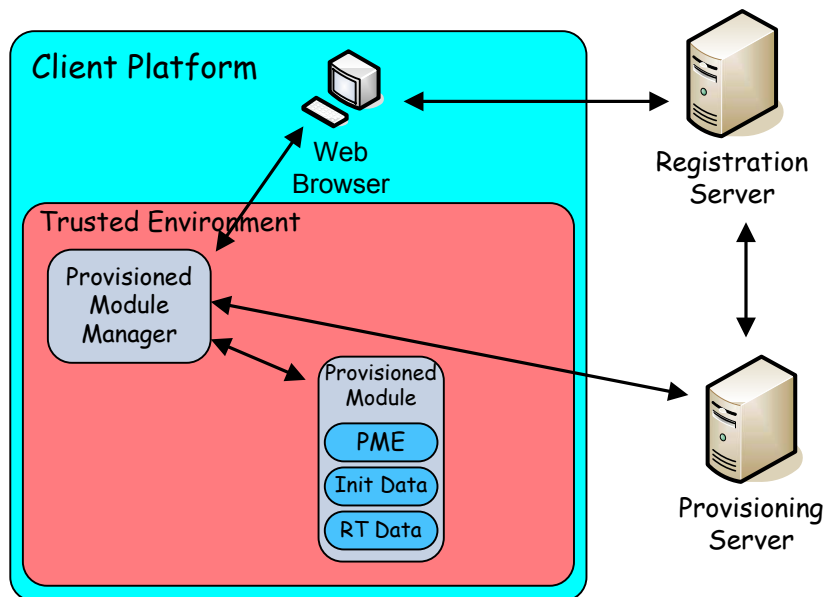
121 The Provisioned Module Manager (PMM) is a component in an system designed to enable run-time provisioning of
122 Provisioned Modules (PM) – functionality modules (executables in some operating environments) to advanced client
123 systems. The Liberty Advanced Client Technologies Overview (see [LibertyACT]) contains a complete description of
124 how the entire system works together. The reader is strongly encouraged to review that document prior to digging
125 into this specification.

126 The provisioning workflow provides for the instantiation of new PM(s) in a run-time environment over the wire. The
127 PMM takes a critical role in this process as a beach head on the target platform which is able to instantiate the PM.

128 The PMM is frequently instantiated within some trusted environment which opens the potential for distributing PMs
129 that contain functionality requiring a high level of trust and tamper resistance. We refer to such a PM as a Trusted
130 Module (TM). For example, the TM may provide some of the IdP extension functionality described within the Liberty
131 Advanced Client Technologies Overview document (see [LibertyACT]).

132 2.1. Provisioning Components

133 The following diagram illustrates the components involved in the provisioning process:



134

135

Figure 1. Provisioning Components

136 Things to note about this diagram:

137 • It is not drawn to any form of scale!

138 • The client platform represents any type of client, such as a personal computer, a device, a smart card, etc..

139 • The trusted environment represents some form of tamper resistant container (thus providing a level of trust for
140 the provisioned components). The trusted environment is **not** a requirement of these protocols – the components
141 shown within the trusted environment could very well exist directly within the relatively untrusted client platform
142 (e.g. the Provisioned Module Manager could run as a service within the Client Platform operating system).

143 • The Provisioned Module Manager (PMM) is a service running on the client platform which provides a beach
144 head for provisioning operations. The PMM exposes the interfaces documented within the Liberty ID-WSF
145 Provisioned Module Manager Service Specification [[LibertyPMM](#)].

146 This document does not address the chicken-vs-egg issue of how the PMM comes into being on the client. It may
147 be built into the platform or it may be manually installed by some party (such as the user). That discussion is
148 out-of-scope.

149 • The Provisioned Module (PM) is a component which performs some set of functionality. For example, a PM
150 could be a TM (a module which provides IdP extension functionality). PMs may also expose functionality that is
151 not defined by Liberty specifications.

152 Each PM is identified using a globally unique identifier called the Provisioned Module IDentifier (PMID). The
153 PMID is used to reference to specific instances of a PM when performing tasks like status updates or module
154 updates.

155 The PM is shown as being composed of 3 distinct parts:

156 • **Provisioned Module Engine (PME)** - the executable code which provides the functionality for the PM.

157 This is defined as a separate component here to enable a provisioning process which allows the PME to preexist
158 in the client platform and so just delivers the data necessary to instantiate the PM using that preexisting engine.
159 Of course, the PME may not preexist and in such cases the PMM will have to retrieve it.

160 During provisioning, the PME is passed by reference (name) so that the PMM can determine whether or not
161 the PME already exists (either because it was pre-installed or because the same PME has been previously
162 provisioned). Should the PMM need to obtain the PME, the passed in reference is used to identify the PME
163 being downloaded.

164 • **Initialization Data (PMInitData)** - the data needed by the PME in order to initialize a new instance of a
165 PM. This may be the actual data needed by the PME or it may be a reference that the PME knows how
166 to dereference and obtain the initialization data at runtime. This data may or may not be needed during the
167 provisioning process. Some PMs are fully individualized and have their PMInitData built in.

168 The format and structure of the PMInitData is out of scope for this document and is specific to the PME. It is
169 up to the Provisioning Service to resolve what data is needed for what PME. The PMM treats PMInitData as
170 an opaque data set that it passes to the PME upon initialization.

171 • **Runtime Data (PMRTData)** - the runtime data created/managed by the PM instance as it performs its tasks.
172 This would include things like MINGs for a TM that is minting assertions, private keys, etc. This is defined
173 separate from the InitData to allow for PM portability (where a previously activated PM is moved to another
174 client platform).

175 • The Web Browser in this diagram represents an enhanced browser (either directly or via a plug-in) with support
176 for the provisioning process. In other provisioning use cases this may be an application or even be the PMM itself
177 can instigate a new provision operation (typically via some direct interaction with the user).

178 • The Registration Server (RegS) is not a Liberty defined entity, but rather a deployment component for a particular
179 set of use cases. In this use case, the RegS interacts with the user through a web browser and then controls the
180 provisioning process using the interfaces on the Provisioning Server.

181 • The Provisioning Server (ProvS) is typically a network hosted service that is the primary entity with which the
182 PMM interacts. This server is an instance of a Liberty ID-WSF Provisioning Service (see [[LibertyPROV](#)]).

183 The primary function of the ProvS is to provide a trusted endpoint for the management and distribution of PMs.

184 3. Provisioned Module Manager Service (PMM)

185 The Provisioned Module Manager Service (PMM) provides the interfaces used by Advanced Clients and/or ProvSs to
186 initialize and/or maintain PMs.

187 An abstract WSDL definition for the PMM is included in this document, see [Section 5: Provisioned Module Manager](#)
188 [Service WSDL](#) . This WSDL document defines all of the "WSDL operations" for the IdP Service.

189 The complete schema for the PMM is included in this document, see [Section 4: Provisioned Module Manager Service](#)
190 [Schema](#) .

191 3.1. Service URIs

192 **Table 1. PMM Service URIs**

Use	URI
Service Type	<i>urn:liberty:pmm:2007-09</i>
Provision wsa:Action	<i>urn:liberty:pmm:2007-09:Provision</i>
ProvisionResponse wsa:Action	<i>urn:liberty:pmm:2007-09:ProvisionResponse</i>
PMActivate wsa:Action	<i>urn:liberty:pmm:2007-09:PMActivate</i>
PMActivateResponse wsa:Action	<i>urn:liberty:pmm:2007-09:PMActivateResponse</i>
PMDeactivate wsa:Action	<i>urn:liberty:pmm:2007-09:PMDeactivate</i>
PMDeactivateResponse wsa:Action	<i>urn:liberty:pmm:2007-09:PMDeactivateResponse</i>
PMDelete wsa:Action	<i>urn:liberty:pmm:2007-09:PMDelete</i>
PMDeleteResponse wsa:Action	<i>urn:liberty:pmm:2007-09:PMDeleteResponse</i>
PMSetStatus wsa:Action	<i>urn:liberty:pmm:2007-09:PMSetStatus</i>
PMSetStatusResponse wsa:Action	<i>urn:liberty:pmm:2007-09:PMSetStatusResponse</i>
PMGetStatus wsa:Action	<i>urn:liberty:pmm:2007-09:PMGetStatus</i>
PMGetStatusResponse wsa:Action	<i>urn:liberty:pmm:2007-09:PMGetStatusResponse</i>
PMUpdate wsa:Action	<i>urn:liberty:pmm:2007-09:PMUpdate</i>
PMUpdateResponse wsa:Action	<i>urn:liberty:pmm:2007-09:PMUpdateResponse</i>

193 3.2. Status Codes

194 The following status code strings are defined:

- 195 • *OK*: message processing succeeded
- 196 • *Failed*: general failure code
- 197 • *Forbidden*: action by invoker is forbidden.
- 198 • *NotFound*: the requested object is not present.
- 199 • *Duplicate*: the object already exists or the task has already been accomplished.

- 200 • *Invalid*: one or more of the specified parameters are invalid.
- 201 • *LimitExceeded*: too much data was sent to the ProvS.
- 202 • *FeatureNotSupported*: the request involves some capability that is not supported.

203 These strings are expected to appear in the "code" attribute of <Status> elements used in SOAP-bound PMM Service
204 protocol messages. Specific uses for the status codes are defined in the processing rules for individual messages. The
205 contents of the comment attribute are not defined by this specification, but it may be used for additional descriptive
206 text intended for human consumption (for example, to carry information that will aid debugging).

207 3.3. Data Definitions

208 The PMM makes extensive use of the data elements defined in the Liberty ID-WSF Provisioning Service Specification
209 [[LibertyPROV](#)] and does not define any of its own data element structures.

210 3.4. Request and Response Abstract Types

211 3.4.1. Complex Type RequestAbstractType

212 All request messages are of types that are derived from the abstract **RequestAbstractType** complex type. This type
213 defines common attributes that are associated with all PMM requests:

- 214 • **anyAttribute [Optional]** - zero or more attributes from a namespace other than that of this specification. One
215 such possibility is an **xs:ID** type attribute such as `xml:id` or `wsu:Id`.

216 The following schema fragment defines the **RequestAbstractType** complex type:

```
217 <!-- RequestAbstractType - common request message structure -->  
218 <xs:complexType name="RequestAbstractType" abstract="true">  
219   <xs:anyAttribute namespace="##other" processContents="lax"/>  
220 </xs:complexType>
```

224 3.4.2. Complex Type ResponseAbstractType

225 All response messages are of types that are derived from the abstract **ResponseAbstractType** complex type. This
226 type defines common attributes and elements that are associated with all PS responses:

- 227 • **<lu:Status> [Required]** - The `<lu:Status>` element is used to convey status codes and related information.
228 The schema fragment is defined in the Liberty ID-WSF Utility schema. The local definition of status codes are
229 described in [Section 3.2](#).

- 230 • **anyAttribute [Optional]** An attribute from a namespace other than that of this specification.

231 The following schema fragment defines the XML **ResponseAbstractType** complex type:

```
232 <!-- ResponseAbstractType - common message response structure -->
233
234 <xs:complexType name="ResponseAbstractType" abstract="true">
235   <xs:sequence>
236     <xs:element ref="lu:Status"/>
237   </xs:sequence>
238   <xs:anyAttribute namespace="##other" processContents="lax"/>
239 </xs:complexType>
240
```

241 **3.5. Operation: *Provision***

242 The *Provision* operation is used by a local application on the Advanced client or a ProvS to instigate the provisioning
243 of a new PM through the PMM.

244 When invoked by a local application on the Advanced Client, a `<prov:ProvisioningHandle>` is provided which
245 provides the necessary information for the PMM to communicate with the ProvS (essentially an indirect provisioning
246 process). When invoked by a ProvS, a `<PMDescriptor>` is typically provided enabling a direct provisioning process
247 (though it is possible for the ProvS to use a `<prov:ProvisioningHandle>` when provisioning).

248 **3.5.1. wsa:Action values for Provision Messages**

249 `<Provision>` request messages MUST include a `<wsa:Action>` SOAP header with the value of
250 "urn:liberty:pmm:2007-09:Provision."

251 `<ProvisionResponse>` messages MUST include a `<wsa:Action>` SOAP header with the value of
252 "urn:liberty:pmm:2007-09:ProvisionResponse."

253 **3.5.2. Provision Message**

254 The `<Provision>` request is called to provide the PMM with a `<prov:PMDescriptor>` or a
255 `<prov:ProvisioningHandle>` (which can be resolved into a `<prov:PMDescriptor>`) instigating the pro-
256 visioning of a new PM at the PMM.

257 The `<pmm:Provision>` request contains the following attributes and/or elements:

- 258 • `<prov:ProvisioningHandle>` **[Optional]** - the provisioning handle for the PM that is to be provisioned. This
259 is the option that is typically specified by an untrusted application running on the same platform.
- 260 • `<prov:PMDescriptor>` **[Optional]** - the provisioning descriptor for the PM that is to be provisioned. This
261 option is typically used by the ProvS when it is able to directly provision a PM to the PMM.
- 262 • `<dp:NotifyTo>` **[Optional]** - an optional endpoint reference for delayed notification completion messages (see
263 [\[LibertyDP\]](#)). This is used to allow the invoker to receive a completion status for a delayed or proxied operation
264 that completes at some point in the future.
- 265 • `wait` **[Optional]** - a boolean attribute indicating whether or not the caller wants to wait until the provisioning
266 process for the PM is complete (the PM is initialized).
267 If not specified, or if the attribute is set to *true*, the PMM SHOULD wait for the provisioning to be complete.
268 If the attribute is set to *false*, the PMM SHOULD return an OK indicating an acknowledgement of the provisioning
269 request as soon as the PMM is satisfied as to the validity of the request and the intent to proceed with the
270 provisioning process.

271 • anyAttribute **[Optional]** Zero or more attributes from a namespace other than that of this specification. One
 272 such possibility is an **xs:ID** type attribute such as `xml:id` or `wsu:Id`.

273 The invoker **MUST** specify the `<prov:ProvisioningHandle>` or the `<prov:PMDescriptor>` element.
 274 Both elements **MUST NOT** be specified.

275 The schema for the `<pmm:Provision>` is shown below.

```

276 <!-- Provision - to instigate the provisioning of a PM at the PMM -->
277
278 <xs:element name="Provision" type="ProvisionType"/>
279
280 <xs:complexType name="ProvisionType">
281   <xs:complexContent>
282     <xs:extension base="RequestAbstractType">
283       <xs:sequence>
284         <xs:choice>
285           <xs:element ref="prov:ProvisioningHandle" />
286           <xs:element ref="prov:PMDescriptor" />
287         </xs:choice>
288         <xs:element ref="dp:NotifyTo" minOccurs="0" />
289       </xs:sequence>
290       <xs:attribute name="wait" type="xs:boolean" use="optional" />
291     </xs:extension>
292   </xs:complexContent>
293 </xs:complexType>
294

```

295 **Figure 2.** `<pmm:Provision>` — Schema Fragment

296 An example message body containing a `<pmm:Provision>` message follows. This request presents a provisioning
 297 handle to the PMM to initiate the Provisioning process.

```

298 <pmm:Provision wait="true" >
299   <prov:ProvisioningHandle xs:id="2302384823023">
300     <prov:PMDArtifact>23asdfhoi323hpo sdf923h9sdfhwiorh2398asdfjweoiha</prov:PMDArtifact>
301     <prov:ProvisioningServiceEPR>
302       <wsa:Address>http://provision.idpsRus.com</wsa:Address>
303       <wsa:Metadata>
304         <ds:Abstract>Provisioning Service</ds:Abstract>
305         <ds:ProviderID>http://provisioning-provider.idpsRus.com/</ds:ProviderID>
306         <ds:ServiceType>urn:liberty:prov:2007-09</ds:ServiceType>
307         <ds:Framework version="2.0" />
308         <ds:SecurityContext>
309           <ds:SecurityMechID>
310             urn:liberty:security:2005-02:TLS:SAMLV2
311           </ds:SecurityMechID>
312           <sec:Token ref="urn:liberty:disco:tokenref:ObtainFromIDP" />
313         </ds:SecurityContext>
314       </wsa:Metadata>
315     </prov:ProvisioningServiceEPR>
316     <ds:Signature>
317       ... signature info here ..
318     </ds:Signature>
319   </prov:ProvisioningHandle>
320 </pmm:Provision>
321

```

322 **Example 1.** Example `<pmm:Provision>` Message

323 3.5.3. ProvisionResponse Message

324 This response to the `<pmm:Provision>` request contains the following elements:

- 325 • `<lu:Status>`: **[Required]** - the status of the response. See the processing rules below for more information.
- 326 • `anyAttribute` **[Optional]** - zero or more attributes from a namespace other than that of this specification. One
- 327 such possibility is an `xs:ID` type attribute such as `xml:id` or `wsu:Id`.

```

328 <!-- ProvisionResponse - response to the Provision request -->
329
330 <xs:element name="ProvisionResponse" type="ProvisionResponseType"/>
331
332 <xs:complexType name="ProvisionResponseType">
333   <xs:complexContent>
334     <xs:extension base="ResponseAbstractType" />
335   </xs:complexContent>
336 </xs:complexType>
337
```

338 **Figure 3. `<pmm:ProvisionResponse>` — Schema Fragment**

339 An example message body containing a `<pmm:ProvisionResponse>` message follows. This is a successful

340 response.

```

341 <pmm:ProvisionResponse>
342   <lu:Status code="OK" />
343 </pmm:ProvisionResponse>
344
```

345 **Example 2. Example `<pmm:ProvisionResponse>` Message**

346 3.5.4. Provision Processing Rules

- 347 • This operation adopts the Delayed Notification design pattern (see [\[LibertyDP\]](#)) and incorporates all of the
- 348 associated processing rules. Delayed notifications are needed for delayed operations (where the `at` attribute
- 349 is used with a time in the future).

350 Delayed Notifications SHOULD NOT to be used for immediate operations (no future `at` attribute). In such cases,

351 the PMM SHOULD wait for the completion of the request to return the appropriate status in its response to the

352 invoker.

353 If any portion of the request IS NOT an immediate operation and the `<dp:NotifyTo>` element IS NOT present

354 on the request, the PMM should validate the request to the extent possible and respond with the results of that

355 validation. If that validation succeeded, the PMM MUST continue with the execution of the operation. In such

356 cases the actual completion status of the operation may be different from what was reported (e.g. the request may

357 fail at the time it is attempted for some reason), but that will not be known to the invoker given that they have

358 chosen to not provide a delayed notification endpoint.

359 If any portion of the request IS NOT an immediate operation and the `<dp:NotifyTo>` element IS present on the

360 request, the PMM MUST report the final completion status (e.g. wait for the completion status of any delayed or

361 indirect operation at the PMM) of the request. If multiple operations are included in the request, the ProvS MAY

362 group the results of some or all of the operations in a single delayed notification message.

363 In the case of a mixed request, the immediate portion of the request SHOULD be handled as an immediate request

364 and the actual results returned with the initial response.

- 365 • If the `wait` attribute is set to a boolean true value, the PMM MUST wait until the provisioning process is complete
- 366 prior to sending the response to this request. This, potentially, includes the steps of obtaining the PMDescriptor
- 367 and the PEngine from the ProvS.

- 368 • If data in the input parameters is invalid (such as a missing PMArtifact in the PH) the PMM MUST treat the request
369 as a failure. The PMM SHOULD return an unsuccessful status in the response. If detailed status codes are being
370 included, the detailed status code for this error MUST be "Invalid."
- 371 If the PMM is unable to return an unsuccessful status in the response, it MUST, instead, generate a SOAP fault
372 message. This allows for tooling situations where the request format error is processed by the SOAP/XML tooling
373 layer prior to getting to the PMM instance.
- 374 • If the input parameters require a capability that is unsupported within the PMM (such as an unsupported Security
375 Mechanism or framework version) such that it is impossible for the PMM to complete the task, the PMM MUST
376 treat the request as a failure. If detailed status codes are being included, the detailed status code for this error
377 MUST be "Unsupported."
- 378 • If request processing succeeded, the top-level status code MUST be "OK." If the request has been accepted **and**
379 is subject to delayed notification, the top-level status code MUST be "WillNotify." Otherwise, the top-level status
380 code MUST be "Failed."
- 381 • If the top-level status code is "Failed," the response MAY also contain *Forbidden* as a second-level status code.
382 The PMM instance may not wish to reveal the reason for failure, in which case no second-level status code will
383 appear.

384 3.6. Operation: *PMActivate*

385 The *PMActivate* operation is used by a provisioning entity to activate a currently inactive PM.

386 3.6.1. *wsa:Action* values for *PMActivate* Messages

387 <PMActivate> request messages MUST include a <wsa:Action> SOAP header with the value of
388 "urn:liberty:pmm:2007-09:PMActivate."

389 <PMActivateResponse> messages MUST include a <wsa:Action> SOAP header with the value of
390 "urn:liberty:pmm:2007-09:PMActivateResponse."

391 3.6.2. *PMActivate* Message

392 The <PMActivate> request is called by a provisioning entity (such as the Liberty ID-WSF Provisioning Service -
393 see [[LibertyPROV](#)]) to activate an inactive PM at the PMM.

394 The <pmm:PMActivate> request contains the following elements/attributes:

395 The <pmm:PMActivate> request contains the following elements/attributes:

- 396 • <pmm:PMActivateItem> [**Required**] - one or more activate request items which contain the following
397 elements/attributes:
- 398 • <PMID> [**Required**] - the identifier of the PM being activated.
 - 399 • <itemID> [**Required**] - a unique (in this request) identifier for this <pmm:PMActivateItem> element. This
400 value is used to correlate the response data to this request data.
 - 401 • <at> [**Optional**] - a time at which the activation should take place. If specified, the value SHOULD be some
402 point in the future at which the PMM would activate the PM.
 - 403 If this attribute is not specified, or it is specified with a time in the past, the PM MUST be activated now.

- 404 • <dp:NotifyTo> **[Optional]** - an optional endpoint reference for delayed notification completion messages (see
- 405 [\[LibertyDP\]](#)). This is used to allow the invoker to receive a completion status for a delayed or proxied operation
- 406 that completes at some point in the future.
- 407 • anyAttribute **[Optional]** Zero or more attributes from a namespace other than that of this specification. One
- 408 such possibility is an **xs:ID** type attribute such as `xml:id` or `wsu:Id`.

409 The schema for the `<pmm:PMActivate>` is shown below.

```

410 <!-- PMActivate - to activate one or more PM(s) at the PMM -->
411
412 <xs:element name="PMActivate" type="PMActivateType"/>
413
414 <xs:complexType name="PMActivateType">
415   <xs:complexContent>
416     <xs:extension base="RequestAbstractType">
417       <xs:sequence>
418         <xs:element ref="PMActivateItem" maxOccurs="unbounded" />
419         <xs:element ref="dp:NotifyTo" minOccurs="0" />
420       </xs:sequence>
421     </xs:extension>
422   </xs:complexContent>
423 </xs:complexType>
424
425 <xs:element name="PMActivateItem" type="PMActivateItemType" />
426
427 <xs:complexType name="PMActivateItemType">
428   <xs:sequence>
429     <xs:element ref="prov:PMID" />
430   </xs:sequence>
431   <xs:attribute name="itemID" type="xs:string" use="required" />
432   <xs:attribute name="at" type="xs:dateTime" use="optional" />
433 </xs:complexType>
434

```

435 **Figure 4. `<pmm:PMActivate>` — Schema Fragment**

436 An example message body containing a `<pmm:PMActivate>` message follows. This request activates 2 PMs, one

437 immediately and one at 1PM on the 18th of Dec, 2006, now and provides a delayed notification EPR.

```

438 <pmm:PMActivate>
439   <pmm:PMActivateItem itemID="1" at="2007-01-18T13:00:00Z" >
440     <prov:PMID issuer="http://provs-r-us.com">uuid:239032-230328-92379-2397923</prov:PMID>
441   </pmm:PMActivateItem>
442   <pmm:PMActivateItem itemID="2">
443     <prov:PMID issuer="http://provs-r-us.com">uuid:392380-948492-18923-2389238</prov:PMID>
444   </pmm:PMActivateItem>
445   <dp:NotifyTo>
446     <wsa:Address>https://provider.com/notifi cations</wsa:Address>
447     <wsa:Metadata>
448       <ds:ProviderID>http://provider.com/</ds:ProviderID>
449       <ds:ServiceType>urn:liberty:dp:2007-09:notification</ds:ServiceType>
450       <ds:Framework version="2.0" />
451       <ds:SecurityContext>
452         <ds:SecurityMechID>urn:liberty:security:2005-02:TLS:null</ds:SecurityMechID>
453       </ds:SecurityContext>
454     </wsa:Metadata>
455   </dp:NotifyTo>
456 </pmm:PMActivate>
457

```

458 **Example 3. Example `<pmm:PMActivate>` Message**

459 3.6.3. PMActivateResponse Message

460 This response to the `<pmm:PMActivate>` request contains the following elements:

- 461 • `<lu:Status>`: **[Required]** - the status of the response. See the processing rules below for more information.
- 462 • `anyAttribute` **[Optional]** - zero or more attributes from a namespace other than that of this specification. One
463 such possibility is an `xs:ID` type attribute such as `xml:id` or `wsu:Id`.

```
464 <!-- PMActivateResponse - the response to the PMActivate request -->  
465  
466 <xs:element name="PMActivateResponse" type="PMActivateResponseType"/>  
467  
468 <xs:complexType name="PMActivateResponseType">  
469   <xs:complexContent>  
470     <xs:extension base="ResponseAbstractType" />  
471   </xs:complexContent>  
472 </xs:complexType>  
473
```

474 **Figure 5. `<pmm:PMActivateResponse>` — Schema Fragment**

475 An example message body containing a `<pmm:PMActivateResponse>` message follows. This is a partial response.
476 The 2nd activation, which is an immediate operation, is successful. The 1st activation is queued for processing and
477 the PMM will notify the invoker when it completes.

```
478 <pmm:PMActivateResponse>  
479   <lu:Status code="Partial">  
480     <lu:Status ref="1" code="WillNotify" />  
481     <lu:Status ref="2" code="OK" />  
482   </lu:Status>  
483 </pmm:PMActivateResponse>  
484
```

485 **Example 4. Example `<pmm:PMActivateResponse>` Message**

486 3.6.4. PMActivate Processing Rules

- 487 • This operation adopts the Delayed Notification design pattern (see [\[LibertyDP\]](#)) and incorporates all of the
488 associated processing rules. Delayed notifications are needed for delayed operations (where the `at` attribute
489 is used with a time in the future).

490 Delayed Notifications SHOULD NOT to be used for immediate operations (no future `at` attribute). In such cases,
491 the PMM SHOULD wait for the completion of the request to return the appropriate status in its response to the
492 invoker.

493 If any portion of the request IS NOT an immediate operation and the `<dp:NotifyTo>` element IS NOT present
494 on the request, the PMM should validate the request to the extent possible and respond with the results of that
495 validation. If that validation succeeded, the PMM MUST continue with the execution of the operation. In such
496 cases the actual completion status of the operation may be different from what was reported (e.g. the request may
497 fail at the time it is attempted for some reason), but that will not be known to the invoker given that they have
498 chosen to not provide a delayed notification endpoint.

499 If any portion of the request IS NOT an immediate operation and the `<dp:NotifyTo>` element IS present on the
500 request, the PMM MUST report the final completion status (e.g. wait for the completion status of any delayed or
501 indirect operation at the PMM) of the request. If multiple operations are included in the request, the ProvS MAY
502 group the results of some or all of the operations in a single delayed notification message.

503 In the case of a mixed request, the immediate portion of the request SHOULD be handled as an immediate request
504 and the actual results returned with the initial response.

- 505 • Requests to activate a PM that is not known to the PMM MUST result in a failure. If detailed status codes are
506 being included in the response, the detailed status code for this error MUST be "NotFound."
- 507 • Requests to activate a PM that is already activated MUST result in a failure unless the request is for a future time
508 when the PMM can determine that the PM will be inactive at that point in time. If detailed status codes are being
509 included in the response, the detailed status code for this error MUST be "AlreadySo."
- 510 • If the timestamp specified in the `at` attribute is in the future and exactly matches the future timestamp for a
511 pending deactivation, this request MUST have the effect of canceling the pending request (such that neither of the
512 two requests remain pending). This is the only means to cancel a pending deactivation request.
- 513 In such cases, if the cancellation is successful, this request should return a successful status.
- 514 • If a pending activation is canceled (via a `<pmm:PMDeactivate>` with the exact same `at` attribute value) and
515 delayed notification is in effect for the activation, the completion status included in the notification message MUST
516 be a failure. If detailed status codes are included, the detailed status code for this case MUST be "Canceled."
- 517 • The behavior of the PMM when multiple pending occurrences of the same operation are requested (such as
518 multiple future activations) is NOT defined by this specification. It is recommended that callers NOT attempt
519 such operations. If an implementation of a PMM chooses to refuse such duplicated operations and treat them as a
520 failure, the detailed status code, if present, MUST be "Duplicate."
- 521 • If the activation is to take place immediately (e.g., there was no `at` attribute specified), the PMM SHOULD wait
522 until the activation process is complete prior to sending the response to this request.
- 523 • If this operation is to take place at some point in the future, the PMM SHOULD, to the extent possible, validate
524 the parameters of the request to detect any errors. Any such errors detected MUST be reported on the response to
525 this request.
- 526 • If **all** items in the request have the same completion status, the top level status MUST reflect that completion status
527 and MUST be "OK", "Failed", or "WillNotify". Otherwise, if the results were mixed, the top-level status MUST
528 be *Partial*. and a second-level status MUST be included indicating which items succeeded, which failed, and
529 which will be subject to delayed notification. The second level status elements MUST include the `ref` attribute
530 containing the `itemID` value for the item. For failures, the second-level status codes MAY simply be "Failed", or
531 they may indicate with more detail the reason for the failure.
- 532 If the top-level status is "Failed", second level status codes MAY be present which contain detailed error
533 information if the PMM wants to share that information with the invoking party.

534 3.7. Operation: *PMDeactivate*

535 The *PMDeactivate* operation is used by a provisioning entity to deactivate a currently active PM.

536 3.7.1. `wsa:Action` values for *PMDeactivate* Messages

537 `<PMDeactivate>` request messages MUST include a `<wsa:Action>` SOAP header with the value of
538 "urn:liberty:pmm:2007-09:PMDeactivate."

539 `<PMDeactivateResponse>` messages MUST include a `<wsa:Action>` SOAP header with the value of
540 "urn:liberty:pmm:2007-09:PMDeactivateResponse."

541 3.7.2. *PMDeactivate* Message

542 The `<PMDeactivate>` request is called by a provisioning entity (such as the Liberty ID-WSF Provisioning Service
543 - see [[LibertyPROV](#)]) to deactivate an active PM at the PMM.

544 The `<pmm:PMDeactivate>` request contains the following attributes and/or elements:

- 545 • <pmm:PMDeactivateItem> **[Required]** - one or more deactivate request items which contain the following
 546 elements/attributes:
- 547 • <PMID> **[Required]** - the identifier of the PM being deactivated.
- 548 • itemID **[Required]** - a unique (in this request) identifier for this <pmm:PMDeactivateItem> element. This
 549 value is used to correlate the response data to this request data.
- 550 • at **[Optional]** - a time at which the deactivation should take place. If specified, the value SHOULD be some
 551 point in the future at which the PMM would deactivate the PM.
- 552 If this attribute is not specified, or it is specified with a time in the past, the PM MUST be deactivated now.
- 553 • <dp:NotifyTo> **[Optional]** - an optional endpoint reference for delayed notification completion messages (see
 554 [\[LibertyDP\]](#)). This is used to allow the invoker to receive a completion status for a delayed or proxied operation
 555 that completes at some point in the future.
- 556 • anyAttribute **[Optional]** Zero or more attributes from a namespace other than that of this specification. One
 557 such possibility is an **xs:ID** type attribute such as `xml:id` or `wsu:Id`.

558 The schema for the <pmm:PMDeactivate> is shown below.

```

559 <!-- PMDeactivate - to deactivate a PM at the PMM -->
560
561 <xs:element name="PMDeactivate" type="PMDeactivateType"/>
562
563 <xs:complexType name="PMDeactivateType">
564   <xs:complexContent>
565     <xs:extension base="RequestAbstractType">
566       <xs:sequence>
567         <xs:element ref="PMDeactivateItem" maxOccurs="unbounded" />
568         <xs:element ref="dp:NotifyTo" minOccurs="0" />
569       </xs:sequence>
570     </xs:extension>
571   </xs:complexContent>
572 </xs:complexType>
573
574 <xs:element name="PMDeactivateItem" type="PMDeactivateItemType" />
575
576 <xs:complexType name="PMDeactivateItemType">
577   <xs:sequence>
578     <xs:element ref="prov:PMID" />
579   </xs:sequence>
580   <xs:attribute name="itemID" type="xs:string" use="required" />
581   <xs:attribute name="at" type="xs:dateTime" use="optional" />
582 </xs:complexType>
583

```

584 **Figure 6. <pmm:PMDeactivate> — Schema Fragment**

585 An example message body containing a <pmm:PMDeactivate> message follows. This request deactivates two PMs,
 586 one now and one near midnight New Year's eve.

```

587 <pmm:PMDeactivate>
588   <pmm:PMDeactivateItem itemID="1" at="2007-01-31T23:59:59Z" >
589     <prov:PMID issuer="http://provs-r-us.com">uuid:239032-230328-92379-2397923</prov:PMID>
590   </pmm:PMDeactivateItem>
591   <pmm:PMDeactivateItem itemID="2">
592     <prov:PMID issuer="http://provs-r-us.com">uuid:392380-948492-18923-2389238</prov:PMID>
593   </pmm:PMDeactivateItem>
594 </pmm:PMDeactivate>
595

```


596 **Example 5. Example `<pmm:PMDeactivate>` Message**

597 **3.7.3. PMDeactivateResponse Message**

598 This response to the `<pmm:PMDeactivate>` request contains the following elements:

- 599 • `<lu:Status>`: **[Required]** - the status of the response. See the processing rules below for more information.
- 600 • `anyAttribute` **[Optional]** - zero or more attributes from a namespace other than that of this specification. One
601 such possibility is an **xs:ID** type attribute such as `xml:id` or `wsu:Id`.

```
602 <!-- PMDeactivateResponse - the response to the PMDeactivate request -->
603
604 <xs:element name="PMDeactivateResponse" type="PMDeactivateResponseType" />
605
606 <xs:complexType name="PMDeactivateResponseType">
607   <xs:complexContent>
608     <xs:extension base="ResponseAbstractType" />
609   </xs:complexContent>
610 </xs:complexType>
611
```

612 **Figure 7. `<pmm:PMDeactivateResponse>` — Schema Fragment**

613 An example message body containing a `<pmm:PMDeactivateResponse>` message follows. This is a partially
614 successful response where one of the PMs was not found.

```
615 <pmm:PMDeactivateResponse>
616   <lu:Status code="Partial">
617     <lu:Status ref="2" code="NotFound" />
618   </lu:Status>
619 </pmm:PMDeactivateResponse>
620
```

621 **Example 6. Example `<pmm:PMDeactivateResponse>` Message**

622 **3.7.4. PMDeactivate Processing Rules**

- 623 • `<dp:NotifyTo>` **[Optional]** - an optional endpoint reference for delayed notification completion messages (see
624 [\[LibertyDP\]](#)). This is used to allow the invoker to receive a completion status for a delayed or proxied operation
625 that completes at some point in the future.
- 626 • If a specified PMID is not known to the PMM, the PMM **MUST** treat that request item as a failure. If detailed
627 status codes are being included in the response, the detailed status code for this error **MUST** be *"NotFound."*
- 628 • If the deactivation is to take place immediately (e.g., there was no `at` attribute specified), the PMM **SHOULD** wait
629 until the deactivation process is complete prior to sending the response to this request.
- 630 • Requests to deactivate a PM that is not active **MUST** result in a failure unless the request is for a future time when
631 the PMM can determine that the PM will be active at that point in time. If detailed status codes are being included
632 in the response, the detailed status code for this error **MUST** be *"AlreadySo."*
- 633 • If this operation is to take place at some point in the future, the PMM **SHOULD**, to the extent possible, validate
634 the parameters of the request to detect any errors. Any such errors detected **MUST** be reported on the response to
635 this request.

- 636 • If the timestamp specified in the `at` attribute is in the future and exactly matches the future timestamp for a pending
637 activation, this request **MUST** have the effect of canceling the pending request (such that neither request remains
638 pending). This is the only means to cancel a pending activation request.
- 639 In such cases, the deactivation request is considered successful (and returns a successful response) but has no effect
640 other than to cancel the pending activation.
- 641 • If a pending deactivation is canceled (via a `<pmm:PMActivate>` with the exact same `at` attribute value) and
642 delayed notification is in effect for the deactivation, the completion status in the notification message **MUST**
643 indicate a failure. If detailed status codes are included, the detailed status code for this case **MUST** be *"Canceled."*
- 644 • The behavior of the PMM when multiple pending occurrences of the same operation are requested (such as multiple
645 future deactivations) is **NOT** defined by this specification. It is recommended that callers **NOT** attempt such
646 operations. If an implementation of a PMM chooses to refuse such duplicated operations and treat them as a
647 failure, the detailed status code, if present, **MUST** be *"Duplicate."*
- 648 • If **all** items in the request have the same completion status, the top level status **MUST** reflect that completion status
649 and **MUST** be *"OK"*, *"Failed"*, or *"WillNotify"*. Otherwise, if the results were mixed, the top-level status **MUST**
650 be *Partial*. and a second-level status **MUST** be included indicating which items succeeded, which failed, and
651 which will be subject to delayed notification. The second level status elements **MUST** include the `ref` attribute
652 containing the `itemID` value for the item. For failures, the second-level status codes **MAY** simply be *"Failed"*, or
653 they may indicate with more detail the reason for the failure.
- 654 If the top-level status is *"Failed"*, second level status codes **MAY** be present which contain detailed error
655 information if the PMM wants to share that information with the invoking party.

656 3.8. Operation: *PMDelete*

657 The *PMDelete* operation is used by a provisioning entity to delete a currently installed PM at the PMM.

658 3.8.1. `wsa:Action` values for *PMDelete* Messages

659 `<PMDelete>` request messages **MUST** include a `<wsa:Action>` SOAP header with the value of
660 `"urn:liberty:pmm:2007-09:PMDelete."`

661 `<PMDeleteResponse>` messages **MUST** include a `<wsa:Action>` SOAP header with the value of
662 `"urn:liberty:pmm:2007-09:PMDeleteResponse."`

663 3.8.2. *PMDelete* Message

664 The `<PMDelete>` request is called by a provisioning entity (such as the Liberty ID-WSF Provisioning Service - see
665 [\[LibertyPROV\]](#)) to delete a PM at the PMM.

666 If the PM is active at the time of deletion, the PM is deactivated and deleted in a single operation (in other words, there
667 is no need to call `<pmm:PMDeactivate>` prior to calling `<pmm:PMDelete>`).

668 The `<pmm:PMDelete>` the following attributes and/or elements:

- 669 • `<pmm:PMDeleteItem>` [**Required**] - one or more containing the following elements and/or attributes:
- 670 • `<PMID>` [**Required**] - the identifier of the PM being deleted.
- 671 • `itemID` [**Required**] - a unique (in this request) identifier for this `<pmm:PMDeleteItem>` element. This value
672 is used to correlate the response data to this request data.
- 673 • `anyAttribute` [**Optional**] - zero or more attributes from a namespace other than that of this specification. One
674 such possibility is an `xs:ID` type attribute such as `xml:id` or `wsu:Id`.

675 The schema for the `<pmm:PMDelete>` is shown below.

```
676 <!-- PMDelete - to delete a PM at the PMM -->
677
678 <xs:element name="PMDelete" type="PMDeleteType"/>
679
680 <xs:complexType name="PMDeleteType">
681   <xs:complexContent>
682     <xs:extension base="RequestAbstractType">
683       <xs:sequence>
684         <xs:element ref="PMDeleteItem" maxOccurs="unbounded" />
685       </xs:sequence>
686     </xs:extension>
687   </xs:complexContent>
688 </xs:complexType>
689
690 <xs:element name="PMDeleteItem" type="PMDeleteItemType" />
691
692 <xs:complexType name="PMDeleteItemType">
693   <xs:sequence>
694     <xs:element ref="prov:PMID" />
695   </xs:sequence>
696   <xs:attribute name="itemID" type="xs:string" use="required" />
697 </xs:complexType>
698
```

699 **Figure 8. `<pmm:PMDelete>` — Schema Fragment**

700 An example message body containing a `<pmm:PMDelete>` message follows. This request deletes one PM.

```
701 <pmm:PMDelete>
702   <pmm:PMDeleteItem itemID="1">
703     <prov:PMID issuer="http://provs-r-us.com">uuid:239032-230328-923 79-2397923</prov:PMID>
704   </pmm:PMDeleteItem>
705 </pmm:PMDelete>
706
```

707 **Example 7. Example `<pmm:PMDelete>` Message**

708 **3.8.3. PMDeleteResponse Message**

709 This response to the `<pmm:PMDelete>` request contains the following elements:

- 710 • `<lu:Status>`: **[Required]** - the status of the response. See the processing rules below for more information.
- 711 • `anyAttribute` **[Optional]** - zero or more attributes from a namespace other than that of this specification. One
- 712 such possibility is an **xs:ID** type attribute such as `xml:id` or `wsu:Id`.

```
713 <!-- PMDeleteResponse - the response to the PMDelete request -->
714
715 <xs:element name="PMDeleteResponse" type="PMDeleteResponseType"/>
716
717 <xs:complexType name="PMDeleteResponseType">
718   <xs:complexContent>
719     <xs:extension base="ResponseAbstractType" />
720   </xs:complexContent>
721 </xs:complexType>
722
```

723 **Figure 9. <pmm:PMDeleteResponse> — Schema Fragment**

724 An example message body containing a <pmm:PMDeleteResponse> message follows. This is a successful
725 response.

```
726 <pmm:PMDeleteResponse>
727   <lu:Status code="OK" />
728 </pmm:PMDeleteResponse>
729
```

730 **Example 8. Example <pmm:PMDeleteResponse> Message**

731 3.8.4. PMDelete Processing Rules

- 732 • If a specified PMID is not known to the PMM, or has already been deleted, the PMM **MUST** treat that request
733 item as a failure. If detailed status codes are being included in the response, the detailed status code for this error
734 **MUST** be *NotFound*.
- 735 • All pending operations (such as a pending future deactivation) related to this PM **MUST** be canceled. If delayed
736 notification is in effect for any of the pending operations, the completion status included in the notification message
737 **MUST** indicate failure. If detailed status codes are included in the notification, the detailed status code for this
738 case **MUST** be *Canceled*.
- 739 • If request processing succeeded for all PMs, the top-level status code **MUST** be *OK*. If the request processing failed
740 for all PMs, the top-level status code **MUST** be *Failed*. Otherwise, if the results were mixed, the top-level status
741 **MUST** be *Partial*. and a second-level status **MUST** be included for the items for which the processing was not
742 successful. The second level status for such items **MUST** indicate that the processing failed and **MUST** include the
743 `ref` attribute containing the `itemID` value for the item. These second-level status codes **MAY** simply be *Failed*,
744 or they may indicate with more detail the reason for the failure.
- 745 • If the top-level status is not *OK*. and second level status codes are present, they **MAY** contain detailed error
746 information if the PMM wants to share that information with the invoking party.

747 **3.9. Operation: *PMUpdate***

748 The *PMUpdate* operation is used by a provisioning entity to update a currently installed PM at the PMM.

749 **3.9.1. *wsa:Action* values for *PMUpdate* Messages**

750 <PMUpdate> request messages MUST include a <wsa:Action> SOAP header with the value of
751 "urn:liberty:pmm:2007-09:PMUpdate."

752 <PMUpdateResponse> messages MUST include a <wsa:Action> SOAP header with the value of
753 "urn:liberty:pmm:2007-09:PMUpdateResponse."

754 **3.9.2. *PMUpdate* Message**

755 The <PMUpdate> request is called by a provisioning entity (such as the Liberty ID-WSF Provisioning Service - see
756 [[LibertyPROV](#)]) to update one or more PM(s) at the PMM.

757 The <pmm:PMUpdate> request contains the following elements/attributes:

758 • <pmm:PMUpdateItem> **[Required]** - one or more update request items which contain the following
759 elements/attributes:

760 • <prov:PMDescriptor> **[Required]** - the replacement/updated information for the PM that is to be updated.
761 The contents of this element will vary depending upon the setting of the *type* attribute.

762 The interpretation of the contents within the <prov:PMDescriptor> is similar to the interpretation of those
763 same elements when processing an initial provisioning operation including:

764 • The <PMID> element identifies the existing PM that is to be updated.

765 • The *activate*, *activateAt*, and *deactivateAt* attributes **SHOULD NOT** be specified when a
766 <PMDescriptor> is used in an update request. The standard interfaces for managing the activation
767 status are the only way to update/change the activation status of a PMD.

768 • The <prov:PMEngineRef> , <prov:PMInitData> , and <prov:PMRTData> elements all contain
769 the updated information, if any, for this update operation.

770 • *type* **[Required]** - the type of update being requested. This attribute has the same values and interpretations as
771 the *type* attribute used on the <prov:PMDUpdateItem> in the <prov:PMDUpdate> interface in the Liberty
772 Provisioning Service Specification ([[LibertyPROV](#)]).

773 • *itemID* **[Required]** - a unique (in this request) identifier for this <pmm:PMUpdateItem> element. This value
774 is used to correlate the response data to this request data.

775 • *at* **[Optional]** - an optional time at which the update should take place. If specified, this **SHOULD** be some
776 time in the future.

777 If this attribute is not specified, or it is specified with a time in the past, the PM **MUST** be updated now.

778 • *anyAttribute* **[Optional]** - zero or more attributes from a namespace other than that of this specification.
779 One such possibility is an **xs:ID** type attribute such as *xml:id* or *wsu:Id*.

780 • <dp:NotifyTo> **[Optional]** - an optional endpoint reference for delayed notification completion messages (see
781 [[LibertyDP](#)]). This is used to allow the invoker to receive a completion status for a delayed or proxied operation
782 that completes at some point in the future.

783 • *anyAttribute* **[Optional]** Zero or more attributes from a namespace other than that of this specification. One
784 such possibility is an **xs:ID** type attribute such as *xml:id* or *wsu:Id*.

785 The schema for the `<pmm:PMUpdate>` is shown below.

```

786 <!-- PMUpdate - update the PM at the PMM -->
787
788 <xs:element name="PMUpdate" type="PMUpdateType"/>
789
790 <xs:complexType name="PMUpdateType">
791   <xs:complexContent>
792     <xs:extension base="RequestAbstractType">
793       <xs:sequence>
794         <xs:element ref="PMUpdateItem" maxOccurs="unbounded" />
795         <xs:element ref="dp:NotifyTo" minOccurs="0" />
796       </xs:sequence>
797     </xs:extension>
798   </xs:complexContent>
799 </xs:complexType>
800
801 <xs:element name="PMUpdateItem" type="PMUpdateItemType" />
802
803 <xs:complexType name="PMUpdateItemType">
804   <xs:sequence>
805     <xs:element ref="prov:PMDescriptor" />
806   </xs:sequence>
807   <xs:attribute name="type" type="xs:anyURI" use="required" />
808   <xs:attribute name="itemID" type="xs:string" use="required" />
809   <xs:attribute name="at" type="xs:dateTime" use="optional" />
810 </xs:complexType>
811

```

812 **Figure 10.** `<pmm:PMUpdate>` — Schema Fragment

813 An example message body containing a `<pmm:PMUpdate>` message follows. This request updates the engine only of
 814 a single PM.

```

815 <pmm:PMUpdate>
816   <pmm:PMUpdateItem itemID="1" type="urn:liberty:prov:2007-09:ut:engine">
817     <prov:PMDescriptor xs:id="2323923900239" >
818       <prov:PMID issuer="http://provs-r-us.com">uuid:778349-283920-88379-5448739</prov:PMID>
819       <prov:PMEngineRef>https://pmsRus.org/VeryTrustedModule/4.0</prov:PMEngineRef>
820       <ds:Signature>
821         ... signature data goes here ...
822       </ds:Signature>
823     </prov:PMDescriptor>
824   </pmm:PMUpdateItem>
825 </pmm:PMUpdate>
826

```

827 **Example 9.** Example `<pmm:PMUpdate>` Message

828 3.9.3. PMUpdateResponse Message

829 This response to the `<pmm:PMUpdate>` request contains the following elements:

- 830 • `<lu:Status>`: **[Required]** - the status of the response. See the processing rules below for more information.
- 831 • `anyAttribute` **[Optional]** - zero or more attributes from a namespace other than that of this specification. One
 832 such possibility is an **xs:ID** type attribute such as `xml:id` or `wsu:Id`.

```

833 <!-- PMUpdateResponse - response to the PMUpdate request -->
834
835 <xs:element name="PMUpdateResponse" type="PMUpdateResponseType"/>
836
837 <xs:complexType name="PMUpdateResponseType">
838   <xs:complexContent>
839     <xs:extension base="ResponseAbstractType" />
840   </xs:complexContent>
841 </xs:complexType>
842

```

Figure 11. <pmm:PMUpdateResponse> — Schema Fragment

844 An example message body containing a <pmm:PMUpdateResponse> message follows. This is a successful
 845 response.

```

846 <pmm:PMUpdateResponse>
847   <lu:Status code="OK" />
848 </pmm:PMUpdateResponse>
849

```

Example 10. Example <pmm:PMUpdateResponse> Message

851 If delayed notifications are in effect for the update, the PMM would need to subsequently send a notification message
 852 with the completion status for the update. The example below shows such a notification.

```

853 <dp:Notification ref="...messageID-of-request...">
854   <pmm:PMUpdateResponse>
855     <lu:Status code="OK" />
856   </pmm:PMUpdateResponse>
857 </dp:Notification>
858
859

```

Example 11. Example <pmm:PMUpdateResponse> Message in a notification

3.9.4. PMUpdate Processing Rules

- This operation adopts the Delayed Notification design pattern (see [LibertyDP]) and incorporates all of the associated processing rules. Delayed notifications are needed for delayed operations (where the `at` attribute is used with a time in the future).

Delayed Notifications SHOULD NOT to be used for immediate operations (no future `at` attribute). In such cases, the PMM SHOULD wait for the completion of the request to return the appropriate status in its response to the invoker.

If any portion of the request IS NOT an immediate operation and the `<dp:NotifyTo>` element IS NOT present on the request, the PMM should validate the request to the extent possible and respond with the results of that validation. If that validation succeeded, the PMM MUST continue with the execution of the operation. In such cases the actual completion status of the operation may be different from what was reported (e.g. the request may fail at the time it is attempted for some reason), but that will not be known to the invoker given that they have chosen to not provide a delayed notification endpoint.

If any portion of the request IS NOT an immediate operation and the `<dp:NotifyTo>` element IS present on the request, the PMM MUST report the final completion status (e.g. wait for the completion status of any delayed or indirect operation at the PMM) of the request. If multiple operations are included in the request, the ProvS MAY group the results of some or all of the operations in a single delayed notification message.

In the case of a mixed request, the immediate portion of the request SHOULD be handled as an immediate request and the actual results returned with the initial response.

- 880 • If a specified PMID is not known to the PMM, the PMM MUST treat that request item as a failure. If detailed
881 status codes are being included in the response, the detailed status code for this error MUST be *"NotFound."*
- 882 • Requests that include invalid parameters, such as an invalid or missing `type` attribute, MUST result in a failure. If
883 detailed status codes are being included in the response, the detailed status code for this error MUST be *"Invalid."*
- 884 • If the update is to take place immediately (e.g., there was no `at` attribute specified), the PMM SHOULD wait until
885 the update process is complete prior to sending the response to this request.
- 886 • A cancellation request for a PM which has NO pending update scheduled to take place at the time specified in the
887 `at` attribute MUST result in a failure. If detailed status codes are included in the response, the detailed status code
888 for this case MUST be *"NotFound."*
- 889 • If a pending update is canceled (via a `<pmm:PMUpdate>` with the exact same `at` attribute value and an update type
890 of `"urn:liberty:prov:2007-09:ut:cancel"`) and delayed notification is in effect for the update, the completion status
891 included in the notification MUST indicate a failure. If detailed status codes are included in the notification, the
892 detailed status code for this case MUST be *"Canceled."*
- 893 • If this operation is to take place at some point in the future, the PMM SHOULD, to the extent possible, validate
894 the parameters of the request to detect any errors. Any such errors detected MUST be reported on the response to
895 this request.
- 896 • If an update request includes any `activate`, `activateAt`, or `deactivateAt` attributes within the
897 `<PMDescriptor>`, these attributes SHOULD be ignored. The update request cannot be used to change
898 the activation status of a PMD.
- 899 • The behavior of the PMM when multiple pending occurrences of the same operation are requested (such as multiple
900 future updates) is NOT defined by this specification. It is recommended that callers NOT attempt such operations.
901 If an implementation of a PMM chooses to refuse such duplicated operations and treat them as a failure, the
902 detailed status code, if present, MUST be *"Duplicate."*
- 903 • If **all** items in the request have the same completion status, the top level status MUST reflect that completion status
904 and MUST be *"OK"*, *"Failed"*, or *"WillNotify"*. Otherwise, if the results were mixed, the top-level status MUST
905 be *Partial*. and a second-level status MUST be included indicating which items succeeded, which failed, and
906 which will be subject to delayed notification. The second level status elements MUST include the `ref` attribute
907 containing the `itemID` value for the item. For failures, the second-level status codes MAY simply be *"Failed"*, or
908 they may indicate with more detail the reason for the failure.
- 909 If the top-level status is *"Failed"*, second level status codes MAY be present which contain detailed error
910 information if the PMM wants to share that information with the invoking party.

911 **3.10. Operation: *PMGetStatus***

912 The *PMGetStatus* operation is used by a provisioning entity to get the current status of a PM at the PMM.

913 **3.10.1. `wsa:Action` values for *PMGetStatus* Messages**

914 `<PMGetStatus>` request messages MUST include a `<wsa:Action>` SOAP header with the value of
915 `"urn:liberty:pmm:2007-09:PMGetStatus."`

916 `<PMGetStatusResponse>` messages MUST include a `<wsa:Action>` SOAP header with the value of
917 `"urn:liberty:pmm:2007-09:PMGetStatusResponse."`

918 **3.10.2. *PMGetStatus* Message**

919 The `<PMGetStatus>` request is called by a provisioning entity (such as the Liberty ID-WSF Provisioning Service
920 - see [[LibertyPROV](#)]) to obtain the current status of one or more PM(s) at the PMM.

921 The `<pmm:PMGetStatus>` request contains the following attributes and/or elements:

922 • `<prov:PMID>` [**Optional**] - zero or more PM identifiers for the PMs whose status is being requested.

923 If no PMs are identified, the request is for the status of all PMs managed by the invoker (typically the ProvS from
924 which the PM was provisioned).

925 • `anyAttribute` [**Optional**] - zero or more attributes from a namespace other than that of this specification. One
926 such possibility is an `xs:ID` type attribute such as `xml:id` or `wsu:Id`.

927 The schema for the `<pmm:PMGetStatus>` is shown below.

```
928 <!-- PMGetStatus - to check the provisioning status of a PM at the PMM -->
929
930 <xs:element name="PMGetStatus" type="PMGetStatusType"/>
931
932 <xs:complexType name="PMGetStatusType">
933   <xs:complexContent>
934     <xs:extension base="RequestAbstractType">
935       <xs:sequence>
936         <xs:element ref="prov:PMID" minOccurs="0" maxOccurs="unbounded" />
937       </xs:sequence>
938     </xs:extension>
939   </xs:complexContent>
940 </xs:complexType>
941
```

942 **Figure 12. `<pmm:PMGetStatus>` — Schema Fragment**

943 An example message body containing a `<pmm:PMGetStatus>` message follows. This requests the status of a single
944 PM.

```
945 <pmm:PMGetStatus>
946   <prov:PMID issuer="http://provs-r-us.com">uuid:239032-230328-92379-2397923</prov:PMID>
947 </pmm:PMGetStatus>
948
```

949 **Example 12. Example `<pmm:PMGetStatus>` Message**

950 **3.10.3. `PMGetStatusResponse` Message**

951 This response to the `<pmm:PMGetStatus>` request contains the following elements:

952 • `<lu:Status>`: [**Required**] - the status of the response. See the processing rules below for more information.

953 • `<prov:PMStatus>`: [**Optional**] - zero or more PM status elements describing the current status of the PM. This
954 element **MUST** be included for any PM that matches the PMID(s) specified in the request.

955 • `anyAttribute` [**Optional**] - zero or more attributes from a namespace other than that of this specification. One
956 such possibility is an `xs:ID` type attribute such as `xml:id` or `wsu:Id`.

```

957 <!-- PMGetStatusResponse - response to the PMGetStatus request -->
958
959 <xs:element name="PMGetStatusResponse" type="PMGetStatusResponseType"/>
960
961 <xs:complexType name="PMGetStatusResponseType">
962   <xs:complexContent>
963     <xs:extension base="ResponseAbstractType">
964       <xs:sequence>
965         <xs:element ref="prov:PMStatus" minOccurs="0" maxOccurs="unbounded" />
966       </xs:sequence>
967     </xs:extension>
968   </xs:complexContent>
969 </xs:complexType>
970

```

971 **Figure 13. <pmm:PMGetStatusResponse> — Schema Fragment**

972 An example message body containing a <pmm:PMGetStatusResponse> message follows. This is a successful
 973 response.

```

974 <pmm:PMGetStatusResponse>
975   <lu:Status code="OK" />
976   <prov:PMStatus>
977     <prov:PMID issuer="http://provs-r-us.com">uid:778349-283920-88379-544 8739</prov:PMID>
978     <prov:State asof="2007-01-14T17:31:11Z">urn:liberty:prov:2007-09:status:Activated</prov:S
979   tate>
980   </prov:PMStatus>
981 </pmm:PMGetStatusResponse>
982

```

983 **Example 13. Example <pmm:PMGetStatusResponse> Message**

984 3.10.4. PMGetStatus Processing Rules

- 985 • The invoking party SHOULD specify a value for the <PMID> elements that match the value of existing, provisioned
 986 PMs. If this is not the case, the PMM MUST ignore such values and treat the request as if they were not present
 987 on the request.
- 988 • When this interface is used by a party other than the PMD issuing authority or the ProvS which provisioned the
 989 PM, the PMM MUST treat the request as if the PM does not exist.
- 990 • If there are NO PMIDs specified in the request, the ProvS MUST return the current status of all PMs that have
 991 been registered by the invoker.
- 992 This rule does NOT apply if PMIDs are specified which refer to nonexistent PMs.
- 993 • If request processing succeeded for **any** of the requested PMs, the top-level status code MUST be *OK*. If the
 994 request processing failed for all PMs, the top-level status code MUST be *Failed*. Partial results, where some
 995 of the items were found and some were not found, are considered a successful response that only include the
 996 <pmm:PMStatus> element(s) for the PMs that were found.
- 997 • If the top-level status is not *OK*, and second level status codes are present, they MAY contain detailed error
 998 information if the PMM wants to share that information with the invoking party.

999 **3.11. Operation: *PMSetStatus***

1000 The *PMSetStatus* operation is used by the PM to set the current status of the PM at the PMM.

1001 **3.11.1. *wsa:Action* values for *PMSetStatus* Messages**

1002 <PMSetStatus> request messages MUST include a <wsa:Action> SOAP header with the value of
1003 "urn:liberty:pmm:2007-09:PMSetStatus."

1004 <PMSetStatusResponse> messages MUST include a <wsa:Action> SOAP header with the value of
1005 "urn:liberty:pmm:2007-09:PMSetStatusResponse."

1006 **3.11.2. *PMSetStatus* Message**

1007 The <PMSetStatus> request is called by a PM to set the current status of the PM at the PMM.

1008 The <pmm:PMSetStatus> request contains the following attributes and/or elements:

- 1009 • <prov:PMStatus> [**Required**] - the updated status information for the PM.
- 1010 • anyAttribute [**Optional**] - zero or more attributes from a namespace other than that of this specification. One
1011 such possibility is an **xs:ID** type attribute such as `xml:id` or `wsu:Id`.

1012 The schema for the <pmm:PMSetStatus> is shown below.

```
1013 <!-- PMSetStatus - to update the status of a PM at the PMM -->
1014
1015 <xs:element name="PMSetStatus" type="PMSetStatusType"/>
1016
1017 <xs:complexType name="PMSetStatusType">
1018   <xs:complexContent>
1019     <xs:extension base="RequestAbstractType">
1020       <xs:sequence>
1021         <xs:element ref="prov:PMStatus" />
1022       </xs:sequence>
1023     </xs:extension>
1024   </xs:complexContent>
1025 </xs:complexType>
1026
```

1027 **Figure 14. <pmm:PMSetStatus> — Schema Fragment**

1028 An example message body containing a <pmm:PMSetStatus> message follows.

```
1029 <pmm:PMSetStatus>
1030   <prov:PMStatus>
1031     <prov:PMID issuer="http://provs-r-us.com">uuid:778349-283920-88379-5448739</prov:PMID>
1032     <prov:State>urn:liberty:prov:2007-09:status:Activated</prov:State>
1033   </prov:PMStatus>
1034 </pmm:PMSetStatus>
1035
```

1036 **Example 14. Example <pmm:PMSetStatus> Message**

1037 **3.11.3. *PMSetStatusResponse* Message**

1038 This response to the <pmm:PMSetStatus> request contains the following elements:

- 1039 • `<lu:Status>`: **[Required]** - the status of the response. See the processing rules below for more information.
- 1040 • `anyAttribute` **[Optional]** - zero or more attributes from a namespace other than that of this specification. One
- 1041 such possibility is an **xs:ID** type attribute such as `xml:id` or `wsu:Id`.

```
1042 <!-- PMSetStatusResponse - response to the PMSetStatus request -->
1043
1044 <xs:element name="PMSetStatusResponse" type="PMSetStatusResponseType"/>
1045
1046 <xs:complexType name="PMSetStatusResponseType">
1047   <xs:complexContent>
1048     <xs:extension base="ResponseAbstractType" />
1049   </xs:complexContent>
1050 </xs:complexType>
1051
```

1052 **Figure 15. `<pmm:PMSetStatusResponse>` — Schema Fragment**

1053 An example message body containing a `<pmm:PMSetStatusResponse>` message follows. This is a successful

1054 response.

```
1055 <pmm:PMSetStatusResponse>
1056   <lu:Status code="OK" />
1057 </pmm:PMSetStatusResponse>
1058
```

1059 **Example 15. Example `<pmm:PMSetStatusResponse>` Message**

1060 **3.11.4. PMSetStatus Processing Rules**

- 1061 • The PMM MUST ensure that only the provisioned PM is allowed to use this interface to directly change the status
- 1062 of the PM. Invocations of this interface by any other party MUST result in a failure.
- 1063 If the invoker is the PM issuing authority or the ProvS which provisioned the PM and detailed status codes are
- 1064 being included in the response, the detailed status code for this error MUST be *"Forbidden."*
- 1065 Otherwise, if detailed status codes are being included in the response, the detailed status code for this error MUST
- 1066 be *"NotFound."*
- 1067 • If a specified PMID is not known to the PMM, the PMM MUST treat that request item as a failure. If detailed
- 1068 status codes are being included in the response, the detailed status code for this error MUST be *"NotFound."*
- 1069 • If the `<State>` element within the `<PMStatus>` contains an `asof` attribute, the PMM MUST ignore this value
- 1070 and, instead, use its understanding of the current time to record as the status time.
- 1071 • If the `<State>` element within the `<PMStatus>` contains a value that is not understood by the PMM, the PMM
- 1072 MAY treat the operation as a failure. PMMs MUST accept all of the Liberty defined status values specified in
- 1073 [\[LibertyPROV\]](#).
- 1074 If the request is to be treated as a failure for this reason and detailed status codes are included in the response, the
- 1075 detailed status code for this error MUST be *"Invalid."*
- 1076 • If request processing succeeded, the top-level status code MUST be *OK*. If the request processing failed, the
- 1077 top-level status code MUST be *Failed*.
- 1078 • If the top-level status is not *OK*, and second level status codes are present, they MAY contain detailed error
- 1079 information if the PMM wants to share that information with the invoking party.

1080 4. Provisioned Module Manager Service Schema

```
1081 <?xml version="1.0" encoding="UTF-8"?>
1082 <xs:schema targetNamespace="urn:liberty:pmm:2007-09"
1083   xmlns:lu="urn:liberty:util:2006-08"
1084   xmlns:prov="urn:liberty:prov:2007-09"
1085   xmlns:dp="urn:liberty:dp:2007-09"
1086   xmlns:xs="http://www.w3.org/2001/XMLSchema"
1087   xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
1088   xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
1089   xmlns:wsa="http://www.w3.org/2005/08/addressing"
1090   xmlns="urn:liberty:pmm:2007-09"
1091   elementFormDefault="qualified"
1092   attributeFormDefault="unqualified"
1093 >
1094
1095 <xs:import namespace="urn:liberty:util:2006-08"
1096   schemaLocation="liberty-idwsf-utility-v2.0.xsd"/>
1097
1098 <xs:import namespace="urn:liberty:prov:2007-09"
1099   schemaLocation="liberty-idwsf-prov-v1.0.xsd"/>
1100
1101 <xs:import namespace="urn:liberty:dp:2007-09"
1102   schemaLocation="liberty-idwsf-dp-v1.0.xsd"/>
1103
1104 <!-- RequestAbstractType - common request message structure -->
1105
1106 <xs:complexType name="RequestAbstractType" abstract="true">
1107   <xs:anyAttribute namespace="##other" processContents="lax"/>
1108 </xs:complexType>
1109
1110
1111 <!-- ResponseAbstractType - common message response structure -->
1112
1113 <xs:complexType name="ResponseAbstractType" abstract="true">
1114   <xs:sequence>
1115     <xs:element ref="lu:Status"/>
1116   </xs:sequence>
1117   <xs:anyAttribute namespace="##other" processContents="lax"/>
1118 </xs:complexType>
1119
1120
1121 <!-- Provision - to instigate the provisioning of a PM at the PMM -->
1122
1123 <xs:element name="Provision" type="ProvisionType"/>
1124
1125 <xs:complexType name="ProvisionType">
1126   <xs:complexContent>
1127     <xs:extension base="RequestAbstractType">
1128       <xs:sequence>
1129         <xs:choice>
1130           <xs:element ref="prov:ProvisioningHandle" />
1131           <xs:element ref="prov:PMDescriptor" />
1132         </xs:choice>
1133         <xs:element ref="dp:NotifyTo" minOccurs="0" />
1134       </xs:sequence>
1135       <xs:attribute name="wait" type="xs:boolean" use="optional" />
1136     </xs:extension>
1137   </xs:complexContent>
1138 </xs:complexType>
1139
1140
1141 <!-- ProvisionResponse - response to the Provision request -->
1142
1143 <xs:element name="ProvisionResponse" type="ProvisionResponseType"/>
1144
1145 <xs:complexType name="ProvisionResponseType">
```

```
1146     <xs:complexContent>
1147     <xs:extension base="ResponseAbstractType" />
1148     </xs:complexContent>
1149 </xs:complexType>
1150
1151
1152 <!-- PMActivate - to activate one or more PM(s) at the PMM -->
1153
1154 <xs:element name="PMActivate" type="PMActivateType"/>
1155
1156 <xs:complexType name="PMActivateType">
1157     <xs:complexContent>
1158     <xs:extension base="RequestAbstractType">
1159     <xs:sequence>
1160     <xs:element ref="PMActivateItem" maxOccurs="unbounded" />
1161     <xs:element ref="dp:NotifyTo" minOccurs="0" />
1162     </xs:sequence>
1163     </xs:extension>
1164     </xs:complexContent>
1165 </xs:complexType>
1166
1167 <xs:element name="PMActivateItem" type="PMActivateItemType" />
1168
1169 <xs:complexType name="PMActivateItemType">
1170     <xs:sequence>
1171     <xs:element ref="prov:PMID" />
1172     </xs:sequence>
1173     <xs:attribute name="itemID" type="xs:string" use="required" />
1174     <xs:attribute name="at" type="xs:dateTime" use="optional" />
1175 </xs:complexType>
1176
1177
1178 <!-- PMActivateResponse - the response to the PMActivate request -->
1179
1180 <xs:element name="PMActivateResponse" type="PMActivateResponseType"/>
1181
1182 <xs:complexType name="PMActivateResponseType">
1183     <xs:complexContent>
1184     <xs:extension base="ResponseAbstractType" />
1185     </xs:complexContent>
1186 </xs:complexType>
1187
1188
1189 <!-- PMDeactivate - to deactivate a PM at the PMM -->
1190
1191 <xs:element name="PMDeactivate" type="PMDeactivateType"/>
1192
1193 <xs:complexType name="PMDeactivateType">
1194     <xs:complexContent>
1195     <xs:extension base="RequestAbstractType">
1196     <xs:sequence>
1197     <xs:element ref="PMDeactivateItem" maxOccurs="unbounded" />
1198     <xs:element ref="dp:NotifyTo" minOccurs="0" />
1199     </xs:sequence>
1200     </xs:extension>
1201     </xs:complexContent>
1202 </xs:complexType>
1203
1204 <xs:element name="PMDeactivateItem" type="PMDeactivateItemType" />
1205
1206 <xs:complexType name="PMDeactivateItemType">
1207     <xs:sequence>
1208     <xs:element ref="prov:PMID" />
1209     </xs:sequence>
1210     <xs:attribute name="itemID" type="xs:string" use="required" />
1211     <xs:attribute name="at" type="xs:dateTime" use="optional" />
1212 </xs:complexType>
```

```
1213
1214
1215 <!-- PMDeactivateResponse - the response to the PMDeactivate request -->
1216
1217 <xs:element name="PMDeactivateResponse" type="PMDeactivateResponseType"/>
1218
1219 <xs:complexType name="PMDeactivateResponseType">
1220   <xs:complexContent>
1221     <xs:extension base="ResponseAbstractType" />
1222   </xs:complexContent>
1223 </xs:complexType>
1224
1225
1226 <!-- PMDelete - to delete a PM at the PMM -->
1227
1228 <xs:element name="PMDelete" type="PMDeleteType"/>
1229
1230 <xs:complexType name="PMDeleteType">
1231   <xs:complexContent>
1232     <xs:extension base="RequestAbstractType">
1233       <xs:sequence>
1234         <xs:element ref="PMDeleteItem" maxOccurs="unbounded" />
1235       </xs:sequence>
1236     </xs:extension>
1237   </xs:complexContent>
1238 </xs:complexType>
1239
1240 <xs:element name="PMDeleteItem" type="PMDeleteItemType" />
1241
1242 <xs:complexType name="PMDeleteItemType">
1243   <xs:sequence>
1244     <xs:element ref="prov:PMID" />
1245   </xs:sequence>
1246   <xs:attribute name="itemID" type="xs:string" use="required" />
1247 </xs:complexType>
1248
1249
1250 <!-- PMDeleteResponse - the response to the PMDelete request -->
1251
1252 <xs:element name="PMDeleteResponse" type="PMDeleteResponseType"/>
1253
1254 <xs:complexType name="PMDeleteResponseType">
1255   <xs:complexContent>
1256     <xs:extension base="ResponseAbstractType" />
1257   </xs:complexContent>
1258 </xs:complexType>
1259
1260
1261 <!-- PMUpdate - update the PM at the PMM -->
1262
1263 <xs:element name="PMUpdate" type="PMUpdateType"/>
1264
1265 <xs:complexType name="PMUpdateType">
1266   <xs:complexContent>
1267     <xs:extension base="RequestAbstractType">
1268       <xs:sequence>
1269         <xs:element ref="PMUpdateItem" maxOccurs="unbounded" />
1270         <xs:element ref="dp:NotifyTo" minOccurs="0" />
1271       </xs:sequence>
1272     </xs:extension>
1273   </xs:complexContent>
1274 </xs:complexType>
1275
1276 <xs:element name="PMUpdateItem" type="PMUpdateItemType" />
1277
1278 <xs:complexType name="PMUpdateItemType">
1279   <xs:sequence>
```

```
1280     <xs:element ref="prov:PMDescriptor" />
1281   </xs:sequence>
1282   <xs:attribute name="type" type="xs:anyURI" use="required" />
1283   <xs:attribute name="itemID" type="xs:string" use="required" />
1284   <xs:attribute name="at" type="xs:dateTime" use="optional" />
1285 </xs:complexType>
1286
1287
1288 <!-- PMUpdateResponse - response to the PMUpdate request -->
1289
1290 <xs:element name="PMUpdateResponse" type="PMUpdateResponseType"/>
1291
1292 <xs:complexType name="PMUpdateResponseType">
1293   <xs:complexContent>
1294     <xs:extension base="ResponseAbstractType" />
1295   </xs:complexContent>
1296 </xs:complexType>
1297
1298
1299 <!-- PMGetStatus - to check the provisioning status of a PM at the PMM -->
1300
1301 <xs:element name="PMGetStatus" type="PMGetStatusType"/>
1302
1303 <xs:complexType name="PMGetStatusType">
1304   <xs:complexContent>
1305     <xs:extension base="RequestAbstractType">
1306       <xs:sequence>
1307         <xs:element ref="prov:PMID" minOccurs="0" maxOccurs="unbounded" />
1308       </xs:sequence>
1309     </xs:extension>
1310   </xs:complexContent>
1311 </xs:complexType>
1312
1313
1314 <!-- PMGetStatusResponse - response to the PMGetStatus request -->
1315
1316 <xs:element name="PMGetStatusResponse" type="PMGetStatusResponseType"/>
1317
1318 <xs:complexType name="PMGetStatusResponseType">
1319   <xs:complexContent>
1320     <xs:extension base="ResponseAbstractType">
1321       <xs:sequence>
1322         <xs:element ref="prov:PMStatus" minOccurs="0" maxOccurs="unbounded" />
1323       </xs:sequence>
1324     </xs:extension>
1325   </xs:complexContent>
1326 </xs:complexType>
1327
1328
1329 <!-- PMSetStatus - to update the status of a PM at the PMM -->
1330
1331 <xs:element name="PMSetStatus" type="PMSetStatusType"/>
1332
1333 <xs:complexType name="PMSetStatusType">
1334   <xs:complexContent>
1335     <xs:extension base="RequestAbstractType">
1336       <xs:sequence>
1337         <xs:element ref="prov:PMStatus" />
1338       </xs:sequence>
1339     </xs:extension>
1340   </xs:complexContent>
1341 </xs:complexType>
1342
1343
1344 <!-- PMSetStatusResponse - response to the PMSetStatus request -->
1345
1346 <xs:element name="PMSetStatusResponse" type="PMSetStatusResponseType"/>
```



```
1347
1348 <xs:complexType name="PMSetStatusResponseType">
1349   <xs:complexContent>
1350     <xs:extension base="ResponseAbstractType" />
1351   </xs:complexContent>
1352 </xs:complexType>
1353
1354
1355 </xs:schema>
1356
```

1357 5. Provisioned Module Manager Service WSDL

```
1358 <?xml version="1.0"?>
1359 <definitions name="pmm-svc"
1360   targetNamespace="urn:liberty:pmm:2007-09"
1361   xmlns:tns="urn:liberty:pmm:2007-09"
1362   xmlns="http://schemas.xmlsoap.org/wsdl/"
1363   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
1364   xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
1365   xmlns:wsaw="http://www.w3.org/2006/02/addressing/wsdl"
1366   xmlns:pmm="urn:liberty:pmm:2007-09"
1367   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
1368   xsi:schemaLocation="http://schemas.xmlsoap.org/wsdl/
1369     http://schemas.xmlsoap.org/wsdl/
1370     http://www.w3.org/2006/02/addressing/wsdl
1371     http://www.w3.org/2006/02/addressing/wsdl/ws-addr-wsdl.xsd">
1372
1373   <types>
1374     <xsd:schema>
1375       <xsd:import namespace="urn:liberty:pmm:2007-09"
1376         schemaLocation="liberty-idwsf-pmm-v1.0.xsd"/>
1377     </xsd:schema>
1378   </types>
1379
1380   <message name="Provision">
1381     <part name="body" element="pmm:Provision"/>
1382   </message>
1383   <message name="ProvisionResponse">
1384     <part name="body" element="pmm:ProvisionResponse"/>
1385   </message>
1386
1387   <message name="PMActivate">
1388     <part name="body" element="pmm:PMActivate"/>
1389   </message>
1390   <message name="PMActivateResponse">
1391     <part name="body" element="pmm:PMActivateResponse"/>
1392   </message>
1393
1394   <message name="PMDeactivate">
1395     <part name="body" element="pmm:PMDeactivate"/>
1396   </message>
1397   <message name="PMDeactivateResponse">
1398     <part name="body" element="pmm:PMDeactivateResponse"/>
1399   </message>
1400
1401   <message name="PMDelete">
1402     <part name="body" element="pmm:PMDelete"/>
1403   </message>
1404   <message name="PMDeleteResponse">
1405     <part name="body" element="pmm:PMDeleteResponse"/>
1406   </message>
1407
1408   <message name="PMGetStatus">
1409     <part name="body" element="pmm:PMGetStatus"/>
1410   </message>
1411   <message name="PMGetStatusResponse">
1412     <part name="body" element="pmm:PMGetStatusResponse"/>
1413   </message>
1414
1415   <message name="PMSetStatus">
1416     <part name="body" element="pmm:PMSetStatus"/>
1417   </message>
1418   <message name="PMSetStatusResponse">
1419     <part name="body" element="pmm:PMSetStatusResponse"/>
1420   </message>
1421
1422   <message name="PMUpdate">
```

```
1423     <part name="body" element="pmm:PMUpdate"/>
1424 </message>
1425 <message name="PMUpdateResponse">
1426   <part name="body" element="pmm:PMUpdateResponse"/>
1427 </message>
1428
1429 <portType name="PMMPort">
1430
1431   <operation name="Provision">
1432     <input message="tns:Provision"
1433       wsaw:Action="urn:liberty:pmm:2007-09:Provision" />
1434     <output message="tns:ProvisionResponse"
1435       wsaw:Action="urn:liberty:pmm:2007-09:ProvisionResponse" />
1436   </operation>
1437
1438   <operation name="PMActivate">
1439     <input message="tns:PMActivate"
1440       wsaw:Action="urn:liberty:pmm:2007-09:PMActivate" />
1441     <output message="tns:PMActivateResponse"
1442       wsaw:Action="urn:liberty:pmm:2007-09:PMActivateResponse" />
1443   </operation>
1444
1445   <operation name="PMDeactivate">
1446     <input message="tns:PMDeactivate"
1447       wsaw:Action="urn:liberty:pmm:2007-09:PMDeactivate" />
1448     <output message="tns:PMDeactivateResponse"
1449       wsaw:Action="urn:liberty:pmm:2007-09:PMDeactivateResponse" />
1450   </operation>
1451
1452   <operation name="PMDelete">
1453     <input message="tns:PMDelete"
1454       wsaw:Action="urn:liberty:pmm:2007-09:PMDelete" />
1455     <output message="tns:PMDeleteResponse"
1456       wsaw:Action="urn:liberty:pmm:2007-09:PMDeleteResponse" />
1457   </operation>
1458
1459   <operation name="PMGetStatus">
1460     <input message="tns:PMGetStatus"
1461       wsaw:Action="urn:liberty:pmm:2007-09:PMGetStatus" />
1462     <output message="tns:PMGetStatusResponse"
1463       wsaw:Action="urn:liberty:pmm:2007-09:PMGetStatusResponse" />
1464   </operation>
1465
1466   <operation name="PMSetStatus">
1467     <input message="tns:PMSetStatus"
1468       wsaw:Action="urn:liberty:pmm:2007-09:PMSetStatus" />
1469     <output message="tns:PMSetStatusResponse"
1470       wsaw:Action="urn:liberty:pmm:2007-09:PMSetStatusResponse" />
1471   </operation>
1472
1473   <operation name="PMUpdate">
1474     <input message="tns:PMUpdate"
1475       wsaw:Action="urn:liberty:pmm:2007-09:PMUpdate" />
1476     <output message="tns:PMUpdateResponse"
1477       wsaw:Action="urn:liberty:pmm:2007-09:PMUpdateResponse" />
1478   </operation>
1479 </portType>
1480
1481 <!--
1482 An example of a binding and service that can be used with this
1483 abstract service description is pmmid below.
1484 -->
1485
1486 <binding name="PMMPort" type="tns:PMMPort">
1487
1488   <soap:binding style="document"
1489
```

```
1490         transport="http://schemas.xmlsoap.org/soap/http"/>
1491
1492     <operation name="Provision">
1493         <input> <soap:body use="literal"/> </input>
1494         <output> <soap:body use="literal"/> </output>
1495     </operation>
1496
1497     <operation name="PMSetStatus">
1498         <input> <soap:body use="literal"/> </input>
1499         <output> <soap:body use="literal"/> </output>
1500     </operation>
1501
1502     <operation name="PMActivate">
1503         <input> <soap:body use="literal"/> </input>
1504         <output> <soap:body use="literal"/> </output>
1505     </operation>
1506
1507     <operation name="PMDeactivate">
1508         <input> <soap:body use="literal"/> </input>
1509         <output> <soap:body use="literal"/> </output>
1510     </operation>
1511
1512     <operation name="PMDelete">
1513         <input> <soap:body use="literal"/> </input>
1514         <output> <soap:body use="literal"/> </output>
1515     </operation>
1516
1517     <operation name="PMGetStatus">
1518         <input> <soap:body use="literal"/> </input>
1519         <output> <soap:body use="literal"/> </output>
1520     </operation>
1521
1522     <operation name="PMUpdate">
1523         <input> <soap:body use="literal"/> </input>
1524         <output> <soap:body use="literal"/> </output>
1525     </operation>
1526
1527 </binding>
1528
1529 <service name="PMMService">
1530
1531     <port name="PMMPort" binding="tns:PMMSBinding">
1532
1533         <!-- Modify with the REAL SOAP endpoint -->
1534
1535         <soap:address location="http://example.com/pmmisioning"/>
1536
1537     </port>
1538
1539 </service>
1540
1541 </definitions>
1542
```

1543 References

1544 Normative

- 1545 [LibertyPMM] Cahill, Conor P., eds. "Liberty ID-WSF Provisioned Module Manager Service Specification," Version
1546 1.0, Liberty Alliance Project (14 December 2007). <http://www.projectliberty.org/specs>
- 1547 [LibertyPROV] Cahill, Conor P., eds. "Liberty ID-WSF Provisioning Service Specification," Version 1.0, Liberty
1548 Alliance Project (14 December 2007). <http://www.projectliberty.org/specs>
- 1549 [LibertyDP] Cahill, Conor P., eds. "Liberty ID-WSF Design Patterns Specification," Version 1.0, Liberty Alliance
1550 Project (14 December 2007). <http://www.projectliberty.org/specs>
- 1551 [LibertyDisco] Cahill, Conor, Hodges, Jeff, eds. "Liberty ID-WSF Discovery Service Specification," Version 2.0-
1552 errata-v1.0, Liberty Alliance Project (29 November, 2006). <http://www.projectliberty.org/specs>
- 1553 [LibertyIDPP] Kellomäki, Sampo, Lockhart, Rob, eds. "Liberty ID-SIS Personal Profile Service Specification,"
1554 Version 1.1, Liberty Alliance Project (29 September, 2005). <http://www.projectliberty.org/specs>
- 1555 [LibertyInteract] Aarts, Robert, Madsen, Paul, eds. "Liberty ID-WSF Interaction Service Specification," Version 2.0-
1556 errata-v1.0, Liberty Alliance Project (21 April, 2007). <http://www.projectliberty.org/specs>
- 1557 [LibertyProtSchema] Cantor, Scott, Kemp, John, eds. "Liberty ID-FF Protocols and Schema Specification," Version
1558 1.2-errata-v3.0, Liberty Alliance Project (14 December 2004). <http://www.projectliberty.org/specs>
- 1559 [LibertyReg] Kemp, John, eds. "Liberty Enumeration Registry Governance," Version 1.1, Liberty Alliance Project (14
1560 December, 2004). <http://www.projectliberty.org/specs>
- 1561 [LibertySOAPAuthn] Hodges, Jeff, Aarts, Robert, Madsen, Paul, Cantor, Scott, eds. "Liberty ID-WSF Authentication,
1562 Single Sign-On, and Identity Mapping Services Specification," v2.0-errata-1.0-01, Liberty Alliance Project
1563 (28 November, 2006). <http://www.projectliberty.org/specs>
- 1564 [LibertySOAPBinding] Hodges, Jeff, Kemp, John, Aarts, Robert, Whitehead, Greg, Madsen, Paul, eds. "Liberty
1565 ID-WSF SOAP Binding Specification," Version 2.0-errata-v1.0, Liberty Alliance Project (21 April, 2007).
1566 <http://www.projectliberty.org/specs>
- 1567 [RFC2119] S. Bradner "Key words for use in RFCs to Indicate Requirement Levels," RFC 2119, The Internet
1568 Engineering Task Force (March 1997). <http://www.ietf.org/rfc/rfc2119.txt>
- 1569 [RFC2251] "Lightweight Directory Access Protocol (v3)," M. Wahl T. Howes S. Kille (December 1997). RFC 2251,
1570 Internet Engineering Task Force <http://www.ietf.org/rfc/rfc2251.txt>
- 1571 [RFC2252] "Lightweight Directory Access Protocol (v3): Attribute Syntax Definitions," M. Wahl A.
1572 Coulbeck T. Howes S. Kille (December 1997). RFC 2252, Internet Engineering Task Force
1573 <http://www.ietf.org/rfc/rfc2252.txt>
- 1574 [RFC2891] Howes, T., Wahl, M., eds. (August 2000). "LDAP Control Extension for Server Side Sorting of Search
1575 Results," RFC 2891, Internet Engineering Task Force <http://www.ietf.org/rfc/rfc2891.txt>
- 1576 [SAMLCore11] Maler, Eve, Mishra, Prateek, Philpott, Rob, eds. (2 September 2003). "Assertions and Pro-
1577 tocol for the OASIS Security Assertion Markup Language (SAML) V1.1," SAML v1.1, OASIS
1578 Standard, Organization for the Advancement of Structured Information Standards [http://www.oasis-
1579 open.org/committees/download.php/3406/oasis-sstc-saml-core-1.1.pdf](http://www.oasis-open.org/committees/download.php/3406/oasis-sstc-saml-core-1.1.pdf)

- 1580 [SAMLCore2] Cantor, Scott, Kemp, John, Philpott, Rob, Maler, Eve, eds. (15 March 2005). "Assertions
1581 and Protocol for the OASIS Security Assertion Markup Language (SAML) V2.0," SAML V2.0, OA-
1582 SIS Standard, Organization for the Advancement of Structured Information Standards [http://docs.oasis-
open.org/security/saml/v2.0/saml-core-2.0-os.pdf](http://docs.oasis-
1583 open.org/security/saml/v2.0/saml-core-2.0-os.pdf)
- 1584 [SAMLProf2] Hughes, John, Cantor, Scott, Hodges, Jeff, Hirsch, Frederick, Mishra, Prateek, Philpott, Rob, Maler,
1585 Eve, eds. (15 March, 2005). "Profiles for the OASIS Security Assertion Markup Language (SAML) V2.0,"
1586 SAML V2.0, OASIS Standard, Organization for the Advancement of Structured Information Standards
1587 <http://docs.oasis-open.org/security/saml/v2.0/saml-profiles-2.0-os.pdf>
- 1588 [Schema1-2] Thompson, Henry S., Beech, David, Maloney, Murray, Mendelsohn, Noah, eds. (28 October
1589 2004). "XML Schema Part 1: Structures Second Edition," Recommendation, World Wide Web Consortium
1590 <http://www.w3.org/TR/xmlschema-1/>
- 1591 [SOAPv1.1] "Simple Object Access Protocol (SOAP) 1.1," Box, Don, Ehnebuske, David , Kakivaya, Gopal, Layman,
1592 Andrew, Mendelsohn, Noah, Nielsen, Henrik Frystyk, Winer, Dave, eds. World Wide Web Consortium W3C
1593 Note (08 May 2000). <http://www.w3.org/TR/2000/NOTE-SOAP-20000508/>
- 1594 [XML] Bray, Tim, Paoli, Jean, Sperberg-McQueen, C. M., Maler, Eve, Yergeau, Francois, eds. (04 February 2004).
1595 "Extensible Markup Language (XML) 1.0 (Third Edition)," Recommendation, World Wide Web Consortium
1596 <http://www.w3.org/TR/2004/REC-xml-20040204>
- 1597 [XMLDsig] Eastlake, Donald, Reagle, Joseph, Solo, David, eds. (12 Feb 2002). "XML-Signature Syntax and
1598 Processing," Recommendation, World Wide Web Consortium <http://www.w3.org/TR/xmlsig-core>
- 1599 [XPath] Clark , J., DeRose , S., eds. (16 November 1999). "XML Path Language (XPath) Version 1.0 ,"
1600 Recommendation, W3C <http://www.w3.org/TR/xpath> [August 2003].

1601 Informative

- 1602 [LibertyACT] Cahill, Conor P., eds. "Liberty Advanced Client Technologies," Version 1.0, Liberty Alliance Project
1603 (14 December 2007). <http://www.projectliberty.org/specs>
- 1604 [LibertyIDWSFGuide] Weitzel, David, eds. "Liberty ID-WSF Implementation Guide," Version 2.0-02, Liberty
1605 Alliance Project (13 January, 2005). <http://www.projectliberty.org/specs>
- 1606 [LibertyIDPPGuide] Kellomäki, Sampo, Lockhart, Rob, eds. "Liberty ID-SIS Personal Profile Service Implementation
1607 Guidelines," Version 1.1, Liberty Alliance Project (29 September, 2005). <http://www.projectliberty.org/specs>
- 1608 [LibertyIDWSFOverview] Tourzan, Jonathan, Koga, Yuzo, eds. "Liberty ID-WSF Web Services Framework
1609 Overview," Version 2.0, Liberty Alliance Project (30 July, 2006). <http://www.projectliberty.org/specs>