



Liberty ID-WSF Provisioning Service Specification

Version: 1.0

Editors:

Conor P. Cahill, Intel Corporation

Contributors:

Hubert Le Van Gong, Sun

Paul Madsen, NTT

Hiroyoshi Takiguchi, NTT

Greg Whitehead, HP

Janice Yang, Intel Corporation

Abstract:

This specification defines the Provisioning Service interfaces.

Filename: liberty-idwsf-prov-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	5
43	1.1. Notation and Conventions	5
44	1.1.1. XML Namespaces	5
45	2. Overview	6
46	2.1. Provisioning Components	6
47	3. Data Definitions	8
48	3.1. Provisioned Module Identifier	8
49	3.2. ProvisioningHandle	8
50	3.3. PMDescriptor	9
51	3.4. PMStatus	12
52	3.5. PMEInfo	13
53	3.6. Callback EPR	15
54	4. Provisioning Service (ProvS)	16
55	4.1. Service URIs	16
56	4.2. Status Codes	17
57	4.3. Request and Response Abstract Types	18
58	4.3.1. Complex Type RequestAbstractType	18
59	4.3.2. Complex Type ResponseAbstractType	18
60	4.4. Operation: <i>PMGetDescriptor</i>	18
61	4.4.1. wsa:Action values for <i>PMGetDescriptor</i> Messages	18
62	4.4.2. <i>PMGetDescriptor</i> Message	18
63	4.4.3. <i>PMGetDescriptorResponse</i> Message	19
64	4.4.4. <i>PMGetDescriptor</i> Processing Rules	20
65	4.5. Operation: <i>PMActivate</i>	21
66	4.5.1. wsa:Action values for <i>PMActivate</i> Messages	21
67	4.5.2. <i>PMActivate</i> Message	21
68	4.5.3. <i>PMActivateResponse</i> Message	22
69	4.5.4. <i>PMActivate</i> Processing Rules	23
70	4.6. Operation: <i>PMDeactivate</i>	24
71	4.6.1. wsa:Action values for <i>PMDeactivate</i> Messages	24
72	4.6.2. <i>PMDeactivate</i> Message	24
73	4.6.3. <i>PMDeactivateResponse</i> Message	26
74	4.6.4. <i>PMDeactivate</i> Processing Rules	26
75	4.7. Operation: <i>PMDelete</i>	27
76	4.7.1. wsa:Action values for <i>PMDelete</i> Messages	27
77	4.7.2. <i>PMDelete</i> Message	27
78	4.7.3. <i>PMDeleteResponse</i> Message	28
79	4.7.4. <i>PMDelete</i> Processing Rules	29
80	4.8. Operation: <i>PMGetStatus</i>	30
81	4.8.1. wsa:Action values for <i>PMGetStatus</i> Messages	30
82	4.8.2. <i>PMGetStatus</i> Message	30
83	4.8.3. <i>PMGetStatusResponse</i> Message	31
84	4.8.4. <i>PMGetStatus</i> Processing Rules	32
85	4.9. Operation: <i>PMRegisterDescriptor</i>	32
86	4.9.1. wsa:Action values for <i>PMRegisterDescriptor</i> Messages	32
87	4.9.2. <i>PMRegisterDescriptor</i> Message	33
88	4.9.3. <i>PMRegisterDescriptorResponse</i> Message	34
89	4.9.4. <i>PMRegisterDescriptor</i> Processing Rules	35
90	4.10. Operation: <i>PMSetStatus</i>	35
91	4.10.1. wsa:Action values for <i>PMSetStatus</i> Messages	36
92	4.10.2. <i>PMSetStatus</i> Message	36
93	4.10.3. <i>PMSetStatusResponse</i> Message	36

94	4.10.4. PMSetStatus Processing Rules	37
95	4.11. Operation: <i>PMUpdate</i>	37
96	4.11.1. wsa:Action values for <i>PMUpdate</i> Messages	38
97	4.11.2. <i>PMUpdate</i> Message	38
98	4.11.3. <i>PMUpdateResponse</i> Message	40
99	4.11.4. <i>PMUpdate</i> Processing Rules	40
100	4.12. Operation: <i>PMDelete</i>	41
101	4.12.1. wsa:Action values for <i>PMDelete</i> Messages	42
102	4.12.2. <i>PMDelete</i> Message	42
103	4.12.3. <i>PMDeleteResponse</i> Message	42
104	4.12.4. <i>PMDelete</i> Processing Rules	43
105	4.13. Operation: <i>PMDisable</i>	43
106	4.13.1. wsa:Action values for <i>PMDisable</i> Messages	43
107	4.13.2. <i>PMDisable</i> Message	43
108	4.13.3. <i>PMDisableResponse</i> Message	44
109	4.13.4. <i>PMDisable</i> Processing Rules	45
110	4.14. Operation: <i>PMEDownload</i>	45
111	4.14.1. wsa:Action values for <i>PMEDownload</i> Messages	45
112	4.14.2. <i>PMEDownload</i> Message	46
113	4.14.3. <i>PMEDownloadResponse</i> Message	47
114	4.14.4. <i>PMEDownload</i> Processing Rules	48
115	4.15. Operation: <i>PMEnable</i>	48
116	4.15.1. wsa:Action values for <i>PMEnable</i> Messages	48
117	4.15.2. <i>PMEnable</i> Message	48
118	4.15.3. <i>PMEnableResponse</i> Message	49
119	4.15.4. <i>PMEnable</i> Processing Rules	50
120	4.16. Operation: <i>PMGetInfo</i>	51
121	4.16.1. wsa:Action values for <i>PMGetInfo</i> Messages	51
122	4.16.2. <i>PMGetInfo</i> Message	51
123	4.16.3. <i>PMGetInfoResponse</i> Message	51
124	4.16.4. <i>PMGetInfo</i> Processing Rules	52
125	4.17. Operation: <i>PMERegister</i>	53
126	4.17.1. wsa:Action values for <i>PMERegister</i> Messages	53
127	4.17.2. <i>PMERegister</i> Message	53
128	4.17.3. <i>PMERegisterResponse</i> Message	54
129	4.17.4. <i>PMERegister</i> Processing Rules	55
130	4.18. Operation: <i>PMEUpload</i>	55
131	4.18.1. wsa:Action values for <i>PMEUpload</i> Messages	55
132	4.18.2. <i>PMEUpload</i> Message	55
133	4.18.3. <i>PMEUploadResponse</i> Message	57
134	4.18.4. <i>PMEUpload</i> Processing Rules	58
135	4.19. Operation: <i>Poll</i>	59
136	4.19.1. wsa:Action values for <i>Poll</i> Messages	59
137	4.19.2. <i>Poll</i> Message	59
138	4.19.3. <i>PollResponse</i> Message	60
139	4.19.4. <i>Poll</i> Processing Rules	61
140	4.20. Operation: <i>UpdateEPR</i>	61
141	4.20.1. wsa:Action values for <i>UpdateEPR</i> Messages	61
142	4.20.2. <i>UpdateEPR</i> Message	61
143	4.20.3. <i>UpdateEPRResponse</i> Message	62
144	4.20.4. <i>UpdateEPR</i> Processing Rules	63
145	5. Provisioning Service Schema	65
146	6. Provisioning Service WSDL	77
147	References	83

148 **1. Introduction**

149 Provisioning, in this context, refers to the distribution, installation and maintenance (update/delete) of some functional
150 module (perhaps a TM) onto a device or platform. The specific capabilities and features of a particular functional
151 module are out of scope here. This process is only concerned with getting the functional module up and running
152 within the target environment.

153 **1.1. Notation and Conventions**

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

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

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

161 **1.1.1. XML Namespaces**

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

163 • The prefix *prov*: represents the Provisioning Service namespace. This namespace is the default for instance
164 fragments, type names, and element names in this document. In schema listings, and in examples of service
165 messages and fragments thereof, this is the default namespace *when* no prefix is shown:

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

167 • The prefix *pmm*: stands for the Liberty ID-WSF Provisioned Module Manager namespace [\[LibertyPMM\]](#):

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

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

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

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

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

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

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

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

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

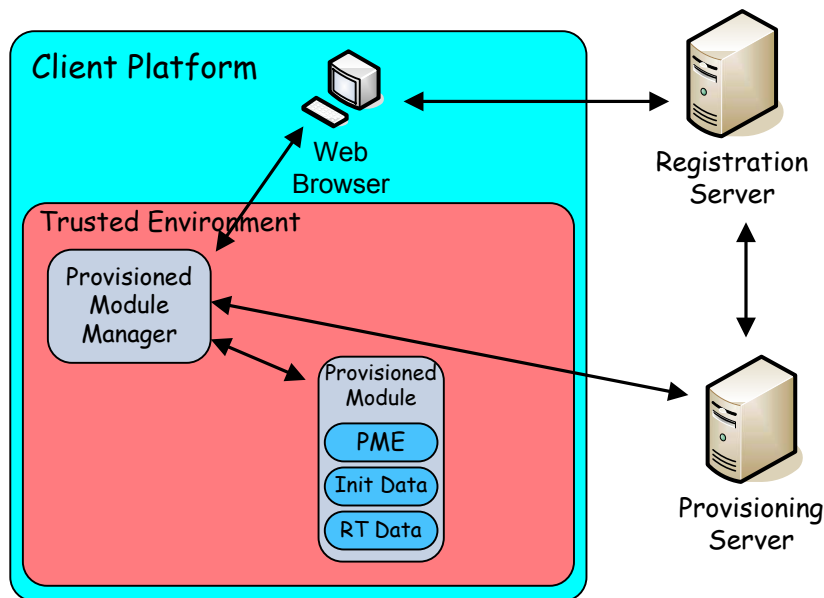
177 2. Overview

178 The Liberty ID-WSF Advanced Client Technologies Overview [[LibertyACT](#)] presents a complete overview of the
179 provisioning process. The reader is strongly encouraged to read through that document (at least the provisioning
180 section) prior to reading this document.

181 This document describes the Liberty ID-WSF Provisioning Service (ProvS). The ProvS is the entity which distributes
182 the data and potentially executable code, to the client platforms. The ProvS also provides a control point for lifecycle
183 management of provisioned modules (PMs), supporting operations such as update, delete, activate and deactivate.

184 2.1. Provisioning Components

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



186

187 **Figure 1. Provisioning Components**

188 Things to note about this diagram:

- 189 • It is not drawn to any form of scale!
- 190 • The client platform represents any type of client, such as a personal computer, a device, a smart card, etc..
- 191 • The trusted environment represents some form of tamper resistant container (thus providing a level of trust for
192 the provisioned components). The trusted environment is **not** a requirement of these protocols – the components
193 shown within the trusted environment could very well exist directly within the relatively untrusted client platform
194 (e.g. the Provisioned Module Manager could run as a service within the Client Platform operating system).
- 195 • The Provisioned Module Manager (PMM) is a service running on the client platform which provides a beach
196 head for provisioning operations. The PMM exposes the interfaces documented within the Liberty ID-WSF
197 Provisioned Module Manager Service Specification [[LibertyPMM](#)].

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

- 201 • The Provisioned Module (PM) is a component which performs some set of functionality. For example, a PM
202 could be a TM (a module which provides IdP extension functionality). PMs may also expose functionality that is
203 not defined by Liberty specifications.
- 204 Each PM is identified using a globally unique identifier called the Provisioned Module Identifier (PMID). The
205 PMID is used to reference to specific instances of a PM when performing tasks like status updates or module
206 updates.
- 207 The PM is shown as being composed of 3 distinct parts:
- 208 • **Provisioned Module Engine (PME)** - the executable code which provides the functionality for the PM.
- 209 This is defined as a separate component here to enable a provisioning process which allows the PME to preexist
210 in the client platform and so just delivers the data necessary to instantiate the PM using that preexisting engine.
211 Of course, the PME may not preexist and in such cases the PMM will have to retrieve it.
- 212 During provisioning, the PME is passed by reference (name) so that the PMM can determine whether or not
213 the PME already exists (either because it was pre-installed or because the same PME has been previously
214 provisioned). Should the PMM need to obtain the PME, the passed in reference is used to identify the PME
215 being downloaded.
- 216 • **Initialization Data (PMInitData)** - the data needed by the PME in order to initialize a new instance of a
217 PM. This may be the actual data needed by the PME or it may be a reference that the PME knows how
218 to dereference and obtain the initialization data at runtime. This data may or may not be needed during the
219 provisioning process. Some PMs are fully individualized and have their PMInitData built in.
- 220 The format and structure of the PMInitData is out of scope for this document and is specific to the PME. It is
221 up to the Provisioning Service to resolve what data is needed for what PME. The PMM treats PMInitData as
222 an opaque data set that it passes to the PME upon initialization.
- 223 • **Runtime Data (PMRTData)** - the runtime data created/managed by the PM instance as it performs its tasks.
224 This would include things like MINGs for a TM that is minting assertions, private keys, etc. This is defined
225 separate from the InitData to allow for PM portability (where a previously activated PM is moved to another
226 client platform).
- 227 • The Web Browser in this diagram represents an enhanced browser (either directly or via a plug-in) with support
228 for the provisioning process. In other provisioning use cases this may be an application or even be the PMM itself
229 can instigate a new provision operation (typically via some direct interaction with the user).
- 230 • The Registration Server (RegS) is not a Liberty defined entity, but rather a deployment component for a particular
231 set of use cases. In this use case, the RegS interacts with the user through a web browser and then controls the
232 provisioning process using the interfaces on the Provisioning Server.
- 233 • The Provisioning Server (ProvS) is typically a network hosted service that is the primary entity with which the
234 PMM interacts. This server is an instance of a Liberty ID-WSF Provisioning Service (see [[LibertyPROV](#)]).
- 235 The primary function of the ProvS is to provide a trusted endpoint for the management and distribution of PMs.

236 3. Data Definitions

237 3.1. Provisioned Module Identifier

238 A `<prov:PMID>` is a unique identifier assigned to the PM when the PMD is first registered with the ProvS (i.e., before
239 the PM has been provisioned). This identifier is used on all subsequent communications with the ProvS relating to
240 that PM from the registering party as well as from the PMM. Therefore the value selected for the PMID MUST be
241 unique across all such values issued by that issuing authority.

242 The `<prov:PMID>` element carries the identifier value as its content and has no other sub-elements. The following
243 attributes are defined:

- 244 • `issuer` [**Required**] - the Provider ID of the party which issued the PMID. This is used to prevent collision of
245 PMID values for PMs issued by different parties.

246 The following schema fragment defines the `<prov:PMID>` element:

```
247  
248 <!-- PMID - the Provisioned Module IDentifier -->  
249  
250 <xs:element name="PMID" type="PMIDType" />  
251 <xs:complexType name="PMIDType">  
252   <xs:simpleContent>  
253     <xs:extension base="xs:anyURI">  
254       <xs:attribute name="issuer" type="xs:anyURI" use="required"/>  
255     </xs:extension>  
256   </xs:simpleContent>  
257 </xs:complexType>  
258
```

259 The following is an example of a PMID element.

```
260 <prov:PMID issuer="http://provs-r-us.com">uuid:239032-230 328-92379-2397923</prov:PMID>  
261
```

262 3.2. ProvisioningHandle

263 A Provisioning Handle provides the PMM with the necessary information to invoke the ProvS and an artifact which
264 refers to a specific PM that is to be provisioned. This is typically handed to the PMM through some non-liberty
265 protocols (such as distribution via email or by being downloaded by the principal).

266 The `<prov:ProvisioningHandle>` element has the following elements/attributes:

- 267 • `<prov:PMDArtifact>` [**Required**] - a token issued by the ProvS which identifies a specific PM that is to be
268 provisioned to the Advanced client.

269 The contents and structure of the PMDArtifact are not defined by this specification and should be treated as an
270 opaque blob by the PMM and any intermediary handlers.

271 Any entity that may pass along or process the PH MUST support a `<PMDArtifact>` length of 2048 bytes. The
272 ProvS, when creating the `<PMDArtifact>` MUST ensure it fits within 2048 bytes.

273 The ProvS should ensure that only appropriate PMMs are able to present the PMDArtifact and it can only be done
274 once (in order to ensure that the Provisioning Handle hasn't been intercepted and used by another party without
275 at least recognizing this when the intended party tries to use it).

- 276 • <prov:ProvisioningServiceEPR> **[optional]** - zero or more ID-WSF EPR(s) describing the endpoint for the
277 entity that is provisioning this PM.
- 278 Normally, the EPR(s) are specified within the PH. However, there MAY be some environments where the PMM
279 has preexisting knowledge of the ProvS EPR and so this element is listed as optional to allow it to be not specified
280 in such cases.
- 281 • <ds:Signature> **[Optional]** - an XML digital signature ([XMLDsig](#)) covering the entire descriptor.
- 282 If a signature is present, the signer MUST follow the normative rules laid out in Section 5 ("SAML and XML
283 Signature Syntax and Processing") of the SAML 2.0 Core Specification ([SAMLCore2](#)).
- 284 • expires **[Optional]** - an optional attribute containing the time at which the data within this handle is deemed to
285 be no longer usable. Artifacts presented after the expiration of this time will not be resolvable.
- 286 • anyAttribute **[Optional]** - zero or more attributes from a namespace other than that of this specification. One
287 such possibility is an **xs:ID** type attribute such as `xml:id` or `wsu:Id`.

288 The following schema fragment defines the <prov:ProvisioningHandle> element:

```
289 <!-- ProvisioningHandle - Info for PMM to initiate provisioning process -->
290
291 <xs:element name="ProvisioningHandle" type="ProvisioningHandleType"/>
292 <xs:complexType name="ProvisioningHandleType">
293   <xs:sequence>
294     <xs:element ref="PMDArtifact" />
295     <xs:element ref="ProvisioningServiceEPR" minOccurs="0"
296               maxOccurs="unbounded" />
297     <xs:element ref="ds:Signature" minOccurs="0"/>
298   </xs:sequence>
299   <xs:attribute name="expires" use="optional" type="xs:dateTime"/>
300   <xs:anyAttribute namespace="##other" processContents="lax"/>
301 </xs:complexType>
302
303 <xs:element name="ProvisioningServiceEPR" type="wsa:EndpointReferenceType"/>
304
305 <xs:element name="PMDArtifact" type="xs:string" />
306
```

307 The following is an example of a <prov:ProvisioningHandle> element.

```
308 <prov:ProvisioningHandle xs:id="2302384823023">
309   <prov:PMDArtifact>23asdfhoi323hposdf923h9sdfhweorh2398asdfjweoiha</prov:PMDArtifact>
310   <prov:ProvisioningServiceEPR>
311     <wsa:Address>http://provision.idpsRus.com</wsa:Address>
312     <wsa:Metadata>
313       <ds:Abstract>Provisioning Service</ds:Abstract>
314       <ds:ProviderID>http://provisioning-provider.idpsRus.com</ds:ProviderID>
315       <ds:ServiceType>urn:liberty:prov:2007-09</ds:ServiceType>
316       <ds:Framework version="2.0" />
317       <ds:SecurityContext>
318         <ds:SecurityMechID>
319           urn:liberty:security:2005-02:TLS:SAMLV2
320         </ds:SecurityMechID>
321         <sec:Token ref="urn:liberty:disco:tokenref:ObtainFromIDP" />
322       </ds:SecurityContext>
323     </wsa:Metadata>
324   </prov:ProvisioningServiceEPR>
325   <ds:Signature>
326     ... signature info here ..
327   </ds:Signature>
328 </prov:ProvisioningHandle>
329
```

330 3.3. PMDescriptor

331 The `<prov:PMDescriptor>`, or Provisioned Module Descriptor (PMD), contains the information necessary for the
332 PMM to instantiate a PM within the Advanced Client.

333 The `<prov:PMDescriptor>` is typically acquired by the PMM following the dereference of the `<PMDArtifact>` at
334 the ProvS.

335 The `<prov:PMDescriptor>` element has the following elements/attributes:

336 • `<ds:Signature>` [**Optional**] - an XML digital signature ([XMLDsig](#)) covering the entire descriptor.

337 If a signature is present, the signer **MUST** follow the normative rules laid out in Section 5 ("SAML and XML
338 Signature Syntax and Processing") of the SAML 2.0 Core Specification ([SAMLCore2](#)).

339 • `<prov:PMID>` [**Required**] - the Provisioned Module Identifier for the PM described by this descriptor.

340 • `<prov:PMEngineRef>` [**Optional**] - a URI reference to a PMEngine. This engine may already exist on the
341 Advanced Client, or it may need to be downloaded and installed by the PMM. This element **MUST** be present in
342 any `<prov:PMDescriptor>` that is describing a complete PM. However, the element **MAY** be absent when the
343 `<prov:PMDescriptor>` is used to describe an update.

344 • `<prov:PMEngine>` [**Optional**] - the complete description and contents of the PM engine. Normally, due to size
345 and processing constraints, the engine itself is not included in the `<prov:PMDescriptor>`. However, when that
346 isn't the case, the executable can be included in this element. If this element is present, the `<prov:PMEngineRef>`
347 element **MUST NOT** be specified.

348 This element contains the following sub-elements:

349 • `<PMEInfo>` [**Required**] - the information about the engine. For a complete description of the contents of this
350 element see [Section 3.5](#).

351 • `<PMEBytes>` [**Required**] - the base64 encoded bytes of the engine (the actual bits of the executable code for
352 the engine).

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

355 • `<prov:PMInitData>` [**Optional**] - initialization data needed by the PMEngine to initialize as a PM instance.
356 The structure and content of this element are defined by, and specific to, the PM being provisioned. In many cases,
357 this will contain an encrypted element containing the initialization data, but many other solutions could be used
358 including a base64 element, an artifact, etc.

359 • `<prov:PMRTData>` [**Optional**] - runtime data saved by a previous instance of the PM associated with this
360 descriptor. Like the `<prov:PMInitData>` element, the content and structure of this element is defined by,
361 and specific to the PM being provisioned.

362 • `activate` [**optional**] - a boolean attribute indicating whether the PM should be activated upon installation. If
363 this attribute is not specified or its value is *true*, the PM is activated. Otherwise the PM is installed, but not
364 activated.

365 See the related `activateAt` and `deactivateAt` attributes below for automated future activation/deactivation.

366 • `activateAt` [**optional**] - an attribute indicating when the PM is to be activated. The value of this attribute is a
367 specific instance in time at which the PM should be activated (functionality enabled).

368 This is typically used when a PM is to be installed now for a future activation.

369 This attribute **MUST ONLY** be specified when the `activate` attribute's value is *true* (explicitly or implicitly).

- 370 • deactivateAt **[optional]** - an attribute indicating when the PM is to be deactivated. The value of this attribute
- 371 is a specific instance in time at which the PM should be deactivated.
- 372 This is typically used when a PM is to be installed now for a future activation.
- 373 This attribute **MUST ONLY** be specified when the activate attribute's value is *true* (explicitly or implicitly).
- 374 If the activateAt attribute is also specified, the value of the deactivateAt attribute value **MUST NOT** be the
- 375 same instant as, or earlier than, the activateAt attribute value.
- 376 • anyAttribute **[Optional]** - zero or more attributes from a namespace other than that of this specification. One
- 377 such possibility is an **xs:ID** type attribute such as xml:id or wsu:Id.

378 The following schema fragment defines the **<prov:PMDescriptor>** element:

```

379 <!-- PMDescriptor - describes/carries the components of a PM -->
380
381 <xs:element name="PMDescriptor" type="PMDescriptorType"/>
382
383 <xs:complexType name="PMDescriptorType">
384   <xs:sequence>
385     <xs:element ref="ds:Signature" minOccurs="0"/>
386     <xs:element ref="PMID" />
387     <xs:element ref="PMEngineRef" minOccurs="0"/>
388     <xs:element ref="PMEngine" minOccurs="0"/>
389     <xs:element ref="PMInitData" minOccurs="0"/>
390     <xs:element ref="PMRTData" minOccurs="0"/>
391     <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
392   </xs:sequence>
393   <xs:attribute name="activate" type="xs:boolean" use="optional"/>
394   <xs:attribute name="activateAt" type="xs:dateTime" use="optional"/>
395   <xs:attribute name="deactivateAt" type="xs:dateTime" use="optional"/>
396   <xs:anyAttribute namespace="##any" processContents="lax"/>
397 </xs:complexType>
398
399 <xs:element name="PMInitData" type="PMDDataType" />
400 <xs:element name="PMRTData" type="PMDDataType" />
401
402 <xs:complexType name="PMDDataType" mixed="false">
403   <xs:sequence>
404     <xs:any namespace="##any" processContents="lax" maxOccurs="unbounded"/>
405   </xs:sequence>
406   <xs:anyAttribute namespace="##other" processContents="lax"/>
407 </xs:complexType>
408
409 <xs:element name="PMEngine" type="PMEngineType" />
410
411 <xs:complexType name="PMEngineType" mixed="false">
412   <xs:sequence>
413     <xs:element ref="PMEInfo" />
414     <xs:element ref="PMEBytes" />
415   </xs:sequence>
416   <xs:anyAttribute namespace="##other" processContents="lax"/>
417 </xs:complexType>
418
419 <xs:element name="PMEBytes" type="xs:base64Binary" />
420

```

421 The following is an example of a **<prov:PMDescriptor>** element. In this case, the descriptor is signed, includes

422 the reference to a PM Engine as well as some initialization data, and the PM is activated upon installation.

```

423 <prov:PMDescriptor xs:id="2323923900239" >
424   <prov:PMID issuer="http://provs-r-us.com" >uuid:239032-230328-92379-2397923</prov:PMID>
425   <prov:PMEngineRef>http://pmsRus.org/Ve ryTrustedModule/3.7</prov:PMEngineRef>
426   <prov:PMInitData>
427     <MyData>

```

```

428     .... initialization data here ...
429     </MyData>
430 </prov:PMInitData>
431 <ds:Signature>
432     ... signature data goes here ...
433 </ds:Signature>
434 </prov:PMDescriptor>
435

```

436 The following is another example of a **<prov:PMDescriptor>** element. In this case, the PM has no initialization
437 data (which would typically mean the PMEngine is somehow uniquely branded with the data) and it is activated at
438 some point in the future.

```

439 <prov:PMDescriptor xs:id="2323923900239"
440     activateAt="2007-01-11T14:52:00Z" >
441     <prov:PMID issuer="http://provs-r-us.com">uuid:239032-230328-92379-2397923</prov:PMID>
442     <prov:PMEngineRef>http://pmsRus.org/ConorsPM/1.0</prov:PMEngineRef>
443     <ds:Signature>
444     ... signature data goes here ...
445     </ds:Signature>
446 </prov:PMDescriptor>
447

```

448 The following is another example of a **<prov:PMDescriptor>** element. In this case, the PM is **not** activated when
449 it is installed. In order to activate it, the provisioning service must invoke the PMM **<pmm:PMActivate** interface.

```

450 <prov:PMDescriptor xs:id="2323923900239"
451     activate="false" >
452     <prov:PMID issuer="http://provs-r-us.com">uuid:239032-230328-92379-2397923</prov:PMID>
453     <prov:PMEngineRef>http://pmsRus.org/VeryTrustedModule/3.7</prov:PMEngineRef>
454     <prov:PMInitData>
455     <MyData>
456     .... initialization data here ...
457     </MyData>
458     </prov:PMInitData>
459     <ds:Signature>
460     ... signature data goes here ...
461     </ds:Signature>
462 </prov:PMDescriptor>
463

```

464 3.4. PMStatus

465 The **<prov:PMStatus>**, (or Provisioned Module Status) describes the provisioning status of a particular provisioned
466 module.

467 The **<prov:PMStatus>** element has the following attributes:

- 468 • **<prov:PMID>** **[Required]** - the PMID for the PM that this status applies to.
- 469 • **<prov:State>** **[Required]** - the current state of the provisioning of this PM. This element contains the URI
470 value representing the current state as well as the following attribute:
 - 471 • **asof** **[Optional]** - the times when the current status of the PM was achieved. This attribute SHOULD NOT
472 be specified when setting a new status at the ProvS or the PMM and SHOULD ONLY be present on status
473 retrievals (e.g., **<PMGetStatus>**).

474 The following status values are defined by Liberty for the provisioning status of a PM (implementations MAY define
475 additional values and attach specific meaning to them):

- 476 • *urn:liberty:prov:2007-09:status:Registered* - the <PMDescriptor> has been registered with the ProvS and the
 477 ProvS is awaiting the <PMGetDescriptor> request from the PMM.
- 478 • *urn:liberty:prov:2007-09:status:Resolved* - the PMM has obtained the <PMDescriptor> via the
 479 <prov:PMGetDescriptor> request and the ProvS is awaiting a status update from the PMM to indicate
 480 the provisioning is complete.
- 481 • *urn:liberty:prov:2007-09:status:Active* - the PM is installed and active.
- 482 • *urn:liberty:prov:2007-09:status:Inactive* - the PM is installed, but is currently inactive.
- 483 • *urn:liberty:prov:2007-09:status:Deleted* - the PM has been deleted. The actual existence of this status is
 484 implementation-defined. Some implementations will immediately remove the PMD data from their database
 485 upon deletion and thus never expose this status as the PMD would not exist. Other implementations will maintain
 486 the PMD in the database for some implementation-defined period of time with this deleted status.
- 487 Implementations of ProvS clients SHOULD NOT depend upon the existence of this status, nor upon the immediate
 488 removal of a PMD upon deletion (e.g., don't expect to be able to reuse the PMID of a deleted PMD although some
 489 implementations MAY allow this).

490 The following schema fragment defines the **PMStatus** element:

```

491 <!-- PMStatus - Provisioning status of the PM -->
492
493 <xs:element name="PMStatus" type="PMStatusType"/>
494 <xs:complexType name="PMStatusType">
495   <xs:sequence>
496     <xs:element ref="PMID"/>
497     <xs:element ref="State"/>
498   </xs:sequence>
499 </xs:complexType>
500
501 <xs:element name="State" type="StateType"/>
502 <xs:complexType name="StateType">
503   <xs:simpleContent>
504     <xs:extension base="xs:anyURI">
505       <xs:attribute name="asof" type="xs:dateTime" use="optional"/>
506     </xs:extension>
507   </xs:simpleContent>
508 </xs:complexType>
509

```

510 The following is an example of a **PMStatus** element. In this case, the provisioning of the PM is complete.

```

511 <prov:PMStatus>
512   <prov:PMID issuer="http://provs-r-us.com">uuid:239032-230328-92379-2 397923</prov:PMID>
513   <prov:State asof="2007-01-14T17:31:11Z">urn:liberty:prov:2007-09:status:Active</prov:State>
514 </prov:PMStatus>
515

```

516 3.5. PMEInfo

517 The <prov:PMEInfo>, (or Provisioned Module Engine Information) describes a provisioned module engine (the
 518 executable code portion of a PM).

519 The <prov:PMEInfo> element has the following attributes:

- 520 • <prov:PMEngineRef> **[Required]** - the identity token (or name of) the provisioned module engine. Each version
 521 of each engine **MUST** have a unique URI string to identify the exact engine.

- 522 • <prov:PMECreatorID> **[Required]** - a URI containing the provider ID of the entity which registered the
523 PMEngine at the ProvS.
- 524 • <prov:PMEWhenCreated> **[Required]** - the moment in time when the PMEngine was registered (created) at the
525 ProvS.
- 526 • <prov:PMEEnabled> **[Required]** - a boolean flag indicating whether or not this PMEngine is enabled for
527 downloading by a PMM.
- 528 • <prov:PMEWhenEnabled> **[Required]** - the moment in time when the enabled status of the PMEngine was last
529 changed.
- 530 • <prov:PMESize> **[Required]** - the size, in bytes, of the PMEngine.
- 531 • <prov:PMEHash> **[Required]** - a hash digest of the PMEngine. This element contains the hash value
532 representing the current state as well as the following attribute:
 - 533 • method **[Optional]** - the algorithm used to generate the hash digest. The possible values are taken from the
534 XML Signature Syntax and Processing, which currently only lists the SHA-1 hash URI:
 - 535 • <http://www.w3.org/2000/09/xmldsig#sha1>

536 The following schema fragment defines the <prov:PMEInfo> element:

```
537 <!-- PMEInfo - The current state of a PMEngine -->
538
539 <xs:element name="PMEInfo" type="PMEInfoType"/>
540 <xs:complexType name="PMEInfoType">
541   <xs:sequence>
542     <xs:element ref="PMEEngineRef"/>
543     <xs:element ref="PMECreatorID"/>
544     <xs:element ref="PMEWhenCreated"/>
545     <xs:element ref="PMEEnabled"/>
546     <xs:element ref="PMEWhenEnabled"/>
547     <xs:element ref="PMESize"/>
548     <xs:element ref="PMEHash"/>
549     <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded" />
550   </xs:sequence>
551   <xs:anyAttribute namespace="##any" processContents="lax" />
552 </xs:complexType>
553
554 <xs:element name="PMEEngineRef" type="xs:anyURI"/>
555
556 <xs:element name="PMECreatorID" type="xs:anyURI"/>
557
558 <xs:element name="PMEWhenCreated" type="xs:dateTime"/>
559
560 <xs:element name="PMEEnabled" type="xs:boolean"/>
561
562 <xs:element name="PMEWhenEnabled" type="xs:dateTime"/>
563
564 <xs:element name="PMESize" type="xs:integer"/>
565
566 <xs:element name="PMEHash" type="PMEHashType"/>
567 <xs:complexType name="PMEHashType">
568   <xs:simpleContent>
569     <xs:extension base="xs:base64Binary">
570       <xs:attribute name="method" type="xs:anyURI" use="required"/>
571     </xs:extension>
572   </xs:simpleContent>
573 </xs:complexType>
574
```

575 The following is an example of a `<prov:PMEInfo>` element. In this case, the provisioning of the PM is complete.

```
576 <prov:PMEInfo>
577   <prov:PMEEngineRef>http://pmsRus.org/VeryTrustedModule/3.6</prov:PMEEngineRef>
578   <prov:PMECreatorID>http://reg.providers.com</prov:PMECreatorID>
579   <prov:PMEWhenCreated>2007-01-18T17:32:14Z</prov:PMEWhenCreated>
580   <prov:PMEEnabled>>true</prov:PMEEnabled>
581   <prov:PMEWhenEnabled>2007-01-18T18:15:22Z</prov:PMEWhenEnabled>
582   <prov:PMESize>185676</prov:PMESize>
583   <prov:PMEHash method="http://www.w3.org/2000/09/xmldsig#sha1">
584     ...SHA1 hash data ...
585   </prov:PMEHash>
586 </prov:PMEInfo>
587
```

588 3.6. Callback EPR

589 The `<prov:CallbackEPR>` is used by the PMM to register its callback location for PM maintenance operations.
590 This EPR is normally a traditional ID-WSF EPR (see [LibertyDisco]).

591 However, in the case where the PMM cannot expose an external endpoint that is visible to the ProvS, the PMM should
592 register an "anonymous" `<prov:CallbackEPR>` which MUST have the following characteristics:

- 593 • The ONLY element present in the EPR is the `<wsa:Address>` element which MUST have the value
594 `http://www.w3.org/2005/08/addressing/anonymous`

595 The schema for the `<prov:CallbackEPR>` is shown below.

```
596 <!-- CallbackEPR - where the PMM can receive Provisionig update requests -->
597
598 <xs:element name="CallbackEPR" type="wsa:EndpointReferenceType"/>
599
```

600 **Figure 2. `<prov:CallbackEPR>` — Schema Fragment**

601 An example "anonymous" `<prov:CallbackEPR>` is shown below.

```
602 <prov:CallbackEPR>
603   <wsa:Address>http://www.w3.org/2005/08/addressing/anonymous</wsa:Address>
604 </prov:CallbackEPR>
605
```

606 **Example 1. Example anonymous `<prov:CallbackEPR>`**

607 **4. Provisioning Service (ProvS)**

608 The Provisioning Service (ProvS) provides the interfaces used by PMMs on Advanced Clients to obtain PMs for
 609 provisioning.

610 An abstract WSDL definition for the Provisioning Service is included in this document, see [Section 6: Provisioning](#)
 611 Service WSDL . This WSDL document defines all of the "WSDL operations" for the IdP Service.

612 The complete schema for the Provisioning Service is included in this document, see [Section 5: Provisioning Service](#)
 613 Schema .

614 **4.1. Service URIs**

615 **Table 1. ProvS Service URIs**

Use	URI
Service Type	<i>urn:liberty:prov:2007-09</i>
PMActivate wsa:Action	<i>urn:liberty:prov:2007-09:PMActivate</i>
PMActivateResponse wsa:Action	<i>urn:liberty:prov:2007-09:PMActivateResponse</i>
PMDeactivate wsa:Action	<i>urn:liberty:prov:2007-09:PMDeactivate</i>
PMDeactivateResponse wsa:Action	<i>urn:liberty:prov:2007-09:PMDeactivateResponse</i>
PMDelete wsa:Action	<i>urn:liberty:prov:2007-09:PMDelete</i>
PMDeleteResponse wsa:Action	<i>urn:liberty:prov:2007-09:PMDeleteResponse</i>
PMGetDescriptor wsa:Action	<i>urn:liberty:prov:2007-09:PMGetDescriptor</i>
PMGetDescriptorResponse wsa:Action	<i>urn:liberty:prov:2007-09:PMGetDescriptorResponse</i>
PMGetStatus wsa:Action	<i>urn:liberty:prov:2007-09:PMGetStatus</i>
PMGetStatusResponse wsa:Action	<i>urn:liberty:prov:2007-09:PMGetStatusResponse</i>
PMRegisterDescriptor wsa:Action	<i>urn:liberty:prov:2007-09:PMRegisterDescriptor</i>
PMRegisterDescriptorResponse wsa:Action	<i>urn:liberty:prov:2007-09:PMRegisterDescriptorResponse</i>
PMSetStatus wsa:Action	<i>urn:liberty:prov:2007-09:PMSetStatus</i>
PMSetStatusResponse wsa:Action	<i>urn:liberty:prov:2007-09:PMSetStatusResponse</i>
PMUpdate wsa:Action	<i>urn:liberty:prov:2007-09:PMUpdate</i>
PMUpdateResponse wsa:Action	<i>urn:liberty:prov:2007-09:PMUpdateResponse</i>
Poll wsa:Action	<i>urn:liberty:prov:2007-09:Poll</i>
PollResponse wsa:Action	<i>urn:liberty:prov:2007-09:PollResponse</i>
UpdateEPR wsa:Action	<i>urn:liberty:prov:2007-09:UpdateEPR</i>
UpdateEPRResponse wsa:Action	<i>urn:liberty:prov:2007-09:UpdateEPRResponse</i>

616

Table 2. ProvS Service URIs, part 2

Use	URI
PMERegister wsa:Action	<i>urn:liberty:prov:2007-09:PMERegister</i>
PMERegisterResponse wsa:Action	<i>urn:liberty:prov:2007-09:PMERegisterResponse</i>
PMEDelete wsa:Action	<i>urn:liberty:prov:2007-09:PMEDelete</i>
PMEDeleteResponse wsa:Action	<i>urn:liberty:prov:2007-09:PMEDeleteResponse</i>
PMEDisable wsa:Action	<i>urn:liberty:prov:2007-09:PMEDisable</i>
PMEDisableResponse wsa:Action	<i>urn:liberty:prov:2007-09:PMEDisableResponse</i>
PMEDownload wsa:Action	<i>urn:liberty:prov:2007-09:PMEDownload</i>
PMEDownloadResponse wsa:Action	<i>urn:liberty:prov:2007-09:PMEDownloadResponse</i>
PMEEenable wsa:Action	<i>urn:liberty:prov:2007-09:PMEEenable</i>
PMEEenableResponse wsa:Action	<i>urn:liberty:prov:2007-09:PMEEenableResponse</i>
PMEGetInfo wsa:Action	<i>urn:liberty:prov:2007-09:PMEGetInfo</i>
PMEGetInfoResponse wsa:Action	<i>urn:liberty:prov:2007-09:PMEGetInfoResponse</i>
PMEUpload wsa:Action	<i>urn:liberty:prov:2007-09:PMEUpload</i>
PMEUploadResponse wsa:Action	<i>urn:liberty:prov:2007-09:PMEUploadResponse</i>

617 **4.2. Status Codes**

618 The following status code strings are defined:

- 619 • *OK*: message processing succeeded
- 620 • *Failed*: general failure code
- 621 • *Forbidden*: action by invoker is forbidden.
- 622 • *Issued*: the specified artifact has already been issued.
- 623 • *NotFound*: the requested object is not present.
- 624 • *Expired*: the presented token/artifact has expired.
- 625 • *Duplicate*: the object already exists or the task has already been accomplished.
- 626 • *Invalid*: one or more of the specified parameters are invalid.
- 627 • *WrongSize*: the engine size is incorrect.
- 628 • *HashMismatch*: the engine hash does not match the registered value.
- 629 • *LimitExceeded*: too much data was sent to the ProvS.
- 630 • *OutOfOrder*: data was sent in an incorrect order.
- 631 • *FeatureNotSupported*: the request involves some capability that is not supported.

632 These strings are expected to appear in the `code` attribute of `<Status>` elements used in Provisioning Service protocol
633 messages. Specific uses for the status codes are defined in the processing rules for individual messages. The contents
634 of the `comment` attribute are not defined by this specification, but it may be used for additional descriptive text intended
635 for human consumption (for example, to carry information that will aid in debugging).

636 4.3. Request and Response Abstract Types

637 4.3.1. Complex Type RequestAbstractType

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

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

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

```
643 <!-- RequestAbstractType - common request message structure -->  
644  
645 <xs:complexType name="RequestAbstractType" abstract="true">  
646   <xs:anyAttribute namespace="##other" processContents="lax"/>  
647 </xs:complexType>  
648
```

649 4.3.2. Complex Type ResponseAbstractType

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

- 652 • **<lu:Status> [Required]** - The `<lu:Status>` element is used to convey status codes and related information.
653 The schema fragment is defined in the Liberty ID-WSF Utility schema. The local definitions of status codes are
654 described in [Section 4.2](#).
- 655 • **anyAttribute [Optional]** - An attribute from a namespace other than that of this specification.

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

```
657 <!-- ResponseAbstractType - common message response structure -->  
658  
659 <xs:complexType name="ResponseAbstractType" abstract="true">  
660   <xs:sequence>  
661     <xs:element ref="lu:Status"/>  
662   </xs:sequence>  
663   <xs:anyAttribute namespace="##other" processContents="lax"/>  
664 </xs:complexType>  
665
```

666 4.4. Operation: *PMGetDescriptor*

667 The *PMGetDescriptor* operation is used by the PMM to exchange a `<PMDArtifact>` for a `<PMDDescriptor>`.

668 4.4.1. `wsa:Action` values for *PMGetDescriptor* Messages

669 `<PMGetDescriptor>` request messages MUST include a `<wsa:Action>` SOAP header with the value of
670 "urn:liberty:prov:2007-09:PMGetDescriptor."

671 `<PMGetDescriptorResponse>` messages MUST include a `<wsa:Action>` SOAP header with the value of
672 "urn:liberty:prov:2007-09:PMGetDescriptorResponse."

673 4.4.2. PMGetDescriptor Message

674 The `<PMGetDescriptor>` request is called to exchange a `<PMDArtifact>` for a `<PMDDescriptor>`.

675 The `<prov:PMGetDescriptor>` request contains the following attributes and/or elements:

- 676 • `PMDArtifact` **[Required]** - the `PMDArtifact` that is being resolved.
- 677 • `<prov:CallbackEPR>` **[Required]** - one or more ID-WSF Endpoint References which describe how the ProvS
678 can communicate with the PMM for maintenance operations (see [Section 3.6](#)).
- 679 If more than one `<prov:CallbackEPR>` element is present, the ProvS may choose any of the elements present
680 (recognizing that the list is in preference order – the most preferred element is first).
- 681 • `anyAttribute` **[Optional]** Zero or more attributes from a namespace other than that of this specification. One
682 such possibility is an `xs:ID` type attribute such as `xml:id` or `wsu:Id`.

683 The schema for the `<prov:PMGetDescriptor>` is shown below.

```
684 <!-- PMGetDescriptor - request to exchange PMHandle for PMGetDescriptor -->
685
686 <xs:element name="PMGetDescriptor" type="PMGetDescriptorType"/>
687
688 <xs:complexType name="PMGetDescriptorType">
689   <xs:complexContent>
690     <xs:extension base="RequestAbstractType">
691       <xs:sequence>
692         <xs:element ref="PMDArtifact"/>
693         <xs:element ref="CallbackEPR" maxOccurs="unbounded"/>
694       </xs:sequence>
695     </xs:extension>
696   </xs:complexContent>
697 </xs:complexType>
698
```

699 **Figure 3. `<prov:PMGetDescriptor>` — Schema Fragment**

700 An example message body containing a `<PMGetDescriptor>` message follows. This request enables two service
701 instances.

```
702 <prov:PMGetDescriptor>
703   <prov:PMDArtifact>23asdfhoi323hpo sdf923h9sdfhweorh2398asdfjweoi ha</prov:PMDArtifact>
704   <prov:CallbackEPR>
705     <wsa:Address>http://www.w3.org/2005/08/addressing/anonymous</wsa:Address>
706   </prov:CallbackEPR>
707 </prov:PMGetDescriptor>
708
```

709 **Example 2. Example `<prov:PMGetDescriptor>` Message**

710 4.4.3. PMGetDescriptorResponse Message

711 This response to the `<prov:PMGetDescriptor>` request contains the following elements:

- 712 • `<lu:Status>` **[Required]** - the status of the response. See the processing rules below for more information.
- 713 • `<prov:PMDDescriptor>` **[Optional]** - the Provisioned Module Descriptor that is associated with the supplied
714 `<prov:PMDArtifact>`. This element will not be present on unsuccessful responses.

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

```

717 <!-- PMGetDescriptorResponse - response to the PMGetDescriptor request -->
718
719 <xs:element name="PMGetDescriptorResponse" type="PMGetDescriptorResponseType"/>
720
721 <xs:complexType name="PMGetDescriptorResponseType">
722   <xs:complexContent>
723     <xs:extension base="ResponseAbstractType">
724       <xs:sequence>
725         <xs:element ref="PMDescriptor" minOccurs="0"/>
726       </xs:sequence>
727     </xs:extension>
728   </xs:complexContent>
729 </xs:complexType>
730
  
```

731 **Figure 4. <prov:PMGetDescriptorResponse> — Schema Fragment**

732 An example message body containing a <PMGetDescriptorResponse> message follows. This is a successful
 733 response.

```

734 <prov:PMGetDescriptorResponse >
735   <lu:Status code="OK" />
736   <prov:PMDescriptor xs:id="2323923900239">
737     <prov:PMID issuer="http://provs-r-us.com">uuid:239032-230328-92379-2397923</prov:PMID>
738     <prov:PMEngineRef>http://pmsRus.org/VeryTrustedModule/3.7</prov:PMEngineRef>
739     <prov:PMInitData>
740       <MyData>
741         .... initialization data here ...
742       </MyData>
743     </prov:PMInitData>
744     <ds:Signature>
745       ... signature data goes here ...
746     </ds:Signature>
747   </prov:PMDescriptor>
748 </prov:PMGetDescriptorResponse>
749
  
```

750 **Example 3. Example <prov:PMGetDescriptorResponse> Message**

751 4.4.4. PMGetDescriptor Processing Rules

- 752 • If the PMDescriptor has already been issued for the specified PMArtifact (e.g., someone has already invoked this
 753 interface to resolve the artifact), the request **MUST** fail. If detailed status codes are to be included in the response,
 754 the detailed error code for this error **MUST** be *"Issued."*
- 755 • If the artifact is presented after the PH has expired, the ProvS **SHOULD** refuse to process it and instead return a
 756 failure. If detailed status codes are to be included in the response, the detailed error code for this error **MUST** be
 757 *"Expired."*
- 758 • If the artifact is not known to the ProvS, the ProvS **MUST** treat the request as a failure. If detailed status codes are
 759 to be included in the response, the detailed error code for this error **MUST** be *"NotFound."*
- 760 • The Provisioning Service **SHOULD** take steps to ensure that the correct party is submitting the request and to
 761 reject requests from inappropriate parties.

762 • If request processing succeeded, the top-level status code MUST be "OK." Otherwise, the top-level status code
763 MUST be "Failed"

764 • If the top-level status code is "Failed," the response MAY also contain *Forbidden*, *Issued*, *Expired*, or *NotFound*
765 as a second-level status code. The Provisioning Service instance may not wish to reveal the reason for failure, in
766 which case, no second-level status code will appear.

767 4.5. Operation: *PMActivate*

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

769 4.5.1. *wsa:Action* values for *PMActivate* Messages

770 <*PMActivate*> request messages MUST include a <*wsa:Action*> SOAP header with the value of
771 "urn:liberty:prov:2007-09:*PMActivate*."

772 <*PMActivateResponse*> messages MUST include a <*wsa:Action*> SOAP header with the value of
773 "urn:liberty:prov:2007-09:*PMActivateResponse*."

774 4.5.2. *PMActivate* Message

775 The <*PMActivate*> request is called by a provisioning entity (such as the RegS show in the Provisioning workflow
776 in the Liberty ID-WSF Advanced Client Technologies Overview [[LibertyACT](#)] to activate an inactive PM.

777 The <*prov:PMActivate*> request contains the following elements/attributes:

778 • <*prov:PMActivateItem*> [**Required**] - one or more activate actions which contain the following
779 elements/attributes:

780 • <*prov:PMID*> [**Required**] - the identifier of the PM being activated.

781 • *itemID* [**Required**] - a unique (within this message) identifier for this request item (used for correlation of
782 result status in the response).

783 • *at* [**Optional**] - a time at which the activation should take place. If specified, the value SHOULD be some
784 point in the future at which the PM would be activated.

785 If this attribute is not specified, or it is specified with a time in the past, the request is treated as an immediate
786 request.

787 This attribute MUST NOT be interpreted or acted upon by the ProvS (with the potential exception of optional
788 scheduling of multiple queued operations). The subsequent <*pmm:PMDactivate*> request to the PMM
789 MUST include this value and SHOULD be sent as soon as possible (e.g., do NOT wait till the specified time
790 to deliver the message).

791 Note that since this request is made to the ProvS and not the PMM, the time it takes to relay the message,
792 especially in a scenario where the PMM is polling the ProvS, may delay the activation time, especially when
793 this attribute has a very short time frame.

794 • <*dp:NotifyTo*> [**Optional**] - an optional endpoint reference for delayed notification completion messages (see
795 [[LibertyDP](#)]). This is used to allow the invoker to receive a completion status on the eventual activation of the PM
796 at the PMM.

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

799 The schema for the `<prov:PMActivate>` is shown below.

```

800 <!-- PMActivate - to activate one or more PM(s) at the PMM -->
801
802 <xs:element name="PMActivate" type="PMActivateType"/>
803
804 <xs:complexType name="PMActivateType">
805   <xs:complexContent>
806     <xs:extension base="RequestAbstractType">
807       <xs:sequence>
808         <xs:element ref="PMActivateItem" maxOccurs="unbounded" />
809         <xs:element ref="dp:NotifyTo" minOccurs="0" />
810       </xs:sequence>
811     </xs:extension>
812   </xs:complexContent>
813 </xs:complexType>
814
815 <xs:element name="PMActivateItem" type="PMActivateItemType" />
816
817 <xs:complexType name="PMActivateItemType">
818   <xs:sequence>
819     <xs:element ref="prov:PMID"/>
820   </xs:sequence>
821   <xs:attribute name="itemID" type="xs:string" use="required"/>
822   <xs:attribute name="at" type="xs:dateTime" use="optional"/>
823 </xs:complexType>
824

```

825 **Figure 5. `<prov:PMActivate>` — Schema Fragment**

826 An example message body containing a `<prov:PMActivate>` message follows. This request activates 2 PMs, one
 827 at 1PM on the 18th of Dec, 2006, and one now and provides a notification endpoint for delayed notification of the
 828 completion status.

```

829 <prov:PMActivate>
830   <prov:PMActivateItem itemID="1" at="2007-01-18T13:00:00Z" >
831     <prov:PMID issuer="http://provs-r-us.com">uuid:239032-23032 8-92379-2397923</prov:PMID>
832   </prov:PMActivateItem>
833   <prov:PMActivateItem itemID="2">
834     <prov:PMID issuer="http://provs-r-us.com">uuid:39238 0-948492-18923-2389238</prov:PMID>
835   </prov:PMActivateItem>
836   <dp:NotifyTo>
837     <wsa:Address>https://provider.com/notifications</wsa:Address>
838     <wsa:Metadata>
839       <ds:ProviderID>http://provider.com/</ds:ProviderID>
840       <ds:ServiceType>urn:liberty:dp:2007-09:notification</ds:ServiceType>
841       <ds:Framework version="2.0" />
842       <ds:SecurityContext>
843         <ds:SecurityMechID>urn:liberty:security:2005-02:TLS:null</ds:SecurityMechID>
844       </ds:SecurityContext>
845     </wsa:Metadata>
846   </dp:NotifyTo>
847 </prov:PMActivate>
848

```

849 **Example 4. Example `<prov:PMActivate>` Message**

850 4.5.3. `PMActivateResponse` Message

851 This response to the `<prov:PMActivate>` request contains the following elements:

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

```

855 <!-- PMActivateResponse - the response to the PMActivate request -->
856
857 <xs:element name="PMActivateResponse" type="PMActivateResponseType"/>
858
859 <xs:complexType name="PMActivateResponseType">
860   <xs:complexContent>
861     <xs:extension base="ResponseAbstractType"/>
862   </xs:complexContent>
863 </xs:complexType>
864
```

865 **Figure 6. `<prov:PMActivateResponse>` — Schema Fragment**

866 An example message body containing a `<prov:PMActivateResponse>` message follows. This is a partial

867 response. The 2nd item in the request succeeded, but the first item is still in process and a delayed notification

868 will be sent upon completion.

```

869 <prov:PMActivateResponse>
870   <lu:Status code="Partial">
871     <lu:Status ref="1" code="WillNotify"/>
872     <lu:Status ref="2" code="OK" />
873   </lu:Status>
874 </prov:PMActivateResponse>
875
```

876 **Example 5. Example `<prov:PMActivateResponse>` Message**

877 In this second example for the `<prov:PMActivateResponse>` message, the delayed notification message contain-

878 ing the completion status for the first item is returned.

```

879 <prov:PMActivateResponse>
880   <lu:Status code="Partial">
881     <lu:Status ref="1" code="OK"/>
882   </lu:Status>
883 </prov:PMActivateResponse>
884
```

885 **Example 6. Example delayed `<prov:PMActivateResponse>` Message**

886 4.5.4. PMActivate Processing Rules

- 887 • This operation adopts the Delayed Notification design pattern (see [LibertyDP]) and incorporates all of the
888 associated processing rules. Delayed notifications are needed for both indirect operations (where the PMM is
889 polling for incoming request) and for delayed operations (where the `at` attribute is used with a time in the future).
- 890 Delayed Notifications SHOULD NOT to be used for immediate operations (no future `at` attribute) targeted at
891 a PMM that has exposed a means of direct communications from the ProvS (e.g., they have registered a non-
892 anonymous CallbackEPR with the ProvS). In such cases, the ProvS SHOULD wait for the completion of the
893 subsequent request to the PMM and return the appropriate status in its response to the invoker.
- 894 If any portion of the request IS NOT an immediate operation and the `<dp:NotifyTo>` element IS NOT present
895 on the request, the ProvS should validate the request to the extent possible and respond with the results of that
896 validation. If that validation succeeded, the ProvS MUST continue with the execution of the operation. In such
897 cases, the actual completion status of the operation may be different from what was reported (e.g., the request may
898 fail at the PMM for some reason), but that will not be known to the invoker, given that they have chosen to not
899 provide a delayed notification endpoint.
- 900 If any portion of the request IS NOT an immediate operation and the `<dp:NotifyTo>` element IS present on the
901 request, the ProvS MUST report the final completion status (e.g., wait for the completion status of any delayed or
902 indirect operation at the PMM) of the request. If multiple operations are included in the request, the ProvS MAY
903 group the results of some or all of the operations in a single delayed notification message.
- 904 In the case of a mixed request, the immediate portion of the request SHOULD be handled as an immediate request
905 and the actual results returned with the initial response.
- 906 • Requests to activate a PM that is not known to the ProvS MUST result in a failure. If detailed status codes are
907 included in the response, the detailed status code for this error MUST be *"NotFound."*
- 908 • Requests to activate a PM that has not yet been provisioned (i.e., the PMDescriptor has been received, but not
909 provisioned to a PMM) MUST result in a failure. If detailed status codes are included in the response, the detailed
910 status code for this error MUST be *"NotProvisioned."*
- 911 • The ProvS SHOULD accept and forward this request regardless of the current status of the PM (provided that it
912 has been provisioned and has not been deleted). The return status of this request should reflect the eventual return
913 status from the PMM.
- 914 • If **all** items in the request have the same completion status, the top level status MUST reflect that completion status
915 and MUST be *"OK," "Failed,"* or *"WillNotify."* Otherwise, if the results were mixed, the top-level status MUST be
916 *Partial* and a second-level status MUST be included indicating which items succeeded, which failed, and which
917 will be subject to delayed notification. The second level status elements MUST include the `ref` attribute containing
918 the `itemID` value for the item. For failures, the second-level status codes MAY simply be *"Failed"* or they may
919 indicate, with more detail, the reason for the failure.
- 920 If the top-level status is *"Failed,"* second level status codes MAY be present which contain detailed error
921 information if the ProvS wants to share that information with the invoking party.

922 **4.6. Operation: PMDeactivate**

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

924 **4.6.1. wsa:Action values for PMDeactivate Messages**

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

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

929 **4.6.2. PMDeactivate Message**

930 The <PMDeactivate> request is called by a provisioning entity (such as the RegS show in the Provisioning
931 workflow in the Liberty ID-WSF Advanced Client Technologies Overview [[LibertyACT](#)] to deactivate an activated
932 PM.

933 The <prov:PMDeactivate> request contains the following elements/attributes:

934 • <prov:PMDeactivateItem> **[Required]** - one or more deactivation actions which contain the following
935 elements/attributes:

936 • <PMID> **[Required]** - the identifier of the PM being deactivated.

937 • itemID **[Required]** - a unique (within this message) identifier for this request item (used for correlation of
938 result status in the response).

939 • at **[Optional]** - a time at which the deactivation should take place. If specified, the value SHOULD be some
940 point in the future at which the PM would be deactivated.

941 If this attribute is not specified, or it is specified with a time in the past, it is interpreted as an immediate request.

942 This attribute MUST NOT be interpreted or acted upon by the ProvS (with the potential exception of optional
943 scheduling of multiple queued operations). The subsequent <pmm:PMDeactivate> request to the PMM
944 MUST include this value and SHOULD be sent as soon as possible (e.g., do NOT wait till the specified time
945 to deliver the message).

946 Note that since this request is made to the ProvS and not the PMM, the time it takes to relay the message,
947 especially in a scenario where the PMM is polling the ProvS, may delay the deactivation time, especially when
948 this attribute has a very short time frame.

949 • <dp:NotifyTo> **[Optional]** - an optional endpoint reference for delayed notification completion messages (see
950 [[LibertyDP](#)]). This is used to allow the invoker to receive a completion status on the eventual activation of the PM
951 at the PMM.

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

954 The schema for the <prov:PMDeactivate> is shown below.

```

955 <!-- PMDeactivate - to deactivate a PM at the PMM -->
956
957 <xs:element name="PMDeactivate" type="PMDeactivateType" />
958
959 <xs:complexType name="PMDeactivateType">
960   <xs:complexContent>
961     <xs:extension base="RequestAbstractType">
962       <xs:sequence>
963         <xs:element ref="PMDeactivateItem" maxOccurs="unbounded" />
964         <xs:element ref="dp:NotifyTo" minOccurs="0" />
965       </xs:sequence>
966     </xs:extension>
967   </xs:complexContent>
968 </xs:complexType>
969
970 <xs:element name="PMDeactivateItem" type="PMDeactivateItemType" />
971
972 <xs:complexType name="PMDeactivateItemType">
973   <xs:sequence>
974     <xs:element ref="prov:PMID" />
975   </xs:sequence>
976   <xs:attribute name="itemID" type="xs:string" use="required" />
977   <xs:attribute name="at" type="xs:dateTime" use="optional" />
978 </xs:complexType>
979

```

980 **Figure 7. <prov:PMDeactivate> — Schema Fragment**

981 An example message body containing a <prov:PMDeactivate> message follows. This request deactivates two
 982 PMs, one now and one near midnight on New Year's Eve.

```
983 <prov:PMDeactivate>
984   <prov:PMDeactivateItem itemID="1" at="2007-01-31T23:59:59Z" >
985     <prov:PMID issuer="http://provs-r-us.com">uuid:239032-2 30328-92379-2397923</prov:PMID>
986   </prov:PMDeactivateItem>
987   <prov:PMDeactivateItem itemID="2">
988     <prov:PMID issuer="http://provs-r-us.com">uid:392380-948492-18923-2389238 </prov:PMID>
989   </prov:PMDeactivateItem>
990 </prov:PMDeactivate>
991
```

992 **Example 7. Example <prov:PMDeactivate> Message**

993 **4.6.3. PMDeactivateResponse Message**

994 This response to the <prov:PMDeactivate> request contains the following elements:

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

```
998 <!-- PMDeactivateResponse - the response to the PMDeactivate request -->
999
1000 <xs:element name="PMDeactivateResponse" type="PMDeactivateResponseType" />
1001
1002 <xs:complexType name="PMDeactivateResponseType">
1003   <xs:complexContent>
1004     <xs:extension base="ResponseAbstractType"/>
1005   </xs:complexContent>
1006 </xs:complexType>
1007
```

1008 **Figure 8. <prov:PMDeactivateResponse> — Schema Fragment**

1009 An example message body containing a <prov:PMDeactivateResponse> message follows. This is a partially
 1010 successful response where one of the PMs was not found.

```
1011 <prov:PMDeactivateResponse>
1012   <lu:Status code="Partial">
1013     <lu:Status ref="1" code="NotFound"/>
1014   </lu:Status>
1015 </prov:PMDeactivateResponse>
1016
```

1017 **Example 8. Example <prov:PMDeactivateResponse> Message**

1018 **4.6.4. PMDeactivate Processing Rules**

- 1019 • This operation adopts the Delayed Notification design pattern (see [LibertyDP]) and incorporates all of the
1020 associated processing rules. Delayed notifications are needed for both indirect operations (where the PMM is
1021 polling for incoming request) and for delayed operations (where the `at` attribute is used with a time in the future).
- 1022 Delayed Notifications SHOULD NOT to be used for immediate operations (no future `at` attribute) targeted at
1023 a PMM that has exposed a means of direct communications from the ProvS (e.g., they have registered a non-
1024 anonymous CallbackEPR with the ProvS). In such cases, the ProvS SHOULD wait for the completion of the
1025 subsequent request to the PMM and return the appropriate status in its response to the invoker.
- 1026 If any portion of the request IS NOT an immediate operation and the `<dp:NotifyTo>` element IS NOT present
1027 on the request, the ProvS should validate the request to the extent possible and respond with the results of that
1028 validation. If that validation succeeded, the ProvS MUST continue with the execution of the operation. In such
1029 cases, the actual completion status of the operation may be different from what was reported (e.g., the request may
1030 fail at the PMM for some reason), but that will not be known to the invoker, given that they have chosen to not
1031 provide a delayed notification endpoint.
- 1032 If any portion of the request IS NOT an immediate operation and the `<dp:NotifyTo>` element IS present on the
1033 request, the ProvS MUST report the final completion status (e.g., wait for the completion status of any delayed or
1034 indirect operation at the PMM) of the request. If multiple operations are included in the request, the ProvS MAY
1035 group the results of some or all of the operations in a single delayed notification message.
- 1036 In the case of a mixed request, the immediate portion of the request SHOULD be handled as an immediate request
1037 and the actual results returned with the initial response.
- 1038 • Requests to deactivate a PM that is not known to the ProvS MUST result in a failure. If detailed status codes are
1039 included in the response, the detailed status code for this error MUST be *"NotFound"*.
- 1040 • Requests to deactivate a PM that has not yet been provisioned (i.e., the PMDescriptor has been received, but not
1041 provisioned to a PMM) MUST result in a failure. If detailed status codes are included in the response, the detailed
1042 status code for this error MUST be *"NotProvisioned"*.
- 1043 • The ProvS SHOULD accept and forward this request regardless of the current state of the PMDescriptor (provided
1044 that it does exist and has not been deleted).
- 1045 • If **all** items in the request have the same completion status, the top level status MUST reflect that completion status
1046 and MUST be *"OK"*, *"Failed"*, or *"WillNotify"*. Otherwise, if the results were mixed, the top-level status MUST be
1047 *Partial* and a second-level status MUST be included indicating which items succeeded, which failed, and which
1048 will be subject to delayed notification. The second level status elements MUST include the `ref` attribute containing
1049 the `itemID` value for the item. For failures, the second-level status codes MAY simply be *"Failed"* or they may
1050 indicate, with more detail, the reason for the failure.
- 1051 If the top-level status is *"Failed"*, second level status codes MAY be present which contain detailed error
1052 information if the ProvS wants to share that information with the invoking party.

1053 **4.7. Operation: PMDelete**

1054 The *PMDelete* operation is used by a PM-issuing party (such as an IdP) to delete previously registered PMDs at the
1055 ProvS. The previously registered PMDs MAY have already been provisioned into running PMs (in which case, the
1056 ProvS would initiate an deletion process with the PMM where the PM has been provisioned).

1057 **4.7.1. wsa:Action values for PMDelete Messages**

1058 `<PMDelete>` request messages MUST include a `<wsa:Action>` SOAP header with the value of
1059 `"urn:liberty:prov:2007-09:PMDelete"`.

1060 `<PMDeleteResponse>` messages MUST include a `<wsa:Action>` SOAP header with the value of
1061 `"urn:liberty:prov:2007-09:PMDeleteResponse"`.

1062 **4.7.2. PMDelete Message**

1063 The <PMDelete> request is called to delete a PMD that was registered with the ProvS.

1064 The <prov:PMDelete> request contains the following elements/attributes:

- 1065 • <PMDeleteItem> [**Required**] - one or more request elements containing the following elements or attributes:
 - 1066 • <PMID> [**Required**] - the identifier for the PMDescriptor that is to be deleted. This element MUST contain a
 - 1067 value that matches the PMID of a previously registered <PMDescriptor>.
 - 1068 • itemID [**Required**] - a unique (within this message) identifier for this request item (used for correlation of
 - 1069 result status in the response).
 - 1070 • <dp:NotifyTo> [**Optional**] - an optional endpoint reference for delayed notification completion messages (see
 - 1071 [[LibertyDP](#)]). This is used to allow the invoker to receive a completion status on the eventual activation of the PM
 - 1072 at the PMM.
 - 1073 • anyAttribute [**Optional**] - zero or more attributes from a namespace other than that of this specification. One
 - 1074 such possibility is an **xs:ID** type attribute such as xml:id or wsu:Id.

1075 The schema for the <prov:PMDelete> is shown below.

```

1076 <!-- PMDelete - to register a new PM at the ProvS -->
1077
1078 <xs:element name="PMDelete" type="PMDeleteType"/>
1079 <xs:complexType name="PMDeleteType">
1080   <xs:complexContent>
1081     <xs:extension base="RequestAbstractType">
1082       <xs:sequence>
1083         <xs:element ref="PMDeleteItem" maxOccurs="unbounded" />
1084         <xs:element ref="dp:NotifyTo" minOccurs="0" />
1085       </xs:sequence>
1086     </xs:extension>
1087   </xs:complexContent>
1088 </xs:complexType>
1089
1090 <xs:element name="PMDeleteItem" type="PMDeleteItemType" />
1091 <xs:complexType name="PMDeleteItemType">
1092   <xs:sequence>
1093     <xs:element ref="prov:PMID"/>
1094   </xs:sequence>
1095   <xs:attribute name="itemID" type="xs:string" use="required"/>
1096 </xs:complexType>
1097

```

1098 **Figure 9. <prov:PMDelete> — Schema Fragment**

1099 An example message body containing a <prov:PMDelete> message follows.

```

1100 <prov:PMDelete>
1101   <prov:PMDeleteItem itemID="1">
1102     <prov:PMID issuer="http://provs-r-us.com">uuid:239032-230328-92379-2397923</prov:PMID>
1103   </prov:PMDeleteItem>
1104   <prov:PMDeleteItem itemID="2">
1105     <prov:PMID issuer="http://provs-r-us.com">uuid:392380-948492-18923-2389238</prov:PMID>
1106   </prov:PMDeleteItem>
1107 </prov:PMDelete>
1108

```

1109 **Example 9. Example <prov:PMDelete> Message**

1110 4.7.3. PMDeleteResponse Message

1111 This response to the <prov:PMDelete> request contains the following elements/attributes:

- 1112 • <lu:Status> **[Required]** - the status of the response. See the processing rules below for more information.
- 1113 • anyAttribute **[Optional]** Zero or more attributes from a namespace other than that of this specification. One
- 1114 such possibility is an **xs:ID** type attribute such as xml:id or wsu:Id.

```

1115 <!-- PMDeleteResponse - the response to the PMDelete request -->
1116
1117 <xs:element name="PMDeleteResponse" type="PMDeleteResponseType"/>
1118
1119 <xs:complexType name="PMDeleteResponseType">
1120   <xs:complexContent>
1121     <xs:extension base="ResponseAbstractType" />
1122   </xs:complexContent>
1123 </xs:complexType>
1124

```

1125 **Figure 10. <prov:PMDeleteResponse> — Schema Fragment**

1126 An example message body containing a <PMDeleteResponse> message follows. In this case, the response indicates
 1127 a partial success and the detailed status codes which indicate that the second PMID was not found.

```

1128 <prov:PMDeleteResponse>
1129   <lu:Status code="Partial">
1130     <lu:Status ref="1" code="NotFound"/>
1131   </lu:Status>
1132 </prov:PMDeleteResponse>
1133

```

1134 **Example 10. Example <prov:PMDeleteResponse> Message**

1135 4.7.4. PMDelete Processing Rules

- 1136 • This operation adopts the Delayed Notification design pattern (see [\[LibertyDP\]](#)) and incorporates all of the
- 1137 associated processing rules. Delayed notifications are needed for both indirect operations (where the PMM is
- 1138 polling for incoming request) and for delayed operations (where the at attribute is used with a time in the future).
- 1139 Delayed Notifications SHOULD NOT to be used for immediate operations (no future at attribute) targeted at
- 1140 a PMM that has exposed a means of direct communications from the ProvS (e.g., they have registered a non-
- 1141 anonymous CallbackEPR with the ProvS). In such cases, the ProvS SHOULD wait for the completion of the
- 1142 subsequent request to the PMM and return the appropriate status in its response to the invoker.
- 1143 If any portion of the request IS NOT an immediate operation and the <dp:NotifyTo> element IS NOT present
- 1144 on the request, the ProvS should validate the request to the extent possible and respond with the results of that
- 1145 validation. If that validation succeeded, the ProvS MUST continue with the execution of the operation. In such
- 1146 cases, the actual completion status of the operation may be different from what was reported (e.g., the request may
- 1147 fail at the PMM for some reason), but that will not be known to the invoker, given that they have chosen to not
- 1148 provide a delayed notification endpoint.
- 1149 If any portion of the request IS NOT an immediate operation and the <dp:NotifyTo> element IS present on the
- 1150 request, the ProvS MUST report the final completion status (e.g., wait for the completion status of any delayed or
- 1151 indirect operation at the PMM) of the request. If multiple operations are included in the request, the ProvS MAY
- 1152 group the results of some or all of the operations in a single delayed notification message.
- 1153 In the case of a mixed request, the immediate portion of the request SHOULD be handled as an immediate request
- 1154 and the actual results returned with the initial response.

- 1155 • Requests to delete a PM that is not known to the ProvS MUST result in a failure. If detailed status codes are
1156 included in the response, the detailed status code for this error MUST be "NotFound."
- 1157 • If the PM has been provisioned into a PMM, the ProvS MUST initiate a <pmm:PMDelete> operation at the PMM
1158 to delete the provisioned PM. This request SHOULD NOT be considered complete until the deletion is completed
1159 at the PMM.
- 1160 • The ProvS SHOULD accept and forward this request regardless of the current state of the PMDescriptor (provided
1161 that it does exist and has not already been deleted).
- 1162 • If **all** items in the request have the same completion status, the top level status MUST reflect that completion status
1163 and MUST be "OK," "Failed," or "WillNotify." Otherwise, if the results were mixed, the top-level status MUST be
1164 *Partial* and a second-level status MUST be included indicating which items succeeded, which failed, and which
1165 will be subject to delayed notification. The second level status elements MUST include the *ref* attribute containing
1166 the *itemID* value for the item. For failures, the second-level status codes MAY simply be "Failed" or they may
1167 indicate, with more detail, the reason for the failure.
- 1168 If the top-level status is "Failed," second level status codes MAY be present which contain detailed error
1169 information if the ProvS wants to share that information with the invoking party.

1170 **4.8. Operation: PMGetStatus**

1171 The *PMGetStatus* operation is used by a PM-issuing party (such as an IdP) to obtain the current status of an issued
1172 PM.

1173 **4.8.1. wsa:Action values for PMGetStatus Messages**

1174 <PMGetStatus> request messages MUST include a <wsa:Action> SOAP header with the value of
1175 "urn:liberty:prov:2007-09:PMGetStatus."

1176 <PMGetStatusResponse> messages MUST include a <wsa:Action> SOAP header with the value of
1177 "urn:liberty:prov:2007-09:PMGetStatusResponse."

1178 **4.8.2. PMGetStatus Message**

1179 The <PMGetStatus> request is called to obtain the status of a PM that was registered with the ProvS.

1180 The <prov:PMGetStatus> request contains the following elements/attributes:

- 1181 • <PMID> [**Optional**] - zero or more identifiers for the PMs who's status is requested. If specified, this element
1182 MUST contain a value that matches the *PMID* of a previously registered <PMDescriptor>.
- 1183 If not specified, the ProvS should return the status for all PMDescriptors registered by the invoker (if any).
- 1184 • *anyAttribute* [**Optional**] - zero or more attributes from a namespace other than that of this specification. One
1185 such possibility is an *xs:ID* type attribute such as *xml:id* or *wsu:Id*.

1186 The schema for the `<prov:PMGetStatus>` is shown below.

```
1187 <!-- PMStatus - to get the status of a PM at the Provs -->
1188
1189 <xs:element name="PMGetStatus" type="PMGetStatusType"/>
1190
1191 <xs:complexType name="PMGetStatusType">
1192   <xs:complexContent>
1193     <xs:extension base="RequestAbstractType">
1194       <xs:sequence>
1195         <xs:element ref="PMID" minOccurs="0" maxOccurs="unbounded"/>
1196       </xs:sequence>
1197     </xs:extension>
1198   </xs:complexContent>
1199 </xs:complexType>
1200
```

1201 **Figure 11. `<prov:PMGetStatus>` — Schema Fragment**

1202 An example message body containing a `<prov:PMGetStatus>` message follows.

```
1203 <prov:PMGetStatus>
1204   <prov:PMID issuer="http://provs-r-us.com">uuid:239032-230328-92379-2397923</prov:PMID>
1205   <prov:PMID issuer="http://provs-r-us.com">uuid:392380-948492-18923-2389238</prov:PMID>
1206 </prov:PMGetStatus>
1207
```

1208 **Example 11. Example `<prov:PMGetStatus>` Message**

1209 **4.8.3. `PMGetStatusResponse` Message**

1210 This response to the `<prov:PMGetStatus>` request contains the following elements/attributes:

- 1211 • `<lu:Status>` **[Required]** - the status of the response. See the processing rules below for more information.
- 1212 • `<prov:PMStatus>` **[Optional]** - zero or more elements describing the status of the requested PMs.
- 1213 • `anyAttribute` **[Optional]** Zero or more attributes from a namespace other than that of this specification. One
- 1214 such possibility is an **xs:ID** type attribute such as `xml:id` or `wsu:Id`.

```

1215 <!-- PMGetStatusResponse - response to the PMGetStatus request -->
1216
1217 <xs:element name="PMGetStatusResponse" type="PMGetStatusResponseType"/>
1218
1219 <xs:complexType name="PMGetStatusResponseType">
1220   <xs:complexContent>
1221     <xs:extension base="ResponseAbstractType">
1222       <xs:sequence>
1223         <xs:element ref="PMStatus" minOccurs="0" maxOccurs="unbounded"/>
1224       </xs:sequence>
1225     </xs:extension>
1226   </xs:complexContent>
1227 </xs:complexType>
1228

```

1229 **Figure 12. <prov:PMGetStatusResponse> — Schema Fragment**

1230 An example message body containing a <PMGetStatusResponse> message follows. In this successful response,
1231 the status of one PM is running and the other PM was not found.

```

1232 <prov:PMGetStatusResponse>
1233   <lu:Status code="OK" />
1234   <prov:PMStatus>
1235     <prov:PMID issuer="http://provs-r-us.com">uuid:239032-230328-92379-23 97923</prov:PMID>
1236     <prov:State asof="2007-01-14T17:31:11Z">urn:liberty:prov:2007-09:status:Active</prov:State>
1237   </prov:PMStatus>
1238 </prov:PMGetStatusResponse>
1239

```

1240 **Example 12. Example <prov:PMGetStatusResponse> Message**

1241 4.8.4. PMGetStatus Processing Rules

- 1242 • The ProvS MUST silently ignore <PMID> values of which it is not aware. If all of the <PMID> values specified
1243 on the request are unknown to the ProvS, the ProvS should return a successful response with no status records.
- 1244 In other words, specifying a non-existent <PMID> does not result in an error. Instead it results in nothing in the
1245 output.
- 1246 • When this interface is used by a party other than the PMD-issuing authority or the PMM where the PM was
1247 provisioned, the ProvS MUST treat the PMD as if the ProvS was not aware of it (i.e., the PMD does not exist as
1248 far as other invoking parties are concerned).
- 1249 • If there are NO PMIDs specified in the request, the ProvS MUST return the current status of all PMs that have
1250 been registered by the invoker.
- 1251 This rule does NOT apply if PMIDs are specified which refer to nonexistent PMs.
- 1252 • If request processing succeeded for **any** of the requested PMDs, the top-level status code MUST be "OK." If the
1253 request processing failed for all PMDs, the top-level status code MUST be "Failed." Partial results, where some
1254 of the items were found and some were not found, are considered a successful response that only include the
1255 <prov:PMStatus> element(s) for the PMDs that were found.
- 1256 • If the top-level status is not OK. and second level status codes are present, they MAY contain detailed error
1257 information if the ProvS wants to share that information with the invoking party.

1258 **4.9. Operation: *PMRegisterDescriptor***

1259 The *PMRegisterDescriptor* operation is used by a PM-issuing party (such as an IdP) to register PMDs with the ProvS
1260 and obtain the PHs that it can issue to the PMM.

1261 **4.9.1. *wsa:Action* values for *PMRegisterDescriptor* Messages**

1262 <PMRegisterDescriptor> request messages MUST include a <wsa:Action> SOAP header with the value of
1263 "urn:liberty:prov:2007-09:PMRegisterDescriptor".

1264 <PMRegisterDescriptorResponse> messages MUST include a <wsa:Action> SOAP header with the value of
1265 "urn:liberty:prov:2007-09:PMRegisterDescriptorResponse."

1266 **4.9.2. *PMRegisterDescriptor* Message**

1267 The <PMRegisterDescriptor> request is called to register a new PMD with the ProvS.

1268 The <prov:PMRegisterDescriptor> request contains one or more <prov:PMRegisterDescriptorItem>
1269 elements which have the following contents:

- 1270 • <PMDescriptor> [**Required**] - the PMD that is being registered.
- 1271 • itemID [**Required**] - a unique (within this message) identifier for this request item (used for correlation of result
1272 status in the response).

1273 The schema for the <prov:PMRegisterDescriptor> is shown below.

```
1274 <!-- PMRegisterDescriptor - to register a new PMD at the ProvS -->
1275
1276 <xs:element name="PMRegisterDescriptor" type="PMRegisterDescriptorType"/>
1277
1278 <xs:complexType name="PMRegisterDescriptorType">
1279   <xs:complexContent>
1280     <xs:extension base="RequestAbstractType">
1281       <xs:sequence>
1282         <xs:element ref="PMRegisterDescriptorItem" maxOccurs="unbounded" />
1283       </xs:sequence>
1284     </xs:extension>
1285   </xs:complexContent>
1286 </xs:complexType>
1287
1288 <xs:element name="PMRegisterDescriptorItem" type="PMRegisterDescriptorItemType" />
1289
1290 <xs:complexType name="PMRegisterDescriptorItemType">
1291   <xs:sequence>
1292     <xs:element ref="PMDescriptor" />
1293   </xs:sequence>
1294   <xs:attribute name="itemID" type="xs:string" use="required" />
1295 </xs:complexType>
1296
```

1297 **Figure 13. <prov:PMRegisterDescriptor> — Schema Fragment**

1298 An example message body containing a <prov:PMRegisterDescriptor> message follows.

```

1299 <prov:PMRegisterDescriptor>
1300   <prov:PMRegisterDescriptorItem itemID="1">
1301     <prov:PMDescriptor xs:id="2323923900239">
1302       <prov:PMID issuer="http://provs-r-us.com">uuid:2 39032-230328-92379-2397923</pr ov:PMID>
1303       <prov:PMEngineRef>https://pmsRus.org/VeryTrustedModule/3.7</prov:PMEngi neRef>
1304     <prov:PMInitData>
1305       <MyData>
1306         .... initialization data here ...
1307       </MyData>
1308     </prov:PMInitData>
1309     <ds:Signature>
1310       ... signature data goes here ...
1311     </ds:Signature>
1312   </prov:PMDescriptor>
1313 </prov:PMRegisterDescriptorItem>
1314 </prov:PMRegisterDescriptor>
1315

```

1316 **Example 13. Example <prov:PMRegisterDescriptor> Message**

1317 4.9.3. PMRegisterDescriptorResponse Message

1318 This response to the <prov:PMRegisterDescriptor> request contains the following elements/attributes:

- 1319 • <lu:Status> **[Required]** - the status of the response. See the processing rules below for more information.
- 1320 • <prov:PMRegisterDescriptorResponseItem> **[Optional]** - zero or more response items which are in-
1321 cluded for successful responses. If present, this element contains the following content:
 - 1322 • <prov:ProvisioningHandle> **[Required]** - the Provisioning Handle created by the ProvS to provide a
1323 PMM with the information necessary to obtain the PMD that was registered.
 - 1324 • ref **[Required]** - a reference to the itemID of the <prov:PMRegisterDescriptorItem> element that
1325 this <prov:PMRegisterDescriptorResponseItem> is in response to.
 - 1326 • anyAttribute **[Optional]** Zero or more attributes from a namespace other than that of this specification. One
1327 such possibility is an **xs:ID** type attribute such as xml:id or wsu:Id.

```

1328 <!-- PMRegisterDescriptorResponse - response to the PMRegisterDescriptor request -->
1329
1330 <xs:element name="PMRegisterDescriptorResponse" type="PMRegisterDescriptorResponseType"/>
1331
1332 <xs:complexType name="PMRegisterDescriptorResponseType">
1333   <xs:complexContent>
1334     <xs:extension base="ResponseAbstractType">
1335       <xs:sequence>
1336         <xs:element ref="PMRegisterDescriptorResponseItem" minOccurs="0"
1337           maxOccurs="unbounded"/>
1338       </xs:sequence>
1339     </xs:extension>
1340   </xs:complexContent>
1341 </xs:complexType>
1342
1343 <xs:element name="PMRegisterDescriptorResponseItem"
1344   type="PMRegisterDescriptorResponseItem"/>
1345
1346 <xs:complexType name="PMRegisterDescriptorResponseItem">
1347   <xs:sequence>
1348     <xs:element ref="ProvisioningHandle"/>
1349   </xs:sequence>
1350   <xs:attribute name="ref" type="xs:string" use="required"/>
1351 </xs:complexType>
1352

```

1353 **Figure 14. <prov:PMRegisterDescriptorResponse> — Schema Fragment**

1354 An example message body containing a <PMRegisterDescriptorResponse> message follows.

```

1355 <prov:PMRegisterDescriptorResponse>
1356   <lu:Status code="OK" />
1357   <prov:PMRegisterDescriptorResponseItem ref="1">
1358     <prov:ProvisioningHandle xs:id="2302384823023">
1359       <prov:PMDArtifact>23asdfhoi323hpos df923h9sdfhweorh2398asdfjweoiha</prov:PMDArtifact>
1360       <prov:ProvisioningServiceEPR>
1361         <wsa:Address>http://provision.idpsRus.com</wsa:Address>
1362         <wsa:Metadata>
1363           <ds:Abstract>Provisioning Service</ds:Abstract>
1364           <ds:ProviderID>http://provisioning-provider.idpsRus.com/</ds:ProviderID>
1365           <ds:ServiceType>urn:liberty:prov:2007-09</ds:ServiceType>
1366           <ds:Framework version="2.0" />
1367           <ds:SecurityContext>
1368             <ds:SecurityMechID>
1369               urn:liberty:security:2005-02:TLS:SAMLV2
1370             </ds:SecurityMechID>
1371             <sec:Token ref="urn:liberty:disco:tokenref:ObtainFromIDP" />
1372           </ds:SecurityContext>
1373         </wsa:Metadata>
1374       </prov:ProvisioningServiceEPR>
1375       <ds:Signature>
1376         ... signature info here ..
1377       </ds:Signature>
1378     </prov:ProvisioningHandle>
1379   </prov:PMRegisterDescriptorResponseItem>
1380 </prov:PMRegisterDescriptorResponse>
1381

```

1382 **Example 14. Example <prov:PMRegisterDescriptorResponse> Message**

1383 **4.9.4. PMRegisterDescriptor Processing Rules**

- 1384 • The invoking party **MUST NOT** specify a value for the <PMID> element in the <PMDescriptor> that duplicates
 1385 a value previously used. In such cases, the request **MUST** result in a failure. If detailed status codes are included
 1386 in the response, the detailed status code for this case **MUST** be *"Duplicate."*
- 1387 • The ProvS **MAY** require that the PMS engine referenced within the <PMDescriptor> first be registered with the
 1388 ProvS before allowing a PMD to be registered. If this is the case, and if an attempt is made to register a
 1389 <PMDescriptor> with an engine that has not been registered, the ProvS **MAY** treat that as a failure. If this is
 1390 the case and if detailed status codes are included in the response, the detailed status code for this case **MUST** be
 1391 *"UnknownEngine"*.
- 1392 • Each <prov:PMRegisterDescriptorItem> is processed independently and may independently succeed or
 1393 fail on its own merits.
- 1394 • If all <prov:PMRegisterDescriptorItem> requests are successfully processed, the top-level status code
 1395 **MUST** be *"OK."* If all of the items failed, the top-level status code **MUST** be *"Failed."* Otherwise, if the results
 1396 were mixed, the top-level status **MUST** be *"Partial"* and the second level status **MUST** be included for items for
 1397 which the processing was not successful indicating so and including the ref attribute containing the itemID value
 1398 for the item. These second-level status codes **MAY** simply be *"Failed,"* or they may indicate, with more detail,
 1399 the reason for the failure.
- 1400 • If the top-level status code is *"Failed,"* the response **MAY** also contain other status codes (such as *NotFound*) as
 1401 a second-level status code. The ProvS instance may not wish to reveal the reason for failure, in which case no
 1402 second-level status code will appear.

1403 **4.10. Operation: *PMSetStatus***

1404 The *PMSetStatus* operation is used by the PMM to notify the Provisioning service of the status of the PM (whether it
 1405 was successfully provisioned or not).

1406 **4.10.1. *wsa:Action* values for *PMSetStatus* Messages**

1407 <PMSetStatus> request messages MUST include a <wsa:Action> SOAP header with the value of
 1408 "urn:liberty:prov:2007-09:PMSetStatus".

1409 <PMSetStatusResponse> messages MUST include a <wsa:Action> SOAP header with the value of
 1410 "urn:liberty:prov:2007-09:PMSetStatusResponse."

1411 **4.10.2. *PMSetStatus* Message**

1412 The <PMSetStatus> request is called to notify the Provisioning Service about the status of the provisioning of a
 1413 PM.

1414 The <prov:PMSetStatus> request contains following attributes and/or elements:

- 1415 • <PMStatus> - the updated status (see [Section 3.4](#)) of the PM. The <State> element SHOULD NOT include an
 1416 asof attribute – the ProvS will assign the time itself.
- 1417 • anyAttribute [**Optional**] Zero or more attributes from a namespace other than that of this specification. One
 1418 such possibility is an **xs:ID** type attribute such as xml:id or wsu:Id.

1419 The schema for the <prov:PMSetStatus> is shown below.

```

1420 <!-- PMSetStatus - update provisioning status of PM -->
1421
1422 <xs:element name="PMSetStatus" type="PMSetStatusType"/>
1423
1424 <xs:complexType name="PMSetStatusType">
1425   <xs:complexContent>
1426     <xs:extension base="RequestAbstractType">
1427       <xs:sequence>
1428         <xs:element ref="PMStatus"/>
1429       </xs:sequence>
1430     </xs:extension>
1431   </xs:complexContent>
1432 </xs:complexType>
1433
    
```

1434 **Figure 15. <prov:PMSetStatus> — Schema Fragment**

1435 An example message body containing a <PMSetStatus> message follows. This request shows that the PM has been
 1436 installed.

```

1437 <prov:PMSetStatus>
1438   <prov:PMStatus>
1439     <prov:PMID issuer="http://provs-r-us.com">uui d:239032-230328-92379-2397923 </prov:PMID>
1440     <prov:State>urn:liberty:prov:2007-09: status:Active</prov:State>
1441   </prov:PMStatus>
1442 </prov:PMSetStatus>
1443
    
```

1444 **Example 15. Example <prov:PMSetStatus> Message**

1445 **4.10.3. *PMSetStatusResponse* Message**

1446 This response to the <prov:PMSetStatus> request contains the following elements:

- 1447 • `<lu:Status>` [**Required**] - the status of the response. See the processing rules below for more information.
- 1448 • `anyAttribute` [**Optional**] - zero or more attributes from a namespace other than that of this specification. One
- 1449 such possibility is an `xs:ID` type attribute such as `xml:id` or `wsu:Id`.

```

1450 <!-- PMSetStatusResonse - response for PMSetStatus request -->
1451
1452 <xs:element name="PMSetStatusResponse" type="PMSetStatusResponseType"/>
1453
1454 <xs:complexType name="PMSetStatusResponseType">
1455   <xs:complexContent>
1456     <xs:extension base="ResponseAbstractType" />
1457   </xs:complexContent>
1458 </xs:complexType>
1459

```

1460 **Figure 16.** `<prov:PMSetStatusResponse>` — Schema Fragment

1461 An example message body containing a `<PMSetStatusResponse>` message follows. This is a successful response.

```

1462 <prov:PMSetStatusResponse>
1463   <lu:Status code="OK" />
1464 </prov:PMSetStatusResponse>
1465

```

1466 **Example 16. Example** `<prov:PMSetStatusResponse>` **Message**

1467 4.10.4. PMSetStatus Processing Rules

- 1468 • If a specified PMID is not known to the ProvS, the ProvS MUST treat that request item as a failure. If detailed
- 1469 status codes are being included in the response, the detailed status code for this error MUST be *"NotFound."*
- 1470 • The ProvS MUST ensure that only the PMM where the PM is provisioned can use this interface to directly change
- 1471 the status of the PM. Invocations of this interface by any other party MUST result in a failure.
- 1472 If the invoker is the PMD-issuing authority, and detailed status codes are being included in the response, the
- 1473 detailed status code for this error MUST be *"Forbidden."* The only way for an issuing authority to change the
- 1474 status of a PMD is indirectly through one of the PMD management interfaces.
- 1475 Otherwise, if detailed status codes are being included in the response, the detailed status code for this error MUST
- 1476 be *"NotFound."*
- 1477 • If the `<State>` element within the `<PMStatus>` contains an `asof` attribute, the ProvS MUST ignore this value
- 1478 and, instead, use its understanding of the current time to mark the status time.
- 1479 • If the `<State>` element within the `<PMStatus>` contains a value that is not understood by the ProvS, the ProvS
- 1480 MAY treat the operation as a failure. ProvSs MUST accept all of the Liberty-defined status values specified in
- 1481 this document.
- 1482 If the request is to be treated as a failure for this reason and detailed status codes are included in the response, the
- 1483 detailed status code for this error MUST be *"Invalid."*
- 1484 • If request processing succeeded, the top-level status code MUST be *"OK."* Otherwise, the top-level status code
- 1485 MUST be *"Failed"*
- 1486 • If the top-level status code is *"Failed,"* the response MAY also contain *Forbidden* or *NotFound* as a second-level
- 1487 status code. The Provisioning Service instance may not wish to reveal the reason for failure. In which case, no
- 1488 second-level status code will appear.

1489 4.11. Operation: *PMUpdate*

1490 The *PMUpdate* operation is used by a PM-issuing party (such as an IdP) to update previously registered PMDs at the
1491 ProvS. The previously registered PMDs MAY have already been provisioned into running PMs (in which case, the
1492 ProvS would initiate an update process with the PMM where the PM has been provisioned).

1493 4.11.1. *wsa:Action* values for *PMUpdate* Messages

1494 <PMUpdate> request messages MUST include a <wsa:Action> SOAP header with the value of
1495 "urn:liberty:prov:2007-09:PMUpdate".

1496 <PMUpdateResponse> messages MUST include a <wsa:Action> SOAP header with the value of
1497 "urn:liberty:prov:2007-09:PMUpdateResponse".

1498 4.11.2. *PMUpdate* Message

1499 The <PMUpdate> request is called to update a PMD that was registered with the ProvS.

1500 The <prov:PMUpdate> request contains the following elements/attributes:

1501 • <prov:PMUpdateItem> **[Required]** - one or more update actions which contain the following
1502 elements/attributes:

1503 • <PMDDescriptor> **[Required]** - the updated PMD information. The <PMID> element MUST contain a value
1504 that matches the <PMID> of a previously registered <PMDDescriptor>.

1505 In some cases, only a portion of the PM is being updated (such as a PMEngine update). Even so, the entire
1506 replacement <PMDDescriptor> must be specified (as opposed to just the changed elements).

1507 The *activate*, *activateAt*, and *deactivateAt* attributes **SHOULD NOT** be specified when a
1508 <PMDDescriptor> is used in an update request. The standard interfaces for managing the activation status
1509 are the only way to update/change the activation status of a PMD.

1510 • *type* **[Required]** - the type of update being applied. The following values are defined for this attribute:

1511 • *urn:liberty:prov:2007-09:ut:replace* - a complete replacement of the existing PMD (and if it has been
1512 provisioned, the PM itself).

1513 • *urn:liberty:prov:2007-09:ut:engine* - an update of the engine, only.

1514 • *urn:liberty:prov:2007-09:ut:initdata* - an update of the initialization data.

1515 • *urn:liberty:prov:2007-09:ut:cancel* - the cancellation of a pending future update. If this type is specified,
1516 the *at* attribute **MUST** be specified, **MUST** contain the exact same value that was used on the update
1517 that is being canceled, and the <PMDDescriptor> element **MUST ONLY** include the <PMID> element
1518 corresponding to the PMID used in the pending update.

1519 • *itemID* **[Required]** - the identifier for this request item (for correlation within the results).

1520 • **at** [**Optional**] - an optional time at which the update should take place. If specified, this SHOULD be some
1521 time in the future.

1522 This attribute MUST NOT be interpreted or acted upon by the ProvS (with the potential exception of optional
1523 scheduling of multiple queued operations). The subsequent `<pmm:PMUpdate>` request to the PMM MUST
1524 include this value and SHOULD be sent as soon as possible (e.g., do NOT wait till the specified time to deliver
1525 the message).

1526 Note that since this request is made to the ProvS and not the PMM, the time it takes to relay the message,
1527 especially in a scenario where the PMM is polling the ProvS, may delay the update time, especially when this
1528 attribute has a very short time frame.

1529 • `<dp:NotifyTo>` [**Optional**] - an optional endpoint reference for delayed notification completion messages (see
1530 [\[LibertyDP\]](#)). This is used to allow the invoker to receive a completion status on the eventual activation of the PM
1531 at the PMM.

1532 • `anyAttribute` [**Optional**] Zero or more attributes from a namespace other than that of this specification. One
1533 such possibility is an `xs:ID` type attribute such as `xml:id` or `wsu:Id`.

1534 The schema for the `<prov:PMUpdate>` is shown below.

```

1535 <!-- PMUpdate - update the PM for a existing PM at the ProvS -->
1536
1537 <xs:element name="PMUpdate" type="PMUpdateType"/>
1538
1539 <xs:complexType name="PMUpdateType">
1540   <xs:complexContent>
1541     <xs:extension base="RequestAbstractType">
1542       <xs:sequence>
1543         <xs:element ref="PMUpdateItem" maxOccurs="unbounded" />
1544         <xs:element ref="dp:NotifyTo" minOccurs="0" />
1545       </xs:sequence>
1546     </xs:extension>
1547   </xs:complexContent>
1548 </xs:complexType>
1549
1550 <xs:element name="PMUpdateItem" type="PMUpdateItemType" />
1551
1552 <xs:complexType name="PMUpdateItemType">
1553   <xs:sequence>
1554     <xs:element ref="PMDescriptor"/>
1555   </xs:sequence>
1556   <xs:attribute name="type" type="xs:anyURI" use="required"/>
1557   <xs:attribute name="itemID" type="xs:string" use="required"/>
1558   <xs:attribute name="at" type="xs:dateTime" use="optional"/>
1559 </xs:complexType>
1560

```

1561 **Figure 17. `<prov:PMUpdate>` — Schema Fragment**

1562 An example message body containing a `<prov:PMUpdate>` message follows. This is an update of the PM Engine to
1563 version 4.0.

```

1564 <prov:PMUpdate>
1565   <prov:PMUpdateItem itemID="1" type="urn:liberty:prov:2007-09:ut:engine">
1566     <prov:PMDescriptor xs:id="2323923900239" >
1567       <prov:PMID issuer="http://provs-r-us.com">uuid:778349-283920-88379-5448739</prov:PMID>
1568       <prov:PMEngineRef>https://pmsRus.org/VeryTrustedModule/4.0</prov:PMEngineRef>
1569       <ds:Signature>
1570         ... signature data goes here ...
1571       </ds:Signature>
1572     </prov:PMDescriptor>
1573   </prov:PMUpdateItem>
1574 </prov:PMUpdate>
1575

```

1576 **Example 17. Example <prov:PMUpdate> Message**

1577 4.11.3. PMUpdateResponse Message

1578 This response to the <prov:PMUpdate> request contains the following elements/attributes:

- 1579 • <lu:Status> **[Required]** - the status of the response. See the processing rules below for more information.
- 1580 • anyAttribute **[Optional]** Zero or more attributes from a namespace other than that of this specification. One
- 1581 such possibility is an **xs:ID** type attribute such as `xml:id` or `wsu:Id`.

```

1582 <!-- PMUpdateResponse - response to the PMUpdate request -->
1583
1584 <xs:element name="PMUpdateResponse" type="PMUpdateResponseType"/>
1585
1586 <xs:complexType name="PMUpdateResponseType">
1587   <xs:complexContent>
1588     <xs:extension base="ResponseAbstractType"/>
1589   </xs:complexContent>
1590 </xs:complexType>
1591

```

1592 **Figure 18. <prov:PMUpdateResponse> — Schema Fragment**

1593 An example message body containing a <PMUpdateResponse> message follows. In this case, the response indicates
 1594 a failure and the detailed status code indicates the PMID was not found.

```

1595 <prov:PMUpdateResponse>
1596   <lu:Status code="Failed">
1597     <lu:Status ref="1" code="NotFound" />
1598   </lu:Status>
1599 </prov:PMUpdateResponse>
1600

```

1601 **Example 18. Example <prov:PMUpdateResponse> Message**

1602 4.11.4. PMUpdate Processing Rules

- 1603 • This operation adopts the Delayed Notification design pattern (see [LibertyDP]) and incorporates all of the
1604 associated processing rules. Delayed notifications are needed for both indirect operations (where the PMM is
1605 polling for incoming request) and for delayed operations (where the `at` attribute is used with a time in the future).
- 1606 Delayed Notifications SHOULD NOT to be used for immediate operations (no future `at` attribute) targeted at
1607 a PMM that has exposed a means of direct communications from the ProvS (e.g., they have registered a non-
1608 anonymous CallbackEPR with the ProvS). In such cases, the ProvS SHOULD wait for the completion of the
1609 subsequent request to the PMM and return the appropriate status in its response to the invoker.
- 1610 If any portion of the request IS NOT an immediate operation and the `<dp:NotifyTo>` element IS NOT present
1611 on the request, the ProvS should validate the request to the extent possible and respond with the results of that
1612 validation. If that validation succeeded, the ProvS MUST continue with the execution of the operation. In such
1613 cases, the actual completion status of the operation may be different from what was reported (e.g., the request may
1614 fail at the PMM for some reason), but that will not be known to the invoker, given that they have chosen to not
1615 provide a delayed notification endpoint.
- 1616 If any portion of the request IS NOT an immediate operation and the `<dp:NotifyTo>` element IS present on the
1617 request, the ProvS MUST report the final completion status (e.g., wait for the completion status of any delayed or
1618 indirect operation at the PMM) of the request. If multiple operations are included in the request, the ProvS MAY
1619 group the results of some or all of the operations in a single delayed notification message.
- 1620 In the case of a mixed request, the immediate portion of the request SHOULD be handled as an immediate request
1621 and the actual results returned with the initial response.
- 1622 • The invoking party MUST specify a value for the `<PMID>` element in the `<PMDescriptor>` that matches a value
1623 previously used **by this issuing party**. If this is not the case, the ProvS MUST reject the update request and return
1624 a failure status code. If detailed status codes are included in that message, the detailed status code MUST be
1625 "*NotFound*."
- 1626 • If a cancellation is received for a pending update that has NOT been transmitted to the PMM yet, the ProvS
1627 SHOULD remove the pending update without transmitting it to the PMM. If the invoker of the update has
1628 asked for delayed notification, the notification message for the canceled update MUST include a failure status
1629 code. If detailed status codes are included in the delayed notification message, the detailed status code MUST be
1630 "*Canceled*." In such case, the response to the cancellation request itself SHOULD indicate success.
- 1631 • If a cancellation is received for an update that is not currently pending (already completed, never seen, etc.), the
1632 request MUST result in a failure. If detailed status codes are included in the delayed notification message, the
1633 detailed status code MUST be "*NotFound*."
- 1634 Update requests that have been forwarded to the PMM, but which are not known by the ProvS to have been
1635 completed, are considered to be pending and the cancellation request MUST be forwarded to the PMM.
- 1636 • If an update request includes any `activate`, `activateAt`, or `deactivateAt` attributes within the
1637 `<PMDescriptor>`, these attributes SHOULD be ignored. The update request cannot be used to change
1638 the activation status of a PMD.
- 1639 • Each `<prov:PMUpdateItem>` is processed independently and may independently succeed or fail on its own
1640 merits.
- 1641 • If **all** items in the request have the same completion status, the top level status MUST reflect that completion status
1642 and MUST be "*OK*," "*Failed*," or "*WillNotify*." Otherwise, if the results were mixed, the top-level status MUST be
1643 *Partial* and a second-level status MUST be included indicating which items succeeded, which failed, and which
1644 will be subject to delayed notification. The second level status elements MUST include the `ref` attribute containing
1645 the `itemID` value for the item. For failures, the second-level status codes MAY simply be "*Failed*" or they may
1646 indicate, with more detail, the reason for the failure.
- 1647 If the top-level status is "*Failed*," second level status codes MAY be present which contain detailed error
1648 information if the ProvS wants to share that information with the invoking party.

1649 **4.12. Operation: *PMDelete***

1650 The *PMDelete* operation is used by the PM-issuing authority to delete an existing PMEngine at the ProvS.

1651 **4.12.1. *wsa:Action* values for *PMDelete* Messages**

1652 <PMDelete> request messages MUST include a <wsa:Action> SOAP header with the value of
1653 "urn:liberty:prov:2007-09:PMDelete."

1654 <PMDeleteResponse> messages MUST include a <wsa:Action> SOAP header with the value of
1655 "urn:liberty:prov:2007-09:PMDeleteResponse."

1656 **4.12.2. *PMDelete* Message**

1657 The <PMDelete> request is called to delete the specified PMEngine(s) at the ProvS.

1658 The <prov:PMDelete> request contains the following attributes and/or elements:

- 1659 • <PMEngineRef> **[Required]** - one or more of the reference(s) to the PMEngine(s) to be deleted.
- 1660 • anyAttribute **[Optional]** - zero or more attributes from a namespace other than that of this specification. One
1661 such possibility is an **xs:ID** type attribute such as `xml:id` or `wsu:Id`.

1662 The schema for the <prov:PMDelete> is shown below.

```
1663 <!-- PMDelete - retrieve the specified PMEngine -->
1664
1665 <xs:element name="PMDelete" type="PMDeleteType"/>
1666
1667 <xs:complexType name="PMDeleteType">
1668   <xs:complexContent>
1669     <xs:extension base="RequestAbstractType">
1670       <xs:sequence>
1671         <xs:element ref="PMEngineRef" maxOccurs="unbounded"/>
1672       </xs:sequence>
1673     </xs:extension>
1674   </xs:complexContent>
1675 </xs:complexType>
1676
```

1677 **Figure 19. <prov:PMDelete> Schema Fragment**

1678 An example message body containing a <PMDelete> message follows.

```
1679 <prov:PMDelete>
1680   <prov:PMEngineRef>http://pmsRus.org/VeryTrustedModule/3.6</prov:PMEngineRef>
1681   <prov:PMEngineRef>http://pmsRus.org/VeryTrustedModule/3.5</prov:PMEngineRef>
1682 </prov:PMDelete>
1683
```

1684 **Example 19. Example <prov:PMDelete> Message**

1685 **4.12.3. *PMDeleteResponse* Message**

1686 This response to the <prov:PMDelete> request contains the following elements:

- 1687 • <lu:Status> **[Required]** - the status of the response. See the processing rules below for more information.

- 1688 • `anyAttribute` **[Optional]** - zero or more attributes from a namespace other than that of this specification. One
 1689 such possibility is an `xs:ID` type attribute such as `xml:id` or `wsu:Id`.

```

1690 <!-- PMEDeleteResponse - response for the PMEDelete Request -->
1691
1692 <xs:element name="PMEDeleteResponse" type="PMEDeleteResponseType"/>
1693
1694 <xs:complexType name="PMEDeleteResponseType">
1695   <xs:complexContent>
1696     <xs:extension base="ResponseAbstractType"/>
1697   </xs:complexContent>
1698 </xs:complexType>
1699
  
```

1700 **Figure 20. <prov:PMEDeleteResponse> Schema Fragment**

1701 An example message body containing a `<PMEDeleteResponse>` message follows. This is a successful response.

```

1702 <prov:PMEDeleteResponse>
1703   <lu:Status code="OK" />
1704 </prov:PMEDeleteResponse>
1705
  
```

1706 **Example 20. Example <prov:PMEDeleteResponse> Message**

1707 4.12.4. PMEDelete Processing Rules

- 1708 • The ProvS SHOULD take steps to ensure that the correct party (typically the party that registered and uploaded
 1709 the PMEngine) is submitting the request and to reject requests from inappropriate parties. If detailed status codes
 1710 are to be included in the response, the detailed error code for this error MUST be *Forbidden*.
- 1711 • Requests which include a `<PMEngineRef>` that is not known to the ProvS, MUST result in a failure. If detailed
 1712 status codes are to be included in the response, the detailed error code for this error MUST be *NotFound*.
- 1713 • If the engines pointed to by all `<PMEngineRef>` elements in the request are successfully deleted, the top-level
 1714 status code MUST be *OK*. If all of the deletes failed, the top-level status code MUST be *Failed*. Otherwise,
 1715 if the results were mixed, the top-level status MUST be *Partial* and the second level status MUST be included
 1716 for items for which the processing was NOT successful indicating so and including the `ref` attribute containing
 1717 the `<prov:PMEngineRef>` value for the item. These second-level status codes MAY simply be *Failed*, or they
 1718 may indicate, with more detail, the reason for the failure.
- 1719 • If the top-level status code is *Failed* or *Partial*, the response MAY also contain *Forbidden* or *NotFound* as a
 1720 second-level status code. The Provisioning Service instance may not wish to reveal the reason for failure, in which
 1721 case, no second-level status code will appear.

1722 4.13. Operation: *PMEDisable*

1723 The *PMEDisable* operation is used by the PM-issuing authority to disable an existing PMEngine at the ProvS. When
1724 a PMEngine is disabled, it is not available for further downloads.

1725 4.13.1. *wsa:Action* values for *PMEDisable* Messages

1726 <PMEDisable> request messages MUST include a <wsa:Action> SOAP header with the value of
1727 "urn:liberty:prov:2007-09:PMEDisable."

1728 <PMEDisableResponse> messages MUST include a <wsa:Action> SOAP header with the value of
1729 "urn:liberty:prov:2007-09:PMEDisableResponse."

1730 4.13.2. *PMEDisable* Message

1731 The <PMEDisable> request is called to disable the specified PMEngine(s) at the ProvS.

1732 The <prov:PMEDisable> request contains the following attributes and/or elements:

- 1733 • <PMEngineRef> **[Required]** - one or more of the reference(s) to the PMEngine(s) to be disabled.
- 1734 • anyAttribute **[Optional]** - zero or more attributes from a namespace other than that of this specification. One
1735 such possibility is an **xs:ID** type attribute such as `xml:id` or `wsu:Id`.

1736 The schema for the <prov:PMEDisable> is shown below.

```
1737 <!-- PMEDisable - retrieve the specified PMEngine -->
1738
1739 <xs:element name="PMEDisable" type="PMEDisableType"/>
1740
1741 <xs:complexType name="PMEDisableType">
1742   <xs:complexContent>
1743     <xs:extension base="RequestAbstractType">
1744       <xs:sequence>
1745         <xs:element ref="PMEngineRef" maxOccurs="unbounded"/>
1746       </xs:sequence>
1747     </xs:extension>
1748   </xs:complexContent>
1749 </xs:complexType>
1750
```

1751 **Figure 21.** <prov:PMEDisable> Schema Fragment

1752 An example message body containing a <PMEDisable> message follows.

```
1753 <prov:PMEDisable>
1754   <prov:PMEngineRef>http://pmsRus.org/VeryTrustedModule/3.6</prov:PMEngineRef>
1755 </prov:PMEDisable>
1756
```

1757 **Example 21.** Example <prov:PMEDisable> Message

1758 4.13.3. *PMEDisableResponse* Message

1759 This response to the <prov:PMEDisable> request contains the following elements:

- 1760 • <lu:Status> **[Required]** - the status of the response. See the processing rules below for more information.

- 1761 • `anyAttribute` **[Optional]** - zero or more attributes from a namespace other than that of this specification. One
1762 such possibility is an `xs:ID` type attribute such as `xml:id` or `wsu:Id`.

```
1763 <!-- PMEDisableResponse - response for the PMEDisable Request -->
1764
1765 <xs:element name="PMEDisableResponse" type="PMEDisableResponseType"/>
1766
1767 <xs:complexType name="PMEDisableResponseType">
1768   <xs:complexContent>
1769     <xs:extension base="ResponseAbstractType"/>
1770   </xs:complexContent>
1771 </xs:complexType>
1772
```

1773 **Figure 22.** `<prov:PMEDisableResponse>` Schema Fragment

1774 An example message body containing a `<PMEDisableResponse>` message follows. This is a successful response.

```
1775 <prov:PMEDisableResponse>
1776   <lu:Status code="OK" />
1777 </prov:PMEDisableResponse>
1778
```

1779 **Example 22.** Example `<prov:PMEDisableResponse>` Message

1780 4.13.4. PMEDisable Processing Rules

- 1781 • The ProvS SHOULD take steps to ensure that the correct party (typically the party that registered and uploaded
1782 the PMEngine) is submitting the request and to reject requests from inappropriate parties. If detailed status codes
1783 are to be included in the response, the detailed error code for this error MUST be *"Forbidden."*
- 1784 • The ProvS MUST NOT allow subsequent downloads of the specified PMEngine following the completion of this
1785 call and SHOULD abandon any in-progress downloads that have not yet completed.
- 1786 • Requests to disable a PMEngine that is not known to the ProvS MUST result in a failure. If detailed status codes
1787 are to be included in the response, the detailed error code for this error MUST be *"NotFound."*
- 1788 • Requests to disable a PMEngine that is not enabled MUST result in a failure. If detailed status codes are to be
1789 included in the response, the detailed error code for this error MUST be *"Duplicate."*
- 1790 • If the engines pointed to by **ALL** `<PMEngineRef>` elements in the request are successfully disabled, the top-level
1791 status code MUST be *"OK."* If all of the disable operations failed, the top-level status code MUST be *"Failed."*
1792 Otherwise, if the results were mixed, the top-level status MUST be *"Partial"* and the second level status MUST
1793 be included for items for which the processing was NOT successful indicating so and including the `ref` attribute
1794 containing the `<prov:PMEngineRef>` value for the item. These second-level status codes MAY simply be
1795 *"Failed,"* or they may indicate, with more detail, the reason for the failure.
- 1796 • If the top-level status code is *"Failed"* or *"Partial,"* the response MAY also contain *Forbidden* or *NotFound* as a
1797 second-level status code. The Provisioning Service instance may not wish to reveal the reason for failure, in which
1798 case, no second-level status code will appear.

1799 4.14. Operation: *PMEDownload*

1800 The *PMEDownload* operation is used by the PMM to obtain the executable code for a PMSngine that is not currently
1801 instantiated within the PMM's domain.

1802 4.14.1. *wsa:Action* values for *PMEDownload* Messages

1803 <PMEDownload> request messages MUST include a <*wsa:Action*> SOAP header with the value of
1804 "urn:liberty:prov:2007-09:PMEDownload."

1805 <PMEDownloadResponse> messages MUST include a <*wsa:Action*> SOAP header with the value of
1806 "urn:liberty:prov:2007-09:PMEDownloadResponse."

1807 4.14.2. *PMEDownload* Message

1808 The <PMEDownload> request is called to obtain the executable code for a PMSngine referenced by a
1809 <PMSngineRef>.

1810 The <prov:PMEDownload> request contains the following attributes and/or elements:

1811 • <PMSngineRef> [**Required**] - the reference to the desired PMSngine. This is typically taken from the element
1812 with the same name in a PMD that is being provisioned or updated at the PMM.

1813 • *dp:BasicPagingAttributeGroup* [**Optional**] - a set of attributes supporting the pagination of results. This
1814 adaptation of the pagination design pattern from the Liberty ID-WSF Design Patterns [[LibertyDP](#)] specification
1815 uses the Basic Paging model. The objects being downloaded are already static and so there is no need for the
1816 static result set support.

1817 For the purpose of pagination, the "item" being paginated is each byte of the executable being downloaded. So a
1818 pagination request with a *count* of 100 is requesting at most 100 bytes of the executable file. This count refers to
1819 the number of bytes prior to the base64 encoding necessary to place the bytes into a message. (The encoded length
1820 for this data is typically 33% larger.)

1821 • *anyAttribute* [**Optional**] - zero or more attributes from a namespace other than that of this specification. One
1822 such possibility is an *xs:ID* type attribute such as *xml:id* or *wsu:Id*.

1823 The schema for the <prov:PMEDownload> is shown below.

```
1824 <!-- PMEDownload - download the specified PMSngine (or part thereof) -->
1825
1826 <xs:element name="PMEDownload" type="PMEDownloadType"/>
1827
1828 <xs:complexType name="PMEDownloadType">
1829   <xs:complexContent>
1830     <xs:extension base="RequestAbstractType">
1831       <xs:sequence>
1832         <xs:element ref="PMSngineRef"/>
1833       </xs:sequence>
1834       <xs:attributeGroup ref="dp:BasicPagingAttributeGroup"/>
1835     </xs:extension>
1836   </xs:complexContent>
1837 </xs:complexType>
1838
```

1839 **Figure 23.** <prov:PMEDownload> Schema Fragment

1840 An example message body containing a <PMEDownload> message follows. This request asks for the first 100K bytes
1841 of the executable.

```

1842 <prov:PMEDownload count="102400">
1843   <prov:PMEEngineRef>http://pmsRus.org/VeryTrustedModule/4.0</prov:PMEEngineRef>
1844 </prov:PMEDownload>
1845

```

1846 **Example 23. Example <prov:PMEDownload> Message**

1847 A second example message body containing a <PMEDownload> message follows. This request asks for the first
1848 remaining bytes of the executable.

```

1849 <prov:PMEDownload count="83276" offset="102400">
1850   <prov:PMEEngineRef>http://pmsRus.org/VeryTrustedModule/3.7</prov:PMEEngineRef>
1851 </prov:PMEDownload>
1852

```

1853 **Example 24. 2nd Example <prov:PMEDownload> Message**

1854 4.14.3. PMEDownloadResponse Message

1855 This response to the <prov:PMEDownload> request contains the following elements:

- 1856 • <lu:Status> **[Required]** - the status of the response. See the processing rules below for more information.
- 1857 • <prov:EngineData> **[Optional]** - the Base64 encoded bytes of the PMEEngine executable. This MAY contain
1858 only a portion of the complete PMEEngine depending upon the settings for the pagination attributes on the request.
- 1859 • dp:BasicPagingResponseAttributeGroup **[Optional]** - a set of response attributes supporting the pagina-
1860 tion of results. This adaptation of the pagination design pattern from the Liberty ID-WSF Design Patterns [[LibertyDP](#)]
1861 specification uses the Basic Pagination model.
1862 For the purpose of pagination, the "item" being paginated is each byte of the executable being downloaded. So a
1863 response with the pagination attribute `remaining` set to `8192` indicates that there are 8,192 bytes of data left, after
1864 the bytes included in the current response, to complete the download of the executable file.
- 1865 • anyAttribute **[Optional]** - zero or more attributes from a namespace other than that of this specification. One
1866 such possibility is an **xs:ID** type attribute such as `xml:id` or `wsu:Id`.

```

1867 <!-- PMEDownloadResponse - response for the PMEDownload Request -->
1868
1869 <xs:element name="PMEDownloadResponse" type="PMEDownloadResponseType"/>
1870
1871 <xs:complexType name="PMEDownloadResponseType">
1872   <xs:complexContent>
1873     <xs:extension base="ResponseAbstractType">
1874       <xs:sequence>
1875         <xs:element ref="EngineData" minOccurs="0" />
1876       </xs:sequence>
1877       <xs:attributeGroup ref="dp:BasicPagingResponseAttributeGroup"/>
1878     </xs:extension>
1879   </xs:complexContent>
1880 </xs:complexType>
1881
1882 <xs:element name="EngineData" type="xs:base64Binary"/>
1883

```

1884 **Figure 24. <prov:PMEDownloadResponse> Schema Fragment**

1885 An example message body containing a <PMEDownloadResponse> message follows. This is a successful response
1886 which includes the first 100K bytes and indicates that there are an additional 83,276 bytes to go in the file.

```
1887 <prov:PMEDownloadResponse remaining="83276" nextOffset="102400">
1888   <lu:Status code="OK"/>
1889   <prov:EngineData>
1890     ... base64 encoded data (100K worth) ...
1891   </prov:EngineData>
1892 </prov:PMEDownloadResponse>
1893
```

1894 **Example 25. Example <prov:PMEDownloadResponse> Message**

1895 A second example message body containing a <PMEDownloadResponse> message follows. This is a successful
1896 response which includes the remaining 83,276 bytes for the file and indicates that there are no further bytes.

```
1897 <prov:PMEDownloadResponse remaining="0">
1898   <lu:Status code="OK"/>
1899   <prov:EngineData>
1900     ... base64 encoded data (83276 bytes worth) ...
1901   </prov:EngineData>
1902 </prov:PMEDownloadResponse>
1903
```

1904 **Example 26. 2nd Example <prov:PMEDownloadResponse> Message**

1905 4.14.4. PMEDownload Processing Rules

- 1906 • All of the processing rules defined for pagination in the Liberty ID-WSF Design Patterns [[LibertyDP](#)] are
1907 incorporated by reference and must be met.
- 1908 • Requests which include a <PMEngineRef> that is not known to the ProvS, or not enabled for download, MUST
1909 result in a failure. If detailed status codes are to be included in the response, the detailed error code for this error
1910 MUST be "NotFound."
- 1911 • Requests which include a <PMEngineRef> that refers to a PMEngine which has no engine data available (i.e.,
1912 the size was zero in registration indicating that this Engine is built-in or available to the PMM through some other
1913 means) MUST result in a failure. If detailed status codes are to be included in the response, the detailed error code
1914 for this error MUST be "NoDownload."
- 1915 • If a PMEngine becomes disabled following the start of a download, but prior to the completion of the download,
1916 the ProvS SHOULD treat any subsequent requests for the additional segments of the file as a failure. In such
1917 cases, if detailed status codes are to be included, the detailed error code for this error MUST be "NotFound."
- 1918 • If request processing succeeded, the top-level status code MUST be "OK." Otherwise, the top-level status code
1919 MUST be "Failed"
- 1920 • If the top-level status code is "Failed," the response MAY also contain *Forbidden*, *NoDownload*, or *NotFound* as
1921 a second-level status code. The Provisioning Service instance may not wish to reveal the reason for failure, in
1922 which case, no second-level status code will appear.

1923 **4.15. Operation: *PMEEEnable***

1924 The *PMEEEnable* operation is used by the PM-issuing authority to enable downloading of a PMEngine at the ProvS.
1925 This operation MUST NOT take place until the PMEngine has been completely uploaded to the ProvS.

1926 **4.15.1. *wsa:Action* values for *PMEEEnable* Messages**

1927 <PMEEEnable> request messages MUST include a <wsa:Action> SOAP header with the value of
1928 "urn:liberty:prov:2007-09:PMEEEnable."

1929 <PMEEEnableResponse> messages MUST include a <wsa:Action> SOAP header with the value of
1930 "urn:liberty:prov:2007-09:PMEEEnableResponse."

1931 **4.15.2. *PMEEEnable* Message**

1932 The <PMEEEnable> request is called to enable downloading of the specified PMEngine(s) at the ProvS.

1933 The <prov:PMEEEnable> request contains the following attributes and/or elements:

- 1934 • <PMEngineRef> **[Required]** - one or more of the reference(s) to the PMEngine(s) to be enabled.
- 1935 • anyAttribute **[Optional]** - zero or more attributes from a namespace other than that of this specification. One
1936 such possibility is an **xs:ID** type attribute such as `xml:id` or `wsu:Id`.

1937 The schema for the <prov:PMEEEnable> is shown below.

```
1938 <!-- PMEEEnable - retrieve the specified PMEngine -->
1939
1940 <xs:element name="PMEEEnable" type="PMEEEnableType"/>
1941
1942 <xs:complexType name="PMEEEnableType">
1943   <xs:complexContent>
1944     <xs:extension base="RequestAbstractType">
1945       <xs:sequence>
1946         <xs:element ref="PMEngineRef" maxOccurs="unbounded"/>
1947       </xs:sequence>
1948     </xs:extension>
1949   </xs:complexContent>
1950 </xs:complexType>
1951
```

1952 **Figure 25. <prov:PMEEEnable> Schema Fragment**

1953 An example message body containing a <PMEEEnable> message follows.

```
1954 <prov:PMEEEnable>
1955   <prov:PMEngineRef>http://pmsRus.org/VeryTrustedModule/3.6</prov:PMEngineRef>
1956 </prov:PMEEEnable>
1957
```

1958 **Example 27. Example <prov:PMEEEnable> Message**

1959 **4.15.3. *PMEEEnableResponse* Message**

1960 This response to the <prov:PMEEEnable> request contains the following elements:

- 1961 • <lu:Status> **[Required]** - the status of the response. See the processing rules below for more information.

- 1962 • anyAttribute **[Optional]** - zero or more attributes from a namespace other than that of this specification. One
 1963 such possibility is an **xs:ID** type attribute such as `xml:id` or `wsu:Id`.

```

1964 <!-- PMEEEnableResponse - response for the PMEEEnable Request -->
1965
1966 <xs:element name="PMEEEnableResponse" type="PMEEEnableResponseType"/>
1967
1968 <xs:complexType name="PMEEEnableResponseType">
1969   <xs:complexContent>
1970     <xs:extension base="ResponseAbstractType"/>
1971   </xs:complexContent>
1972 </xs:complexType>
1973

```

1974 **Figure 26. <prov:PMEEEnableResponse> Schema Fragment**

1975 An example message body containing a `<PMEEEnableResponse>` message follows. This is a successful response.

```

1976 <prov:PMEEEnableResponse>
1977   <lu:Status code="OK"/>
1978 </prov:PMEEEnableResponse>
1979

```

1980 **Example 28. Example <prov:PMEEEnableResponse> Message**

1981 **4.15.4. PMEEEnable Processing Rules**

- 1982 • The ProvS SHOULD take steps to ensure that the correct party (typically the party that registered and uploaded
 1983 the PMEngine) is submitting the request and to reject requests from inappropriate parties. If detailed status codes
 1984 are to be included in the response, the detailed error code for this error MUST be *"Forbidden."*
- 1985 • Attempts to enable a PMEngine that is already enabled MUST result in a failure. If detailed status codes are to be
 1986 included in the response, the detailed error code for this error MUST be *"Duplicate."*
- 1987 • The ProvS MUST NOT allow a PMEngine to be enabled if the registered size is greater than zero and the engine
 1988 data of the same size is not available. If this is attempted, the enable attempt MUST be treated as a failure. If
 1989 detailed status codes are to be included in the response, the detailed error code for this error MUST be *"WrongSize."*
 1990 Note that how the engine data is available to the ProvS is not an issue here – the data may have been uploaded
 1991 using the `<PMEUpload>` interface or the ProvS may have some other means of acquiring the data. However, the
 1992 size MUST still match regardless of how the engine data is acquired.
- 1993 • If the size specified on the registration of the PMEngine is non-zero, the ProvS MUST NOT allow a PMEngine
 1994 to be enabled when the hash value of the available engine data (calculated using the hash method specified during
 1995 registration) does not match the registered hash value. If the calculated and registered hash values do not match,
 1996 the enable attempt MUST be treated as a failure. If detailed status codes are to be included in the response, the
 1997 detailed error code for this error MUST be *"HashMismatch."*
 1998 The hash digest MUST be calculated on the binary (decoded) bytes of the executable, NOT the Base64 encoded
 1999 bytes.
- 2000 • The ProvS MUST NOT allow subsequent uploads targeted at the specified PMEngine(s) following the completion
 2001 of this call.
- 2002 • Attempts to enable a PMEngine that is not known to the ProvS, MUST result in a failure. If detailed status codes
 2003 are to be included in the response, the detailed error code for this error MUST be *"NotFound."*

- 2004 • If the engines pointed to by **ALL** of the `<PMEngineRef>` elements in the request are successfully enabled, the
2005 top-level status code **MUST** be "OK." If all of the enable operations failed, the top-level status code **MUST** be
2006 "Failed." Otherwise, if the results were mixed, the top-level status **MUST** be "Partial" and the second level status
2007 **MUST** be included for items for which the processing was NOT successful indicating so and including the `ref`
2008 attribute containing the `<prov:PMEngineRef>` value for the item. These second-level status codes **MAY** simply
2009 be "Failed" or they may indicate, with more detail, the reason for the failure.
- 2010 • If the top-level status code is "Failed" or "Partial," the response **MAY** also contain *Forbidden*, *Duplicate*,
2011 *WrongSize*, *HashMismatch*, or *NotFound* as a second-level status code. The Provisioning Service instance may
2012 not wish to reveal the reason for failure, in which case, no second-level status code will appear.

2013 **4.16. Operation: *PMEGetInfo***

2014 The *PMEGetInfo* operation is used to obtain information about a registered PMEngine.

2015 **4.16.1. *wsa:Action* values for *PMEGetInfo* Messages**

2016 `<PMEGetInfo>` request messages **MUST** include a `<wsa:Action>` SOAP header with the value of
2017 "urn:liberty:prov:2007-09:PMEGetInfo."

2018 `<PMEGetInfoResponse>` messages **MUST** include a `<wsa:Action>` SOAP header with the value of
2019 "urn:liberty:prov:2007-09:PMEGetInfoResponse."

2020 **4.16.2. *PMEGetInfo* Message**

2021 The `<PMEGetInfo>` request is called to get information about the specified PMEngine(s) at the ProvS.

2022 The `<prov:PMEGetInfo>` request contains the following attributes and/or elements:

- 2023 • `<PMEngineRef>` [**Optional**] - zero or more reference(s) to the PMEngine(s) for which the current information is
2024 desired. If this element is not specified, the request is a query for all PMEngine(s) registered by the invoker.
- 2025 • `anyAttribute` [**Optional**] - zero or more attributes from a namespace other than that of this specification. One
2026 such possibility is an `xs:ID` type attribute such as `xml:id` or `wsu:Id`.

2027 The schema for the `<prov:PMEGetInfo>` is shown below.

```
2028 <!-- PMEGetInfo - to get information about the specifed PME(s) -->
2029
2030 <xs:element name="PMEGetInfo" type="PMEGetInfoType"/>
2031 <xs:complexType name="PMEGetInfoType">
2032   <xs:complexContent>
2033     <xs:extension base="RequestAbstractType">
2034       <xs:sequence>
2035         <xs:element ref="PMEngineRef" minOccurs="0" maxOccurs="unbounded"/>
2036       </xs:sequence>
2037     </xs:extension>
2038   </xs:complexContent>
2039 </xs:complexType>
2040
```

2041 **Figure 27. `<prov:PMEGetInfo>` Schema Fragment**

2042 An example message body containing a `<PMEGetInfo>` message follows.

```
2043 <prov:PMEGetInfo>
2044   <prov:PMEEngineRef>http://pmsRus.org/VeryTrustedModule/4.0</prov:PMEEngineRef>
2045 </prov:PMEGetInfo>
2046
```

2047 **Example 29. Example <prov:PMEGetInfo> Message**

2048 4.16.3. PMEGetInfoResponse Message

2049 This response to the <prov:PMEGetInfo> request contains the following elements:

- 2050 • <lu:Status> **[Required]** - the status of the response. See the processing rules below for more information.
- 2051 • <prov:PMEInfo> **[Optional]** - zero or more elements containing the information about each PMEngine (see
2052 [Section 3.5](#)).
- 2053 • anyAttribute **[Optional]** - zero or more attributes from a namespace other than that of this specification. One
2054 such possibility is an **xs:ID** type attribute such as xml:id or wsu:Id.

```
2055 <!-- PMEGetInfoResponse - response to the PMEGetInfo request -->
2056
2057 <xs:element name="PMEGetInfoResponse" type="PMEGetInfoResponseType"/>
2058
2059 <xs:complexType name="PMEGetInfoResponseType">
2060   <xs:complexContent>
2061     <xs:extension base="ResponseAbstractType">
2062       <xs:sequence>
2063         <xs:element ref="PMEInfo" minOccurs="0" maxOccurs="unbounded"/>
2064       </xs:sequence>
2065     </xs:extension>
2066   </xs:complexContent>
2067 </xs:complexType>
2068
```

2069 **Figure 28. <prov:PMEGetInfoResponse> Schema Fragment**

2070 An example message body containing a <PMEGetInfoResponse> message follows. This is a successful response.

```
2071 <prov:PMEGetInfoResponse>
2072   <lu:Status code="OK" />
2073   <prov:PMEInfo>
2074     <prov:PMEEngineRef>http://pmsRus.org/VeryTrustedModule/4.0</prov:PMEEngineRef>
2075     <prov:PMECreatorID>http://reg.providers.com</prov:PMECreatorID>
2076     <prov:PMEWhenCreated>2007-01-18T17:32:14Z</prov:PMEWhenCreated>
2077     <prov:PMEEnabled>true</prov:PMEEnabled>
2078     <prov:PMEWhenEnabled>2007-01-18T18:15:22Z</prov:PMEWhenEnabled>
2079     <prov:PMESize>185676</prov:PMESize>
2080     <prov:PMEHash method="...SHA-1">...SHA1 hash data ...</prov:PMEHash>
2081   </prov:PMEInfo>
2082 </prov:PMEGetInfoResponse>
2083
```

2084 **Example 30. Example <prov:PMEGetInfoResponse> Message**

2085 4.16.4. PMEGetInfo Processing Rules

- 2086 • Requests which do not include at least one PMEngineRef element are interpreted as a request for all PMEngine(s)
2087 registered by the invoker.

- 2088 • Attempts to obtain information about a PMEngine that is not known to the ProvS MUST result in a failure. If
2089 detailed status codes are to be included in the response, the detailed error code for this error MUST be "NotFound."
- 2090 • If the engines pointed to by **all** of the <PMEngineRef> elements in the request are successfully enabled, the
2091 top-level status code MUST be "OK." If all of the enable operations failed, the top-level status code MUST be
2092 "Failed." Otherwise, if the results were mixed, the top-level status MUST be "Partial" and the second level status
2093 MUST be included for items for which the processing was NOT successful indicating so and including the *ref*
2094 attribute containing the <prov:PMEngineRef> value for the item. These second-level status codes MAY simply
2095 be "Failed," or they may indicate, with more detail, the reason for the failure.
- 2096 • If the top-level status code is "Failed" or "Partial," the response MAY also contain *Forbidden* or *NotFound* as a
2097 second-level status code. The Provisioning Service instance may not wish to reveal the reason for failure, in which
2098 case, no second-level status code will appear.

2099 **4.17. Operation: *PMERegister***

2100 The *PMERegister* operation is used by the PM-issuing authority to register a new PMEngine at the ProvS. The
2101 PMEngine must first be registered before it can be uploaded to the ProvS.

2102 **4.17.1. *wsa:Action* values for *PMERegister* Messages**

2103 <PMERegister> request messages MUST include a <wsa:Action> SOAP header with the value of
2104 "urn:liberty:prov:2007-09:PMERegister."

2105 <PMERegisterResponse> messages MUST include a <wsa:Action> SOAP header with the value of
2106 "urn:liberty:prov:2007-09:PMERegisterResponse."

2107 **4.17.2. *PMERegister* Message**

2108 The <PMERegister> request is called to register the specified PMEngine at the ProvS.

2109 The <prov:PMERegister> request contains the following attributes and/or elements:

- 2110 • <PMEngineRef> [**Required**] - the reference id assigned to the PMEngine to be registered.
- 2111 • <PMESize> [**Required**] - the size, in bytes, of the PMEngine's executable. This is the number of bytes that will
2112 be uploaded via the <prov:PMUpload> interface for this PMEngine. If a PMEngine is not to be uploaded, but
2113 is, in fact, built-in to the target platform(s), the size should be defined as zero.
- 2114 • <PMEHash> [**Required**] - the digest hash of the PMEngine's executable. This is used to verify that the executable
2115 has been uploaded properly and/or by the PMM to ensure that its local image is correct. The hash methods
2116 supported are the same methods defined for XML Digital Signature, most notably SHA-1 and MD5.
- 2117 • *anyAttribute* [**Optional**] - zero or more attributes from a namespace other than that of this specification. One
2118 such possibility is an **xs:ID** type attribute such as `xml:id` or `wsu:Id`.

2119 The schema for the `<prov:PMERegister>` is shown below.

```

2120 <!-- PMERegister - retrieve the specified PMSession -->
2121
2122 <xs:element name="PMERegister" type="PMERegisterType"/>
2123
2124 <xs:complexType name="PMERegisterType">
2125   <xs:complexContent>
2126     <xs:extension base="RequestAbstractType">
2127       <xs:sequence>
2128         <xs:element ref="PMSessionRef"/>
2129         <xs:element ref="PMESize" />
2130         <xs:element ref="PMEHash" />
2131       </xs:sequence>
2132     </xs:extension>
2133   </xs:complexContent>
2134 </xs:complexType>
2135

```

2136 **Figure 29. `<prov:PMERegister>` Schema Fragment**

2137 An example message body containing a `<PMERegister>` message follows.

```

2138 <prov:PMERegister>
2139   <prov:PMSessionRef>http://pmsRus.org/VeryTrustedModule/3.6</prov:PMSessionRef>
2140   <prov:PMESize>185676</prov:PMESize>
2141   <prov:PMEHash method="http://www.w3.org/2000/09/xmldsig#sha-1">
2142     ...SHA1 hash data ...
2143   </prov:PMEHash>
2144 </prov:PMERegister>
2145

```

2146 **Example 31. Example `<prov:PMERegister>` Message**

2147 4.17.3. `PMERegisterResponse` Message

2148 This response to the `<prov:PMERegister>` request contains the following elements:

- 2149 • `<lu:Status>` **[Required]** - the status of the response. See the processing rules below for more information.
- 2150 • `<prov:PMEUploadMax>` **[Optional]** - the upper limit on the number of bytes that the caller should send in a single
 2151 `<prov:PMEUpload>` invocation. This element **MUST** be specified if the request was successful and it included
 2152 a non-zero `<prov:PMESize>` value.
- 2153 • `anyAttribute` **[Optional]** - zero or more attributes from a namespace other than that of this specification. One
 2154 such possibility is an **xs:ID** type attribute such as `xml:id` or `wsu:Id`.

```

2155 <!-- PMERegisterResponse - response for the PMERegister Request -->
2156
2157 <xs:element name="PMERegisterResponse" type="PMERegisterResponseType"/>
2158 <xs:complexType name="PMERegisterResponseType">
2159   <xs:complexContent>
2160     <xs:extension base="ResponseAbstractType">
2161       <xs:sequence>
2162         <xs:element ref="PMEUploadMax" minOccurs="0"/>
2163       </xs:sequence>
2164     </xs:extension>
2165   </xs:complexContent>
2166 </xs:complexType>
2167
2168 <xs:element name="PMEUploadMax" type="xs:integer"/>
2169

```

2170 **Figure 30. <prov:PMERegisterResponse> Schema Fragment**

2171 An example message body containing a <PMERegisterResponse> message follows. This is a successful response.

```

2172 <prov:PMERegisterResponse>
2173   <lu:Status code="OK"/>
2174   <prov:PMEUploadMax>102400</prov:PMEUploadMax>
2175 </prov:PMERegisterResponse>
2176

```

2177 **Example 32. Example <prov:PMERegisterResponse> Message**

2178 4.17.4. PMERegister Processing Rules

- 2179 • The ProvS SHOULD take steps to ensure that the correct party is submitting the request and to reject requests
 2180 from inappropriate parties. If detailed status codes are to be included in the response, the detailed error code for
 2181 this error MUST be *"Forbidden."*
- 2182 • Requests which include a <PMEngineRef> that is already known to the ProvS, MUST result in a failure. If detailed
 2183 status codes are to be included in the response, the detailed error code for this error MUST be *"Duplicate."*
- 2184 • If request processing succeeded, the top-level status code MUST be *"OK."* Otherwise, the top-level status code
 2185 MUST be *"Failed"*
- 2186 • If the top-level status code is *"Failed,"* the response MAY also contain *Forbidden* or *Duplicate* as a second-level
 2187 status code. The Provisioning Service instance may not wish to reveal the reason for failure, in which case, no
 2188 second-level status code will appear.

2189 **4.18. Operation: *PMEUpload***

2190 The *PMEUpload* operation is used by the PM-issuing authority to upload the executable code for a PMEngine to the
2191 ProvS.

2192 **4.18.1. wsa:Action values for PMEUpload Messages**

2193 <PMEUpload> request messages MUST include a <wsa:Action> SOAP header with the value of
2194 "urn:liberty:prov:2007-09:PMEUpload."

2195 <PMEUploadResponse> messages MUST include a <wsa:Action> SOAP header with the value of
2196 "urn:liberty:prov:2007-09:PMEUploadResponse."

2197 **4.18.2. PMEUpload Message**

2198 The <PMEUpload> request is called to store the executable code module for a PMEngine at the ProvS so that it can
2199 be retrieved by the PMM using the <PMEDownload> interface.

2200 The <prov:PMEUpload> request contains the following attributes and/or elements:

- 2201 • <PMEngineRef> **[Required]** - the identity of the PMEngine assigned by the issuer.
- 2202 • <prov:EngineData> **[Required]** - the Base64 encoded bytes of the PMEngine executable. This MAY contain
2203 only a portion of the complete PMEngine if the file is larger than the number of bytes allowed for an upload.
2204 All counts of bytes and offsets are related to the unencoded bytes of the executable. The Base64 encoding will
2205 typically add approximately 33% additional bytes (e.g., a count of 300 bytes would result in an encoded string of
2206 approximately 400 bytes).
2207 The number of executable bytes (prior to encoding) SHOULD NOT exceed the limit specified in the
2208 <prov:PMEUploadMax> element in the <prov:PMERegisterResponse> message where this PMEngine
2209 was registered.
- 2210 • offset **[Required]** - the zero based offset in the uploaded file for the first byte of data in the enclosed
2211 <prov:EngineData> element.
- 2212 • remaining **[Required]** - the number of bytes remaining to be uploaded for the file after the enclosed data bytes
2213 are added to the file.
- 2214 • anyAttribute **[Optional]** - zero or more attributes from a namespace other than that of this specification. One
2215 such possibility is an **xs:ID** type attribute such as `xml:id` or `wsu:Id`.

2216 The schema for the `<prov:PMEUpload>` is shown below.

```

2217 <!-- PMEUpload - retrieve the specified PMSengine -->
2218
2219 <xs:element name="PMEUpload" type="PMEUploadType"/>
2220
2221 <xs:complexType name="PMEUploadType">
2222   <xs:complexContent>
2223     <xs:extension base="RequestAbstractType">
2224       <xs:sequence>
2225         <xs:element ref="PMSengineRef"/>
2226         <xs:element ref="EngineData" />
2227       </xs:sequence>
2228       <xs:attribute name="offset" use="required" type="xs:integer"/>
2229       <xs:attribute name="remaining" use="required" type="xs:integer"/>
2230     </xs:extension>
2231   </xs:complexContent>
2232 </xs:complexType>
2233
  
```

2234 **Figure 31. `<prov:PMEUpload>` Schema Fragment**

2235 An example message body containing a `<PMEUpload>` message follows. This request contains the first (offset 0)
 2236 64K bytes of a 185,676 byte file (65,536 in this message plus 120,140 remaining).

```

2237 <prov:PMEUpload offset="0" remaining="120140">
2238   <prov:PMSengineRef>http://pmsRus.org/VeryTrustedModule/3.7</prov:PMSengineRef>
2239   <prov:EngineData>
2240     ... 64K of base 64 encoded data
2241   </prov:EngineData>
2242 </prov:PMEUpload>
2243
  
```

2244 **Example 33. Example `<prov:PMEUpload>` Message**

2245 A second example message body containing a `<PMEUpload>` message follows. This request includes the next 64K
 2246 of the same 185,676 byte executable.

```

2247 <prov:PMEUpload offset="65536" remaining="54604">
2248   <prov:PMSengineRef>http://pmsRus.org/VeryTrustedModule/3.7</prov:PMSengineRef>
2249   <prov:EngineData>
2250     ... base64 encoded data (64K worth) ...
2251   </prov:EngineData>
2252 </prov:PMEUpload>
2253
  
```

2254 **Example 34. Example `<prov:PMEUpload>` Message**

2255 A third example message body containing a `<PMEUpload>` message follows. This request includes the remaining
 2256 bytes of the same 185,676 byte executable.

```

2257 <prov:PMEUpload offset="131072" remaining="0">
2258   <prov:PMSengineRef>http://pmsRus.org/VeryTrustedModule/3.7</prov:PMSengineRef>
2259   <prov:EngineData>
2260     ... base64 encoded data (54604 bytes worth) ...
2261   </prov:EngineData>
2262 </prov:PMEUpload>
2263
  
```

2264 **Example 35. Example `<prov:PMEUpload>` Message**

2265 **4.18.3. PMEUploadResponse Message**

2266 This response to the <prov:PMEUpload> request contains the following elements:

- 2267 • <lu:Status> **[Required]** - the status of the response. See the processing rules below for more information.
- 2268 • anyAttribute **[Optional]** - zero or more attributes from a namespace other than that of this specification. One
- 2269 such possibility is an **xs:ID** type attribute such as `xml:id` or `wsu:Id`.

```
2270 <!-- PMEUploadResponse - response for the PMEUpload Request -->
2271
2272 <xs:element name="PMEUploadResponse" type="PMEUploadResponseType"/>
2273
2274 <xs:complexType name="PMEUploadResponseType">
2275   <xs:complexContent>
2276     <xs:extension base="ResponseAbstractType"/>
2277   </xs:complexContent>
2278 </xs:complexType>
2279
```

2280 **Figure 32. <prov:PMEUploadResponse> Schema Fragment**

2281 An example message body containing a <PMEUploadResponse> message follows. This is a successful response.

```
2282 <prov:PMEUploadResponse>
2283   <lu:Status code="OK"/>
2284 </prov:PMEUploadResponse>
2285
```

2286 **Example 36. Example <prov:PMEUploadResponse> Message**

2287 **4.18.4. PMEUpload Processing Rules**

- 2288 • Attempts to upload data for a PMEngine that is not known to the ProvS **MUST** result in a failure. If detailed status
- 2289 codes are to be included in the response, the detailed error code for this error **MUST** be *"NotFound."*
- 2290 • The invoking party **SHOULD NOT** exceed the limit placed on the maximum number of upload bytes of data set
- 2291 by the ProvS in the <prov:PMEUploadMax> element in the <prov:PMERegisterResponse> message when
- 2292 this PMEngine was registered. If the invoker does exceed the limit, the ProvS **MAY** accept the invocation as-is,
- 2293 **MAY** generate a SOAP fault, or **MAY** otherwise treat this as a failure. If this is treated as a failure and not a
- 2294 SOAP fault and the ProvS is including detailed error status codes, the detailed status code for this error **MUST** be
- 2295 *"LimitExceeded."*
- 2296 • The invoking party **MAY** restart the uploading of a file if the PMEngine has not been enabled for download
- 2297 (perhaps because their upload process was reset) by sending an upload request with an offset of zero. When this
- 2298 occurs, the ProvS must throw away previously uploaded data and start anew with the data from the current packet.
- 2299 • If the calculated (using the remaining count, offset and data length) or actual size of the uploaded data differs from
- 2300 the size that was registered for the PMEngine, the ProvS **SHOULD** treat the upload as a failure. If it is treated
- 2301 as a failure and detailed status codes are included in the results, the detailed status code for this error **MUST** be
- 2302 *"WrongSize."*

- 2303 • Other than an upload restart as outlined above, the invoking party MUST send each segment of the uploaded file
2304 in sequential order, without skipping a section or otherwise transmitting the segments out-of-order. If a segment
2305 is transmitted out-of-order (where the first byte of the new segment is not adjacent to the last byte of the previous
2306 segment), the ProvS MUST treat this operation as a failure. If the ProvS is including detailed error status codes,
2307 the detailed status code for this error MUST be "OutOfOrder."
- 2308 • Once all segments of an executable have been uploaded and the executable has been enabled, the ProvS SHOULD
2309 NOT allow the executable to be changed. If an attempt is made to upload new data for an already uploaded
2310 executable, the ProvS SHOULD treat such requests as a failure. If the ProvS is including detailed error status
2311 codes, the detailed status code for this error MUST be "Duplicate."
- 2312 Note that if the invoking party, for some reason, decides that they have uploaded the wrong file, they
2313 would have to disable the existing PMEngine and upload the correct file with a new, non-conflicting,
2314 <prov:PMEngineRef>. This is to prevent the possibility of ever having two different executable files with the
2315 exact same <prov:PMEngineRef>.
- 2316 • The Provisioning Service SHOULD take steps to ensure that the correct party is submitting the request and to
2317 reject requests from inappropriate parties.
- 2318 • If request processing succeeded, the top-level status code MUST be "OK." Otherwise, the top-level status code
2319 MUST be "Failed."
- 2320 • If the top-level status code is "Failed," the response MAY also contain *Forbidden*, *Duplicate*, *LimitExceeded*,
2321 *WrongSize*, or *OutOfOrder* as a second-level status code. The ProvS may not wish to reveal the reason for failure,
2322 in which case, no second-level status code will appear.

2323 4.19. Operation: *Poll*

2324 The *Poll* operation is used by the PMM to poll for any new provisioning maintenance requests when the PMM is
2325 unable to expose an externally visible endpoint for direct access by the ProvS.

2326 This operation is an adaptation of the *Poll* design pattern and inherits all of its structure and processing rules.

2327 4.19.1. wsa:Action values for Poll Messages

2328 <Poll> messages MUST include a <wsa:Action> SOAP header with the value of "urn:liberty:prov:2007-09:Poll."

2329 <PollResponse> messages MUST include a <wsa:Action> SOAP header with the value of
2330 "urn:liberty:prov:2007-09:PollResponse."

2331 4.19.2. Poll Message

2332 The <Poll> request is called by the PMM to ask the ProvS for any queued provisioning maintenance requests and
2333 to return any responses to prior requests.

2334 The structure of the <prov:Poll> request is derived from the structure of the <dp:PollType> without
2335 modification. See [LibertyDP] for a complete description of the structure and meaning of the elements.

2336 The schema for the <prov:Poll> is shown below.

```
2337 <!-- Poll - Poll for new service requests -->  
2338  
2339 <xs:element name="Poll" type="dp:PollType"/>  
2340
```

2341 **Figure 33. <prov:Poll> — Schema Fragment**

2342 An example message body containing a `<prov:Poll>` message follows. This is a request asking for
 2343 `<pmm:PMUpdate>`, `<pmm:PMDelete>`, or `<pmm:PMGetStatus>` requests and asks ProvS to wait for 5 min-
 2344 utes if none are immediately available.

```
2345 <prov:Poll wait="300">
2346   <wsa:Action>urn:liberty:pmm:2007-09:PMUpdate</wsa:Action>
2347   <wsa:Action>urn:liberty:pmm:2007-09:PMDelete</wsa:Action>
2348   <wsa:Action>urn:liberty:pmm:2007-09:PMGetStatus</wsa:Action>
2349 </prov:Poll>
2350
```

2351 **Example 37. Example `<prov:Poll>` Message**

2352 Another example message body containing a `<prov:Poll>` message follows. This example also includes a
 2353 `<prov:UpdatePMResponse>` from a prior request received at the PMM.

```
2354 <prov:Poll wait="300">
2355   <wsa:Action>urn:liberty:pmm:2007-09:PMUpdate</wsa:Action>
2356   <wsa:Action>urn:liberty:pmm:2007-09:PMDelete</wsa:Action>
2357   <wsa:Action>urn:liberty:pmm:2007-09:PMGetStatus</wsa:Action>
2358   <dp:Response ref="1">
2359     <prov:UpdatePMResponse>
2360       <lu:Status code="WillNotify" />
2361     </prov:UpdatePMResponse>
2362   </dp:Response>
2363 </prov:Poll>
2364
```

2365 **Example 38. Example `<prov:Poll>` Message with a `<prov:UpdatePMResponse>`.**

2366 4.19.3. PollResponse Message

2367 This response to the `<prov:Poll>` request is derived from the `<dp:PollResponseType>` without modification.
 2368 See [\[LibertyDP\]](#) for a complete description of the structure and meaning of the elements.

```
2369 <!--PollResponse - response for the Poll request -->
2370
2371 <xs:element name="PollResponse" type="dp:PollResponseType"/>
2372
```

2373 **Figure 34. `<prov:PollResponse>` — Schema Fragment**

2374 An example message body containing a `<ps:PollResponse>` message follows. This is a successful response
 2375 without an embedded request (and therefore there were no queued requests) and ProvS is advising the PMM to poll
 2376 again in 10 minutes (600 seconds).

```
2377 <prov:PollResponse nextPoll="600">
2378   <lu:Status code="OK" />
2379 </prov:PollResponse>
2380
```

2381 **Example 39. Example `<prov:PollResponse>` Message**

2382 Another example message body containing a `<prov:PollResponse>` message follows. This is a successful
 2383 response with an embedded `pmm:UpdatePM` request.

```

2384 <prov:PollResponse>
2385   <lu:Status code="OK" />
2386   <dp:Request itemID="1">
2387     <pmm:PMUpdate>
2388       <pmm:PMUpdateItem itemID="1" type="urn:liberty:prov:2007-09:ut:engine">
2389         <prov:PMDescriptor xs:id="2323923900239" >
2390           <prov:PMID issuer="http://provs-r-us.com">uuid:778349-283920-88379-5448739</prov:PMID>
2391           <prov:PMEngineRef>https://pmsRus.org/VeryTrustedModule/4.0</prov:PMEngineRef>
2392           <ds:Signature>
2393             ... signature data goes here ...
2394           </ds:Signature>
2395         </prov:PMDescriptor>
2396       </pmm:PMUpdateItem>
2397     <dp:NotifyTo>
2398       <wsa:Address>http://www.w3.org/2005/08/addressing/anonymous</wsa:Address>
2399     </dp:NotifyTo>
2400   </pmm:PMUpdate>
2401 </dp:Request>
2402 </prov:PollResponse>
2403
2404

```

2405 **Example 40. Example <prov:PollResponse> Message**

2406 4.19.4. Poll Processing Rules

- 2407 • All of the processing rules defined for the *Poll* design pattern MUST be followed. See [LibertyDP] for further
2408 information.

2409 4.20. Operation: *UpdateEPR*

2410 The *UpdateEPR* operation is used by the PMM to update the <prov:CallbackEPR> used by the ProvS to contact
2411 the PMM.

2412 4.20.1. wsa:Action values for UpdateEPR Messages

2413 <UpdateEPR> messages MUST include a <wsa:Action> SOAP header with the value of "urn:liberty:prov:2007-09:UpdateEPR"

2414 <UpdateEPRResponse> messages MUST include a <wsa:Action> SOAP header with the value of
2415 "urn:liberty:prov:2007-09:UpdateEPRResponse."

2416 4.20.2. UpdateEPR Message

2417 The <UpdateEPR> request is called to update the <prov:CallbackEPR> that was registered when the PMM used
2418 the <prov:PMGetDescriptor> interface to obtain a PMDescriptor.

2419 The <prov:UpdateEPR> request contains one or more <prov:UpdateEPRItem> elements which have the
2420 following contents:

- 2421 • <PMID> **[Required]** - the identifier of the PM to which this update is related. This identifier is included in the
2422 <prov:PMDescriptor> returned in the <prov:PMGetDescriptorResponse>.
- 2423 • <prov:CallbackEPR> **[Required]** - one or more updated ID-WSF EPR(s). This is a complete replacement of
2424 any formerly registered EPRs related to this PM.
- 2425 • itemID **[Required]** - the identifier for this request item (for correlation within the results).

2426 The schema for the `<prov:UpdateEPR>` is shown below.

```

2427 <!-- UpdateEPR - update the EPR for PM maintenance operations -->
2428
2429 <xs:element name="UpdateEPR" type="UpdateEPRTYPE" />
2430
2431 <xs:complexType name="UpdateEPRTYPE">
2432   <xs:complexContent>
2433     <xs:extension base="RequestAbstractType">
2434       <xs:sequence>
2435         <xs:element ref="UpdateEPRItem" maxOccurs="unbounded" />
2436       </xs:sequence>
2437     </xs:extension>
2438   </xs:complexContent>
2439 </xs:complexType>
2440
2441 <xs:element name="UpdateEPRItem" type="UpdateEPRItemTYPE" />
2442
2443 <xs:complexType name="UpdateEPRItemTYPE">
2444   <xs:sequence>
2445     <xs:element ref="PMID" />
2446     <xs:element ref="CallbackEPR" />
2447   </xs:sequence>
2448   <xs:attribute name="itemID" type="xs:string" use="required" />
2449 </xs:complexType>
2450

```

2451 **Figure 35. `<prov:UpdateEPR>` — Schema Fragment**

2452 An example message body containing a `<prov:UpdateEPR>` message follows. This is an update of the `CallbackEPR`
 2453 for two different PMs.

```

2454 <prov:UpdateEPR>
2455   <prov:UpdateEPRItem itemID="1" >
2456     <prov:PMID issuer="http://provs-r-us.com">uuid:778349-283920-88379-5448739</prov:PMID>
2457     <prov:CallbackEPR>
2458       <wsa:Address>http://im-a-pmm.com</wsa:Address>
2459       <wsa:Metadata>
2460         <ds:ServiceType>urn:liberty:pmm:2007-09</ds:ServiceType>
2461         <ds:Framework version="2.0" />
2462         <ds:SecurityContext>
2463           <ds:SecurityMechID>urn:liberty:security:2006-08:TLS:SAMLV2</ds:SecurityMechID>
2464         </ds:SecurityContext>
2465       </wsa:Metadata>
2466     </prov:CallbackEPR>
2467   </prov:UpdateEPRItem>
2468   <prov:UpdateEPRItem itemID="2">
2469     <prov:PMID issuer="http://provs-r-us.com">uuid:239032-230328-92379-2397923</prov:PMID>
2470     <prov:CallbackEPR>
2471       <wsa:Address>http://im-a-pmm.com</wsa:Address>
2472       <wsa:Metadata>
2473         <ds:ServiceType>urn:liberty:pmm:2007-09</ds:ServiceType>
2474         <ds:Framework version="2.0" />
2475         <ds:SecurityContext>
2476           <ds:SecurityMechID>urn:liberty:security:2006-08:TLS:SAMLV2</ds:SecurityMechID>
2477         </ds:SecurityContext>
2478       </wsa:Metadata>
2479     </prov:CallbackEPR>
2480   </prov:UpdateEPRItem>
2481 </prov:UpdateEPR>
2482

```

2483 **Example 41. Example `<prov:UpdateEPR>` Message**

2484 4.20.3. UpdateEPRResponse Message

2485 This response to the <prov:UpdateEPR> request contains the following elements/attributes:

- 2486 • <lu:Status>: **[Required]** - The status of the response. See the processing rules below for more information.
- 2487 • anyAttribute **[Optional]** - Zero or more attributes from a namespace other than that of this specification. One
- 2488 such possibility is an **xs:ID** type attribute such as `xml:id` or `wsu:Id`.

```
2489 <!-- UpdateEPRResponse - the response to the UpdateEPR request -->
2490
2491 <xs:element name="UpdateEPRResponse" type="UpdateEPRResponseType"/>
2492
2493 <xs:complexType name="UpdateEPRResponseType">
2494   <xs:complexContent>
2495     <xs:extension base="ResponseAbstractType" />
2496   </xs:complexContent>
2497 </xs:complexType>
2498
```

2499 **Figure 36.** <prov:UpdateEPRResponse> — Schema Fragment

2500 An example message body containing an <UpdateEPRResponse> message follows. This is a partial success message
2501 for an attempted update of the CallbackEPR for 2 PMs where the 1st succeeded and the 2nd failed.

```
2502 <prov:UpdateEPRResponse>
2503   <lu:Status code="Partial">
2504     <lu:Status code="OK" ref="1" />
2505     <lu:Status code="NotFound" ref="2" />
2506   </lu:Status>
2507 </prov:UpdateEPRResponse>
2508
```

2509 **Example 42.** Example <prov:UpdateEPRResponse> Message

2510 4.20.4. UpdateEPR Processing Rules

- 2511 • If, for any reason, an update is not accepted, the existing service instance at the ProvS **MUST NOT** be affected.
- 2512 • PMs instantiated in one PMM **MUST NOT** be visible to other PMMs. If a PMM presents a PMID for a PM
2513 instance that has been instantiated in a different PMM, the ProvS **MUST** behave as if that PM does not exist and,
2514 accordingly, any attempt to update the associated CallbackEPR should fail. In such cases, the request **MUST** result
2515 in a failure. If detailed status codes are included in the response, the detailed status code for this case **MUST** be
2516 *"NotFound."*
- 2517 • The ProvS **SHOULD** generate an error if the CallbackEPR is such that the ProvS is unable to meet the requirements
2518 within the ID-WSF EPR and, therefore, the ProvS would be unable to dereference the ID-WSF EPR (for example,
2519 if the ProvS does not support the framework version(s) specified in the CallbackEPR). If detailed status codes are
2520 included in the response, the detailed status code for this case **MUST** be *"FeatureNotSupported."*
- 2521 • Each <prov:UpdateEPRItem> is processed independently and may, independently, succeed or fail on its own
2522 merits.

- 2523 • If all `<prov:UpdateEPRItem>` requests are successfully processed, the top-level status code MUST be "OK."
2524 If all of the items failed, the top-level status code MUST be "Failed." Otherwise, if the results were mixed, the top-
2525 level status MUST be "Partial" and the second level status MUST be included for items for which the processing
2526 was not successful indicating so and including the `ref` attribute containing the `itemID` value for the item. These
2527 second-level status codes MAY simply be "Failed" or they may indicate, with more detail, the reason for the
2528 failure.
- 2529 • If the top-level status code is "Failed," the response MAY also contain other status codes (such as *NotFound* or
2530 *FeatureNotSupported*) as a second-level status code. The ProvS instance may not wish to reveal the reason for
2531 failure, in which case, no second-level status code will appear.

2532 5. Provisioning Service Schema

```

2533 <?xml version="1.0" encoding="UTF-8"?>
2534 <xs:schema targetNamespace="urn:liberty:prov:2007-09"
2535   xmlns:lu="urn:liberty:util:2006-08"
2536   xmlns:prov="urn:liberty:prov:2007-09"
2537   xmlns:dp="urn:liberty:dp:2007-09"
2538   xmlns:xs="http://www.w3.org/2001/XMLSchema"
2539   xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
2540   xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
2541   xmlns:wsa="http://www.w3.org/2005/08/addressing"
2542   xmlns="urn:liberty:prov:2007-09"
2543   elementFormDefault="qualified"
2544   attributeFormDefault="unqualified"
2545 >
2546
2547 <xs:import namespace="urn:liberty:util:2006-08"
2548   schemaLocation="liberty-idwsf-utility-v2.0.xsd"/>
2549
2550 <xs:import namespace="urn:liberty:dp:2007-09"
2551   schemaLocation="liberty-idwsf-dp-v1.0.xsd"/>
2552
2553 <xs:import namespace="http://www.w3.org/2000/09/xmldsig#"
2554   schemaLocation="http://www.w3.org/TR/2002/REC-xmldsig-core-20020212/xmldsig-core-schema.
2555 xsd"/>
2556
2557 <xs:import namespace="http://www.w3.org/2001/04/xmlenc#"
2558   schemaLocation="http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/xenc-schema.xsd"/>
2559 <xs:import namespace="http://www.w3.org/2005/08/addressing"
2560   schemaLocation="http://www.w3.org/2005/08/addressing/ws-addr.xsd" />
2561
2562 <!-- PMID - the Provisioned Module Identifier -->
2563
2564 <xs:element name="PMID" type="PMIDType" />
2565 <xs:complexType name="PMIDType">
2566   <xs:simpleContent>
2567     <xs:extension base="xs:anyURI">
2568       <xs:attribute name="issuer" type="xs:anyURI" use="required"/>
2569     </xs:extension>
2570   </xs:simpleContent>
2571 </xs:complexType>
2572
2573
2574 <!-- PMDescriptor - describes/carries the components of a PM -->
2575
2576 <xs:element name="PMDescriptor" type="PMDescriptorType"/>
2577
2578 <xs:complexType name="PMDescriptorType">
2579   <xs:sequence>
2580     <xs:element ref="ds:Signature" minOccurs="0"/>
2581     <xs:element ref="PMID" />
2582     <xs:element ref="PMEngineRef" minOccurs="0"/>
2583     <xs:element ref="PMEngine" minOccurs="0"/>
2584     <xs:element ref="PMInitData" minOccurs="0"/>
2585     <xs:element ref="PMRTData" minOccurs="0"/>
2586     <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
2587   </xs:sequence>
2588   <xs:attribute name="activate" type="xs:boolean" use="optional"/>
2589   <xs:attribute name="activateAt" type="xs:dateTime" use="optional"/>
2590   <xs:attribute name="deactivateAt" type="xs:dateTime" use="optional"/>
2591   <xs:anyAttribute namespace="##any" processContents="lax"/>
2592 </xs:complexType>
2593
2594 <xs:element name="PMInitData" type="PMDataType" />
2595 <xs:element name="PMRTData" type="PMDataType" />
2596
2597 <xs:complexType name="PMDataType" mixed="false">

```

```

2598     <xs:sequence>
2599         <xs:any namespace="##any" processContents="lax" maxOccurs="unbounded"/>
2600     </xs:sequence>
2601     <xs:anyAttribute namespace="##other" processContents="lax"/>
2602 </xs:complexType>
2603
2604 <xs:element name="PMEngine"    type="PMEngineType" />
2605
2606 <xs:complexType name="PMEngineType" mixed="false">
2607     <xs:sequence>
2608         <xs:element ref="PMEInfo" />
2609         <xs:element ref="PMEBytes" />
2610     </xs:sequence>
2611     <xs:anyAttribute namespace="##other" processContents="lax"/>
2612 </xs:complexType>
2613
2614 <xs:element name="PMEBytes"    type="xs:base64Binary" />
2615
2616
2617 <!-- ProvisioningHandle - Info for PMM to initiate provisioning process -->
2618
2619 <xs:element name="ProvisioningHandle" type="ProvisioningHandleType"/>
2620 <xs:complexType name="ProvisioningHandleType">
2621     <xs:sequence>
2622         <xs:element ref="PMDArtifact" />
2623         <xs:element ref="ProvisioningServiceEPR" minOccurs="0"
2624             maxOccurs="unbounded" />
2625         <xs:element ref="ds:Signature"    minOccurs="0"/>
2626     </xs:sequence>
2627     <xs:attribute name="expires" use="optional" type="xs:dateTime"/>
2628     <xs:anyAttribute namespace="##other" processContents="lax"/>
2629 </xs:complexType>
2630
2631 <xs:element name="ProvisioningServiceEPR" type="wsa:EndpointReferenceType"/>
2632
2633 <xs:element name="PMDArtifact" type="xs:string" />
2634
2635
2636 <!-- CallbackEPR - where the PMM can receive Provisionig update requests -->
2637
2638 <xs:element name="CallbackEPR" type="wsa:EndpointReferenceType"/>
2639
2640
2641 <!-- PMStatus - Provisioning status of the PM -->
2642
2643 <xs:element name="PMStatus" type="PMStatusType"/>
2644 <xs:complexType name="PMStatusType">
2645     <xs:sequence>
2646         <xs:element ref="PMID"/>
2647         <xs:element ref="State"/>
2648     </xs:sequence>
2649 </xs:complexType>
2650
2651 <xs:element name="State" type="StateType"/>
2652 <xs:complexType name="StateType">
2653     <xs:simpleContent>
2654         <xs:extension base="xs:anyURI">
2655             <xs:attribute name="asof" type="xs:dateTime" use="optional"/>
2656         </xs:extension>
2657     </xs:simpleContent>
2658 </xs:complexType>
2659
2660
2661 <!-- PMEInfo - The current state of a PMEngine -->
2662
2663 <xs:element name="PMEInfo" type="PMEInfoType"/>
2664 <xs:complexType name="PMEInfoType">
    
```

```

2665     <xs:sequence>
2666     <xs:element ref="PMEngineRef" />
2667     <xs:element ref="PMECreatorID" />
2668     <xs:element ref="PMEWhenCreated" />
2669     <xs:element ref="PMEEnabled" />
2670     <xs:element ref="PMEWhenEnabled" />
2671     <xs:element ref="PMESize" />
2672     <xs:element ref="PMEHash" />
2673     <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded" />
2674 </xs:sequence>
2675 <xs:anyAttribute namespace="##any" processContents="lax" />
2676 </xs:complexType>
2677
2678 <xs:element name="PMEngineRef" type="xs:anyURI" />
2679
2680 <xs:element name="PMECreatorID" type="xs:anyURI" />
2681
2682 <xs:element name="PMEWhenCreated" type="xs:dateTime" />
2683
2684 <xs:element name="PMEEnabled" type="xs:boolean" />
2685
2686 <xs:element name="PMEWhenEnabled" type="xs:dateTime" />
2687
2688 <xs:element name="PMESize" type="xs:integer" />
2689
2690 <xs:element name="PMEHash" type="PMEHashType" />
2691 <xs:complexType name="PMEHashType">
2692     <xs:simpleContent>
2693         <xs:extension base="xs:base64Binary">
2694             <xs:attribute name="method" type="xs:anyURI" use="required" />
2695         </xs:extension>
2696     </xs:simpleContent>
2697 </xs:complexType>
2698
2699
2700 <!-- RequestAbstractType - common request message structure -->
2701
2702 <xs:complexType name="RequestAbstractType" abstract="true">
2703     <xs:anyAttribute namespace="##other" processContents="lax" />
2704 </xs:complexType>
2705
2706
2707 <!-- ResponseAbstractType - common message response structure -->
2708
2709 <xs:complexType name="ResponseAbstractType" abstract="true">
2710     <xs:sequence>
2711         <xs:element ref="lu:Status" />
2712     </xs:sequence>
2713     <xs:anyAttribute namespace="##other" processContents="lax" />
2714 </xs:complexType>
2715
2716
2717 <!--*****-->
2718 <!-- -->
2719 <!--     PMEEngine Maintenance interfaces     -->
2720 <!-- -->
2721 <!-- These interfaces are used to manage PMEEngine objects at -->
2722 <!-- the ProvS and include register/delete,enable/disable, -->
2723 <!-- upload/download, and getInfo operations -->
2724 <!-- -->
2725 <!--*****-->
2726 <!-- PMERegister - retrieve the specified PMEEngine -->
2727
2728 <xs:element name="PMERegister" type="PMERegisterType" />
2729
2730 <xs:complexType name="PMERegisterType">
2731     <xs:complexContent>
    
```

```

2732     <xs:extension base="RequestAbstractType">
2733     <xs:sequence>
2734         <xs:element ref="PMEEngineRef"/>
2735         <xs:element ref="PMESize" />
2736         <xs:element ref="PMEHash" />
2737     </xs:sequence>
2738 </xs:extension>
2739 </xs:complexContent>
2740 </xs:complexType>
2741
2742
2743 <!-- PMERegisterResponse - response for the PMERegister Request -->
2744
2745 <xs:element name="PMERegisterResponse" type="PMERegisterResponseType"/>
2746 <xs:complexType name="PMERegisterResponseType">
2747     <xs:complexContent>
2748         <xs:extension base="ResponseAbstractType">
2749             <xs:sequence>
2750                 <xs:element ref="PMEUploadMax" minOccurs="0"/>
2751             </xs:sequence>
2752         </xs:extension>
2753     </xs:complexContent>
2754 </xs:complexType>
2755
2756 <xs:element name="PMEUploadMax" type="xs:integer"/>
2757
2758
2759 <!-- PMEUpload - retrieve the specified PMEngine -->
2760
2761 <xs:element name="PMEUpload" type="PMEUploadType"/>
2762
2763 <xs:complexType name="PMEUploadType">
2764     <xs:complexContent>
2765         <xs:extension base="RequestAbstractType">
2766             <xs:sequence>
2767                 <xs:element ref="PMEEngineRef"/>
2768                 <xs:element ref="EngineData" />
2769             </xs:sequence>
2770             <xs:attribute name="offset" use="required" type="xs:integer"/>
2771             <xs:attribute name="remaining" use="required" type="xs:integer"/>
2772         </xs:extension>
2773     </xs:complexContent>
2774 </xs:complexType>
2775
2776
2777 <!-- PMEUploadResponse - response for the PMEUpload Request -->
2778
2779 <xs:element name="PMEUploadResponse" type="PMEUploadResponseType"/>
2780
2781 <xs:complexType name="PMEUploadResponseType">
2782     <xs:complexContent>
2783         <xs:extension base="ResponseAbstractType"/>
2784     </xs:complexContent>
2785 </xs:complexType>
2786
2787
2788 <!-- PMEDownload - download the specified PMEngine (or part thereof) -->
2789
2790 <xs:element name="PMEDownload" type="PMEDownloadType"/>
2791
2792 <xs:complexType name="PMEDownloadType">
2793     <xs:complexContent>
2794         <xs:extension base="RequestAbstractType">
2795             <xs:sequence>
2796                 <xs:element ref="PMEEngineRef"/>
2797             </xs:sequence>
2798             <xs:attributeGroup ref="dp:BasicPagingAttributeGroup"/>
    
```

```
2799     </xs:extension>
2800   </xs:complexContent>
2801 </xs:complexType>
2802
2803
2804 <!-- PMEDownloadResponse - response for the PMEDownload Request -->
2805
2806 <xs:element name="PMEDownloadResponse" type="PMEDownloadResponseType"/>
2807
2808 <xs:complexType name="PMEDownloadResponseType">
2809   <xs:complexContent>
2810     <xs:extension base="ResponseAbstractType">
2811       <xs:sequence>
2812         <xs:element ref="EngineData" minOccurs="0" />
2813       </xs:sequence>
2814       <xs:attributeGroup ref="dp:BasicPagingResponseAttributeGroup"/>
2815     </xs:extension>
2816   </xs:complexContent>
2817 </xs:complexType>
2818
2819 <xs:element name="EngineData" type="xs:base64Binary"/>
2820
2821
2822 <!-- PMEEnable - retrieve the specified PMServer -->
2823
2824 <xs:element name="PMEEnable" type="PMEEnableType"/>
2825
2826 <xs:complexType name="PMEEnableType">
2827   <xs:complexContent>
2828     <xs:extension base="RequestAbstractType">
2829       <xs:sequence>
2830         <xs:element ref="PMServerRef" maxOccurs="unbounded"/>
2831       </xs:sequence>
2832     </xs:extension>
2833   </xs:complexContent>
2834 </xs:complexType>
2835
2836
2837 <!-- PMEEnableResponse - response for the PMEEnable Request -->
2838
2839 <xs:element name="PMEEnableResponse" type="PMEEnableResponseType"/>
2840
2841 <xs:complexType name="PMEEnableResponseType">
2842   <xs:complexContent>
2843     <xs:extension base="ResponseAbstractType"/>
2844   </xs:complexContent>
2845 </xs:complexType>
2846
2847
2848 <!-- PMEDisable - retrieve the specified PMServer -->
2849
2850 <xs:element name="PMEDisable" type="PMEDisableType"/>
2851
2852 <xs:complexType name="PMEDisableType">
2853   <xs:complexContent>
2854     <xs:extension base="RequestAbstractType">
2855       <xs:sequence>
2856         <xs:element ref="PMServerRef" maxOccurs="unbounded"/>
2857       </xs:sequence>
2858     </xs:extension>
2859   </xs:complexContent>
2860 </xs:complexType>
2861
2862
2863 <!-- PMEDisableResponse - response for the PMEDisable Request -->
2864
2865 <xs:element name="PMEDisableResponse" type="PMEDisableResponseType"/>
```

```
2866
2867 <xs:complexType name="PMEDisableResponseType">
2868   <xs:complexContent>
2869     <xs:extension base="ResponseAbstractType"/>
2870   </xs:complexContent>
2871 </xs:complexType>
2872
2873
2874 <!-- PMEDelete - retrieve the specified PMSngine -->
2875
2876 <xs:element name="PMEDelete" type="PMEDeleteType"/>
2877
2878 <xs:complexType name="PMEDeleteType">
2879   <xs:complexContent>
2880     <xs:extension base="RequestAbstractType">
2881       <xs:sequence>
2882         <xs:element ref="PMSngineRef" maxOccurs="unbounded"/>
2883       </xs:sequence>
2884     </xs:extension>
2885   </xs:complexContent>
2886 </xs:complexType>
2887
2888
2889 <!-- PMEDeleteResponse - response for the PMEDelete Request -->
2890
2891 <xs:element name="PMEDeleteResponse" type="PMEDeleteResponseType"/>
2892
2893 <xs:complexType name="PMEDeleteResponseType">
2894   <xs:complexContent>
2895     <xs:extension base="ResponseAbstractType"/>
2896   </xs:complexContent>
2897 </xs:complexType>
2898
2899
2900 <!-- PMEGetInfo - to get information about the specified PME(s) -->
2901
2902 <xs:element name="PMEGetInfo" type="PMEGetInfoType"/>
2903 <xs:complexType name="PMEGetInfoType">
2904   <xs:complexContent>
2905     <xs:extension base="RequestAbstractType">
2906       <xs:sequence>
2907         <xs:element ref="PMSngineRef" minOccurs="0" maxOccurs="unbounded"/>
2908       </xs:sequence>
2909     </xs:extension>
2910   </xs:complexContent>
2911 </xs:complexType>
2912
2913
2914 <!-- PMEGetInfoResponse - response to the PMEGetInfo request -->
2915
2916 <xs:element name="PMEGetInfoResponse" type="PMEGetInfoResponseType"/>
2917
2918 <xs:complexType name="PMEGetInfoResponseType">
2919   <xs:complexContent>
2920     <xs:extension base="ResponseAbstractType">
2921       <xs:sequence>
2922         <xs:element ref="PMEInfo" minOccurs="0" maxOccurs="unbounded"/>
2923       </xs:sequence>
2924     </xs:extension>
2925   </xs:complexContent>
2926 </xs:complexType>
2927
2928
2929 <!--*****-->
2930 <!-- -->
2931 <!-- Provisioned Module Maintenance interfaces -->
2932 <!-- -->
```

```
2933 <!-- These interfaces are used to manage PMs during and after -->
2934 <!-- the initial provisioning process. Actions taken with -->
2935 <!-- these interfaces typically result in indirect actions -->
2936 <!-- within a PMM where the PM has been provisioned. -->
2937 <!-- -->
2938 <!--*****-->
2939 <!-- PMGetDescriptor - request to exchange PMHandle for PMGetDescriptor -->
2940
2941 <xs:element name="PMGetDescriptor" type="PMGetDescriptorType"/>
2942
2943 <xs:complexType name="PMGetDescriptorType">
2944 <xs:complexContent>
2945 <xs:extension base="RequestAbstractType">
2946 <xs:sequence>
2947 <xs:element ref="PMDArtifact"/>
2948 <xs:element ref="CallbackEPR" maxOccurs="unbounded"/>
2949 </xs:sequence>
2950 </xs:extension>
2951 </xs:complexContent>
2952 </xs:complexType>
2953
2954
2955 <!-- PMGetDescriptorResponse - response to the PMGetDescriptor request -->
2956
2957 <xs:element name="PMGetDescriptorResponse" type="PMGetDescriptorResponseType"/>
2958
2959 <xs:complexType name="PMGetDescriptorResponseType">
2960 <xs:complexContent>
2961 <xs:extension base="ResponseAbstractType">
2962 <xs:sequence>
2963 <xs:element ref="PMDescriptor" minOccurs="0"/>
2964 </xs:sequence>
2965 </xs:extension>
2966 </xs:complexContent>
2967 </xs:complexType>
2968
2969
2970 <!-- PMActivate - to activate one or more PM(s) at the PMM -->
2971
2972 <xs:element name="PMActivate" type="PMActivateType"/>
2973
2974 <xs:complexType name="PMActivateType">
2975 <xs:complexContent>
2976 <xs:extension base="RequestAbstractType">
2977 <xs:sequence>
2978 <xs:element ref="PMActivateItem" maxOccurs="unbounded" />
2979 <xs:element ref="dp:NotifyTo" minOccurs="0" />
2980 </xs:sequence>
2981 </xs:extension>
2982 </xs:complexContent>
2983 </xs:complexType>
2984
2985 <xs:element name="PMActivateItem" type="PMActivateItemType" />
2986
2987 <xs:complexType name="PMActivateItemType">
2988 <xs:sequence>
2989 <xs:element ref="prov:PMID"/>
2990 </xs:sequence>
2991 <xs:attribute name="itemID" type="xs:string" use="required"/>
2992 <xs:attribute name="at" type="xs:dateTime" use="optional"/>
2993 </xs:complexType>
2994
2995
2996 <!-- PMActivateResponse - the response to the PMActivate request -->
2997
2998 <xs:element name="PMActivateResponse" type="PMActivateResponseType"/>
2999
```

```

3000 <xs:complexType name="PMActivateResponseType">
3001   <xs:complexContent>
3002     <xs:extension base="ResponseAbstractType"/>
3003   </xs:complexContent>
3004 </xs:complexType>
3005
3006
3007 <!-- PMDeactivate - to deactivate a PM at the PMM -->
3008
3009 <xs:element name="PMDeactivate" type="PMDeactivateType"/>
3010
3011 <xs:complexType name="PMDeactivateType">
3012   <xs:complexContent>
3013     <xs:extension base="RequestAbstractType">
3014       <xs:sequence>
3015         <xs:element ref="PMDeactivateItem" maxOccurs="unbounded" />
3016         <xs:element ref="dp:NotifyTo" minOccurs="0" />
3017       </xs:sequence>
3018     </xs:extension>
3019   </xs:complexContent>
3020 </xs:complexType>
3021
3022 <xs:element name="PMDeactivateItem" type="PMDeactivateItemType" />
3023
3024 <xs:complexType name="PMDeactivateItemType">
3025   <xs:sequence>
3026     <xs:element ref="prov:PMID"/>
3027   </xs:sequence>
3028   <xs:attribute name="itemID" type="xs:string" use="required"/>
3029   <xs:attribute name="at" type="xs:dateTime" use="optional"/>
3030 </xs:complexType>
3031
3032
3033 <!-- PMDeactivateResponse - the response to the PMDeactivate request -->
3034
3035 <xs:element name="PMDeactivateResponse" type="PMDeactivateResponseType"/>
3036
3037 <xs:complexType name="PMDeactivateResponseType">
3038   <xs:complexContent>
3039     <xs:extension base="ResponseAbstractType"/>
3040   </xs:complexContent>
3041 </xs:complexType>
3042
3043
3044 <!-- PMRegisterDescriptor - to register a new PMD at the ProvS -->
3045
3046 <xs:element name="PMRegisterDescriptor" type="PMRegisterDescriptorType"/>
3047
3048 <xs:complexType name="PMRegisterDescriptorType">
3049   <xs:complexContent>
3050     <xs:extension base="RequestAbstractType">
3051       <xs:sequence>
3052         <xs:element ref="PMRegisterDescriptorItem" maxOccurs="unbounded" />
3053       </xs:sequence>
3054     </xs:extension>
3055   </xs:complexContent>
3056 </xs:complexType>
3057
3058 <xs:element name="PMRegisterDescriptorItem" type="PMRegisterDescriptorItemType" />
3059
3060 <xs:complexType name="PMRegisterDescriptorItemType">
3061   <xs:sequence>
3062     <xs:element ref="PMDescriptor" />
3063   </xs:sequence>
3064   <xs:attribute name="itemID" type="xs:string" use="required" />
3065 </xs:complexType>
3066

```



```

3067
3068 <!-- PMRegisterDescriptorResponse - response to the PMRegisterDescriptor request -->
3069
3070 <xs:element name="PMRegisterDescriptorResponse" type="PMRegisterDescriptorResponseType"/>
3071
3072 <xs:complexType name="PMRegisterDescriptorResponseType">
3073   <xs:complexContent>
3074     <xs:extension base="ResponseAbstractType">
3075       <xs:sequence>
3076         <xs:element ref="PMRegisterDescriptorResponseItem" minOccurs="0"
3077           maxOccurs="unbounded"/>
3078       </xs:sequence>
3079     </xs:extension>
3080   </xs:complexContent>
3081 </xs:complexType>
3082
3083 <xs:element name="PMRegisterDescriptorResponseItem"
3084   type="PMRegisterDescriptorResponseItemType"/>
3085
3086 <xs:complexType name="PMRegisterDescriptorResponseItemType">
3087   <xs:sequence>
3088     <xs:element ref="ProvisioningHandle"/>
3089   </xs:sequence>
3090   <xs:attribute name="ref" type="xs:string" use="required"/>
3091 </xs:complexType>
3092
3093
3094 <!-- PMUpdate - update the PM for a existing PM at the ProvS -->
3095
3096 <xs:element name="PMUpdate" type="PMUpdateType"/>
3097
3098 <xs:complexType name="PMUpdateType">
3099   <xs:complexContent>
3100     <xs:extension base="RequestAbstractType">
3101       <xs:sequence>
3102         <xs:element ref="PMUpdateItem" maxOccurs="unbounded" />
3103         <xs:element ref="dp:NotifyTo" minOccurs="0" />
3104       </xs:sequence>
3105     </xs:extension>
3106   </xs:complexContent>
3107 </xs:complexType>
3108
3109 <xs:element name="PMUpdateItem" type="PMUpdateItemType" />
3110
3111 <xs:complexType name="PMUpdateItemType">
3112   <xs:sequence>
3113     <xs:element ref="PMDescriptor"/>
3114   </xs:sequence>
3115   <xs:attribute name="type" type="xs:anyURI" use="required"/>
3116   <xs:attribute name="itemID" type="xs:string" use="required"/>
3117   <xs:attribute name="at" type="xs:dateTime" use="optional"/>
3118 </xs:complexType>
3119
3120
3121 <!-- PMUpdateResponse - response to the PMUpdate request -->
3122
3123 <xs:element name="PMUpdateResponse" type="PMUpdateResponseType"/>
3124
3125 <xs:complexType name="PMUpdateResponseType">
3126   <xs:complexContent>
3127     <xs:extension base="ResponseAbstractType"/>
3128   </xs:complexContent>
3129 </xs:complexType>
3130
3131
3132 <!-- PMDelete - to register a new PM at the ProvS -->
3133

```

```

3134 <xs:element name="PMDelete" type="PMDeleteType"/>
3135 <xs:complexType name="PMDeleteType">
3136   <xs:complexContent>
3137     <xs:extension base="RequestAbstractType">
3138       <xs:sequence>
3139         <xs:element ref="PMDeleteItem" maxOccurs="unbounded" />
3140         <xs:element ref="dp:NotifyTo" minOccurs="0" />
3141       </xs:sequence>
3142     </xs:extension>
3143   </xs:complexContent>
3144 </xs:complexType>
3145
3146 <xs:element name="PMDeleteItem" type="PMDeleteItemType" />
3147 <xs:complexType name="PMDeleteItemType">
3148   <xs:sequence>
3149     <xs:element ref="prov:PMID"/>
3150   </xs:sequence>
3151   <xs:attribute name="itemID" type="xs:string" use="required"/>
3152 </xs:complexType>
3153
3154 <!-- PMDeleteResponse - the response to the PMDelete request -->
3155
3156 <xs:element name="PMDeleteResponse" type="PMDeleteResponseType"/>
3157
3158 <xs:complexType name="PMDeleteResponseType">
3159   <xs:complexContent>
3160     <xs:extension base="ResponseAbstractType" />
3161   </xs:complexContent>
3162 </xs:complexType>
3163
3164
3165 <!-- PMStatus - to get the status of a PM at the ProvS -->
3166
3167 <xs:element name="PMGetStatus" type="PMGetStatusType"/>
3168
3169 <xs:complexType name="PMGetStatusType">
3170   <xs:complexContent>
3171     <xs:extension base="RequestAbstractType">
3172       <xs:sequence>
3173         <xs:element ref="PMID" minOccurs="0" maxOccurs="unbounded"/>
3174       </xs:sequence>
3175     </xs:extension>
3176   </xs:complexContent>
3177 </xs:complexType>
3178
3179
3180 <!-- PMGetStatusResponse - response to the PMGetStatus request -->
3181
3182 <xs:element name="PMGetStatusResponse" type="PMGetStatusResponseType"/>
3183
3184 <xs:complexType name="PMGetStatusResponseType">
3185   <xs:complexContent>
3186     <xs:extension base="ResponseAbstractType">
3187       <xs:sequence>
3188         <xs:element ref="PMStatus" minOccurs="0" maxOccurs="unbounded"/>
3189       </xs:sequence>
3190     </xs:extension>
3191   </xs:complexContent>
3192 </xs:complexType>
3193
3194
3195 <!-- PMSetStatus - update provisioning status of PM -->
3196
3197 <xs:element name="PMSetStatus" type="PMSetStatusType"/>
3198
3199 <xs:complexType name="PMSetStatusType">
3200

```

```

3201     <xs:complexContent>
3202     <xs:extension base="RequestAbstractType">
3203     <xs:sequence>
3204     <xs:element ref="PMStatus"/>
3205     </xs:sequence>
3206     </xs:extension>
3207     </xs:complexContent>
3208 </xs:complexType>
3209
3210
3211 <!-- PMSetStatusResonse - response for PMSetStatus request -->
3212
3213 <xs:element name="PMSetStatusResponse" type="PMSetStatusResponseType"/>
3214
3215 <xs:complexType name="PMSetStatusResponseType">
3216 <xs:complexContent>
3217 <xs:extension base="ResponseAbstractType" />
3218 </xs:complexContent>
3219 </xs:complexType>
3220
3221
3222 <!-- Poll - Poll for new service requests -->
3223
3224 <xs:element name="Poll" type="dp:PollType"/>
3225
3226
3227 <!--PollResponse - response for the Poll request -->
3228
3229 <xs:element name="PollResponse" type="dp:PollResponseType"/>
3230
3231
3232 <!-- UpdateEPR - update the EPR for PM maintenance operations -->
3233
3234 <xs:element name="UpdateEPR" type="UpdateEPRTYPE"/>
3235
3236 <xs:complexType name="UpdateEPRTYPE">
3237 <xs:complexContent>
3238 <xs:extension base="RequestAbstractType">
3239 <xs:sequence>
3240 <xs:element ref="UpdateEPRItem" maxOccurs="unbounded" />
3241 </xs:sequence>
3242 </xs:extension>
3243 </xs:complexContent>
3244 </xs:complexType>
3245
3246 <xs:element name="UpdateEPRItem" type="UpdateEPRItemTYPE" />
3247
3248 <xs:complexType name="UpdateEPRItemTYPE">
3249 <xs:sequence>
3250 <xs:element ref="PMID" />
3251 <xs:element ref="CallbackEPR" />
3252 </xs:sequence>
3253 <xs:attribute name="itemID" type="xs:string" use="required" />
3254 </xs:complexType>
3255
3256
3257 <!-- UpdateEPRResponse - the response to the UpdateEPR request -->
3258
3259 <xs:element name="UpdateEPRResponse" type="UpdateEPRResponseType"/>
3260
3261 <xs:complexType name="UpdateEPRResponseType">
3262 <xs:complexContent>
3263 <xs:extension base="ResponseAbstractType" />
3264 </xs:complexContent>
3265 </xs:complexType>
3266
3267
  
```

3268 </xs:schema>
3269

3270 6. Provisioning Service WSDL

```
3271 <?xml version="1.0"?>
3272 <definitions name="prov-svc"
3273   targetNamespace="urn:liberty:prov:2007-09"
3274   xmlns:tns="urn:liberty:prov:2007-09"
3275   xmlns="http://schemas.xmlsoap.org/wsdl/"
3276   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
3277   xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
3278   xmlns:wsaw="http://www.w3.org/2006/02/addressing/wsdl"
3279   xmlns:prov="urn:liberty:prov:2007-09"
3280   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3281   xsi:schemaLocation="http://schemas.xmlsoap.org/wsdl/
3282     http://schemas.xmlsoap.org/wsdl/
3283     http://www.w3.org/2006/02/addressing/wsdl
3284     http://www.w3.org/2006/02/addressing/wsdl/ws-addr-wsdl.xsd">
3285
3286   <types>
3287     <xsd:schema>
3288       <xsd:import namespace="urn:liberty:prov:2007-09"
3289         schemaLocation="liberty-idwsf-prov-v1.0.xsd" />
3290     </xsd:schema>
3291   </types>
3292
3293   <message name="PMGetDescriptor">
3294     <part name="body" element="prov:PMGetDescriptor"/>
3295   </message>
3296
3297   <message name="PMGetDescriptorResponse">
3298     <part name="body" element="prov:PMGetDescriptorResponse"/>
3299   </message>
3300
3301   <message name="PMActivate">
3302     <part name="body" element="prov:PMActivate"/>
3303   </message>
3304
3305   <message name="PMActivateResponse">
3306     <part name="body" element="prov:PMActivateResponse"/>
3307   </message>
3308
3309   <message name="PMDeactivate">
3310     <part name="body" element="prov:PMDeactivate"/>
3311   </message>
3312
3313   <message name="PMDeactivateResponse">
3314     <part name="body" element="prov:PMDeactivateResponse"/>
3315   </message>
3316
3317   <message name="PMDelete">
3318     <part name="body" element="prov:PMDelete"/>
3319   </message>
3320
3321   <message name="PMDeleteResponse">
3322     <part name="body" element="prov:PMDeleteResponse"/>
3323   </message>
3324
3325   <message name="PMGetStatus">
3326     <part name="body" element="prov:PMGetStatus"/>
3327   </message>
3328
3329   <message name="PMGetStatusResponse">
3330     <part name="body" element="prov:PMGetStatusResponse"/>
3331   </message>
3332
3333   <message name="PMRegisterDescriptor">
3334     <part name="body" element="prov:PMRegisterDescriptor"/>
3335   </message>
3336
3337   <message name="PMRegisterDescriptorResponse">
3338     <part name="body" element="prov:PMRegisterDescriptorResponse"/>
3339   </message>
```

```
3336 <message name="PMSetStatus">
3337   <part name="body" element="prov:PMSetStatus"/>
3338 </message>
3339 <message name="PMSetStatusResponse">
3340   <part name="body" element="prov:PMSetStatusResponse"/>
3341 </message>
3342
3343 <message name="PMUpdate">
3344   <part name="body" element="prov:PMUpdate"/>
3345 </message>
3346 <message name="PMUpdateResponse">
3347   <part name="body" element="prov:PMUpdateResponse"/>
3348 </message>
3349
3350 <message name="PMDelete">
3351   <part name="body" element="prov:PMDelete"/>
3352 </message>
3353 <message name="PMDeleteResponse">
3354   <part name="body" element="prov:PMDeleteResponse"/>
3355 </message>
3356
3357 <message name="PMDisable">
3358   <part name="body" element="prov:PMDisable"/>
3359 </message>
3360 <message name="PMDisableResponse">
3361   <part name="body" element="prov:PMDisableResponse"/>
3362 </message>
3363
3364 <message name="PMEDownload">
3365   <part name="body" element="prov:PMEDownload"/>
3366 </message>
3367 <message name="PMEDownloadResponse">
3368   <part name="body" element="prov:PMEDownloadResponse"/>
3369 </message>
3370
3371 <message name="PMEEnable">
3372   <part name="body" element="prov:PMEEnable"/>
3373 </message>
3374 <message name="PMEEnableResponse">
3375   <part name="body" element="prov:PMEEnableResponse"/>
3376 </message>
3377
3378 <message name="PMEGetInfo">
3379   <part name="body" element="prov:PMEGetInfo"/>
3380 </message>
3381 <message name="PMEGetInfoResponse">
3382   <part name="body" element="prov:PMEGetInfoResponse"/>
3383 </message>
3384
3385 <message name="PMERegister">
3386   <part name="body" element="prov:PMERegister"/>
3387 </message>
3388 <message name="PMERegisterResponse">
3389   <part name="body" element="prov:PMERegisterResponse"/>
3390 </message>
3391
3392 <message name="PMEUpload">
3393   <part name="body" element="prov:PMEUpload"/>
3394 </message>
3395 <message name="PMEUploadResponse">
3396   <part name="body" element="prov:PMEUploadResponse"/>
3397 </message>
3398
3399 <message name="Poll">
3400   <part name="body" element="prov:Poll"/>
3401 </message>
3402 <message name="PollResponse">
```

```
3403     <part name="body" element="prov:PollResponse"/>
3404 </message>
3405
3406 <message name="UpdateEPR">
3407   <part name="body" element="prov:UpdateEPR"/>
3408 </message>
3409 <message name="UpdateEPRResponse">
3410   <part name="body" element="prov:UpdateEPRResponse"/>
3411 </message>
3412
3413 <portType name="ProvisioningPort">
3414
3415   <operation name="PMGetDescriptor">
3416     <input message="tns:PMGetDescriptor"
3417       wsaw:Action="urn:liberty:prov:2007-09:PMGetDescriptor" />
3418     <output message="tns:PMGetDescriptorResponse"
3419       wsaw:Action="urn:liberty:prov:2007-09:PMGetDescriptorResponse" />
3420   </operation>
3421
3422   <operation name="PMActivate">
3423     <input message="tns:PMActivate"
3424       wsaw:Action="urn:liberty:prov:2007-09:PMActivate" />
3425     <output message="tns:PMActivateResponse"
3426       wsaw:Action="urn:liberty:prov:2007-09:PMActivateResponse" />
3427   </operation>
3428
3429   <operation name="PMDeactivate">
3430     <input message="tns:PMDeactivate"
3431       wsaw:Action="urn:liberty:prov:2007-09:PMDeactivate" />
3432     <output message="tns:PMDeactivateResponse"
3433       wsaw:Action="urn:liberty:prov:2007-09:PMDeactivateResponse" />
3434   </operation>
3435
3436   <operation name="PMDelete">
3437     <input message="tns:PMDelete"
3438       wsaw:Action="urn:liberty:prov:2007-09:PMDelete" />
3439     <output message="tns:PMDeleteResponse"
3440       wsaw:Action="urn:liberty:prov:2007-09:PMDeleteResponse" />
3441   </operation>
3442
3443   <operation name="PMGetStatus">
3444     <input message="tns:PMGetStatus"
3445       wsaw:Action="urn:liberty:prov:2007-09:PMGetStatus" />
3446     <output message="tns:PMGetStatusResponse"
3447       wsaw:Action="urn:liberty:prov:2007-09:PMGetStatusResponse" />
3448   </operation>
3449
3450   <operation name="PMRegisterDescriptor">
3451     <input message="tns:PMRegisterDescriptor"
3452       wsaw:Action="urn:liberty:prov:2007-09:PMRegisterDescriptor" />
3453     <output message="tns:PMRegisterDescriptorResponse"
3454       wsaw:Action="urn:liberty:prov:2007-09:PMRegisterDescriptorResponse" />
3455   </operation>
3456
3457   <operation name="PMSetStatus">
3458     <input message="tns:PMSetStatus"
3459       wsaw:Action="urn:liberty:prov:2007-09:PMSetStatus" />
3460     <output message="tns:PMSetStatusResponse"
3461       wsaw:Action="urn:liberty:prov:2007-09:PMSetStatusResponse" />
3462   </operation>
3463
3464   <operation name="PMUpdate">
3465     <input message="tns:PMUpdate"
3466       wsaw:Action="urn:liberty:prov:2007-09:PMUpdate" />
3467     <output message="tns:PMUpdateResponse"
3468       wsaw:Action="urn:liberty:prov:2007-09:PMUpdateResponse" />
3469   </operation>
```

```

3470
3471 <operation name="PMEDelete">
3472   <input message="tns:PMEDelete"
3473     wsaw:Action="urn:liberty:prov:2007-09:PMEDelete" />
3474   <output message="tns:PMEDeleteResponse"
3475     wsaw:Action="urn:liberty:prov:2007-09:PMEDeleteResponse" />
3476 </operation>
3477
3478 <operation name="PMEDisable">
3479   <input message="tns:PMEDisable"
3480     wsaw:Action="urn:liberty:prov:2007-09:PMEDisable" />
3481   <output message="tns:PMEDisableResponse"
3482     wsaw:Action="urn:liberty:prov:2007-09:PMEDisableResponse" />
3483 </operation>
3484
3485 <operation name="PMEDownload">
3486   <input message="tns:PMEDownload"
3487     wsaw:Action="urn:liberty:prov:2007-09:PMEDownload" />
3488   <output message="tns:PMEDownloadResponse"
3489     wsaw:Action="urn:liberty:prov:2007-09:PMEDownloadResponse" />
3490 </operation>
3491
3492 <operation name="PMEEnable">
3493   <input message="tns:PMEEnable"
3494     wsaw:Action="urn:liberty:prov:2007-09:PMEEnable" />
3495   <output message="tns:PMEEnableResponse"
3496     wsaw:Action="urn:liberty:prov:2007-09:PMEEnableResponse" />
3497 </operation>
3498
3499 <operation name="PMEGetInfo">
3500   <input message="tns:PMEGetInfo"
3501     wsaw:Action="urn:liberty:prov:2007-09:PMEGetInfo" />
3502   <output message="tns:PMEGetInfoResponse"
3503     wsaw:Action="urn:liberty:prov:2007-09:PMEGetInfoResponse" />
3504 </operation>
3505
3506 <operation name="PMERegister">
3507   <input message="tns:PMERegister"
3508     wsaw:Action="urn:liberty:prov:2007-09:PMERegister" />
3509   <output message="tns:PMERegisterResponse"
3510     wsaw:Action="urn:liberty:prov:2007-09:PMERegisterResponse" />
3511 </operation>
3512
3513 <operation name="PMEUpload">
3514   <input message="tns:PMEUpload"
3515     wsaw:Action="urn:liberty:prov:2007-09:PMEUpload" />
3516   <output message="tns:PMEUploadResponse"
3517     wsaw:Action="urn:liberty:prov:2007-09:PMEUploadResponse" />
3518 </operation>
3519
3520 <operation name="Poll">
3521   <input message="tns:Poll"
3522     wsaw:Action="urn:liberty:prov:2007-09:Poll" />
3523   <output message="tns:PollResponse"
3524     wsaw:Action="urn:liberty:prov:2007-09:PollResponse" />
3525 </operation>
3526
3527 <operation name="UpdateEPR">
3528   <input message="tns:UpdateEPR"
3529     wsaw:Action="urn:liberty:prov:2007-09:UpdateEPR" />
3530   <output message="tns:UpdateEPRResponse"
3531     wsaw:Action="urn:liberty:prov:2007-09:UpdateEPRResponse" />
3532 </operation>
3533
3534 </portType>
3535
3536 <!--

```



```
3537 An example of a binding and service that can be used with this
3538 abstract service description is provided below.
3539 -->
3540
3541 <binding name="ProvisioningBinding" type="tns:ProvisioningPort">
3542
3543   <soap:binding style="document"
3544     transport="http://schemas.xmlsoap.org/soap/http"/>
3545
3546   <operation name="PMGetDescriptor">
3547     <input> <soap:body use="literal"/> </input>
3548     <output> <soap:body use="literal"/> </output>
3549   </operation>
3550
3551   <operation name="PMActivate">
3552     <input> <soap:body use="literal"/> </input>
3553     <output> <soap:body use="literal"/> </output>
3554   </operation>
3555
3556   <operation name="PMDeactivate">
3557     <input> <soap:body use="literal"/> </input>
3558     <output> <soap:body use="literal"/> </output>
3559   </operation>
3560
3561   <operation name="PMDelete">
3562     <input> <soap:body use="literal"/> </input>
3563     <output> <soap:body use="literal"/> </output>
3564   </operation>
3565
3566   <operation name="PMGetStatus">
3567     <input> <soap:body use="literal"/> </input>
3568     <output> <soap:body use="literal"/> </output>
3569   </operation>
3570
3571   <operation name="PMRegisterDescriptor">
3572     <input> <soap:body use="literal"/> </input>
3573     <output> <soap:body use="literal"/> </output>
3574   </operation>
3575
3576   <operation name="PMSetStatus">
3577     <input> <soap:body use="literal"/> </input>
3578     <output> <soap:body use="literal"/> </output>
3579   </operation>
3580
3581   <operation name="PMUpdate">
3582     <input> <soap:body use="literal"/> </input>
3583     <output> <soap:body use="literal"/> </output>
3584   </operation>
3585
3586   <operation name="PMDelete">
3587     <input> <soap:body use="literal"/> </input>
3588     <output> <soap:body use="literal"/> </output>
3589   </operation>
3590
3591   <operation name="PMEDisable">
3592     <input> <soap:body use="literal"/> </input>
3593     <output> <soap:body use="literal"/> </output>
3594   </operation>
3595
3596   <operation name="PMEDownload">
3597     <input> <soap:body use="literal"/> </input>
3598     <output> <soap:body use="literal"/> </output>
3599   </operation>
3600
3601   <operation name="PMEEnable">
3602     <input> <soap:body use="literal"/> </input>
3603     <output> <soap:body use="literal"/> </output>
```

```
3604     </operation>
3605
3606     <operation name="PMEGetInfo">
3607         <input> <soap:body use="literal"/> </input>
3608         <output> <soap:body use="literal"/> </output>
3609     </operation>
3610
3611     <operation name="PMERegister">
3612         <input> <soap:body use="literal"/> </input>
3613         <output> <soap:body use="literal"/> </output>
3614     </operation>
3615
3616     <operation name="PMEUpload">
3617         <input> <soap:body use="literal"/> </input>
3618         <output> <soap:body use="literal"/> </output>
3619     </operation>
3620
3621     <operation name="Poll">
3622         <input> <soap:body use="literal"/> </input>
3623         <output> <soap:body use="literal"/> </output>
3624     </operation>
3625
3626     <operation name="UpdateEPR">
3627         <input> <soap:body use="literal"/> </input>
3628         <output> <soap:body use="literal"/> </output>
3629     </operation>
3630
3631 </binding>
3632
3633 <service name="ProvisioningService">
3634
3635     <port name="ProvisioningPort" binding="tns:ProvisioningBinding">
3636
3637         <!-- Modify with the REAL SOAP endpoint -->
3638
3639         <soap:address location="http://example.com/provisioning"/>
3640
3641     </port>
3642
3643 </service>
3644
3645 </definitions>
3646
```

3647 **References**

3648 **Normative**

- 3649 [LibertyPMM] Cahill, Conor P., eds. "Liberty ID-WSF Provisioned Module Manager Service Specification," Version
3650 1.0, Liberty Alliance Project (14 December 2007). <http://www.projectliberty.org/specs>
- 3651 [LibertyPROV] Cahill, Conor P., eds. "Liberty ID-WSF Provisioning Service Specification," Version 1.0, Liberty
3652 Alliance Project (14 December 2007). <http://www.projectliberty.org/specs>
- 3653 [LibertyDisco] Cahill, Conor, Hodges, Jeff, eds. "Liberty ID-WSF Discovery Service Specification," Version 2.0-
3654 errata-v1.0, Liberty Alliance Project (29 November, 2006). <http://www.projectliberty.org/specs>
- 3655 [LibertyIDPP] Kellomäki, Sampo, Lockhart, Rob, eds. "Liberty ID-SIS Personal Profile Service Specification,"
3656 Version 1.1, Liberty Alliance Project (29 September, 2005). <http://www.projectliberty.org/specs>
- 3657 [LibertyDP] Cahill, Conor P., eds. "Liberty ID-WSF Design Patterns Specification," Version 1.0, Liberty Alliance
3658 Project (14 December 2007). <http://www.projectliberty.org/specs>
- 3659 [LibertyInteract] Aarts, Robert, Madsen, Paul, eds. "Liberty ID-WSF Interaction Service Specification," Version 2.0-
3660 errata-v1.0, Liberty Alliance Project (21 April, 2007). <http://www.projectliberty.org/specs>
- 3661 [LibertyProtSchema] Cantor, Scott, Kemp, John, eds. "Liberty ID-FF Protocols and Schema Specification," Version
3662 1.2-errata-v3.0, Liberty Alliance Project (14 December 2004). <http://www.projectliberty.org/specs>
- 3663 [LibertyReg] Kemp, John, eds. "Liberty Enumeration Registry Governance," Version 1.1, Liberty Alliance Project (14
3664 December, 2004). <http://www.projectliberty.org/specs>
- 3665 [LibertySOAPAauthn] Hodges, Jeff, Aarts, Robert, Madsen, Paul, Cantor, Scott, eds. "Liberty ID-WSF Authentication,
3666 Single Sign-On, and Identity Mapping Services Specification," v2.0-errata-1.0-01, Liberty Alliance Project
3667 (28 November, 2006). <http://www.projectliberty.org/specs>
- 3668 [LibertySOAPBinding] Hodges, Jeff, Kemp, John, Aarts, Robert, Whitehead, Greg, Madsen, Paul, eds. "Liberty
3669 ID-WSF SOAP Binding Specification," Version 2.0-errata-v1.0, Liberty Alliance Project (21 April, 2007).
3670 <http://www.projectliberty.org/specs>
- 3671 [RFC2119] S. Bradner "Key words for use in RFCs to Indicate Requirement Levels," RFC 2119, The Internet
3672 Engineering Task Force (March 1997). <http://www.ietf.org/rfc/rfc2119.txt>
- 3673 [RFC2251] "Lightweight Directory Access Protocol (v3)," M. Wahl T. Howes S. Kille (December 1997). RFC 2251,
3674 Internet Engineering Task Force <http://www.ietf.org/rfc/rfc2251.txt>
- 3675 [RFC2252] "Lightweight Directory Access Protocol (v3): Attribute Syntax Definitions," M. Wahl A.
3676 Coulbeck T. Howes S. Kille (December 1997). RFC 2252, Internet Engineering Task Force
3677 <http://www.ietf.org/rfc/rfc2252.txt>
- 3678 [RFC2891] Howes, T., Wahl, M., eds. (August 2000). "LDAP Control Extension for Server Side Sorting of Search
3679 Results," RFC 2891, Internet Engineering Task Force <http://www.ietf.org/rfc/rfc2891.txt>
- 3680 [SAMLCore11] Maler, Eve, Mishra, Prateek, Philpott, Rob, eds. (2 September 2003). "Assertions and Pro-
3681 tocol for the OASIS Security Assertion Markup Language (SAML) V1.1," SAML v1.1, OASIS
3682 Standard, Organization for the Advancement of Structured Information Standards [http://www.oasis-](http://www.oasis-open.org/committees/download.php/3406/oasis-sstc-saml-core-1.1.pdf)
3683 [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)

- 3684 [SAMLCore2] Cantor, Scott, Kemp, John, Philpott, Rob, Maler, Eve, eds. (15 March 2005). "Assertions
3685 and Protocol for the OASIS Security Assertion Markup Language (SAML) V2.0," SAML V2.0, OA-
3686 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-
3687 open.org/security/saml/v2.0/saml-core-2.0-os.pdf)
- 3688 [SAMLProf2] Hughes, John, Cantor, Scott, Hodges, Jeff, Hirsch, Frederick, Mishra, Prateek, Philpott, Rob, Maler,
3689 Eve, eds. (15 March, 2005). "Profiles for the OASIS Security Assertion Markup Language (SAML) V2.0,"
3690 SAML V2.0, OASIS Standard, Organization for the Advancement of Structured Information Standards
3691 <http://docs.oasis-open.org/security/saml/v2.0/saml-profiles-2.0-os.pdf>
- 3692 [Schema1-2] Thompson, Henry S., Beech, David, Maloney, Murray, Mendelsohn, Noah, eds. (28 October
3693 2004). "XML Schema Part 1: Structures Second Edition," Recommendation, World Wide Web Consortium
3694 <http://www.w3.org/TR/xmlschema-1/>
- 3695 [SOAPv1.1] "Simple Object Access Protocol (SOAP) 1.1," Box, Don, Ehnebuske, David, Kakivaya, Gopal, Layman,
3696 Andrew, Mendelsohn, Noah, Nielsen, Henrik Frystyk, Winer, Dave, eds. World Wide Web Consortium W3C
3697 Note (08 May 2000). <http://www.w3.org/TR/2000/NOTE-SOAP-20000508/>
- 3698 [XML] Bray, Tim, Paoli, Jean, Sperberg-McQueen, C. M., Maler, Eve, Yergeau, Francois, eds. (04 February 2004).
3699 "Extensible Markup Language (XML) 1.0 (Third Edition)," Recommendation, World Wide Web Consortium
3700 <http://www.w3.org/TR/2004/REC-xml-20040204>
- 3701 [XMLDsig] Eastlake, Donald, Reagle, Joseph, Solo, David, eds. (12 Feb 2002). "XML-Signature Syntax and
3702 Processing," Recommendation, World Wide Web Consortium <http://www.w3.org/TR/xmlsig-core>
- 3703 [XPath] Clark, J., DeRose, S., eds. (16 November 1999). "XML Path Language (XPath) Version 1.0,"
3704 Recommendation, W3C <http://www.w3.org/TR/xpath> [August 2003].

3705 Informative

- 3706 [LibertyACT] Cahill, Conor P., eds. "Liberty Advanced Client Technologies," Version 1.0, Liberty Alliance Project
3707 (14 December 2007). <http://www.projectliberty.org/specs>
- 3708 [LibertyIDWSFGuide] Weitzel, David, eds. "Liberty ID-WSF Implementation Guide," Version 2.0-02, Liberty
3709 Alliance Project (13 January, 2005). <http://www.projectliberty.org/specs>
- 3710 [LibertyIDPPGuide] Kellomäki, Sampo, Lockhart, Rob, eds. "Liberty ID-SIS Personal Profile Service Implementation
3711 Guidelines," Version 1.1, Liberty Alliance Project (29 September, 2005). <http://www.projectliberty.org/specs>
- 3712 [LibertyIDWSFOverview] Tourzan, Jonathan, Koga, Yuzo, eds. "Liberty ID-WSF Web Services Framework
3713 Overview," Version 2.0, Liberty Alliance Project (30 July, 2006). <http://www.projectliberty.org/specs>