



PPS (Production Planning and Scheduling) Part 3: Profile Specifications, Version 1.0

Committee Specification 01

11 April 2008

Specification URIs:

<http://docs.oasis-open.org/pps/v1.0/cs01/pps-profile-specifications-1.0-cs01.doc>
<http://docs.oasis-open.org/pps/v1.0/cs01/pps-profile-specifications-1.0-cs01.html>
<http://docs.oasis-open.org/pps/v1.0/cs01/pps-profile-specifications-1.0-cs01.pdf>

Previous Version:

<http://docs.oasis-open.org/pps/v1.0/pr02/pps-profile-specifications-1.0-pr02.doc>
<http://docs.oasis-open.org/pps/v1.0/pr02/pps-profile-specifications-1.0-pr02.html>
<http://docs.oasis-open.org/pps/v1.0/pr02/pps-profile-specifications-1.0-pr02.pdf>

Latest Version:

<http://docs.oasis-open.org/pps/v1.0/pps-profile-specifications-1.0.doc>
<http://docs.oasis-open.org/pps/v1.0/pps-profile-specifications-1.0.html>
<http://docs.oasis-open.org/pps/v1.0/pps-profile-specifications-1.0.pdf>

Technical Committee:

OASIS Production Planning and Scheduling TC

Chair(s):

Yasuyuki Nishioka, PSLX Forum / Hosei University

Editor(s):

Yasuyuki Nishioka, PSLX Forum / Hosei University
Koichi Wada, PSLX Forum

Related work:

This specification is related to:

- Universal Business Language 2.0

Declared XML Namespace(s):

<http://docs.oasis-open.org/pps/ns>

Abstract:

OASIS PPS (Production Planning and Scheduling) Standard deals with problems in all manufacturing companies who want to have a sophisticated information system for production planning and scheduling. PPS standard provides XML schema and communication protocols for information exchange among manufacturing application programs in the web-services environment. This document especially focuses on profiles of application programs that may exchange the messages defined in this standard. The profile shows capability of application programs in terms of services for message exchange. The profile can be used for definition of a minimum level of implementation of application programs who are involved in a community of data exchange.

Status:

This document was last revised or approved by the PPS TC on the above date. The level of approval is also listed above. Check the "Latest Version" or "Latest Approved Version" location noted above for possible later revisions of this document.

Technical Committee members should send comments on this specification to the Technical Committee's email list. Others should send comments to the Technical Committee by using the "Send A Comment" button on the Technical Committee's web page at <http://www.oasis-open.org/committees/pps/>.

For information on whether any patents have been disclosed that may be essential to implementing this specification, and any offers of patent licensing terms, please refer to the Intellectual Property Rights section of the Technical Committee web page (<http://www.oasis-open.org/committees/pps/ipr.php>).

The non-normative errata page for this specification is located at <http://www.oasis-open.org/committees/pps/>.

Notices

Copyright © OASIS® 2007. All Rights Reserved.

All capitalized terms in the following text have the meanings assigned to them in the OASIS Intellectual Property Rights Policy (the "OASIS IPR Policy"). The full Policy may be found at the OASIS website.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published, and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this section are included on all such copies and derivative works. However, this document itself may not be modified in any way, including by removing the copyright notice or references to OASIS, except as needed for the purpose of developing any document or deliverable produced by an OASIS Technical Committee (in which case the rules applicable to copyrights, as set forth in the OASIS IPR Policy, must be followed) or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by OASIS or its successors or assigns.

This document and the information contained herein is provided on an "AS IS" basis and OASIS DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY OWNERSHIP RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

OASIS requests that any OASIS Party or any other party that believes it has patent claims that would necessarily be infringed by implementations of this OASIS Committee Specification or OASIS Standard, to notify OASIS TC Administrator and provide an indication of its willingness to grant patent licenses to such patent claims in a manner consistent with the IPR Mode of the OASIS Technical Committee that produced this specification.

OASIS invites any party to contact the OASIS TC Administrator if it is aware of a claim of ownership of any patent claims that would necessarily be infringed by implementations of this specification by a patent holder that is not willing to provide a license to such patent claims in a manner consistent with the IPR Mode of the OASIS Technical Committee that produced this specification. OASIS may include such claims on its website, but disclaims any obligation to do so.

OASIS takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Information on OASIS' procedures with respect to rights in any document or deliverable produced by an OASIS Technical Committee can be found on the OASIS website. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this OASIS Committee Specification or OASIS Standard, can be obtained from the OASIS TC Administrator. OASIS makes no representation that any information or list of intellectual property rights will at any time be complete, or that any claims in such list are, in fact, Essential Claims.

The names "OASIS", PPS are trademarks of OASIS, the owner and developer of this specification, and should be used only to refer to the organization and its official outputs. OASIS welcomes reference to, and implementation and use of, specifications, while reserving the right to enforce its marks against misleading uses. Please see <http://www.oasis-open.org/who/trademark.php> for above guidance.

Table of Contents

1	Introduction.....	6
1.1	Terminology.....	6
1.2	Normative References.....	6
1.3	Non-Normative References.....	6
1.4	Conformance.....	6
1.5	Terms and definitions.....	7
2	Application profile Definitions.....	8
2.1	General.....	8
2.2	Structure of profile definitions.....	8
2.3	Standard profile definitions.....	9
2.4	Extended profile definitions.....	11
2.5	Revision rule.....	12
3	Implementation profiles.....	13
3.1	General.....	13
3.2	Structure of implementation profiles.....	13
3.3	Level of implementation.....	15
3.4	Profile inquiry.....	15
4	XML Elements.....	17
4.1	AppProfile Element.....	17
4.2	AppDocument Element.....	18
4.3	AppObject Element.....	18
4.4	AppProperty Element.....	19
4.5	Data Type Element.....	19
4.6	Unit Type Element.....	20
4.7	Enumeration Element.....	21
4.8	EnumElement Element.....	21
4.9	ImplementProfile Element.....	22
4.10	ImplementDocument Element.....	23
4.11	ImplementAction Element.....	24
4.12	ImplementProperty Element.....	24
4.13	ImplementEvent Element.....	25
A.	Acknowledgements.....	27
B.	Revision History.....	28

Figures

Figure 1 Structure of profile specifications.....	8
Figure 2 Profile definition	9
Figure 3 Concept of communication availability between implementations	13
Figure 4 Structure of ImplementProfile	14

1 Introduction

This part of PPS specification prescribes definition of implementation profile that shows capability of information exchange with other application programs using PPS transaction messages [PPS02]. In order to define an implementation profile for each application program, this document also defines and prescribes application profile specification that should be consistent with all implementation profiles. An application profile allows each individual program to describe their capability.

Application profile shows a set of domain documents, domain objects and domain properties, which may be used in a message of production planning and scheduling application programs. Implementation profile shows domain documents, domain objects and domain properties that the application program can deal with correctly. The implementation profile also shows an implementation level of the application program. By collecting implementation profiles, a system integrator can arrange particular messaging in particular application scenario.

1.1 Terminology

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

1.2 Normative References

- [RFC2119] S. Bradner, *Key words for use in RFCs to Indicate Requirement Levels*, <http://www.ietf.org/rfc/rfc2119.txt>, IETF RFC 2119, March 1997.
- [PPS01] PPS (Production Planning and Scheduling) Part 1: Core Elements, Version 1.0, Public Review Draft 01, <http://www.oasis-open.org/committees/pps/>
- [PPS02] PPS (Production Planning and Scheduling) Part 2: Transaction Messages, Version 1.0, Public Review Draft 01, <http://www.oasis-open.org/committees/pps/>
- [PPS04] PPS (Production Planning and Scheduling) Standard Profile Definition, Version 1.0, Committee Draft 01, <http://www.oasis-open.org/committees/pps/>
- [PATH] XML Path Language (XPath) Version 1.0, <http://www.w3.org/TR/xpath>

1.3 Non-Normative References

- [PSLXWP] PSLX Consortium, PSLX White Paper - APS Conceptual definition and implementation, <http://www.pslx.org/>
- [PSLX001] PSLX Technical Standard, Version 2, Part 1: Enterprise Model (in Japanese), Recommendation of PSLX Forum, <http://www.pslx.org/>
- [PSLX002] PSLX Technical Standard, Version 2, Part 2: Activity Model (in Japanese), Recommendation of PSLX Forum, <http://www.pslx.org/>
- [PSLX003] PSLX Technical Standard, Version 2, Part 3: Object Model (in Japanese), Recommendation of PSLX Forum, <http://www.pslx.org/>

1.4 Conformance

A document of profile confirms OASIS PPS Profile Specifications if all elements in the artifact are consistent with the normative text of this specification, and the document can be processed properly with the XML schema that can be downloaded from the following URI.

<http://docs.oasis-open.org/pps/v1.0/pps-profile-specifications.xsd>

43 **1.5 Terms and definitions**

44 **Domain document**

45 Document that is the content of message sent or received between application programs.
46 Message document consists of verb and noun part. Verbs such as add, change and remove
47 provide types of messages, while nouns show the classes of domain objects.

48 **Domain object**

49 Object that corresponds to production planning and scheduling information in a view of operations
50 management. Domain objects are the contents of transaction elements, and represented by
51 primitive elements.

52 **Domain property**

53 Any parameters that show a property of a domain object. A domain property is represented by
54 XML attributes of the primitive element, or XML child elements of the primitive elements. A
55 domain object may have multiple domain properties with the same property name, if the XML
56 child element is multiple.

57 **Primitive element**

58 XML element that represents a primitive object in the production planning and scheduling domain,
59 nine primitive elements are defined in the part 1 of PPS standard. Every domain objects are
60 represented by the primitive elements.

61 **Transaction element**

62 XML element that represents a message document to sent or received between application
63 programs. Transaction element has primitive elements as contents of domain information.
64 Transaction element also has a header information as well as application specific information.

65 **Implementation profile**

66 Specification of capability of application program including a list of available documents and
67 transactions that may be exchanged in PPS transaction messages in production planning and
68 scheduling problems.

69 **Application profile definition**

70 Collections of profile specifications for all application programs that may be involved in the
71 communication exchanging PPS transaction messages. This provides all available domain
72 documents, domain objects and domain primitives.

73 **Messaging model**

74 Simple patterns of messaging between sender and receiver, or initiator and responder, The 5
75 message models are defined from an application independent perspective, by defining 7
76 message types considering the verbs aspect of message documents.

77 2 Application profile Definitions

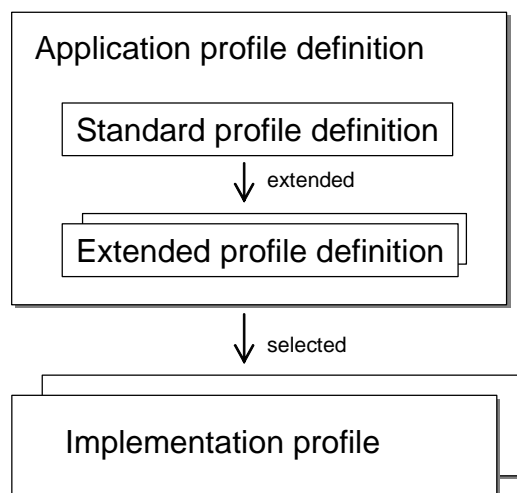
78 2.1 General

79 Application profile definition is a set of specifications for all application programs that may be involved in
80 the communication exchanging PPS transaction messages. Each application program may send and
81 receive messages that consist of domain documents, domain objects and domain properties. The
82 application profile definition provides all available domain documents, domain objects and domain
83 primitives.

84 Several application profile definitions may exist developing by particular group for particular problem
85 domains. Regarding such situation, this part of standard defines a standard profile definition as a
86 template of application profile definitions. A standard profile definition can be extended to an extended
87 profile definition for particular group in local domain.

88 Figure 1 shows the structure of application profile definitions. Application profile definitions consist of
89 standard profile definitions and extended profile definitions. Figure also shows the relation of application
90 profile definitions with implementation profiles that are defined in the next section.

91



92

93

Figure 1 Structure of profile specifications

94

95 Application programs can exchange their messages correctly when they understand the semantics of
96 information in the message. In order to do this, application profile definition helps agreement of common
97 usage and understanding of domain documents, domain objects and domain properties.

98 As an instance of standard profile definition, PPS TC provides the PPS standard profile definition [PPS04].
99 However, this part of standard only shows general rules and structures of a standard profile definition.

100 2.2 Structure of profile definitions

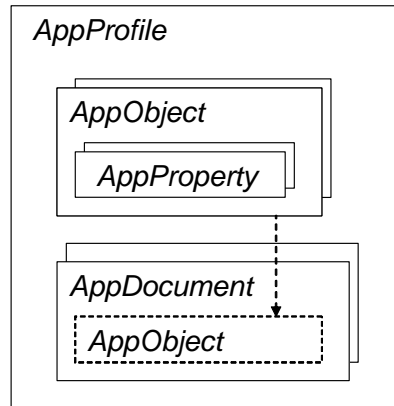
101 Application profile definition SHOULD have a list of domain documents, a list of domain objects. In
102 addition, application profile definition MAY have a list of enumerations, data types and unit types, which
103 shows available value set, data types and unit types of a domain property of a domain object, respectively.

104 Application profile definition SHOULD be specified by *AppProfile* element defined in Section 4.1. This
105 element SHOULD appear in the top level of the XML document.

106 All candidates of domain documents, which may be used by any application program who sends or
107 receives a message in the whole system, SHOULD be specified in *AppDocument* element under the
108 *AppProfile* element.

109 All candidates of domain objects, which may be used in any domain objects defined in *AppDocument*
110 elements, SHOULD be specified in *AppObject* element under the *AppProfile* element. An *AppObject* has
111 a list of properties that represent the characteristics of the object. Each property SHOULD be specified in
112 *AppProperty* under the *AppObject*.

113



114

115

Figure 2 Profile definition

116

117 The structure of application profile definition is illustrated in Figure 2. Domain document represented by
118 *AppDocument* has domain objects represented by *AppObject*. These domain objects are the same class
119 objects defined as a class in the application profile. The profile defines domain objects independent from
120 domain documents, because the domain objects may be referred from different kinds of domain
121 documents.

122

123 Example 1 Application profile definition

```
124 <AppProfile prefix="pps" name="http://www.oasis-open.org/committees/pps/profile-1.0">
125 <AppObject name="Product" primitive="Item">
126 <AppProperty name="id" path="@id"/>
127 <AppProperty name="name" path="@name"/>
128 ...
129 <AppProperty name="Size" path="Spec[@type="size"]/@value"/>
130 <AppProperty name="Color" path="Spec[@type="color"]/@value"/>
131 ...
132 </AppObject>
133 ...
134 <AppDocument name="ProductRecord" object="Product"/>
135 <AppDocument name="ProductInventory" object="Product"/>
136 <AppDocument name="BillOfMaterials" object="Product"/>
137 <AppDocument name="BillOfResources" object="Product"/>
138 ...
139 </AppProfile>
```

140

141

142 2.3 Standard profile definitions

143 An application profile definition that does not have a base profile SHOULD be a standard profile definition.
144 Standard profile definition SHOULD specified in consistent with the following rules:

- 145 • Standard profile definition SHOULD have a name to identify the definition among all application
146 programs in the past, current and future.

- 147 • The name of standard profile definition contains information of revision, and the revision of the
148 definition SHOULD follow the rule defined in Section 2.5.
- 149 • Standard profile definition SHOULD NOT have a base definition as a reference of other standard
150 profile definitions.
- 151 • Standard profile definition SHOULD be published and accessible by all application programs in the
152 problem domain via Internet by announcing the URL the application can download the document.
- 153 • Standard profile definition SHOULD have at least all the domain objects in Table 1. The domain
154 objects correspond to the primitive elements defined in [PPS01].
- 155 • Every domain object in a standard profile definition SHOULD have at least all the domain properties
156 in Table 2 and Table 3. The properties in Table 2 correspond to attributes in the primitive elements in
157 [PPS01], and the properties in Table 3 correspond to sub-elements and its attribute in the primitive
158 elements.

159

160

Table 1 Domain objects required in standard profile definitions

Object Name	XML Element	Description
Party	<i>Party</i>	Party such as customers and suppliers
Plan	<i>Plan</i>	Plan of production, capacity, inventory, etc.
Order	<i>Order</i>	Request of products and services
Item	<i>Item</i>	Items to produce or consume
Resource	<i>Resource</i>	Production resource such as machine and personnel
Process	<i>Process</i>	Production process
Lot	<i>Lot</i>	Actual lots produced in the plant
Task	<i>Task</i>	Actual tasks on certain resources
Operation	<i>Operation</i>	Actual operations in the plant

161

162

Table 2 Domain properties required in each domain object in standard profile definitions (part 1 of 2)

Property Name	Attribute Name	Description	Data Type
id	<i>id</i>	Identifier	string
key	<i>key</i>	Key in the data storage	string
name	<i>name</i>	Name of the object	string
parent	<i>parent</i>	Parent name of the object	string
type	<i>type</i>	Qualifier to make a subclass	string
status	<i>status</i>	Status of the object	string

163

164

Table 3 Domain properties required in each domain object in standard profile definitions (part 2 of 2)

Property Name	XML Element	Attribute Name	Description	Data Type
priority	<i>Priority</i>	value	Priority of the object	string

display	<i>Display</i>	value	Display value	string
description	<i>Description</i>	value	Description value	string
author	<i>Author</i>	value	Author of the object	string
date	<i>Date</i>	value	Date of the object	string

165

166 2.4 Extended profile definitions

167 Standard profile definition MAY be extended by an extended profile definition. This is also represented by
168 *AppProfile* element.

169 Extended profile definition SHOULD select domain documents, domain objects and domain properties
170 from a standard profile definition. Extended profile definition MAY add new domain documents, domain
171 objects and domain properties.

172 Additional information of domain documents, domain objects and domain properties SHOULD be defined
173 in the same way as the definition in standard profile definitions. However, the domain documents, domain
174 objects and domain properties selected from the standard profile are defined only by the name of the
175 document, object or property.

176 Extended profile definitions MAY NOT have all the contents of corresponding standard profile definition.
177 When some contents have not been selected, it means that the availability of the application profile
178 definition is restricted.

179 Extended profile definitions MAY modify the domain documents, domain objects and domain properties
180 addressed in the standard profile. In order to modify the definition, extended profile SHOULD describe
181 new contents with the same identification name of the document, object or property.

182 Enumerations, data types and unit types MAY be selected from the standard profile or added to the
183 standard profile. In order to select an enumeration, data type and unit type, extended profile SHOULD
184 only have the name of the definition.

185 All Extended profile definitions SHOULD have a reference of a standard profile definition, which is the
186 base of extension.

187

188 Example 2 Extended application profile

```

189 <AppProfile prefix="ex1" name="http://www.pslx.org/profile-1" base="http://www.oasis-open.org/committees/pps/profile-
190 1.0">
191   <Enumeration name="groupType">
192     <EnumElement name="high" description="description of a"/>
193     <EnumElement name="low" description="description of b"/>
194   </Enumeration>
195   <AppObject name="Customer"/>
196   <AppObject name="Supplier"/>
197   <AppObject name="Consumer">
198     <AppProperty name="id"/>
199     <AppProperty name="name"/>
200     <AppProperty name="group" path="Spec[type='pslx:group']/@value" enumeration="groupType"/>
201   </AppObject>
202   <AppDocument name="CustomerRecord"/>
203   <AppDocument name="SupplierRecord"/>
204   <AppDocument name="ConsumerRecord" />
205 </AppProfile>

```

206

207 Example 2 shows an application profile extended from the standard profile. The new profile has additional
208 enumeration named “groupType”, and then a new Consumer object is defined with a new property named
209 group that has the enumeration type.

210 2.5 Revision rule

211 After an application profile definition has been created, many application programs are developed
212 according to the specification. By experiences from industries, the specification may be required to modify
213 for certain reasons and/or new findings in the application domain.

214 Any application profile SHOULD NOT be changed without keeping the following rules after when the
215 profile definition once published. Otherwise, the new profile SHOULD have a new name that doesn't have
216 any relation with the previous one.

217 There are two revision levels. One is a revision that the system developers SHOULD deal with the new
218 specification. The other is editorial revision where the any program doesn't need to care. To inform the
219 former cases, the name of profile SHOULD be changed by adding the revision numbers. For the latter
220 cases, instead of changing the name of profile, the actual file name of the profile, specified at the *location*
221 attribute in the *AppProfile* element SHOULD be changed.

222 In order to represent the revision status in the profile name, there are two portions of digits in the name of
223 profile definitions: major revision and minor revision. They are following the original identification name or
224 the profile separated by dash "-" mark. The two portion is separated by the dot "." character.

225 When the major version increases, it:

- 226 • SHOULD NOT change the name of the profile excepting the portion representing the revision status.
- 227 • SHOULD NOT change the prefix name in the attribute of *AppProfile* element.

228 When the minor version increases, it:

- 229 • SHOULD follow the rule of major version increasing,
- 230 • SHOULD NOT remove the domain objects,
- 231 • SHOULD NOT remove the domain properties,
- 232 • SHOULD NOT remove the domain documents,
- 233 • SHOULD NOT change the domain object in any domain document.

234

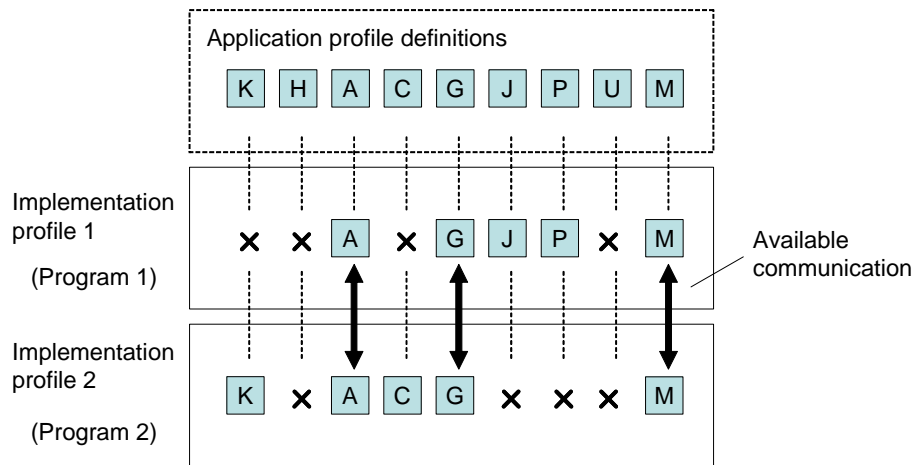
235 **3 Implementation profiles**

236 **3.1 General**

237 Application program may not have all capability in dealing with the domain documents, domain objects
238 and domain properties defined in the application profile definitions. Implementation profiles are the
239 selection of domain documents, domain objects and domain properties from application profile definitions
240 by application programs depending on the capability of the program.

241 When an application program tries to send a message to another application program, system integrator
242 may need to confirm whether or not the receiving application program has capability to response the
243 message. To confirm the capability of any application program, section 3.4 provides the method how to
244 get the information by receiving an implementation profile from the program.

245



246

247 *Figure 3 Concept of communication availability between implementations*

248

249 Figure 3 explains a concept of communication availability between two application programs. Each
250 application program that refers a same application profile definition has a set of capabilities by selecting
251 from the all items defined in the application profile. Two application programs can exchange a message
252 properly if the both implementations have the corresponding capability.

253 An application program MAY have two or more than two implementation profiles each of which
254 corresponding different application profile definitions. An implementation profile SHOULD have a
255 corresponding application profile definition.

256 **3.2 Structure of implementation profiles**

257 Implementation profiles defined for application programs SHOULD be specified by *ImplementProfile*
258 element in XML format. The information includes domain documents, domain objects and domain
259 properties available to process by the application program. Every domain document has the level of
260 implementation that shows the application program have all functions or not in terms of transactions
261 defined in [PPS02].

262 As an implementation profile has a reference to an application profile definition, it doesn't show whether
263 the application profile is a standard or extended. From the perspective of application programs, standard
264 profile definition and extended profile definition are equivalent.

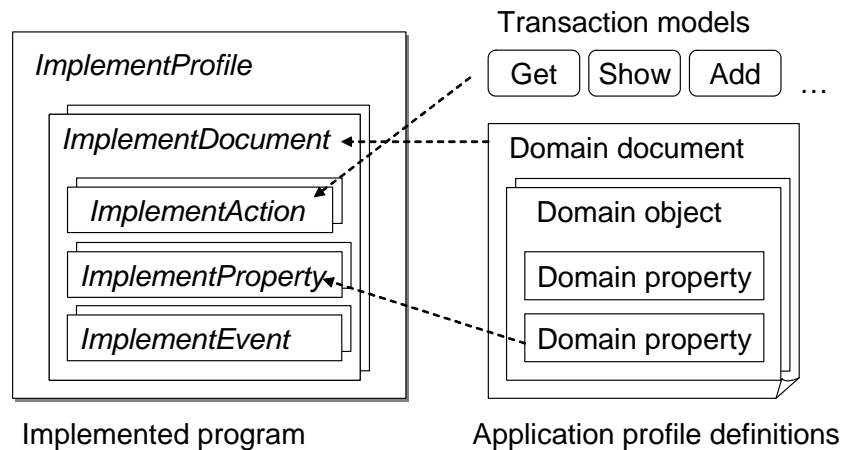
265 The structure of *ImplementProfile* element is a special case of the transaction element defined in [PPS02].
266 Therefore, this can be regarded as a transaction document that is send or receive through a PPS

267 transaction process. Using Get and Show transactions, two application programs can exchange the
268 implementation profile.

269 An *ImplementationProfile* element has *ImplementDocument* elements each of which represents
270 availability for any domain document. An *ImplementDocument* element has *ImplementAction*,
271 *ImplementProperty* and *ImplementEvent*.

272 *ImplementAction* element represents information of implemented transaction, and *ImplementProperty*
273 element represents implemented properties of the domain object. *ImplementEvent* represents any event
274 definitions that the application program monitors properties and publish notifications of event defined on
275 the property. Figure 4 shows the structure of *ImplementationProfile*, *ImplementDocument*, *ImplementAction*,
276 and *ImplementProperty* elements.

277



278

279

Figure 4 Structure of *ImplementationProfile*

280

281 All domain documents represented by *ImplementationProfile* SHOULD be in the list of the corresponding
282 application profile definition.

283 Example 3 shows an implementation profile of an application program that communicates with other
284 program under an application profile. Then the implementation profile of Example 3 is the selection of the
285 application profile representing domain documents, transaction types and domain properties.

286

287 Example 3. Implementation profile of a program for an application profile shown in Example 1

```
288 <ImplementationProfile id="001" transaction="001" action="Notify" sender="APP1"  
289 profile="http://www.oasis-open.org/committees/pps/profile-1.0">  
290 <ImplementDocument name="ProductRecord">  
291 <ImplementAction action="Get" level="1"/>  
292 <ImplementAction action="Show" level="1"/>  
293 <ImplementAction action="Add" level="2"/>  
294 <ImplementProperty name="id" title="Company ID"/>  
295 <ImplementProperty name="name" title="Company name"/>  
296 </ImplementDocument>  
297 <ImplementDocument name="ProductInventory">  
298 ...  
299 </ImplementDocument>  
300 ...  
301 </ImplementationProfile>
```

302

303 In accordance with the implementation profile, the application program sends or receives a message that
304 SHOULD have a domain document listed in the implementation profile. The domain properties in the
305 object SHOULD be one of the domain properties defined in the application profile.

306

307 Example 4. A message created on the implementation profile

```
308 <ProductMaster id="001" transaction="A001-001" action="Get" sender="P01"  
309 profile="http://www.oasis-open.org/committees/pps/profile-1.0">  
310 <Condition>  
311 <Property name="pps:name" value="MX-001"/>  
312 <Property name="pps:color" value="white"/>  
313 </Condition>  
314 <Selection type="all"/>  
315 </ProductMaster>
```

316

317 Example 4 shows a message of a Get transaction created by an application program. The properties
318 referred to as "name" and "color" are specified in this message. The properties are defined in the
319 implementation profile as well as the application profile. The prefix "pps" and colon mark are added at the
320 front of the name to notify that the name is defined in the profile.

321 3.3 Level of implementation

322 Domain documents can be sent or received by application programs in any types of transaction including
323 Add, Change, Remove, Get, Show, Notify and Sync. These transactions are prescribed in [PPS02]. Level
324 of implementation represents whether or not the functions prescribed in [PPS02] are fully implemented or
325 partially implemented. The level includes Full, Partial and None.

326 The certain level of Partial implementation is defined in [PPS02] depending on the type of transaction.
327 When the application program informs Partial implementation, it SHOULD have full capability of functions
328 defined in the partial implementation in [PPS02].

329 An application program MAY define a level of implementation for each pair of document and transaction
330 type for each application profile definition.

331 3.4 Profile inquiry

332 All application programs SHOULD send implementation profile as a Show transaction message or Notify
333 transaction message. Application programs SHOULD have capability to response implementation profile
334 as Show message when it receives a Get message of *ImplementProfile* transaction.

335 When responding the Get message of implementation profile in Get transaction, then the program
336 SHOULD send corresponding Show message that has available set of transactions or error information.
337 The name of application profile definition is the same as the name in the Get message.

338 Get-Show transaction of *ImplementProfile* MAY NOT be in the list of profile inquiry by *ImplementProfile*.
339 Transaction message of *ImplementProfile*, such as the response Show message, SHOULD NOT have a
340 *Header* element.

341 Any *Condition* and *Selection* element SHOULD NOT be in *ImplementProfile*. The inquiry of
342 implementation profile can only be added the condition to identify the name of application profile by
343 setting the name on *profile* attribute.

344

345 Example 2 Inquiry of implementation profile for PPS standard profile definition

```
346 <ImplementProfile id="A01" transaction="T1" action="Get" sender="A"  
347 profile="http://www.oasis-open.org/committees/pps/profile-1.0"/>
```

348

349 Example 3 Answer of the inquiry in Example 2

```
350 <ImplementProfile id="B01" transaction="T1" action="Show" sender="B">  
351 <ImplementDocument name="SupplierList">  
352 <ImplementAction action="Get" level="1"/>  
353 <ImplementAction action="Add"/>  
354 <ImplementProperty name="id" display="NO"/>  
355 <ImplementProperty name="name" display="NAME"/>  
356 ...
```

357 </ImplementDocument>
358
359 </ImplementProfile >

360

361 Example 2 and 3 are the request of implementation profile and its response. By the message in Example
362 2, the responder needs to answer its capability on PPS standard application profile.

363

4 XML Elements

364

4.1 AppProfile Element

365

AppProfile element SHOULD represent an application profile. Standard application profile and extended application profile are both represented by this element. This is a top level element in an application profile, and has *Enumeration* element, *AppObject* element, and *AppDocument* element.

366

367

This information SHOULD be specified in the following XML schema. The XML documents generated by the schema SHOULD be consistent with the following arguments.

368

369

370

371

372

373

374

375

376

377

378

379

380

381

382

383

384

385

386

387

388

```
<xsd:element name="AppProfile">
<xsd:complexType>
  <xsd:sequence>
    <xsd:element ref="DataType" minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element ref="UnitType" minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element ref="Enumeration" minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element ref="AppObject" minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element ref="AppDocument" minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
  <xsd:attribute name="name" type="xsd:string" use="required"/>
  <xsd:attribute name="base" type="xsd:string"/>
  <xsd:attribute name="location" type="xsd:string"/>
  <xsd:attribute name="prefix" type="xsd:string"/>
  <xsd:attribute name="namespace" type="xsd:string"/>
  <xsd:attribute name="create" type="xsd:string"/>
  <xsd:attribute name="description" type="xsd:string"/>
</xsd:complexType>
</xsd:element>
```

389

390

- *Data Type* element SHOULD represent any data type that is used as a type of property data.

391

- *UnitType* element SHOULD represent any unit type of property defined in domain object in this profile.

392

- *Enumeration* element SHOULD represent any enumeration type that is used as a special type of

393

properties.

394

- *AppObject* element SHOULD represent any domain objects used in the domain documents defined in this profile.

395

396

- *AppDocument* element SHOULD represent any application documents that the applications may send or receive on this profile.

397

398

- *name* attribute SHOULD represent the name of this application profile. By including an URL texts, the name SHOULD be unique and even in the future. This attribute is REQUIRED.

399

400

- *base* attribute SHOULD represent the base application profile when this profile is an extended application profile.

401

402

- *location* attribute SHOULD represent the location where the profile can be downloaded via Internet.

403

- *prefix* attribute SHOULD represent the prefix text that is added in the name of values that are qualified by this profile.

404

405

- *namespace* attribute SHOULD represent the namespace when this profile is used in a specific namespace.

406

407

- *create* attribute SHOULD represent the date of creation of the profile

408

- *description* attribute SHOULD represent any description related to this profile.

4.2 AppDocument Element

AppDocument element SHOULD represent a domain document that is contained in a message of any transactions. All domain documents that may appear in messages SHOULD be specified in *AppApplication* element that corresponds to the application profile.

This information SHOULD be specified in the following XML schema. The XML documents generated by the schema SHOULD be consistent with the following arguments.

```
<xsd:element name="AppDocument">
  <xsd:complexType>
    <xsd:attribute name="name" type="xsd:string" use="required"/>
    <xsd:attribute name="object" type="xsd:string"/>
    <xsd:attribute name="category" type="xsd:string"/>
    <xsd:attribute name="description" type="xsd:string"/>
  </xsd:complexType>
</xsd:element>
```

- *name* attribute SHOULD represent the name of the domain document. The name SHOULD be unique to identify the type of the document. This attribute is REQUIRED.
- *object* attribute SHOULD represent the name of domain object that the document MAY have in the body as its content. One document SHOULD have one kind of domain object. All object names, which are defined in the *AppDocument* elements, SHOULD be defined in the same application profile definition. This attribute is REQUIRED.
- *category* attribute SHOULD represent any category of the domain document. This information is used for making any group by categorizing various documents. Same group documents have same value of category. This attribute is OPTIONAL.
- *description* attribute SHOULD represent any description of the domain document. Any comments and additional information of the document may be specified there. This attribute is OPTIONAL.

4.3 AppObject Element

AppObject element SHOULD represent a domain object corresponding to any actual object in the target problem domain. All domain objects that are referred from domain documents in the same application profile SHOULD be specified in the *AppObject* element.

This information SHOULD be specified in the following XML schema. The XML documents generated by the schema SHOULD be consistent with the following arguments.

```
<xsd:element name="AppObject">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="AppProperty" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="name" type="xsd:string" use="required"/>
    <xsd:attribute name="primitive" type="xsd:string" use="required"/>
    <xsd:attribute name="description" type="xsd:string"/>
  </xsd:complexType>
</xsd:element>
```

- *AppProfile* element SHOULD represent a property that may be specified in the domain objects of the application profile definition. All possible profiles SHOULD be specified in the domain object represented by *AppObject*.
- *name* attribute SHOULD represent the name of the property. The name SHOULD be unique under the same domain object defined by *AppObject*. This attribute is REQUIRED.

- 459 • *primitive* attribute SHOULD represent a primitive element name selected from the primitive element
460 list defined in [PPS01]. Since every domain object is a subclass of one in the primitive object, all
461 *AppObject* elements SHOULD have a primitive attribute. This attribute is REQUIRED.
- 462 • *description* attribute SHOULD represent any description of the domain object. This attribute is
463 OPTIONAL.

464 4.4 AppProperty Element

465 *AppProperty* element SHOULD represent a domain property of a particular domain object. All properties
466 that may be defined to represent the characteristics of the domain object SHOULD be defined in the
467 *AppObject* corresponding to the domain object.

468 This information SHOULD be specified in the following XML schema. The XML documents generated by
469 the schema SHOULD be consistent with the following arguments.

470

```
471 <xsd:element name="AppProperty">
472   <xsd:complexType>
473     <xsd:attribute name="name" type="xsd:string" use="required"/>
474     <xsd:attribute name="path" type="xsd:string"/>
475     <xsd:attribute name="multiple" type="xsd:string"/>
476     <xsd:attribute name="enumeration" type="xsd:string"/>
477     <xsd:attribute name="dataType" type="xsd:string"/>
478     <xsd:attribute name="unitType" type="xsd:string"/>
479     <xsd:attribute name="use" type="xsd:string"/>
480     <xsd:attribute name="description" type="xsd:string"/>
481   </xsd:complexType>
482 </xsd:element>
```

483

- 484 • *name* attribute SHOULD represent the name of the property. The name SHOULD be unique in the
485 domain object defined by *AppObject* to identify the property. This attribute is REQUIRED.
- 486 • *path* attribute SHOULD represent the location of the attribute data in the primitive XML description
487 defined in [PPS01]. The specification of the path SHOULD conform to [PATH]. If the profile is a
488 standard application profile, this attribute is REQUIRED, and otherwise OPTIONAL.
- 489 • *multiple* attribute SHOULD represent whether the property can have multiple values or not. If the
490 value of this attribute is positive integer or “unbounded”, actual message described by [PPS01]
491 specification can have corresponding number of values for this property. This attribute is OPTIONAL.
- 492 • *enumeration* attribute SHOULD represent the name of enumeration type when the property has a
493 value in the enumeration list. The name of enumeration type SHOULD be specified in *Enumeration*
494 elements in the same application profile definition. This attribute is OPTIONAL.
- 495 • *dataType* attribute SHOULD represent the data type of the property. The value of this attribute
496 SHOULD be “qty”, “char”, “time”, or any data type name that may be specified in the *DataType*
497 element. The data type specified in the attribute SHOULD be the same as the data type of attribute
498 on the body elements identified by the path attribute.
- 499 • *unitType* attribute SHOULD represent the unit of measure for the property. The value of this attribute
500 is any texts that show the unit name. The name may be defined in *UnitType* element. This attribute is
501 OPTIONAL.
- 502 • *use* attribute SHOULD represent that the property is mandatory for any implementation, if the value of
503 this attribute is “required”.
- 504 • *description* attribute SHOULD represent any description of the domain property. This attribute is
505 OPTIONAL.

506 4.5 DataType Element

507 *DataType* element SHOULD represent a data type of domain property. This is information to map the
508 data of each property to local database managed by the application program.

509 This information SHOULD be specified in the following XML schema. The XML documents generated by
510 the schema SHOULD be consistent with the following arguments.

511

```
512 <xsd:element name="DataType">  
513 <xsd:complexType>  
514 <xsd:attribute name="name" type="xsd:string" use="required"/>  
515 <xsd:attribute name="type" type="xsd:string" use="required"/>  
516 <xsd:attribute name="size" type="xsd:int"/>  
517 <xsd:attribute name="description" type="xsd:string"/>  
518 </xsd:complexType>  
519 </xsd:element>
```

520

- 521 • *name* attribute SHOULD represent a name of data type. The name SHOULD be unique in the
522 application profile definition. This attribute is REQUIRED.
- 523 • *type* attribute SHOULD represent a data type that is grouped into quantitative, qualitative and
524 temporal data. The value of this attribute SHOULD be “qty”, “char”, or “time”, respectively. This
525 attribute is REQUIRED.
- 526 • *size* attribute SHOULD represent size of data. If the data is string, then this shows length of the string.
527 This attribute is OPTIONAL.
- 528 • *description* attribute SHOULD represent any description of the data type. This attribute is OPTIONAL.

529

530 4.6 UnitType Element

531 *UnitType* element SHOULD represent a unit definition for domain properties. This information is used by
532 property referring as its unit of measure. More than one property MAY refers a unit definition. Especially,
533 discrete time scales such as day, week and month are defined precisely by this element. *UnitType*
534 element MAY allow application programs to convert the data between two units.

535 This information SHOULD be specified in the following XML schema. The XML documents generated by
536 the schema SHOULD be consistent with the following arguments.

537

```
538 <xsd:element name="UnitType">  
539 <xsd:complexType>  
540 <xsd:attribute name="name" type="xsd:string" use="required"/>  
541 <xsd:attribute name="base" type="xsd:string"/>  
542 <xsd:attribute name="size" type="xsd:int"/>  
543 <xsd:attribute name="time" type="xsd:datetime"/>  
544 <xsd:attribute name="duration" type="xsd:duration"/>  
545 <xsd:attribute name="description" type="xsd:string"/>  
546 </xsd:complexType>  
547 </xsd:element>
```

548

- 549 • *name* attribute SHOULD represent a name of data type. The name SHOULD be unique in the
550 application profile definition. This attribute is REQUIRED.
- 551 • *base* attribute SHOULD represent a unit name that is referred to as base unit. The unit MAY be
552 calculated by the base unit and size. The base name SHOULD be defined in other unit type definition
553 in the same application profile. This attribute is OPTIONAL.
- 554 • *size* attribute SHOULD represent size of data. The size SHOULD show how much times as big as the
555 size of the base unit. This information is used to convert the unit from the base unit, and vice versa.
556 This attribute is OPTIONAL.
- 557 • *time* attribute SHOULD represent a origin time of data. If the unit is on a discrete time scale, this data
558 shows the origin of the time scale. This attribute is OPTIONAL.

- 559 • *duration* attribute SHOULD represent a time unit of data. If the unit is on a discrete time scale, this
560 data shows the unit time duration of the time scale. This attribute is OPTIONAL.
- 561 • *description* attribute SHOULD represent any description of the data type. This attribute is OPTIONAL.
562

563 4.7 Enumeration Element

564 *Enumeration* element SHOULD represent an enumeration type that has several items in a list format. If a
565 property of a domain object has the enumeration type, then the property SHOULD have one of any items
566 in the enumeration list.

567 Enumeration type is independent from any domain object in the application profile definition. Therefore,
568 many different domain objects MAY have different properties that has the same enumeration type.
569 Instead of this, the name of enumeration type may have wide-variety.

570 This information SHOULD be specified in the following XML schema. The XML documents generated by
571 the schema SHOULD be consistent with the following arguments.

572

```
573 <xsd:element name="Enumeration">
574 <xsd:complexType>
575 <xsd:sequence>
576 <xsd:element ref="EnumElement" minOccurs="0" maxOccurs="unbounded"/>
577 </xsd:sequence>
578 <xsd:attribute name="name" type="xsd:string" use="required"/>
579 <xsd:attribute name="description" type="xsd:string"/>
580 </xsd:complexType>
581 </xsd:element>
```

582

- 583 • *EnumElement* element SHOULD represent an item of the list that the enumeration type has as
584 candidates of property value.
- 585 • *name* attribute SHOULD represent a name of this enumeration type. The name SHOULD be unique
586 in the application profile definition. This attribute is REQUIRED.
- 587 • *description* attribute SHOULD represent any description of the enumeration type. This attribute is
588 OPTIONAL.

589 4.8 EnumElement Element

590 *EnumElement* element SHOULD represent an item of enumeration list. A property that is defined with the
591 enumeration type SHOULD select one of the items from the enumeration list.

592 This information SHOULD be specified in the following XML schema. The XML documents generated by
593 the schema SHOULD be consistent with the following arguments.

594

```
595 <xsd:element name="EnumElement">
596 <xsd:complexType>
597 <xsd:attribute name="value" type="xsd:string" use="required"/>
598 <xsd:attribute name="primary" type="xsd:boolean"/>
599 <xsd:attribute name="alias" type="xsd:int"/>
600 <xsd:attribute name="description" type="xsd:string"/>
601 </xsd:complexType>
602 </xsd:element>
```

603

- 604 • *value* attribute SHOULD represent value texts that can be selected from the enumeration list. The
605 value SHOULD be unique in the value list of the enumeration type. This attribute is REQUIRED.

- 606 • *primary* attribute SHOULD represent the primary item in the enumeration list. Only the primary
607 attribute SHOULD have “true” value for this attribute. No more than one item in the item list SHOULD
608 have “true” value. This attribute is OPTIONAL, and the default value is “false”.
- 609 • *alias* attribute SHOULD represent a numerical value instead of the text value specified in the value
610 attribute. The value SHOULD be unique integer among the items in the enumeration type.
- 611 • *description* attribute SHOULD represent any description of the enumeration type. This attribute is
612 OPTIONAL.

613 4.9 ImplementProfile Element

614 *ImplementProfile* element SHOULD represent an implementation profile for an application program.
615 *ImplementProfile* SHOULD be defined for each application profile that the application program supports.
616 This information MAY be created as a transaction message, and MAY be sent or received by the party to
617 inform the implementation profile. Therefore, the structure of this element is almost the same as the
618 structure of transaction element defined in [PPS02].

619 This information SHOULD be specified in the following XML schema. The XML documents generated by
620 the schema SHOULD be consistent with the following arguments.

621

```
622 <xsd:element name="ImplementProfile">
623   <xsd:complexType>
624     <xsd:sequence>
625       <xsd:element ref="Error" minOccurs="0" maxOccurs="unbounded"/>
626       <xsd:element ref="App" minOccurs="0"/>
627       <xsd:element ref="Spec" minOccurs="0" maxOccurs="unbounded"/>
628       <xsd:element ref="ImplementDocument" minOccurs="0" maxOccurs="unbounded"/>
629     </xsd:sequence>
630     <xsd:attribute name="id" type="xsd:string"/>
631     <xsd:attribute name="action" type="xsd:string"/>
632     <xsd:attribute name="transaction" type="xsd:string"/>
633     <xsd:attribute name="profile" type="xsd:string" use="required"/>
634     <xsd:attribute name="sender" type="xsd:string" use="required"/>
635     <xsd:attribute name="create" type="xsd:dateTime"/>
636     <xsd:attribute name="description" type="xsd:string"/>
637   </xsd:complexType>
638 </xsd:element>
```

639

- 640 • *Error* element SHOULD represent error information, when any errors occur during the transaction of
641 message exchange of this implementation profile. The specification of this element is defined in
642 [PPS02].
- 643 • *App* element SHOULD represent any information for the application program concerning the
644 transaction of profile exchange. The use of this element SHOULD be consistent with all cases of
645 transactions while the other messages are exchanged. The specification of this element is defined in
646 [PPS02].
- 647 • *Spec* element SHOULD represent an additional specification about the transaction of exchanging this
648 implementation profile. The use of this element SHOULD be consistent with all cases of transactions
649 while the other messages are exchanged. The specification of this element is defined in [PPS02].
- 650 • *ImplementDocument* element SHOULD represent a domain document that the application program
651 may send or receive. All available documents in the application profile SHOULD be listed using this
652 element.
- 653 • *id* attribute SHOULD represent identifier of the message. The id SHOULD be unique in all messages
654 the sender has sent. Then the receiver of this message can identify the message by combination of
655 the id and the sender name. When the element is created as a message for exchange, this attribute
656 is REQUIRED. Otherwise, such as for a XML document file, this attribute is OPTIONAL.
- 657 • *action* attribute SHOULD represent a name of action during transaction models defined in [PPS02].
658 The value of this attribute SHOULD be “Add”, “Change”, “Remove”, “Notify”, “Sync”, “Get” or “Show”.

- 659 When the element is created as a message for exchange, this attribute is REQUIRED. Otherwise,
660 such as for a XML document file, this attribute is OPTIONAL.
- 661 • *transaction* attribute SHOULD represent ID of transaction that several messages in a certain
662 transaction can be grouped. Response message SHOULD have the same transaction ID as the
663 request message. When the element is created as a message for exchange, this attribute is
664 REQUIRED. Otherwise, such as for a XML document file, this attribute is OPTIONAL.
 - 665 • *profile* attribute SHOULD represent the name of application profile that the implementation profile is
666 depending. The application profile MAY be either a standard application profile or an extended
667 application profile. This attribute is REQUIRED.
 - 668 • *sender* attribute SHOULD represent a name of application program who has functions of message
669 transactions. Definition of this implementation profile SHOULD certify the functionality of the program.
670 The name of sender SHOULD be unique in the table of party who may participate the message
671 exchange. This attribute is REQUIRED.
 - 672 • *create* attribute SHOULD represent the date of creation of the implementation profile. This attribute is
673 OPTIONAL.
 - 674 • *description* attribute SHOULD represent any description of the enumeration type. This attribute is
675 OPTIONAL.

676 4.10 ImplementDocument Element

677 *ImplementDocument* element SHOULD represent a domain document selected from the application
678 profile. All available domain documents SHOULD be listed by this element. Available domain documents
679 MAY be defined for each application profile that the program can support.

680 This information SHOULD be specified in the following XML schema. The XML documents generated by
681 the schema SHOULD be consistent with the following arguments.

682

```
683 <xsd:element name="ImplementDocument">
684 <xsd:complexType>
685 <xsd:sequence>
686 <xsd:element ref="ImplementAction" minOccurs="0" maxOccurs="unbounded"/>
687 <xsd:element ref="ImplementProperty" minOccurs="0" maxOccurs="unbounded"/>
688 <xsd:element ref="ImplementEvent" minOccurs="0" maxOccurs="unbounded"/>
689 </xsd:sequence>
690 <xsd:attribute name="name" type="xsd:string" use="required"/>
691 <xsd:attribute name="profile" type="xsd:string"/>
692 </xsd:complexType>
693 </xsd:element>
```

694

- 695 • *ImplementAction* element SHOULD represent an action that the program can perform for the domain
696 document. This element MAY represent a role of the program in the transaction.
- 697 • *ImplementProperty* element SHOULD represent a property that the program can deal with in the
698 domain object. All properties defined in this element SHOULD be defined as a property of a domain
699 object in the corresponding application profile.
- 700 • *ImplementEvent* element SHOULD represent an event that the program can monitor a property in
701 order to notify the change of the data to subscribers. This information MAY be defined by each
702 application programs.
- 703 • *name* attribute SHOULD represent the name of the domain document. The name SHOULD be
704 defined in the list of domain document in the corresponding application profile. This attribute is
705 REQUIRED.
- 706 • *profile* attribute SHOULD represent the name of application profile that this implementation profile is
707 referring to select the available part in the definition. This attribute is required if the profile is different
708 from the name defined in *ImplementProfile* element. Therefore, this attribute is OPTIONAL.

709 4.11 ImplementAction Element

710 *ImplementAction* element SHOULD represent an action that the program can perform for the domain
711 document. The actions include the transaction model referred to as “Add”, “Change”, “Remove”, “Notify”,
712 “Sync”, “Get” or “Show”. This element MAY represent a role of the program in the transaction such as
713 sender or receiver.

714 This information SHOULD be specified in the following XML schema. The XML documents generated by
715 the schema SHOULD be consistent with the following arguments.

716

```
717 <xsd:element name="ImplementAction">  
718 <xsd:complexType>  
719 <xsd:attribute name="action" type="xsd:string" use="required"/>  
720 <xsd:attribute name="level" type="xsd:int"/>  
721 <xsd:attribute name="role" type="xsd:string"/>  
722 </xsd:complexType>  
723 </xsd:element>
```

724

- 725 • *action* attribute SHOULD represent the action performed by the application program. The value of this
726 attribute SHOULD be one of “Add”, “Change”, “Remove”, “Notify”, “Sync”, “Get” and “Show”, or any
727 combination of those. If more than one action are specified, all available names of action SHOULD be
728 separated by “|” character. This attribute is REQUIRED.
- 729 • *level* attribute SHOULD represent an implementation level defined in [PPS02] for each document
730 processed by the application program. Level 0 shows no implementation, while level 1 and 2 are
731 partially and fully implemented, respectively. Default value is the highest number that shows the fully
732 implemented. This attribute is OPTIONAL.
- 733 • *role* attribute SHOULD represent a role in the transaction. Every transaction has its available roles
734 that can be selected as a value of this attribute. Default value is “receiver” or “responder”. This
735 attribute is OPTIONAL.

736 4.12 ImplementProperty Element

737 *ImplementProperty* element SHOULD represent a domain property that can be processed in the
738 application program. Some properties SHOULD be defined in the corresponding domain object in the
739 application profile definition. The properties that are not defined in the application profile SHOULD be
740 specified in this element as a user extended property. Properties extended by application programs
741 SHOULD have additional definitions similar to the definitions by *AppProperty* element.

742 This information SHOULD be specified in the following XML schema. The XML documents generated by
743 the schema SHOULD be consistent with the following arguments.

744

```
745 <xsd:element name="ImplementProperty">  
746 <xsd:complexType>  
747 <xsd:attribute name="name" type="xsd:string" use="required"/>  
748 <xsd:attribute name="title" type="xsd:string"/>  
749 <xsd:attribute name="extend" type="xsd:string"/>  
750 <xsd:attribute name="path" type="xsd:string"/>  
751 <xsd:attribute name="dataType" type="xsd:string"/>  
752 <xsd:attribute name="unitType" type="xsd:string"/>  
753 <xsd:attribute name="enumeration" type="xsd:string"/>  
754 <xsd:attribute name="description" type="xsd:string"/>  
755 </xsd:complexType>  
756 </xsd:element>
```

757

- 758 • *name* attribute SHOULD represent the name of the property. The name SHOULD be defined in the
759 corresponding application profile. This attribute is REQUIRED.

- 760 • *title* attribute SHOULD represent the header title of the property. This value MAY be a short
761 description to show the property relating to the actual world. This attribute is OPTIONAL.
- 762 • *extend* attribute SHOULD represent qualifier string that is specified as prefix of the property name, if
763 this property is extended by the local program. For example, if the value is "user", then the
764 description of this property will have "user:" prefix in the actual messages. This attribute is OPTIONAL.
- 765 • *path* attribute SHOULD represent the location of the attribute data in the primitive XML description
766 defined in [PPS01]. The specification of the path SHOULD conform to the syntax of [PATH]. If the
767 attribute value of *extend* is true, this attribute is REQUIRED, and otherwise OPTIONAL.
- 768 • *dataType* attribute SHOULD represent the data type of the property. This attribute is used to map the
769 data to the local database that is managed by the application program. This attribute is OPTIONAL.
- 770 • *unitType* attribute SHOULD represent a unit of the property. This attribute is used to clarify the data
771 unit shown in *unit* attribute in transaction messages. This attribute is OPTIONAL.
- 772 • *enumeration* attribute SHOULD represent the name of enumeration type when the property is
773 extended property by the local program, and has a value in the enumeration list. The name of
774 enumeration type SHOULD be specified in *Enumeration* elements in the application profile definition.
775 This attribute is OPTIONAL.
- 776 • *description* attribute SHOULD represent any description of the property. This attribute is OPTIONAL.
777

778 4.13 ImplementEvent Element

779 *ImplementEvent* element SHOULD represent any event definitions that the application program monitors
780 on a particular property and detects occurrence on it. When the event occurs, the application program
781 SHOULD publish a notification of the event to all the parties who are on the list of subscription. This
782 information is defined by each application program, then any users of information MAY request of
783 publication as a subscriber.

784 *ImplementEvent* element SHOULD allow an application program to define the unit size of data differences,
785 maximum and minimum data value, duration of one monitoring cycle and expire date of notifications to
786 determine the event occurrence.

787 This information SHOULD be specified in the following XML schema. The XML documents generated by
788 the schema SHOULD be consistent with the following arguments.

789

```
790 <xsd:element name="ImplementEvent">
791 <xsd:complexType>
792 <xsd:sequence>
793 <xsd:element ref="Qty" minOccurs="0" maxOccurs="unbounded"/>
794 <xsd:element ref="Char" minOccurs="0" maxOccurs="unbounded"/>
795 <xsd:element ref="Time" minOccurs="0" maxOccurs="unbounded"/>
796 </xsd:sequence>
797 <xsd:attribute name="name" type="xsd:string" use="required"/>
798 <xsd:attribute name="property" type="xsd:string" use="required"/>
799 <xsd:attribute name="cycle" type="xsd:duration"/>
800 <xsd:attribute name="expire" type="xsd:datetime"/>
801 <xsd:attribute name="description" type="xsd:string"/>
802 </xsd:complexType>
803 </xsd:element>
```

804

- 805 • *Qty* element SHOULD represent the quantitative data concerned as a condition of the event
806 occurrence.
- 807 • *Char* element SHOULD represent the qualitative data concerned as a condition of the event
808 occurrence.
- 809 • *Time* element SHOULD represent the temporal data concerned as a condition of the event
810 occurrence.

- 811 • *name* attribute SHOULD represent the name of the event. The name SHOULD be unique in the
812 domain object defined in the application profile. This attribute is REQUIRED.
- 813 • *property* attribute SHOULD represent the name of the property that is monitored by the application
814 program. The property name SHOULD be defined in the domain object. This attribute is REQUIRED.
- 815 • *cycle* attribute SHOULD represent the duration of monitoring of the property value to detect the event
816 occurrence. The application program SHOULD monitor the value until the expiration date. This
817 attribute is OPTIONAL.
- 818 • *expire* attribute SHOULD represent expire time and date of the event notification. After the time of
819 expiration, the application will stop monitoring the event occurrence. If this attribute is not defined, it
820 SHOULD represent that there is no expiration date. This attribute is OPTIONAL.
- 821 • *description* attribute SHOULD represent any description of the property. This attribute is OPTIONAL.

822 **A. Acknowledgements**

823 The following individuals have participated in the creation of this specification and are gratefully
824 acknowledged:

825 **Participants:**

826 Shinya Matsukawa, Hitachi
827 Tomohiko Maeda, Fujitsu
828 Masahiro Mizutani, Unisys Corporation
829 Akihiro Kawauchi, Individual Member
830 Yuto Banba, PSLX Forum
831 Hideichi Okamune, PSLX Forum

832

833

B. Revision History

834

Revision	Date	Editor	Changes Made

835

836