



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

Public Review Draft 02

24 Oct 2009

Specification URIs:

<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>

Previous Version:

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

Latest Version:

<http://docs.oasis-open.org/pps/v1.0/pr02/pps-profile-specifications-1.0.doc>
<http://docs.oasis-open.org/pps/v1.0/pr02/pps-profile-specifications-1.0.html>
<http://docs.oasis-open.org/pps/v1.0/pr02/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/2009>

Abstract:

OASIS PPS (Production Planning and Scheduling) specifications deal with problems of decision-making in all manufacturing companies who want to have a sophisticated information system for production planning and scheduling. PPS specifications provide XML schema and communication protocols for information exchange among manufacturing application programs in the web-services environment. This specification entitled "Part 3: Profile Specifications" especially focuses on profiles of application programs that may exchange the messages. Application profile and implementation profile are defined. Implementation profile shows capability of application programs in terms of services for message exchange, selecting from all exchange items defined in the application profile. 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.....	10
2.5	Revision rule	11
3	Implementation profiles	12
3.1	General	12
3.2	Structure of implementation profiles	12
3.3	Level of implementation	14
3.4	Profile inquiry	14
4	XML Elements	16
4.1	AppProfile Element	16
4.2	AppDocument Element.....	16
4.3	AppObject Element.....	17
4.4	AppProperty Element.....	18
4.5	Enumeration Element	18
4.6	EnumElement Element	19
4.7	ImplementProfile Element.....	19
4.8	ImplementDocument Element	21
4.9	ImplementAction Element.....	21
4.10	ImplementProperty Element	22
4.11	ImplementEvent Element.....	23
A.	Acknowledgements	25
B.	Revision History.....	26

Figures

Figure 1 Structure of profile specifications.....	8
Figure 2 Application Profile.....	9
Figure 3 Concept of communication availability between implementations	12
Figure 4 Structure of ImplementProfile	13

1 Introduction

This specification prescribes definition of application profile and implementation profile. Implementation profile 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 application specific scenarios.

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/>
- [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/>
- [PROFILE] PSLX Application Profile, Version 1.0 (printed edition is in Japanese), <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-schema-1.0.xsd>

43 **1.5 Terms and definitions**

44 **Application profile**

45 Collections of profile specifications for all application programs that may be involved in the
46 communication group who exchanges PPS messages. This information is defined by platform
47 designer to provide all available domain documents, domain objects and domain properties.

48 **Domain document**

49 Document that is a content of message sent or received between application programs, and is
50 processed by a transaction. Domain document consists of a verb part and a noun part. Verbs
51 such as add, change and remove affect the types of messages, while nouns represented by
52 domain objects show the classes of domain objects. Specific classes of domain documents can
53 be defined by platform designer to share the domain information.

54 **Domain object**

55 Object necessary for representing production planning and scheduling information in
56 manufacturing operations management. Domain objects are contents of a domain document, and
57 represented by primitive elements. Specific classes of domain objects can be defined by platform
58 designer to share the domain information.

59 **Domain property**

60 Any parameters that show a property of a domain object. A domain property is represented by
61 XML attributes of the primitive element, or XML child elements of the primitive elements. A
62 domain object may have multiple domain properties that has same property name. Specific
63 properties of domain objects can be defined by platform designer to share the domain information,
64 and additionally defined by each application designer.

65 **Implementation profile**

66 Specification of capability of an application program in terms of exchanging PPS messages. The
67 profile includes a list of available documents and their properties that may be exchanged in PPS
68 messages among production planning and scheduling applications.

69 **Messaging model**

70 Simple patterns of messaging between sender and receiver, or requester and responder. Four
71 message models: NOTIFY, PUSH, PULL, SYNC are defined from an application independent
72 perspective.

73 **Primitive element**

74 XML element that represents a primitive object in the production planning and scheduling domain.
75 Nine primitive elements are defined in [PPS01]. Every domain objects are represented by the
76 primitive elements.

77 **Transaction element**

78 XML element that represents a transaction to process message documents which is sent or
79 received between application programs. Transaction element can control a transaction process of
80 application program database by commitment and rollback. Transaction element may request
81 confirmation from receiver if the message has been received properly.

82

2 Application profile Definitions

2.1 General

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

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

Several application profile definitions can exist independently for the same problem domain. Two application programs cannot communicate each other if they don't refer a common application profile. In order to avoid such a situation, this specification provides an extension mechanism in which a standard profile definition can be extended to an extended profile definition for particular group in local domain.

Figure 1 shows the structure of application profiles. Application profile is either a standard profile definition or an extended profile definition. Figure also shows that an implementation profile refers an application profile without regarding distinction of standard profile definition and extended profile definition.

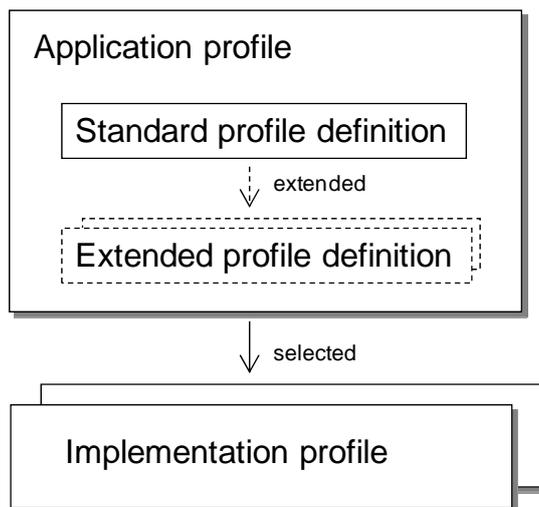


Figure 1 Structure of profile specifications

As an example of standard profile definition, PPS TC supports the PSLX profile [PROFILE] for this planning and scheduling domain. However, this specification only shows general rules and structures of a standard profile definition.

2.2 Structure of profile definitions

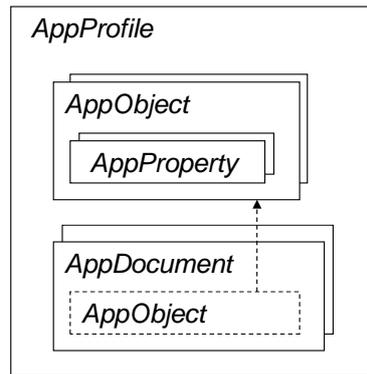
Application profile SHOULD have a list of domain documents and a list of domain objects. In addition, application profile MAY have a list of enumerations, which shows available value set of a domain property of a domain object.

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

113 All candidates of domain documents, which may be used by any application program who sends or
114 receives a message in the target domain, SHOULD be specified using *AppDocument* element under the
115 *AppProfile* element.

116 All domain objects, which are used in any domain document defined in *AppDocument* elements,
117 SHOULD be specified in *AppObject* element under the *AppProfile* element. An *AppObject* has a list of
118 properties that represent the characteristics of the object. Each property SHOULD be described in
119 *AppProperty* under the *AppObject*.

120



121

122

Figure 2 Application Profile

123

124 The structure of application profile is illustrated in Figure 2. Domain document represented by
125 *AppDocument* has domain objects represented by *AppObject*. The domain objects that is listed in the
126 same document SHOULD be the same class objects defined in one *AppObject* in the application profile.
127 The application profile defines domain objects independent from domain documents, because the domain
128 objects may be referred from several different kinds of domain documents.

129

130 **Example:** Application profile definition

```
131 <AppProfile name="pps-profile" prefix="pps" namespace="http://www.oasis-open.org/committees/pps/profile-1.0">  
132 <AppObject name="Product" primitive="Item">  
133 <AppProperty name="id" path="@id"/>  
134 <AppProperty name="name" path="@name"/>  
135 ...  
136 <AppProperty name="Size" path="Spec[@type="size"]/@value"/>  
137 <AppProperty name="Color" path="Spec[@type="color"]/@value"/>  
138 ...  
139 </AppObject>  
140 ...  
141 <AppDocument name="ProductRecord" object="Product"/>  
142 <AppDocument name="ProductInventory" object="Product"/>  
143 <AppDocument name="BillOfMaterials" object="Product"/>  
144 <AppDocument name="BillOfResources" object="Product"/>  
145 ...  
146 </AppProfile>
```

147

148

149 2.3 Standard profile definitions

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

- 152 • Standard profile definition SHOULD have a name to identify the definition among all application
153 programs in world-wide. Unique identifier such as URI is required.

- 154 • The name of standard profile definition contains information of revision, and the revision of the
155 definition SHOULD follow the rule defined in Section 2.5.
- 156 • Standard profile definition SHOULD NOT have a base definition as a reference of other standard
157 profile definitions.
- 158 • Standard profile definition SHOULD be published among application programs and accessible by all
159 the application programs in the problem domain via Internet by announcing the URL the application
160 can download the document.
- 161 • Standard profile definition SHOULD have the domain object in Table 1 or sub-class of Table 1
162 domain objects. The domain objects SHOULD be represented by the primitive elements [PPS01]
163 determined by the table.
- 164 • Every domain object in a standard profile definition SHOULD have a domain property that shows
165 identifier of the object. The domain property SHOULD be represented by id attribute of the primitive
166 XML element in Table 1.

167
168

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

169

170 2.4 Extended profile definitions

171 Standard profile definition MAY be extended by an extended profile definition. Extended profile definition
172 MAY also be extended recursively. This is also represented by *AppProfile* element. Extended profile
173 definitions SHOULD have a reference of a standard profile definition, which is the base of extension.

174 Extended profile definition MAY add domain documents, domain objects and domain properties which
175 have not been defined in the standard profile definition. Additional information of domain documents,
176 domain objects and domain properties SHOULD be defined in the same way as the definition in standard
177 profile definitions.

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

181 Extended profile definitions SHOULD NOT remove the domain documents, domain objects and domain
182 properties addressed in the standard profile.

183 Enumerations MAY be added or modified to the standard profile definition. When extended profile
184 describes enumeration name which is in the standard profile, the candidates of the enumeration are
185 replaced to those in the standard. Extended profile definitions SHOULD NOT remove any enumeration in
186 the application profile.

187

188 **Example:** Extended application profile

```
189 <AppProfile prefix="ex1" name="pps-profile-1.1" namespace="http://www.pslx.org/profile-1" base="pps-profile-1.0">
190 <Enumeration name="groupType">
191 <EnumElement name="high" description="description of a"/>
192 <EnumElement name="low" description="description of b"/>
193 </Enumeration>
194 <AppObject name="Consumer">
195 <AppProperty name="group" path="Spec[type='pslx:group']/@value" enumeration="groupType"/>
196 </AppObject>
197 </AppProfile>
```

198

199 Example shows an application profile extended from the standard profile. The new profile has additional
200 enumeration named “groupType”, and then a new Consumer object is defined with a new property which
201 has a name “group” and the additional enumeration type.

202 2.5 Revision rule

203 After an application profile definition has been created, many application programs are developed
204 according to the profile definition. In accordance with the industrial experiences, the old definition may be
205 required to modify for domain specific reasons in the application domain.

206 Any application profile SHOULD NOT be changed without keeping the following rules after when the
207 profile definition has been published. Otherwise, the new profile SHOULD have a new name that doesn't
208 have any relation with the previous one.

209 There are two revision levels. One is a revision that the system developers have to deal with the new
210 specification and change if necessary. The other is editorial revision where the any program doesn't need
211 to care in terms of interoperability. To inform the former cases, the name of profile SHOULD be changed
212 by adding the revision numbers. For the latter cases, instead of changing the name of profile, the actual
213 file name of the profile, specified at the *location* attribute in the *AppProfile* element SHOULD be changed.

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

217 When the major version increases, it:

- 218 • SHOULD NOT change the name of the profile excepting the portion representing the revision status.
- 219 • SHOULD NOT change the prefix and namespace in the attribute of *AppProfile* element.
- 220 • SHOULD NOT change the domain object in *AppDocument* element.

221 When the minor version increases, it:

- 222 • SHOULD follow the rule of major version increasing,
- 223 • SHOULD NOT change the domain properties in the domain objects.
- 224 • SHOULD NOT change the enumeration definition in the *AppProfile* element.

225

226

3 Implementation profiles

227

3.1 General

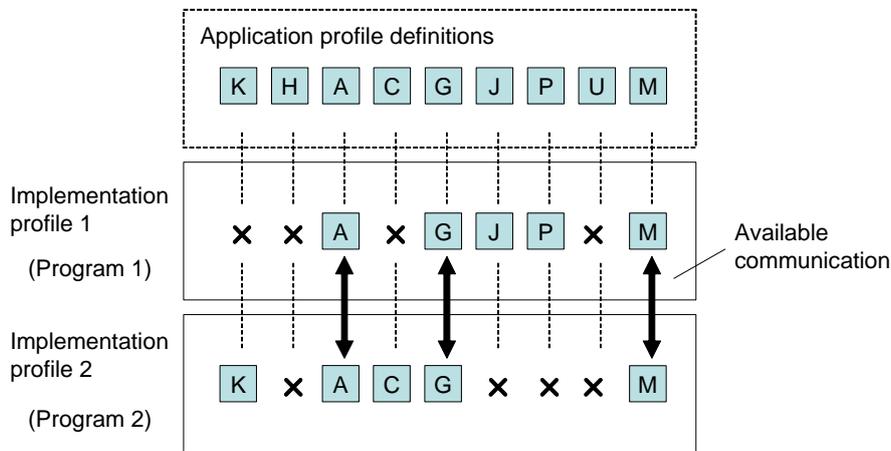
228

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

232

When an application program tries to send a message to another application program, system integrator may need to confirm whether or not the receiving application program has capability to response the message. Then an implementation profile of an application program shows such capability to send or receive information.

236



237

Figure 3 Concept of communication availability between implementations

238

239

240

Figure 3 explains a concept of communication availability between two application programs. Each application program that refers a same application profile has an implementation profile that has a list of items available to communicate, by selecting from the candidates defined in the application profile. Two application programs can exchange a message properly if the both implementations have the corresponding capability.

245

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

248

To confirm the capability of any application program, section 3.4 provides the method of how to get the information by receiving an implementation profile from the program.

249

250

3.2 Structure of implementation profiles

251

Implementation profiles defined for application programs SHOULD be described by *ImplementProfile* element in XML format. The information includes domain documents, domain objects and domain properties available to process by the application program. For each domain document, implementation level, which shows the application program have all functions or not in terms of transactions defined in [PPS02], can be defined.

256

Every implementation profile has a reference to a certain application profile. However, it doesn't show whether the application profile is a standard or extended. From the perspective of application programs, distinction between standard profile definition and extended profile definition has no sense.

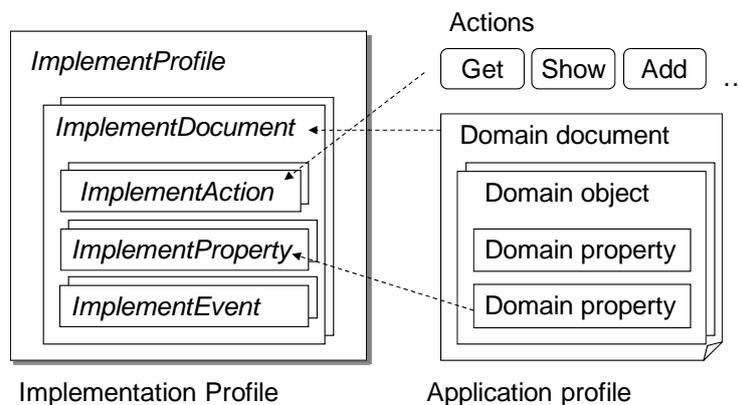
258

259 *ImplementProfile* element MAY be described under *Transaction* element defined in [PPS02]. Therefore,
 260 this can be send or receive through a PPS transaction process. Using Get and Show transactions, two
 261 application programs can exchange the implementation profile.

262 An *ImplementationProfile* element has *ImplementDocument* elements each of which represents
 263 availability for any domain document. An *ImplementDocument* element has *ImplementAction*,
 264 *ImplementProperty* and *ImplementEvent*.

265 *ImplementAction* element represents information of implemented type of transaction such as Get, Show,
 266 Add, and so forth. *ImplementProperty* element represents implemented properties of the domain object.
 267 *ImplementEvent* represents any event definitions that the application program monitors properties and
 268 publish notifications of event defined on the property. Figure 4 shows the structure of *ImplementProfile*,
 269 *ImplementDocument*, *ImplementAction*, and *ImplementProperty* elements.

270



271

272

Figure 4 Structure of *ImplementProfile*

273

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

276 Domain documents in implementation profile SHOULD have a domain property if the property is defined
 277 in the application profile as a primary key of the object or as a property that is always required.

278 The following example shows an implementation profile of an application program that communicates
 279 with other program under an application profile. Then the implementation profile of the example is the
 280 selection of the application profile representing domain documents, transaction types and domain
 281 properties.

282

283 **Example:** Implementation profile of a program for an application profile

```

284 <ImplementProfile id="AP001" action="Notify">
285   <ImplementDocument name="Product">
286     <ImplementAction action="Get" level="1"/>
287     <ImplementAction action="Show" level="1"/>
288     <ImplementAction action="Add" level="2"/>
289     <ImplementProperty name="id" title="Company ID"/>
290     <ImplementProperty name="name" title="Company name"/>
291   </ImplementDocument>
292   <ImplementDocument name="ProductInventory">
293     ...
294   </ImplementDocument>
295   ....
296 </ImplementProfile>
  
```

297

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

301

302 **Example:** A message created on the implementation profile

```
303 <Document name="Product" id="001" action="Get"  
304 namespace="http://www.oasis-open.org/committees/pps/profile-1.0">  
305 <Condition>  
306 <Property name="pps:name" value="MX-001"/>  
307 <Property name="pps:color" value="white"/>  
308 </Condition>  
309 <Selection type="All"/>  
310 </Document>
```

311

312 Above example shows a message of a Get document created by an application program. The properties
313 referred to as "name" and "color" are specified in this message. The properties are defined in the
314 implementation profile as well as the application profile. The prefix "pps" and colon mark are added at the
315 front of the name to notify that the name is defined in the profile.

316 3.3 Level of implementation

317 Domain documents can be sent or received by application programs in any types of action including Add,
318 Change, Remove, Get, Show, Notify and Sync. These actions are prescribed in [PPS02]. Level of
319 implementation represents whether or not the functions prescribed in [PPS02] are fully implemented or
320 partially implemented

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

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

326 3.4 Profile inquiry

327 All application programs SHOULD send implementation profile as a Show transaction message or Notify
328 transaction message. Application programs SHOULD have capability to response implementation profile
329 as Show message when it receives an *ImplementProfile* inquiry in a form of Get message.

330 When responding to the Get message of implementation profile in PULL model, the program SHOULD
331 send corresponding Show message that is made of *ImplementProfile* element or *Error* element.

332 This capability of implement profile inquiry SHOULD NOT be in the available list of *ImplementProfile* by
333 itself. Since any *Condition* and *Selection* element cannot be described in *ImplementProfile*, the inquiry of
334 implementation profile can only request all the information of implement profiles.

335

336 **Example:** Inquiry of implementation profile for PPS standard profile definition

```
337 <Message id="A01" sender="A">  
338 <ImplementProfile action="Get" />  
339 </Message>
```

340

341 **Example:** Answer of the inquiry in above example

```
342 <Message id="B01" sender="B">  
343 <ImplementProfile id="B01" action="Show" >  
344 <ImplementDocument name="Supplier">  
345 <ImplementAction action="Get" level="1"/>  
346 <ImplementAction action="Add"/>  
347 <ImplementProperty name="id" display="NO"/>
```

```
348 <ImplementProperty name="name" display="NAME"/>
349 ...
350 </ImplementDocument>
351
352 </ImplementProfile >
353 </Message>
```

354

355 Examples are the request of implementation profile and its response. By the message in the first
356 example , the responder needs to answer its capability on the application profiles.

357

4 XML Elements

358

4.1 AppProfile Element

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

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

364

365

366

367

368

369

370

371

372

373

374

375

376

377

378

379

380

```
<xsd:element name="AppProfile">
  <xsd:complexType>
    <xsd:sequence>
      <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>
```

381

382 • *Enumeration* element SHOULD represent any enumeration type that is used as a special type of
383 properties.

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

386 • *AppDocument* element SHOULD represent any domain documents that the applications may send or
387 receive on this profile.

388

389 • *name* attribute SHOULD represent the name of this application profile. The name SHOULD be unique
390 in the namespace. This attribute is REQUIRED.

391 • *base* attribute SHOULD represent the base application profile when this profile is an extended
392 application profile.

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

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

396 • *namespace* attribute SHOULD represent the namespace when this profile is used in a specific
397 namespace.

398 • *create* attribute SHOULD represent the date of creation of the profile

399 • *description* attribute SHOULD represent any description related to this profile.

4.2 AppDocument Element

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

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

406

```
407 <xsd:element name="AppDocument">  
408   <xsd:complexType>  
409     <xsd:attribute name="name" type="xsd:string" use="required"/>  
410     <xsd:attribute name="object" type="xsd:string"/>  
411     <xsd:attribute name="category" type="xsd:string"/>  
412     <xsd:attribute name="description" type="xsd:string"/>  
413   </xsd:complexType>  
414 </xsd:element>
```

415

- 416 • *name* attribute SHOULD represent the name of the domain document. The name SHOULD be unique
417 in the namespace to identify the type of the document. This attribute is REQUIRED.
- 418 • *object* attribute SHOULD represent the name of domain object that the document MAY have in the
419 body as its content. One document SHOULD have one kind of domain object. All objects referred by
420 this attribute SHOULD be defined in the same application profile or base application profile. This
421 attribute is REQUIRED.
- 422 • *category* attribute SHOULD represent any category of the domain document. This information is used
423 for making any group by categorizing various documents. Same group documents have same value
424 for this attribute. This attribute is OPTIONAL.
- 425 • *description* attribute SHOULD represent any description of the domain document. Any comments and
426 additional information of the document may be specified there. This attribute is OPTIONAL.

427 4.3 AppObject Element

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

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

433

```
434 <xsd:element name="AppObject">  
435   <xsd:complexType>  
436     <xsd:sequence>  
437       <xsd:element ref="AppProperty" minOccurs="0" maxOccurs="unbounded"/>  
438     </xsd:sequence>  
439     <xsd:attribute name="name" type="xsd:string" use="required"/>  
440     <xsd:attribute name="primitive" type="xsd:string" use="required"/>  
441     <xsd:attribute name="description" type="xsd:string"/>  
442   </xsd:complexType>  
443 </xsd:element>
```

444

- 445 • *AppProperty* element SHOULD represent a property that may be described in the domain objects of
446 the application profile definition. All possible properties SHOULD be described in the domain object
447 represented by *AppObject*.
- 448
- 449 • *name* attribute SHOULD represent the name of the object. The name SHOULD be unique under the
450 application profile definition in the selected namespace. This attribute is REQUIRED.
- 451 • *primitive* attribute SHOULD represent a primitive element name selected from the primitive element
452 list defined in [PPS01]. Since every domain object is a subclass of one in the primitive objects, all
453 *AppObject* elements SHOULD have a primitive attribute. This attribute is REQUIRED.

- 454 • *description* attribute SHOULD represent any description of the domain object. This attribute is
455 OPTIONAL.

456 4.4 AppProperty Element

457 *AppProperty* element SHOULD represent a domain property of a domain object. All properties that may
458 be defined to represent the characteristics of the domain object SHOULD be described under the
459 *AppObject* corresponding to the domain object.

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

462

```
463 <xsd:element name="AppProperty">  
464 <xsd:complexType>  
465 <xsd:attribute name="name" type="xsd:string"/>  
466 <xsd:attribute name="path" type="xsd:string"/>  
467 <xsd:attribute name="multiple" type="xsd:string"/>  
468 <xsd:attribute name="key" type="xsd:string"/>  
469 <xsd:attribute name="enumeration" type="xsd:string"/>  
470 <xsd:attribute name="dataType" type="xsd:string"/>  
471 <xsd:attribute name="use" type="xsd:string"/>  
472 <xsd:attribute name="description" type="xsd:string"/>  
473 </xsd:complexType>  
474 </xsd:element>
```

475

- 476 • *name* attribute SHOULD represent the name of the property. The name SHOULD be unique in the
477 domain object defined by *AppObject* to identify the property. This attribute is REQUIRED.
- 478 • *path* attribute SHOULD represent the location of the attribute data in the primitive XML description
479 defined in [PPS01]. The specification of the path SHOULD conform to [PATH]. If the profile is a
480 standard application profile, this attribute is REQUIRED, and otherwise OPTIONAL.
- 481 • *multiple* attribute SHOULD represent whether the property can have multiple values or not. If the
482 value of this attribute is positive integer or “Unbounded”, actual message described by [PPS01]
483 specification can have corresponding number of values for this property. This attribute is OPTIONAL.
- 484 • *key* attribute SHOULD represent whether or not this property is primary key of the domain object to
485 identify the target object from the instances in the database. If the value is “True”, then this property is
486 primary key. Primary key SHOULD NOT defined more than one in the same domain object.
- 487 • *enumeration* attribute SHOULD represent the name of enumeration type when the property has a
488 value in the enumeration list. The name of enumeration type SHOULD be specified in *Enumeration*
489 elements in the same application profile definition. This attribute is OPTIONAL.
- 490 • *dataType* attribute SHOULD represent the data type of the property. The value of this attribute
491 SHOULD be “Qty”, “Char” or “Time”. The data type described in the attribute SHOULD be the same
492 as the data type of attribute on the body elements identified by the path attribute.
- 493 • *use* attribute SHOULD represent that the property is mandatory for any implementation, if the value of
494 this attribute is “Required”.
- 495 • *description* attribute SHOULD represent any description of the domain property. This attribute is
496 OPTIONAL.

497 4.5 Enumeration Element

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

501 Enumeration type is independent from any domain object in the application profile definition. Therefore,
502 several different domain objects MAY have different properties that has the same enumeration type.

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

505

```
506 <xsd:element name="Enumeration">  
507   <xsd:complexType>  
508     <xsd:sequence>  
509       <xsd:element ref="EnumElement" maxOccurs="unbounded"/>  
510     </xsd:sequence>  
511     <xsd:attribute name="name" type="xsd:string" use="required"/>  
512     <xsd:attribute name="description" type="xsd:string"/>  
513   </xsd:complexType>  
514 </xsd:element>
```

515

516 • *EnumElement* element SHOULD represent an item of the list that the enumeration type has as
517 candidates of property value.

518

519 • *name* attribute SHOULD represent a name of this enumeration type. The name SHOULD be unique
520 in the application profile definition. This attribute is REQUIRED.

521 • *description* attribute SHOULD represent any description of the enumeration type. This attribute is
522 OPTIONAL.

523 4.6 EnumElement Element

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

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

528

```
529 <xsd:element name="EnumElement">  
530   <xsd:complexType>  
531     <xsd:attribute name="value" type="xsd:string" use="required"/>  
532     <xsd:attribute name="primary" type="xsd:boolean"/>  
533     <xsd:attribute name="alias" type="xsd:int"/>  
534     <xsd:attribute name="description" type="xsd:string"/>  
535   </xsd:complexType>  
536 </xsd:element>
```

537

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

540 • *primary* attribute SHOULD represent the primary item in the enumeration list. Only the primary
541 attribute SHOULD have "True" value for this attribute. No more than one item in the item list SHOULD
542 have "true" value. This attribute is OPTIONAL, and the default value is "False".

543 • *alias* attribute SHOULD represent a numerical value instead of the text value specified in the *value*
544 attribute. The value SHOULD be unique integer among the items in the enumeration type.

545 • *description* attribute SHOULD represent any description of the enumeration element. This attribute is
546 OPTIONAL.

547 4.7 ImplementProfile Element

548 *ImplementProfile* element SHOULD represent an implementation profile for an application program.
549 *ImplementProfile* SHOULD be defined for each application program what the application program
550 supports. This information MAY be sent by the application program and received by the party who wants
551 to know the capability of the application program. Therefore, in order to make transactions, some
552 attributes and sub-elements are the same as the attributes of Document element defined in [PPS02].

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

555

```
556 <xsd:element name="ImplementProfile">  
557   <xsd:complexType>  
558     <xsd:sequence>  
559       <xsd:element ref="Error" minOccurs="0" maxOccurs="unbounded"/>  
560       <xsd:element ref="App" minOccurs="0"/>  
561       <xsd:element ref="ImplementDocument" minOccurs="0" maxOccurs="unbounded"/>  
562     </xsd:sequence>  
563     <xsd:attribute name="id" type="xsd:string"/>  
564     <xsd:attribute name="name" type="xsd:string"/>  
565     <xsd:attribute name="action" type="xsd:string"/>  
566     <xsd:attribute name="profile" type="xsd:string"/>  
567     <xsd:attribute name="location" type="xsd:string"/>  
568     <xsd:attribute name="namespace" type="xsd:string"/>  
569     <xsd:attribute name="create" type="xsd:dateTime"/>  
570     <xsd:attribute name="description" type="xsd:string"/>  
571   </xsd:complexType>  
572 </xsd:element>
```

573

- 574 • *Error* element SHOULD represent error information, when any errors occur during the transaction of
575 message exchange of this implementation profile. The specification of this element is defined in
576 [PPS02].
- 577 • *App* element SHOULD represent any information for the application program concerning the
578 transaction of profile exchange. The use of this element SHOULD be consistent with all cases of
579 transactions while the other messages are exchanged. The specification of this element is defined in
580 [PPS02].
- 581 • *ImplementDocument* element SHOULD represent a domain document that the application program
582 may send or receive. All available documents in the application profile SHOULD be listed using this
583 element.
- 584
- 585 • *id* attribute SHOULD represent identifier of the application program. The *id* SHOULD be unique in all
586 application programs that can be accessed in the network. In order to guarantee the uniqueness,
587 system integrator must assigns the unique number and manages it in the network configuration. This
588 *id* is the same as the sender name when the application will send a message. This attribute is
589 REQUIRED.
- 590 • *name* attribute SHOULD represent a name that the application program shows its name for an
591 explanation by natural texts. This attribute is OPTIONAL.
- 592 • *action* attribute SHOULD represent a name of action during transaction models defined in [PPS02].
593 The value of this attribute SHOULD be "Notify", "Get" or "Show". When the element is created as a
594 message for exchange, this attribute is REQUIRED. Otherwise, such as for a XML document file, this
595 attribute is OPTIONAL.
- 596 • *profile* attribute SHOULD represent the name of application profile that this implementation profile is
597 referring to select the available part in the definition. This attribute is OPTIONAL.
- 598 • *location* attribute SHOULD represent the location of the application profile to get the actual file by the
599 party who want to know the content of the application profile. This attribute is OPTIONAL.
- 600 • *namespace* attribute SHOULD represent the namespace of the application profile. This attribute is
601 necessary to identify the profile in world-wide basis. This attribute is OPTIONAL.
- 602 • *create* attribute SHOULD represent the date of creation of the implementation profile. This attribute is
603 OPTIONAL.
- 604 • *description* attribute SHOULD represent any description of the implementation profile. This attribute is
605 OPTIONAL.
- 606

607 4.8 ImplementDocument Element

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

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

613

```
614 <xsd:element name="ImplementDocument">  
615 <xsd:complexType>  
616 <xsd:sequence>  
617 <xsd:element ref="ImplementAction" minOccurs="0" maxOccurs="unbounded"/>  
618 <xsd:element ref="ImplementProperty" minOccurs="0" maxOccurs="unbounded"/>  
619 <xsd:element ref="ImplementEvent" minOccurs="0" maxOccurs="unbounded"/>  
620 </xsd:sequence>  
621 <xsd:attribute name="name" type="xsd:string" use="required"/>  
622 <xsd:attribute name="option" type="xsd:string"/>  
623 <xsd:attribute name="profile" type="xsd:string"/>  
624 <xsd:attribute name="location" type="xsd:string"/>  
625 <xsd:attribute name="namespace" type="xsd:string"/>  
626 <xsd:attribute name="description" type="xsd:string"/>  
627 </xsd:complexType>  
628 </xsd:element>
```

629

630 • *ImplementAction* element SHOULD represent an action that the program can perform for the domain
631 document. This element MAY represent a role of the program in the transaction.

632 • *ImplementProperty* element SHOULD represent a property that the program can deal with in the
633 domain object. All properties defined in this element SHOULD be defined as a property of a domain
634 object in the corresponding application profile.

635 • *ImplementEvent* element SHOULD represent an event that the program can monitor a property in
636 order to notify the change of the data to subscribers. This information MAY be defined by each
637 application programs.

638

639 • *name* attribute SHOULD represent the name of the domain document. The name SHOULD be
640 defined in the list of domain document in the corresponding application profile. This attribute is
641 REQUIRED.

642 • *option* attribute SHOULD represent optional process to deal with the domain document data. There
643 can be several domain document of same document name if the document has different option value.
644 According to the option process, the required implement properties may be different.

645 • *profile* attribute SHOULD represent the name of application profile that this *ImplementDocument* is
646 referring to select the available part in the definition. This attribute is OPTIONAL.

647 • *location* attribute SHOULD represent the location of the application profile to get the actual file by the
648 party who want to know the content of the application profile. This attribute is OPTIONAL.

649 • *namespace* attribute SHOULD represent the namespace of the *ImplementDocument*. This attribute is
650 necessary to identify the document name in world-wide basis. This attribute is OPTIONAL.

651 • *description* attribute SHOULD represent any description of the implemented document. This attribute
652 is OPTIONAL.

653 4.9 ImplementAction Element

654 *ImplementAction* element SHOULD represent an action that the program can perform for the domain
655 document. The actions include the transaction model referred to as "Add", "Change", "Remove", "Notify",
656 "Sync", "Get" or "Show". This element MAY represent a role of the program in the transaction such as
657 sender or receiver.

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

660

```
661 <xsd:element name="ImplementAction">  
662   <xsd:complexType>  
663     <xsd:attribute name="action" type="xsd:string" use="required"/>  
664     <xsd:attribute name="level" type="xsd:int"/>  
665     <xsd:attribute name="role" type="xsd:string"/>  
666     <xsd:attribute name="description" type="xsd:string"/>  
667   </xsd:complexType>  
668 </xsd:element>
```

669

- 670 • *action* attribute SHOULD represent the action performed by the application program. The value of this
671 attribute SHOULD be one of “Add”, “Change”, “Remove”, “Notify”, “Sync”, “Get” and “Show”. This
672 attribute is REQUIRED.
- 673 • *level* attribute SHOULD represent an implementation level defined in [PPS02] for each document
674 processed by the application program. Level 0 shows no implementation, while level 1 and 2 are
675 partially and fully implemented, respectively. Default value is 1 that minimum implementation is
676 supported. This attribute is OPTIONAL.
- 677 • *role* attribute SHOULD represent a role in the transaction. The value of this attribute is either “Server”
678 or “Client”. Every transaction has its available roles that can be selected as a value of this attribute.
679 Default value is “Server”. This attribute is OPTIONAL.
- 680 • *description* attribute SHOULD represent any description of the implement action. This attribute is
681 OPTIONAL.

682 4.10 ImplementProperty Element

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

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

690

```
691 <xsd:element name="ImplementProperty">  
692   <xsd:complexType>  
693     <xsd:attribute name="name" type="xsd:string" use="required"/>  
694     <xsd:attribute name="title" type="xsd:string"/>  
695     <xsd:attribute name="extend" type="xsd:string"/>  
696     <xsd:attribute name="link" type="xsd:string"/>  
697     <xsd:attribute name="multiple" type="xsd:string"/>  
698     <xsd:attribute name="path" type="xsd:string"/>  
699     <xsd:attribute name="dataType" type="xsd:string"/>  
700     <xsd:attribute name="enumeration" type="xsd:string"/>  
701     <xsd:attribute name="type" type="xsd:string"/>  
702     <xsd:attribute name="use" type="xsd:string"/>  
703     <xsd:attribute name="description" type="xsd:string"/>  
704   </xsd:complexType>  
705 </xsd:element>
```

706

- 707 • *name* attribute SHOULD represent the name of the property. The name SHOULD be defined in the
708 corresponding application profile. This attribute is REQUIRED.
- 709 • *title* attribute SHOULD represent the header title of the property. This value MAY be a short
710 description to show the property relating to the actual world. This attribute is OPTIONAL.

- 711 • *extend* attribute SHOULD represent qualifier string that is specified as prefix of the property name, if
712 this property is extended by the local program. For example, if the value is “user”, then the description
713 of this property will have “user:” prefix in the actual messages. This attribute is OPTIONAL.
- 714 • *link* attribute SHOULD represent that this property is also defined in other domain document that can
715 be linked to this document. The value of this attribute MAY has the name of domain document.
- 716 • *multiple* attribute SHOULD represent whether the property can have multiple values or not. If the
717 value of this attribute is positive integer or “Unbounded”, actual message can have corresponding
718 number of values for this property. The value number SHOULD be less or equal than the number
719 defined in the application profile.
- 720 • *path* attribute SHOULD represent the location of the attribute data in the primitive XML description
721 defined in [PPS01]. The specification of the path SHOULD conform to the syntax of [PATH]. If the
722 attribute value of *extend* is defined and this attribute is not described, then the default path data
723 SHOULD be “Spce[@type='aaa:bbb']/CCC/@value”, where aaa denotes the value of *extend* attribute
724 and bbb denotes the value of *name* attribute, and CCC is the value of *dataType* attribute.
- 725 • *dataType* attribute SHOULD represent the data type of the property. The expecting value of this
726 attribute is Qty, Char and Time. This attribute is REQUIRED if the value of *extend* has data.
727 Otherwise it is OPTIONAL.
- 728 • *enumeration* attribute SHOULD represent the name of enumeration type when the property is
729 extended by the local program, and has a value in the enumeration list. The name of enumeration
730 type SHOULD be specified in *Enumeration* elements in the application profile definition. This attribute
731 is OPTIONAL.
- 732 • *type* attribute SHOULD represent that the type of this property in terms of usage. When the value is
733 “Typical”, then the usage of this property is typical.
- 734 • *use* attribute SHOULD whether the property is mandatory. When the value “Required” represents
735 mandatory, while the value “Optional” represents optional. This value SHOULD be “Required” if the
736 corresponding property in the application profile has “Required” value. Default value of this attribute is
737 “Optional”.
- 738 • *description* attribute SHOULD represent any description of the property. This attribute is OPTIONAL.
739

740 4.11 ImplementEvent Element

741 *ImplementEvent* element SHOULD represent any event definitions that the application program monitors
742 on a particular property and detects the event occurrence on it. When the event occurs, the application
743 program SHOULD publish a notification of the event to all the parties who are on the list of subscription.
744 This information is defined by each application program, then clients of the event notification service MAY
745 request for the publication as a subscriber.

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

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

751

```

752 <xsd:element name="ImplementEvent">
753   <xsd:complexType>
754     <xsd:sequence>
755       <xsd:element ref="App" minOccurs="0"/>
756       <xsd:element ref="Condition" minOccurs="0" maxOccurs="unbounded"/>
757       <xsd:element ref="Selection" minOccurs="0" maxOccurs="unbounded"/>
758       <xsd:element ref="Property" minOccurs="0" maxOccurs="unbounded"/>
759     </xsd:sequence>
760     <xsd:attribute name="name" type="xsd:string" use="required"/>
761     <xsd:attribute name="type" type="xsd:string"/>
762     <xsd:attribute name="cycle" type="xsd:duration"/>

```

763
764
765
766
767

```
<xsd:attribute name="start" type="xsd:dateTime"/>  
<xsd:attribute name="expire" type="xsd:dateTime"/>  
<xsd:attribute name="description" type="xsd:string"/>  
</xsd:complexType>  
</xsd:element>
```

768

- 769 • *App* element SHOULD represent the application specific information about event monitoring, event
770 processing, transaction control and so forth. The specification of *App* element is defined in [PPS01].
- 771 • *Condition* element SHOULD represent the condition to select the target domain objects the
772 application is monitoring the event. The specification of this element is defined in [PPS02].
- 773 • *Selection* element SHOULD represent the condition of selecting the target property in the domain
774 object. The selected property values are reported to the subscribers when event occurs. When the
775 target property is multiple, *Condition* element under this element can restrict the properties. The
776 specification of this element is defined in [PPS02].
- 777 • *Property* element SHOULD represent the target property and constraints to detect event on the
778 property. The target property is monitored by the program. When there is more than one *Property*
779 element under the *ImplementEvent*, it SHOULD represent that more than one conditions need to be
780 checked to detect the event occurrence. Each *Property* element MAY have a different target property
781 on the domain object to others. Conditions of these properties SHOULD be conjunctive. The
782 specification of this element is defined in [PPS02].
- 783
- 784 • *name* attribute SHOULD represent the name of the event. The name SHOULD be unique in the
785 domain object defined in the application profile. This attribute is REQUIRED.
- 786 • *type* attribute SHOULD represent a method to detect this event. Value candidates of this attribute
787 SHOULD include "True", "False", "Enter", "Leave", "Change", "Add", and "Remove". If the value is
788 "True", then event occurs when all the conditions are true. If the value is "False", then event occurs
789 when at least one condition is false. If the value is "Enter", then event occurs when the status
790 changes from false to true, while "Leave" means that the status changes from true to false. If the
791 value is "Change", then event occurs when the value of the target property is change. "Add"
792 represents that event occurs when a new domain object which satisfies the conditions is added, and
793 "Remove" shows that event occurs when any objects which satisfies the conditions is removed. If the
794 target property is multiple and *Selection* element is described, then "Add" and "Remove" mean that
795 one of the multiple properties is added and removed, respectively. Default value is "Change". This
796 attribute is OPTIONAL.
- 797 • *cycle* attribute SHOULD represent the duration of monitoring of the property value to detect the event
798 occurrence. The application program SHOULD monitor the value until the expiration date. This
799 attribute is OPTIONAL.
- 800 • *start* attribute SHOULD represent starting time of the monitoring and notification service. After this
801 date and time, application program start monitoring the properties. If this attribute is not described,
802 then it represent the service has already started. The origin of cyclic procedure defined by *cycle*
803 attribute SHOULD be this start time. This attribute is OPTIONAL.
- 804 • *expire* attribute SHOULD represent expire time and date of the event notification. After the time of
805 expiration, the application will stop monitoring the event occurrence. If this attribute is not defined, it
806 SHOULD represent that there is no expiration date. This attribute is OPTIONAL.
- 807 • *description* attribute SHOULD represent any description of the event. This attribute is OPTIONAL.
- 808

809 **A. Acknowledgements**

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

812 **Participants:**

813 Shinya Matsukawa, Hitachi
814 Tomohiko Maeda, Fujitsu
815 Masahiro Mizutani, Unisys Corporation
816 Akihiro Kawauchi, Individual Member
817 Yuto Banba, PSLX Forum
818 Hideichi Okamune, PSLX Forum

819

820 **B. Revision History**

821

Revision	Date	Editor	Changes Made

822

823