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

Public Review Draft 02

17 Dec 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

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 22
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

1.5 Terms and definitions

Application profile

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

Domain document

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

Domain object

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

Domain property

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

Implementation profile

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

Messaging model

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

Primitive element

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

Transaction element

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

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.

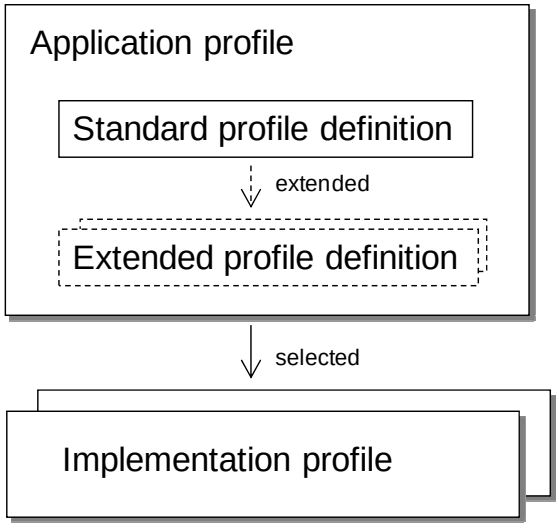

Figure 1 Structure of profile specifications

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.

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

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

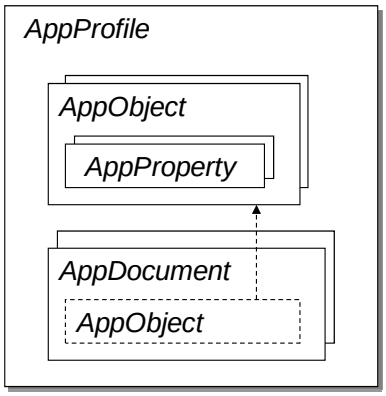

Figure 2 Application Profile

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

Example: Application profile definition

```
<AppProfile name=”pps-profile” prefix="pps" namespace="http:www.oasis-open.org/committees/pps/profile-1.0">
  <AppObject name="Product" primitive=”Item”>
    <AppProperty name="id" path="@id"/>
    <AppProperty name="name" path="@name"/>
    …
    <AppProperty name="Size" path="Spec[@type=”size”]/@value"/>
    <AppProperty name="Color" path="Spec[@type=”color”]/@value"/>
    …
  </AppObject>
  …
  <AppDocument name="ProductRecord" object=”Product”/>
  <AppDocument name="ProductInventory" object=”Product”/>
  <AppDocument name="BillOfMaterials" object=”Product”/>
  <AppDocument name="BillOfResources" object=”Product”/>
  …
</AppProfile>
```

2.3 Standard profile definitions

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

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

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

Table 1 Domain objects required in standard profile definitions

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

2.4 Extended profile definitions

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

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

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

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

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

Example: Extended application profile

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

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

2.5 Revision rule

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

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

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

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

When the major version increases, it:

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

When the minor version increases, it:

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

3 Implementation profiles

3.1 General

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.

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.

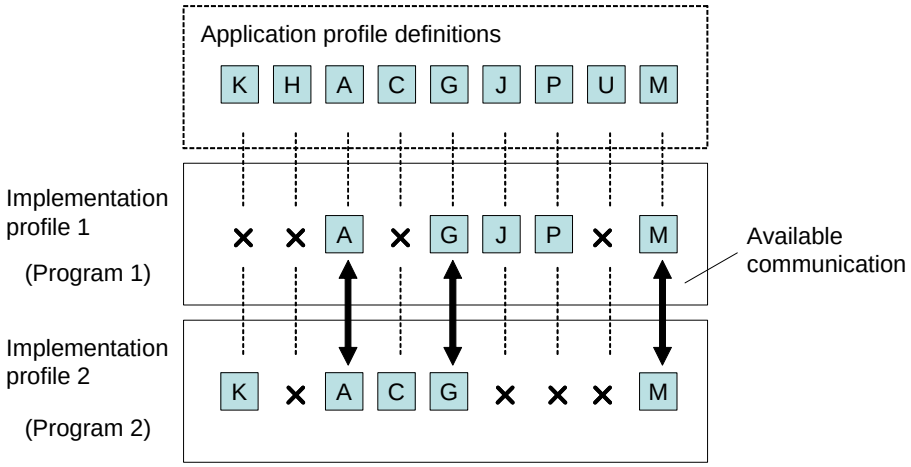

Figure 3 Concept of communication availability between implementations

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. Tow application programs can exchange a message properly if the both implementations have the corresponding capability.

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.

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.

3.2 Structure of implementation profiles

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.

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.

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

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

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

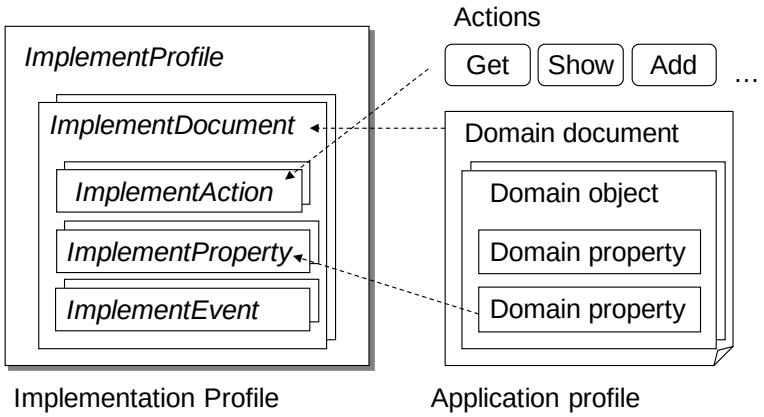

Figure 4 Structure of ImplementProfile

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

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

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

Example: Implementation profile of a program for an application profile

```
<ImplementProfile id=”AP001” action=”Notify”>
  <ImplementDocument name=”Product”>
    <ImplementAction action=”Get” level=”1”/>
    <ImplementAction action=”Show” level=”1”/>
    <ImplementAction action=”Add” level=”2”/>
    <ImplementProperty name=”id” title=”Company ID”/>
    <ImplementProperty name=”name” title=”Company name”/>
  </ImplementDocument>
  <ImplementDocument name=”ProductInventory”>
    …
  </ImplementDocument>
  ….
</ImplementProfile>
```

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

Example: A message created on the implementation profile

```
<Document name=”Product” id=”001” action=”Get”
 namespace="http:www.oasis-open.org/committees/pps/profile-1.0">
 <Condition>
  <Property name=”pps:name” value=”MX-001”/>
  <Property name=”pps:color” value=”white”/>
 </Condition>
 <Selection type=”All”/>
</Document>
```

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

3.3 Level of implementation

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

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

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

3.4 Profile inquiry

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

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

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

Example: Inquiry of implementation profile for PPS standard profile definition

```
<Message id=”A01” sender=”A”>
 <ImplementProfile action=”Get” />
</Message>
```

Example: Answer of the inquiry in above example

```
<Message id=”B01” sender=”B”>
<ImplementProfile id=”B01” action=”Show” >
 <ImplementDocument name="Supplier">
  <ImplementAction action="Get" level="1"/>
  <ImplementAction action="Add"/>
  <ImplementProperty name="id" display="NO"/>
```

```
    <ImplementProperty name="name" display="NAME"/>
    …
  </ImplementDocument>

</ImplementProfile >
</Message>
```

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

4 XML Elements

4.1 AppProfile Element

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.

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="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>
```

- *Enumeration* element SHOULD represent any enumeration type that is used as a special type of properties.
- *AppObject* element SHOULD represent any domain objects used in the domain documents defined in this profile.
- *AppDocument* element SHOULD represent any domain documents that the applications may send or receive on this profile.

- *name* attribute SHOULD represent the name of this application profile. The name SHOULD be unique in the namespace. This attribute is REQUIRED.
- *base* attribute SHOULD represent the base application profile when this profile is an extended application profile.
- *location* attribute SHOULD represent the location where the profile can be downloaded via Internet.
- *prefix* attribute SHOULD represent the prefix text that is added in the name of values that are qualified by this profile.
- *namespace* attribute SHOULD represent the namespace when this profile is used in a specific namespace.
- *create* attribute SHOULD represent the date of creation of the profile
- *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 described in *AppApplication* element that corresponds to an 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="descirption" type="xsd:string"/>
  </xsd:complexType>
</xsd:element>
```

- *name* attribute SHOULD represent the name of the domain document. The name SHOULD be unique in the namespace 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 objects referred by this attribute SHOULD be defined in the same application profile or base application profile. This attribute is REQUIRED.
- *category* attribute SHULD 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 for this attribute. 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 to from domain documents in the application profile SHOULD be described 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="descirption" type="xsd:string"/>
  </xsd:complexType>
</xsd:element>
```

- *AppProperty* element SHOULD represent a property that may be described in the domain objects of the application profile definition. All possible properties SHOULD be described in the domain object represented by *AppObject*.

- *name* attribute SHOULD represent the name of the object. The name SHOULD be unique under the application profile definition in the selected namespace. This attribute is REQUIRED.
- *primitive* attribute SHOULD represent a primitive element name selected from the primitive element list defined in [PPS01]. Since every domain object is a subclass of one in the primitive objects, all *AppObject* elements SHOULD have a primitive attribute. This attribute is REQUIRED.

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

4.4 AppProperty Element

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

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="AppProperty">
  <xsd:complexType>
    <xsd:attribute name="name" type="xsd:string"/>
    <xsd:attribute name="path" type="xsd:string"/>
    <xsd:attribute name="multiple" type="xsd:string"/>
    <xsd:attribute name="key" type="xsd:string"/>
    <xsd:attribute name="enumeration" type="xsd:string"/>
    <xsd:attribute name="dataType" type="xsd:string"/>
    <xsd:attribute name="use" type="xsd:string"/>
    <xsd:attribute name="description" type="xsd:string"/>
  </xsd:complexType>
</xsd:element>
```

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

4.5 Enumeration Element

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

Enumeration type is independent from any domain object in the application profile definition. Therefore, 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="descirption" 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].

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="ImplementProfile">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="Error" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="App" minOccurs="0"/>
      <xsd:element ref="ImplementDocument" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="id" type="xsd:string"/>
    <xsd:attribute name="name" type="xsd:string"/>
    <xsd:attribute name="action" type="xsd:string"/>
    <xsd:attribute name="profile" type="xsd:string"/>
    <xsd:attribute name="location" type="xsd:string"/>
    <xsd:attribute name="namespace" type="xsd:string"/>
    <xsd:attribute name="create" type="xsd:dateTime"/>
    <xsd:attribute name="description" type="xsd:string"/>
  </xsd:complexType>
</xsd:element>
```

- *Error* element SHOULD represent error information, when any errors occur during the transaction of message exchange of this implementation profile. The specification of this element is defined in [PPS02].
- *App* element SHOULD represent any information for the application program concerning the transaction of profile exchange. The use of this element SHOULD be consistent with all cases of transactions while the other messages are exchanged. The specification of this element is defined in [PPS02].
- *ImplementDocument* element SHOULD represent a domain document that the application program may send or receive. All available documents in the application profile SHOULD be listed using this element.

- *id* attribute SHOULD represent identifier of the application program. The id SHOULD be unique in all application programs that can be accessed in the network. In order to guarantee the uniqueness, system integrator must assigns the unique number and manages it in the network configuration. This id is the same as the sender name when the application will send a message. This attribute is REQUIRED.
- *name* attribute SHOULD represent a name that the application program shows its name for an explanation by natural texts. This attribute is OPTIONAL
- *action* attribute SHOULD represent a name of action during transaction models defined in [PPS02]. The value of this attribute SHOULD be “Notify”, “Get” or “Show”. When the element is created as a message for exchange, this attribute is REQUIRED. Otherwise, such as for a XML document file, this attribute is OPTIONAL.
- *profile* attribute SHOULD represent the name of application profile that this implementation profile is referring to select the available part in the definition. This attribute is OPTIONAL.
- *location* attribute SHOULD represent the location of the application profile to get the actual file by the party who want to know the content of the application profile. This attribute is OPTIONAL.
- *namespace* attribute SHOULD represent the namespace of the application profile. This attribute is necessary to identify the profile in world-wide basis. This attribute is OPTIONAL.
- *create* attribute SHOULD represent the date of creation of the implementation profile. This attribute is OPTIONAL.
- *description* attribute SHOULD represent any description of the implementation profile. This attribute is OPTIONAL.

4.8 ImplementDocument Element

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

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="ImplementDocument">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="ImplementAction" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="ImplementProperty" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="ImplementEvent" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="name" type="xsd:string" use="required"/>
    <xsd:attribute name="title" type="xsd:string"/>
    <xsd:attribute name="option" type="xsd:string"/>
    <xsd:attribute name="profile" type="xsd:string"/>
    <xsd:attribute name="location" type="xsd:string"/>
    <xsd:attribute name="namespace" type="xsd:string"/>
    <xsd:attribute name="description" type="xsd:string"/>
  </xsd:complexType>
</xsd:element>
```

- *ImplementAction* element SHOULD represent an action that the program can perform for the domain document. This element MAY represent a role of the program in the transaction.
- *ImplementProperty* element SHOULD represent a property that the program can deal with in the domain object. All properties defined in this element SHOULD be defined as a property of a domain object in the corresponding application profile.
- *ImplementEvent* element SHOULD represent an event that the program can monitor a property in order to notify the change of the data to subscribers. This information MAY be defined by each application programs.

- *name* attribute SHOULD represent the name of the domain document. The name SHOULD be defined in the list of domain document in the corresponding application profile. This attribute is REQUIRED.
- *title* attribute SHOULD represent the header title of the document. This value MAY be a short description to show the property relating to the actual world. This attribute is OPTIONAL.
- *option* attribute SHOULD represent optional process to deal with the domain document data. There can be several domain document of same document name if the document has different option value. According to the option process, the required implement properties may be different.
- *profile* attribute SHOULD represent the name of application profile that this *ImplementDocument* is referring to select the available part in the definition. This attribute is OPTIONAL.
- *location* attribute SHOULD represent the location of the application profile to get the actual file by the party who want to know the content of the application profile. This attribute is OPTIONAL.
- *namespace* attribute SHOULD represent the namespace of the *ImplementDocument*. This attribute is necessary to identify the document name in world-wide basis. This attribute is OPTIONAL.
- *description* attribute SHOULD represent any description of the implemented document. This attribute is OPTIONAL.

4.9 ImplementAction Element

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

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="ImplementAction">
  <xsd:complexType>
    <xsd:attribute name="action" type="xsd:string" use="required"/>
    <xsd:attribute name="level" type="xsd:int"/>
    <xsd:attribute name="role" type="xsd:string"/>
    <xsd:attribute name="description" type="xsd:string"/>
  </xsd:complexType>
</xsd:element>
```

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

4.10 ImplementProperty Element

ImplementProperty element SHOULD represent a domain property that can be processed in the application program. Some properties SHULD be defined in the corresponding domain object in the application profile definition. The properties that are not defined in the application profile SHOULD be specified in this element as a user extended property. Properties extended by application programs SHOULD have additional definitions similar to the definitions by *AppProperty* 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="ImplementProperty">
  <xsd:complexType>
    <xsd:attribute name="name" type="xsd:string" use="required"/>
    <xsd:attribute name="title" type="xsd:string"/>
    <xsd:attribute name="extend" type="xsd:string"/>
    <xsd:attribute name="link" type="xsd:string"/>
    <xsd:attribute name="multiple" type="xsd:string"/>
    <xsd:attribute name="path" type="xsd:string"/>
    <xsd:attribute name="dataType" type="xsd:string"/>
    <xsd:attribute name="enumeration" type="xsd:string"/>
    <xsd:attribute name="type" type="xsd:string"/>
    <xsd:attribute name="use" type="xsd:string"/>
    <xsd:attribute name="description" type="xsd:string"/>
  </xsd:complexType>
</xsd:element>
```

- *name* attribute SHOULD represent the name of the property. The name SHOULD be defined in the corresponding application profile. This attribute is REQUIRED.
- *title* attribute SHOULD represent the header title of the property. This value MAY be a short description to show the property relating to the actual world. This attribute is OPTIONAL.
- *extend* attribute SHOULD represent qualifier string that is specified as prefix of the property name, if this property is extended by the local program. For example, if the value is “user”, then the description of this property will have “user:” prefix in the actual messages. This attribute is OPTIONAL.
- *link* attribute SHOULD represent that this property is also defined in other domain document that can be linked to this document. The value of this attribute MAY has the name of domain document.
- *multiple* attribute SHOULD represent whether the property can have multiple values or not. If the value of this attribute is positive integer or “Unbounded”, actual message can have corresponding number of values for this property. The value number SHOULD be less or equal than the number defined in the application profile.
- *path* attribute SHOULD represent the location of the attribute data in the primitive XML description defined in [PPS01]. The specification of the path SHOULD conform to the syntax of [PATH]. If the attribute value of *extend* is defined and this attribute is not described, then the default path data SHOULD be “Spce[@type=’aaa:bbb’]/CCC/@value”, where aaa denotes the value of *extend* attribute and bbb denotes the value of *name* attribute, and CCC is the value of *dataType* attribute.
- *dataType* attribute SHOULD represent the data type of the property. The expecting value of this attribute is Qty, Char and Time. This attribute is REQUIRED if the value of *extend* has data. Otherwise it is OPTIONAL.
- *enumeration* attribute SHOULD represent the name of enumeration type when the property is extended by the local program, and has a value in the enumeration list. The name of enumeration type SHOULD be specified in *Enumeration* elements in the application profile definition. This attribute is OPTIONAL.
- *type* attribute SHOULD represent that the type of this property in terms of usage. When the value is “Typical”, then the usage of this property is typical.
- *use* attribute SHOULD whether the property is mandatory. When the value “Required” represents mandatory, while the value “Optional” represents optional. This value SHOULD be “Required” if the corresponding property in the application profile has “Required” value. Default value of this attribute is “Optional”.
- *description* attribute SHOULD represent any description of the property. This attribute is OPTIONAL.

4.11 ImplementEvent Element

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

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

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="ImplementEvent">
  <xsd:complexType>
    <xsd:sequence>
```

```
      <xsd:element ref="App" minOccurs="0"/>
      <xsd:element ref="Condition" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="Selection" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="Property" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="name" type="xsd:string" use="required"/>
    <xsd:attribute name="type" type="xsd:string"/>
    <xsd:attribute name="cycle" type="xsd:duration"/>
    <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>
```

- *App* element SHOULD represent the application specific information about event monitoring, event processing, transaction control and so forth. The specification of *App* element is defined in [PPS01].
- *Condition* element SHOULD represent the condition to select the target domain objects the application is monitoring the event. The specification of this element is defined in [PPS02].
- *Selection* element SHOULD represent the condition of selecting the target property in the domain object. The selected property values are reported to the subscribers when event occurs. When the target property is multiple, Condition element under this element can restrict the properties. The specification of this element is defined in [PPS02].
- *Property* element SHOULD represent the target property and constraints to detect event on the property. The target property is monitored by the program. When there is more than one Property element under the *ImplementEvent*, it SHOULD represent that more than one conditions need to be checked to detect the event occurrence. Each Property element MAY have a different target property on the domain object to others. Conditions of these properties SHOULD be conjunctive. The specification of this element is defined in [PPS02].

- *name* attribute SHOULD represent the name of the event. The name SHOULD be unique in the domain object defined in the application profile. This attribute is REQUIRED.
- *type* attribute SHOULD represent a method to detect this event. Value candidates of this attribute SHOULD include “True”, “False”, “Enter”, “Leave”, “Change”, “Add”, and “Remove”. If the value is “True”, then event occurs when all the conditions are true. If the value is “False”, then event occurs when at least one condition is false. If the value is “Enter”, then event occurs when the status changes from false to true, while “Leave” means that the status changes from true to false. If the value is “Change”, then event occurs when the value of the target property is change. “Add” represents that event occurs when a new domain object which satisfies the conditions is added, and “Remove” shows that event occurs when any objects which satisfies the conditions is removed. If the target property is multiple and *Selection* element is described, then “Add” and “Remove” mean that one of the multiple properties is added and removed, respectively. Default value is “Change”. This attribute is OPTIONAL.
- *cycle* attribute SHOULD represent the duration of monitoring of the property value to detect the event occurrence. The application program SHOULD monitor the value until the expiration date. This attribute is OPTIONAL.
- *start* attribute SHOULD represent starting time of the monitoring and notification service. After this date and time, application program start monitoring the properties. If this attribute is not described, then it represent the service has already started. The origin of cyclic procedure defined by cycle attribute SHOULD be this start time. This attribute is OPTIONAL.
- *expire* attribute SHOULD represent expire time and date of the event notification. After the time of expiration, the application will stop monitoring the event occurrence. If this attribute is not defined, it SHOULD represent that there is no expiration date. This attribute is OPTIONAL.
- *description* attribute SHOULD represent any description of the event. This attribute is OPTIONAL.

A. Acknowledgements

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

Participants:

Shinya Matsukawa, Hitachi
Tomohiko Maeda, Fujitsu
Masahiro Mizutani, Unisys Corporation
Akihiro Kawauchi, Individual Member
Yuto Banba, PSLX Forum
Hideichi Okamune, PSLX Forum

B. Revision History

Revision	Date	Editor	Changes Made