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

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

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

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

The non-normative errata page for this specification is located at http://www.oasis-open.org/committees/pps/.
# 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 .............................................................................................................. 12
   2.5 Revision rule ..................................................................................................................................... 12
3 Implementation profiles ...................................................................................................................... 14
   3.1 General ............................................................................................................................................. 14
   3.2 Structure of implementation profiles ................................................................................................. 14
   3.3 Level of implementation .................................................................................................................... 16
   3.4 Profile inquiry .................................................................................................................................... 16
4 XML Elements .................................................................................................................................... 18
   4.1 AppProfile Element ........................................................................................................................... 18
   4.2 AppDocument Element ..................................................................................................................... 19
   4.3 AppObject Element ........................................................................................................................... 19
   4.4 AppProperty Element ....................................................................................................................... 20
   4.5 DataType Element ............................................................................................................................ 20
   4.6 UnitType Element ............................................................................................................................. 21
   4.7 Enumeration Element ....................................................................................................................... 22
   4.8 EnumElement Element ..................................................................................................................... 22
   4.9 ImplementProfile Element ................................................................................................................. 23
   4.10 ImplementDocument Element ........................................................................................................ 24
   4.11 ImplementAction Element ............................................................................................................. 24
   4.12 ImplementProperty Element ......................................................................................................... 25
   4.13 ImplementEvent Element ................................................................................................................ 26
A Acknowledgements ............................................................................................................................ 28
B Revision History .................................................................................................................................. 29
Figures

Figure 1 Structure of profile specifications ................................................................. 8
Figure 2 Profile definition ............................................................................................ 9
Figure 3 Concept of communication availability between implementations ................ 14
Figure 4 Structure of ImplementProfile ........................................................................ 15
1 Introduction

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

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

1.1 Terminology

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

1.2 Normative References


1.3 Non-Normative References


1.4 Conformance

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

http://docs.oasis-open.org/pps/v1.0/pps-profile-specifications.xsd
1.5 Terms and definitions

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

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

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 with the same property name, if the XML child element is multiple.

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

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

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

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

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

Several application profile definitions may exist developing by particular group for particular problem domains. Regarding such situation, this part of standard defines a standard profile definition as a template of application profile definitions. 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 profile definitions. Application profile definitions consist of standard profile definitions and extended profile definitions. Figure also shows the relation of application profile definitions with implementation profiles that are defined in the next section.

![Figure 1 Structure of profile specifications]

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.

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

2.2 Structure of profile definitions

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

Application profile definition SHOULD be specified 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 whole system, SHOULD be specified in `AppDocument` element under the `AppProfile` element.

All candidates of domain objects, which may be used in any domain objects 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 specified in `AppProperty` under the `AppObject`.

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

Example 1 Application profile definition

```xml
<AppProfile prefix="pps" name="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" />
  ...
</AppProfile>
```

2.3 Standard profile definitions

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

- Standard profile definition SHOULD have a name to identify the definition among all application programs in the past, current and future.
• 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 and accessible by all application programs in the problem domain via Internet by announcing the URL the application can download the document.

• Standard profile definition SHOULD have at least all the domain objects in Table 1. The domain objects correspond to the primitive elements defined in [PPS01].

• Every domain object in a standard profile definition SHOULD have at least all the domain properties in Table 2 and Table 3. The properties in Table 2 correspond to attributes in the primitive elements in [PPS01], and the properties in Table 3 correspond to sub-elements and its attribute in the primitive elements.

Table 1 Domain objects required in standard profile definitions

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

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

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Attribute Name</th>
<th>Description</th>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>id</td>
<td>Identifier</td>
<td>string</td>
</tr>
<tr>
<td>key</td>
<td>key</td>
<td>Key in the data storage</td>
<td>string</td>
</tr>
<tr>
<td>name</td>
<td>name</td>
<td>Name of the object</td>
<td>string</td>
</tr>
<tr>
<td>parent</td>
<td>parent</td>
<td>Parent name of the object</td>
<td>string</td>
</tr>
<tr>
<td>type</td>
<td>type</td>
<td>Qualifier to make a subclass</td>
<td>string</td>
</tr>
<tr>
<td>status</td>
<td>status</td>
<td>Status of the object</td>
<td>string</td>
</tr>
<tr>
<td>party</td>
<td>party</td>
<td>Relation to a party</td>
<td>string</td>
</tr>
<tr>
<td>plan</td>
<td>plan</td>
<td>Relation to a plan</td>
<td>string</td>
</tr>
<tr>
<td>order</td>
<td>order</td>
<td>Relation to an order</td>
<td>string</td>
</tr>
<tr>
<td>item</td>
<td>item</td>
<td>Relation to an item</td>
<td>string</td>
</tr>
<tr>
<td>Property Name</td>
<td>XML Element</td>
<td>Attribute Name</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>-------------</td>
<td>----------------</td>
<td>--------------------------------------</td>
</tr>
<tr>
<td>resource</td>
<td>resource</td>
<td>Relation to a resource</td>
<td>string</td>
</tr>
<tr>
<td>process</td>
<td>process</td>
<td>Relation to a process</td>
<td>string</td>
</tr>
<tr>
<td>lot</td>
<td>lot</td>
<td>Relation to a lot</td>
<td>string</td>
</tr>
<tr>
<td>task</td>
<td>task</td>
<td>Relation to a task</td>
<td>string</td>
</tr>
<tr>
<td>operation</td>
<td>operation</td>
<td>Relation to an operation</td>
<td>string</td>
</tr>
</tbody>
</table>

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

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

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

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

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

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.

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

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

Example 2 Extended application profile

```
<AppProfile prefix="ex1" name="http://www.pslx.org/profile-1" base="http://www.oasis-open.org/committees/pps/profile-1.0">
  <Enumeration name="groupType">
    <EnumElement name="high" description="description of a"/>
    <EnumElement name="low" description="description of b"/>
  </Enumeration>
  <AppObject name="Customer"/>
  <AppObject name="Supplier"/>
  <AppObject name="Consumer">
    <AppProperty name="id"/>
    <AppProperty name="name"/>
    <AppProperty name="group" path="Spec[type='pslx:group']/@value" enumeration="groupType"/>
  </AppObject>
  <AppDocument name="CustomerRecord"/>
  <AppDocument name="SupplierRecord"/>
  <AppDocument name="ConsumerRecord"/>
</AppProfile>
```

Example 2 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 named
group that has the enumeration type.

2.5 Revision rule

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

Any application profile SHOULD NOT be changed without keeping the following rules after when the
profile definition once 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 SHOULD deal with the new specification. The other is editorial revision where the any program doesn’t need to care. 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 SHOULD 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 name in the attribute of AppProfile element.

When the minor version increases, it:

- SHOULD follow the rule of major version increasing,
- SHOULD NOT remove the domain objects,
- SHOULD NOT remove the domain properties,
- SHOULD NOT remove the domain documents,
- SHOULD NOT change the domain object in any domain document.
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. To confirm the capability of any application program, section 3.4 provides the method how to get the information by receiving an implementation profile from the program.

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 definition has a set of capabilities by selecting from the all items 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 different application profile definitions. An implementation profile SHOULD have a corresponding application profile definition.

3.2 Structure of implementation profiles

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

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

The structure of ImplementProfile element is a special case of the transaction element defined in [PPS02]. Therefore, this can be regarded as a transaction document that is 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 transaction, and 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.

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

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

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

```xml
<ImplementProfile id="001" transaction="001" action="Notify" sender="APP1"
    profile="http:www.oasis-open.org/committees/pps/profile-1.0">
    <ImplementDocument name="ProductRecord">
        <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 4. A message created on the implementation profile

```
<ProductMaster id="001" transaction="A001-001" action="Get" sender="P01"
profile="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"/>
</ProductMaster>
```

Example 4 shows a message of a Get transaction 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 transaction including Add, Change, Remove, Get, Show, Notify and Sync. These transactions are prescribed in [PPS02]. Level of implementation represents whether or not the functions prescribed in [PPS02] are fully implemented or partially implemented. The level includes Full, Partial and None.

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 a Get message of *ImplementProfile* transaction.

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

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

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

Example 2 Inquiry of implementation profile for PPS standard profile definition

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

Example 3 Answer of the inquiry in Example 2

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

Example 2 and 3 are the request of implementation profile and its response. By the message in Example 2, the responder needs to answer its capability on PPS standard application profile.
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="DataType" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="UnitType" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="Enumeration" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="AppObject" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="AppDocument" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="name" type="xsd:string" use="required"/>
    <xsd:attribute name="base" type="xsd:string"/>
    <xsd:attribute name="location" type="xsd:string"/>
    <xsd:attribute name="prefix" type="xsd:string"/>
    <xsd:attribute name="namespace" type="xsd:string"/>
    <xsd:attribute name="create" type="xsd:string"/>
    <xsd:attribute name="description" type="xsd:string"/>
  </xsd:complexType>
</xsd:element>
```

- **DataType** element SHOULD represent any data type that is used as a type of property data.
- **UnitType** element SHOULD represent any unit type of property defined in domain object in this profile.
- **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 application documents that the applications may send or receive on this profile.
- **name** attribute SHOULD represent the name of this application profile. By including an URL texts, the name SHOULD be unique and even in the future. This attribute is REQUIRED.
- **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 specified in *AppApplication* element that corresponds to the application profile. This information SHOULD be specified in the following XML schema. The XML documents generated by the schema SHOULD be consistent with the following arguments.

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

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

4.3 AppObject Element

*AppObject* element SHOULD represent a domain object corresponding to any actual object in the target problem domain. All domain objects that are referred from domain documents in the same application profile SHOULD be specified in the *AppObject* element. This information SHOULD be specified in the following XML schema. The XML documents generated by the schema SHOULD be consistent with the following arguments.

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

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

```xml
<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:boolean"/>
    <xsd:attribute name="enumeration" type="xsd:string"/>
    <xsd:attribute name="dataType" type="xsd:string"/>
    <xsd:attribute name="unitType" 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 this attribute has true value, actual message described by [PPS01] specification can have several values for this property. This attribute is OPTIONAL, and the default value is “false”.

• *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 “int”, “double”, “string”, “datetime” or “duration”. This attribute is OPTIONAL. If there is no value, application program MAY determine the value according to the path attribute, because the data type is the same as the data type of attribute on the body elements represented according to [PPS01].

• *unitType* attribute SHOULD represent the unit of measure for the property. The value of this attribute is any texts that show the unit name. This attribute is OPTIONAL.

• *description* attribute SHOULD represent any description of the domain property. This attribute is OPTIONAL.

### 4.5 DataType Element

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

This information SHOULD be specified in the following XML schema. The XML documents generated by the schema SHOULD be consistent with the following arguments.
• **name** attribute SHOULD represent a name of data type. The name SHOULD be unique in the application profile definition. This attribute is REQUIRED.

• **type** attribute SHOULD represent a type that is prescribed by the application program or commercial database packages. This attribute is OPTIONAL.

• **size** attribute SHOULD represent size of data. If the data is string, then this shows length of the string. This attribute is OPTIONAL.

• **description** attribute SHOULD represent any description of the data type. This attribute is OPTIONAL.

### 4.6 UnitType Element

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

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

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

• **base** attribute SHOULD represent a unit name that is referred to as base unit. The unit MAY be calculated by the base unit and size. The base name SHOULD be defined in other unit type definition in the same application profile. This attribute is OPTIONAL.

• **size** attribute SHOULD represent size of data. The size SHOULD show how much times as big as the size of the base unit. This information is used to convert the unit from the base unit, and vice versa. This attribute is OPTIONAL.

• **time** attribute SHOULD represent a origin time of data. If the unit is on a discrete time scale, this data shows the origin of the time scale. This attribute is OPTIONAL.

• **duration** attribute SHOULD represent a time unit of data. If the unit is on a discrete time scale, this data shows the unit time duration of the time scale. This attribute is OPTIONAL.

• **description** attribute SHOULD represent any description of the data type. This attribute is OPTIONAL.
4.7 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, many different domain objects MAY have different properties that has the same enumeration type. Instead of this, the name of enumeration type may have wide-variety.

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

```xml
<xsd:element name="Enumeration">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="EnumElement" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="name" type="xsd:string" use="required"/>
    <xsd:attribute name="description" type="xsd:string"/>
  </xsd:complexType>
</xsd:element>
```

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

4.8 EnumElement Element

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

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

```xml
<xsd:element name="EnumElement">
  <xsd:complexType>
    <xsd:attribute name="value" type="xsd:string" use="required"/>
    <xsd:attribute name="primary" type="xsd:boolean"/>
    <xsd:attribute name="alias" type="xsd:int"/>
    <xsd:attribute name="description" type="xsd:string"/>
  </xsd:complexType>
</xsd:element>
```

- **value** attribute SHOULD represent value texts that can be selected from the enumeration list. The value SHOULD be unique in the value list of the enumeration type. This attribute is REQUIRED.
- **primary** attribute SHOULD represent the primary item in the enumeration list. Only the primary attribute SHOULD have “true” value for this attribute. No more than one item in the item list SHOULD have “true” value. This attribute is OPTIONAL, and the default value is “false”.
- **alias** attribute SHOULD represent a numerical value instead of the text value specified in the value attribute. The value SHOULD be unique integer among the items in the enumeration type.
- **description** attribute SHOULD represent any description of the enumeration type. This attribute is OPTIONAL.
4.9 ImplementProfile Element

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

```xml
<xsd:element name="ImplementProfile">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element ref="Error" minOccurs="0" maxOccurs="unbounded"/>
            <xsd:element ref="App" minOccurs="0" maxOccurs="1"/>
            <xsd:element ref="Spec" minOccurs="0" maxOccurs="unbounded"/>
            <xsd:element ref="ImplementDocument" minOccurs="0" maxOccurs="unbounded"/>
        </xsd:sequence>
        <xsd:attribute name="id" type="xsd:string"/>
        <xsd:attribute name="action" type="xsd:string"/>
        <xsd:attribute name="transaction" type="xsd:string"/>
        <xsd:attribute name="profile" type="xsd:string" use="required"/>
        <xsd:attribute name="sender" type="xsd:string" use="required"/>
        <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].
- Spec element SHOULD represent an additional specification about the transaction of exchanging this
  implementation profile. 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 message. The id SHOULD be unique in all messages
  the sender has sent. Then the receiver of this message can identify the message by combination of
  the id and the sender name. 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.
- action attribute SHOULD represent a name of action during transaction models defined in [PPS02].
  The value of this attribute SHOULD be “Add”, “Change”, “Remove”, “Notify”, “Sync”, “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.
- transaction attribute SHOULD represent ID of transaction that several messages in a certain
  transaction can be grouped. Response message SHOULD have the same transaction ID as the
  request message. 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 the implementation profile is depending. The application profile MAY be either a standard application profile or an extended application profile. This attribute is REQUIRED.
• sender attribute SHOULD represent a name of application program who has functions of message transactions. Definition of this implementation profile SHOULD certify the functionality of the program. The name of sender SHOULD be unique in the table of party who may participate the message exchange. This attribute is REQUIRED.
• create attribute SHOULD represent the date of creation of the implementation profile. This attribute is OPTIONAL.
• description attribute SHOULD represent any description of the enumeration type. This attribute is OPTIONAL.

4.10 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.

```
<xs:element name="ImplementDocument">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="ImplementAction" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element ref="ImplementProperty" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element ref="ImplementEvent" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="name" type="xsd:string" use="required"/>
    <xs:attribute name="profile" type="xsd:string"/>
  </xs:complexType>
</xs: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.
• 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 required if the profile is different from the name defined in ImplementPforile element. Therefore, this attribute is OPTIONAL.

4.11 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.

```xml
<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: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”, or any combination of those. If more than one action are specified, all available names of action SHOULD be separated by “|” character. 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 the highest number that shows the fully implemented. This attribute is OPTIONAL.

- `role` attribute SHOULD represent a role in the transaction. Every transaction has its available roles that can be selected as a value of this attribute. Default value is “receiver” or “responder”. This attribute is OPTIONAL.

### 4.12 ImplementProperty Element

ImplementProperty element SHOULD represent a domain property that can be processed in the application program. Some properties SHOULD 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.

```xml
<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:boolean"/>
        <xsd:attribute name="path" type="xsd:string"/>
        <xsd:attribute name="dataType" type="xsd:string"/>
        <xsd:attribute name="unitType" type="xsd:string"/>
        <xsd:attribute name="enumeration" 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 whether this property is extended by the local program or not. If the value of this attribute is “true”, then the description of this property will have “user:" prefix in the actual messages. This attribute is OPTIONAL.
• **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 true, this attribute is REQUIRED, and otherwise OPTIONAL.

• **dataType** attribute SHOULD represent the data type of the property. This attribute is used to map the data to the local database that is managed by the application program. This attribute is OPTIONAL.

• **unitType** attribute SHOULD represent a unit of the property. This attribute is used to clarify the data unit shown in **unit** attribute in transaction massages. This attribute is OPTIONAL.

• **enumeration** attribute SHOULD represent the name of enumeration type when the property is extended property 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.

• **description** attribute SHOULD represent any description of the property. This attribute is OPTIONAL.

**4.13 ImplementEvent Element**

**ImplementEvent** element SHOULD represent any event definitions that the application program monitors on a particular property and detects 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 any users of information MAY request of 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.

```xml
<xsd:element name="ImplementEvent">
    <xsd:complexType>
        <xsd:attribute name="name" type="xsd:string" use="required"/>
        <xsd:attribute name="property" type="xsd:string" use="required"/>
        <xsd:attribute name="size" type="xsd:double"/>
        <xsd:attribute name="min" type="xsd:double"/>
        <xsd:attribute name="max" type="xsd:double"/>
        <xsd:attribute name="exclusive" type="xsd:boolean"/>
        <xsd:attribute name="cycle" type="xsd:duration"/>
        <xsd:attribute name="expire" type="xsd:datetime"/>
        <xsd:attribute name="description" type="xsd:string"/>
    </xsd:complexType>
</xsd:element>
```

• **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.

• **property** attribute SHOULD represent the name of the property that is monitored by the application program. The property name SHOULD be defined in the domain object. This attribute is REQUIRED.

• **size** attribute SHOULD represent unit size of data to recognize any changes. When the event is defined to detect changes, the application program SHOULD omit any changes of data value less than the unit size. This attribute is OPTIONAL.

• **min** attribute SHOULD represent a minimum constraint of the property value. This constraint shows that the data value of property SHOULD be greater than or equal to the value of this attribute. If this attribute is defined, then the constraint is added to the event condition. This attribute is OPTIONAL.
• `max` attribute SHOULD represent a maximum constraint of the property value. This constraint shows that the data value of property must be less than or equal to the value of this attribute. If this attribute is defined, then the constraint is added to the event condition. This attribute is OPTIONAL.

• `exclusive` attribute SHOULD represent whether the values defined in the maximum and minimum constraint are exclusive. If the value of this attribute is “true”, then the maximum and minimum conditions SHOULD be redefined without equal condition, that is, the value of property must be less (greater) than the value of the maximum (minimum) attribute. The default value of this attribute is “false”. 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.

• `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 property. 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

<table>
<thead>
<tr>
<th>Revision</th>
<th>Date</th>
<th>Editor</th>
<th>Changes Made</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>