psXML Specification

Draft, Oct, 2009

Document identifier: psxml-0.4

Chair(s):

Editor(s):

Abstract:

This document specifies an object model and XML format used for remotely monitoring and controlling device for electric services.

Status:
# Table of Contents

1 Introduction ......................................................................................................................... 4
   1.1 Purpose.......................................................................................................................... 4
   1.2 Glossary.......................................................................................................................... 4

2 psXML Overview and Architecture .................................................................................. 5
   2.1 Overview ......................................................................................................................... 5
   2.1.1 Requirements ............................................................................................................. 5
   2.2 Architecture .................................................................................................................. 6
   2.2.1 Device ....................................................................................................................... 6
   2.2.2 Device Gateway ......................................................................................................... 7
   2.2.3 Device Portal ............................................................................................................ 7
   2.2.4 Consumer Portal ....................................................................................................... 7
   2.2.5 Services ................................................................................................................... 7

3 psXML Modeling ................................................................................................................. 8
   3.1 Overview ....................................................................................................................... 8
   3.2 Object Model ................................................................................................................ 9
   3.2.1 Primary Data Type..................................................................................................... 9
   3.2.2 Reference Data Type ................................................................................................ 12
   3.2.3 Method or Function .................................................................................................. 13
   3.2.4 Derived Data Type .................................................................................................... 13
   3.3 Event Model ................................................................................................................. 16
   3.3.1 Alarm Event .............................................................................................................. 17
   3.3.2 Error Event .............................................................................................................. 18
   3.3.3 Device Handle(Control) Event .................................................................................. 20
   3.3.4 Acknowledge Event .................................................................................................. 21
   3.3.5 Remote Meter Reading Event ................................................................................. 22
   3.3.6 Sample for Historical Data ..................................................................................... 22

4 psXML-Security ................................................................................................................. 26
   4.1 Overview ....................................................................................................................... 26
   4.2 Authentication .............................................................................................................. 26
   4.3 Encryption ..................................................................................................................... 26
   4.3.1 HTTP ....................................................................................................................... 26
   4.3.2 WAP ......................................................................................................................... 27

5 psXML - Networking ......................................................................................................... 28
   5.1 Overview ....................................................................................................................... 28
   5.1.1 Get ............................................................................................................................. 28
   5.1.2 Post ............................................................................................................................ 29
### A. Implementation of psXML Architecture

5.2 Protocol
5.2.1 Consumer Portal to Client

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>A.1</td>
<td>Overview of the Portal System</td>
</tr>
<tr>
<td>A.2</td>
<td>Definition and Role of Each Component</td>
</tr>
<tr>
<td>A.2.1</td>
<td>Device</td>
</tr>
<tr>
<td>A.2.2</td>
<td>Device Gateway</td>
</tr>
<tr>
<td>A.2.3</td>
<td>Device Portal Server</td>
</tr>
<tr>
<td>A.2.3.1</td>
<td>DP Controller</td>
</tr>
<tr>
<td>A.2.3.2</td>
<td>Arbiter</td>
</tr>
<tr>
<td>A.2.3.3</td>
<td>Signal Receiver</td>
</tr>
<tr>
<td>A.2.3.4</td>
<td>Data Extractor</td>
</tr>
<tr>
<td>A.2.3.5</td>
<td>Data Format Standardization</td>
</tr>
<tr>
<td>A.2.3.6</td>
<td>Query Engine</td>
</tr>
<tr>
<td>A.2.4</td>
<td>Consumer Portal Server</td>
</tr>
<tr>
<td>A.2.4.1</td>
<td>CP Controller</td>
</tr>
<tr>
<td>A.2.4.2</td>
<td>Query Engine</td>
</tr>
<tr>
<td>A.2.4.3</td>
<td>On-Demand Signal Collector</td>
</tr>
<tr>
<td>A.2.4.4</td>
<td>psXML Generator</td>
</tr>
<tr>
<td>A.2.4.5</td>
<td>SOAP Server</td>
</tr>
<tr>
<td>A.2.4.6</td>
<td>Admin Module</td>
</tr>
<tr>
<td>A.2.5</td>
<td>Client</td>
</tr>
<tr>
<td>A.2.5.1</td>
<td>SOAP Client</td>
</tr>
</tbody>
</table>
1 Introduction

1.1 Purpose

psXML (power system eXtensible Markup Language) is a set of XML standards for accessing and controlling electric devices within customers’ premises for Energy Service Provider (ESP) to provide them with a variety of value-added services including demand management, energy information service, power quality monitoring, real time pricing and etc. Currently in the power industry, there are an increasing number of devices available and under development that can enable those services with the use of various emerging IT technologies. However, there is not yet a dominant standard for describing the data and controls on various devices for those services. XML is widely used internet-based language and due to its interoperability and expandability enable data to be structurally organized and unambiguously shared irrespective of platforms, programming languages and application programs. Thus XML can be used for representing a set of necessary information for those services. In this document, this specification describes psXML standard to monitor and control in-premises’ electric devices.

1.2 Glossary

- Device Gateway : DG
- Device Portal : DP
- Document Object Model : DOM
- Consumer Portal : CP
- Energy Service Provider : ESP
- Smart Cabinet Panel : SCP
- Simple Object Access Protocol : SOAP
- Universal Plug and Play : UPnP
- Wireless Application Protocol : WAP
2 psXML Overview and Architecture

2.1 Overview

In this section, the schematic of the Electrical IT System is presented and the definition and role of their logical components is briefly described. The Electrical IT System is defined as a system that can remotely monitor and control electric devices within customers’ premises to provide them a variety of value-added services. psXML can be viewed as the standard for information exchange within the Electrical IT System.

2.1.1 Requirements

2.1.1.1 Independence on Communications Protocols and Media

psXML should have communications protocol and media independence. As a result, while psXML is primarily defined on the basis of HTTP/Ethernet using XML, it can also be used on the lower level protocols including UDP, TCP, RS-232 and various communications media such as Ethernet, Wireless, RS-232, RS-485 and etc. The issues concerning characteristics of the protocol and media will be considered in the implementation.

2.1.1.2 Common Modeling

psXML should have the capability to remotely monitor and control electric devices by utilizing XML-based device and event model. psXML consists of the hierarchical structure of the object model which models electrical devices and possible events in the Electrical IT System.

2.1.1.3 Meta Information

psXML should model only the abstract meta information, but not the specific features of device. That is, psXML does not define the specific tag, but the generic meta tag related to electrical devices and events. Providing the meta information allows device monitoring and control to be possible in application programming and thus the actual system design and implementation to be facilitated.

2.1.1.4 Independence on the Operating Systems and Programming Languages

psXML should be implemented irrespective of the particular operating systems and/or programming languages. psXML is based on the current version of the XML specification and can be used in different development environments.

2.1.1.5 Internet based Technology

psXML should utilize commonly used internet-based technologies, allowing the various internet tools currently being used to be reused and the programmers to develop a wide range of useful systems based on the psXML through leveraging the internet-based technologies. For example, if HTTP as a basic protocol is used, then the existing web servers such as Apache, WebLogic and etc. may be utilized.
2.1.6 Expandability

psXML should have expandability to model newly developing devices with minimizing changes in its basic specifications or without any changes. Currently all the most electric devices do not have capability to handle psXML, but it can be assumed that those devices may have intelligence and deal with psXML, and newly developed devices may have intrinsic capability in the future.

2.1.7 Security and Reliability

psXML should have security and reliability related specifications that were not enough in the SOAP and/or UPnP specifications. psXML is a set of standards for remotely monitoring and controlling electric devices in the various facilities. In general the internet is public network in which anybody can freely access without the particular security measures. psXML should permit access only for relevant and authenticated operators and from reliable devices without communications packets including possibly confidential information being intercepted by any hackers malicious or not.

2.2 Architecture

The Electrical IT System is an infrastructure and an integrated system of consumer portal, device portal and ESP’s IT system that can together provide customers with a variety of value-added electric services. Consumer portal is an intelligent bilateral communication and service infrastructure component between ESP and customers. Device portal is an infrastructure component to collect information from and control devices in customer site. ESP’s IT system may be currently employed utility’s information systems. The schematic architecture of the Electrical IT System is represented in the following figure.

![Schematic Architecture of Electrical IT System](image)

Figure. The schematic architecture of the Electrical IT System

The Electrical IT System consists of various logical components: Device, Device Gateway, Device Portal, Consumer Portal and Services. As shown in above figure, psXML is used for communications between the logical components. The brief description of the definition and role of each logical component is as the following.

2.2.1 Device

Device is an object monitored and controlled in the Electrical IT system and is logically modeled by psXML. Each device may have various object types, methods and events in the hierarchical way.
There are many different types of devices available and under development including SCP (Smart Cabinet Panel), meter, HVAC (Heating, Ventilation, Air-Conditioning) equipment, lighting fixtures and etc.

2.2.2 Device Gateway

Device Gateway is a physical and/or logical component to aggregate information about and sends control commands to devices. Device and device gateway can communicate using psXML and/or legacy protocols. Because all the most devices currently available do not support psXML specification, it should be assumed that communications between device and device gateway are based on legacy protocols including RS-232, RS-485, PLC, Wireless and device gateway converts data from devices to psXML format and data from device portal to device specific data format. Device gateway may be a software based emulator that is implemented with device portal or it may be a dedicated embedded hardware system.

2.2.3 Device Portal

Device Portal is a physical platform on behalf of the particular customer and may be the local server system to monitor and control devices in customer site via multiple device gateways. Device portal may have many number of device gateways within customer site and externally connect to consumer portal. Device gateway and device portal communicates using psXML over TCP/IP. If device gateway seems to be not necessary, the function of the device gateway may be implemented in device portal.

2.2.4 Consumer Portal

Consumer Portal is a communications and services infrastructure component and may be a physical platform such as the central server system on behalf of ESP. Depending on the type of services offered by ESP, consumer portal appropriately collects information from and controls devices in customer site through multiple device portals connected to it. Device Portal and consumer Portal communicates using psXML or SOAP over TCP/IP.

2.2.5 Services

Services mean the value-added electric services offered to customers by ESP. These may include demand management, energy information service, power quality monitoring, real time pricing and etc. Consumer portal and services communicate using HTTP/Web.
3 psXML Modeling

3.1 Overview

psXML model all the elements related to electric devices as the hierarchical structure of an object model based on widely used XML. In order to remotely monitor and control electric devices within customers’ premises by using psXML, it is required that physical devices monitored and controlled be logically modeled and also events including alarms from devices and control commands for electric services be modeled. In general, device can be represented as the set of various information, but not simply one data and each element of the data set can be represented as an object. psXML uses these basic objects to define derived data type to model different type of devices and various operation environments and can also model newly developing devices by applying the object-oriented modeling method. In addition, psXML define event model to deal with various events associate with devices and services. In this specification, each devices and events are designed and modeled as different types of the standardized objects respectively so that psXML can still be used with minimum changes or without any changes although new devices are developed and innovative services are explored. Also using the hierarchical structure of an object can improve the inefficiency and complexity of object management associated with the number of objects and flexibly deal with future changes and upgrades of the Electrical IT system because the object can be easily added and removed in the corresponding hierarchy. psXML is designed to support various communications protocols and media and ensure the security and reliability such as authentication, encryption, access control and etc. The following table represents the brief description of the psXML components.

Table. psXML Components

<table>
<thead>
<tr>
<th>psXML Component</th>
<th>Details</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Device Model</td>
<td>Object Model</td>
<td>XML schema which defines the device characteristic and feature</td>
</tr>
<tr>
<td></td>
<td>- Primary Data type</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Reference Data type</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Method or Function</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Derived Data Type</td>
<td></td>
</tr>
<tr>
<td>Event Model</td>
<td>Events from devices and services</td>
<td>Event occurrence and its processing</td>
</tr>
<tr>
<td></td>
<td>- Alarm Event</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Error Event</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Control Event</td>
<td></td>
</tr>
<tr>
<td>Security</td>
<td>Authentication</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Encryption</td>
<td></td>
</tr>
<tr>
<td></td>
<td>WAP</td>
<td></td>
</tr>
<tr>
<td>Networking</td>
<td>Invocation Method(Get/Post ) Protocol(HTTP/WAP/SOAP)</td>
<td>Communication functions with HTTP and SOAP</td>
</tr>
</tbody>
</table>
3.2 Object Model

Object model means XML schema which defines the device characteristic and feature and is represented as the set of elements. Each element consists of the abstract data types such as the primary data type and the reference data type. As shown in the section 3.2.4 Derived Data Type, object can contain other objects. Also object may have some methods or functions related to device functions.

3.2.1 Primary Data Type

psXML represent modeling information using the following primary data type: bool, int, real, str, enum, list, datetime. The primary data types can be grouped as different types depending on the size of data type.

3.2.1.1 bool

The bool type represents a Boolean condition of either true or false. The literal value of a bool must be “true” or “false” and the literals “1” or “0” are not permitted.

<table>
<thead>
<tr>
<th>Type</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>bool</td>
<td>&quot;true&quot; or &quot;false&quot;</td>
</tr>
</tbody>
</table>

| Schema Definition | <element name="RightNow", type="bool", default="true"/> |
| Usage            | <RightNow>true</RightNow> |

3.2.1.2 int

The int type represents an integer number. It is classified into int8, int16, int32 and int64 depending on the size with its size property. The int8 type can be used to store one character. Also it can represent negative integer number with its signed property whose value is true. If the value of its properties is not specified, then the default values of the size and signed properties are “16” and “true” respectively.

<table>
<thead>
<tr>
<th>Type</th>
<th>Property</th>
<th>size (bits)</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>signed</td>
<td></td>
<td></td>
</tr>
<tr>
<td>int</td>
<td>true</td>
<td>8</td>
<td>-128(2^7) ~ 127(2^7-1)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>16</td>
<td>-32768(-2^15) ~ 32767(2^15-1) / default</td>
</tr>
<tr>
<td></td>
<td></td>
<td>32</td>
<td>-2147483648(-2^31) ~ 2147483647(2^31-1)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>64</td>
<td>-2^63 ~ 2^63-1</td>
</tr>
<tr>
<td></td>
<td>false</td>
<td>8</td>
<td>0 ~ 255(2^8-1)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>16</td>
<td>0 ~ 65535(2^16-1)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>32</td>
<td>0 ~ 4294967295(2^32-1)</td>
</tr>
</tbody>
</table>
3.2.1.3 real

The real type represents a floating point number. It is classified into real 32 and real 64 depending on the size with its size property and can represent the single precision and double precision formats with its fixed property. If the values of its properties are not specified, then the default values of the size and fixed properties are “32” and “true” respectively.

<table>
<thead>
<tr>
<th>Type</th>
<th>Property</th>
<th>Precision</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>real</td>
<td>fixed</td>
<td>32</td>
<td>7 decimal digits</td>
</tr>
<tr>
<td></td>
<td>true/false</td>
<td>64</td>
<td>16 decimal digits</td>
</tr>
</tbody>
</table>

3.2.1.4 str

The str type represents a string of Unicode characters. It is classified into UTF-8 and UTF-16 with its code property. For example, UTF-8 is a variable-length character encoding for Unicode and is backward compatible with ASCII which is the most representative code to express character. UTF-8 and UTF-16 are the standard encoding of XML documents. If the code property is not specified, then the default value of the code property is UTF-8.

<table>
<thead>
<tr>
<th>Type</th>
<th>Property / Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>str</td>
<td>UTF-8 / default</td>
</tr>
</tbody>
</table>
### 3.2.1.5 enum or list

The enum or list type is data type for storing a list of other data types. It can have order with its ordered property. Its intype property represents what data type the enum or list type can have as value. The default values of the enum or list are set with its val property.

<table>
<thead>
<tr>
<th>Type</th>
<th>Property</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>list/enum</td>
<td>ordered</td>
<td>true</td>
<td>A set with order</td>
</tr>
<tr>
<td></td>
<td></td>
<td>false</td>
<td>A set without order</td>
</tr>
<tr>
<td></td>
<td>intype</td>
<td>Int/real/str/ID</td>
<td>Can have int, real, str and ID type as an internal data type</td>
</tr>
<tr>
<td></td>
<td>val</td>
<td>...</td>
<td>Default values of the list</td>
</tr>
</tbody>
</table>

#### Schema Definition

```xml
<element name="DeviceName", type="str", code="UTF-8"/>
```

#### Usage

```xml
<DeviceName>Home SCP</DeviceName>
```

#### 3.2.1.6 datetime

The datetime type represents a date and time. Its default expression is YYYY-MM-DD hh:mm:ss. It can be differently expressed with its format property.

<table>
<thead>
<tr>
<th>Type</th>
<th>Property</th>
<th>Value</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>datetime</td>
<td>format</td>
<td>default</td>
<td>2007-01-01 01:01:01</td>
</tr>
<tr>
<td></td>
<td></td>
<td>YYYYMMDD</td>
<td>2007-01-01</td>
</tr>
<tr>
<td></td>
<td></td>
<td>YYYY&quot;년&quot;MM&quot;월&quot;DD&quot;일&quot;</td>
<td>2007 년 1 월 1 일</td>
</tr>
<tr>
<td></td>
<td></td>
<td>h:mm:ss AM/PM</td>
<td>1:01:01 AM</td>
</tr>
<tr>
<td></td>
<td></td>
<td>hh:mm:ss.0</td>
<td>01:01:01.1</td>
</tr>
</tbody>
</table>
### 3.2.2 Reference Data Type

The reference data type is a data type for referring to other psXML object. In general, the identity object and reference object can be included.

#### 3.2.2.1 ID(Identity)

The ID object is used to identify psXML object. It can be expressed in hierarchical structure with its Hierarchy property. Also it can be used duplicately with its Unique property. It basically uses a string of mixed characters and integers. The hierarchical structure is expressed by the literals “@” and “.”. The literal “@” distinguishes device and where device is. The literal “.” represents the hierarchical structure of the device itself or the location.

<table>
<thead>
<tr>
<th>Type</th>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID(Identify)</td>
<td>Name</td>
<td>Value</td>
</tr>
<tr>
<td></td>
<td>Unique</td>
<td>true</td>
</tr>
<tr>
<td></td>
<td></td>
<td>false</td>
</tr>
<tr>
<td></td>
<td>Hierarchy</td>
<td>true</td>
</tr>
<tr>
<td></td>
<td></td>
<td>false</td>
</tr>
</tbody>
</table>

#### Schema Definition

<table>
<thead>
<tr>
<th></th>
<th>&lt;element name=&quot;Install_1&quot;, type=&quot;datetime&quot;, format=&quot;default&quot;/&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>&lt;element name=&quot;Install_2&quot;, type=&quot;datetime&quot;, format=&quot;YYMMDD&quot;/&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;element name=&quot;Install_3&quot;, type=&quot;datetime&quot;, format=&quot;hh:mm:ss.0&quot;/&gt;</td>
</tr>
</tbody>
</table>

#### Usage

<table>
<thead>
<tr>
<th></th>
<th>&lt;Install_1&gt;2007-01-01 01:01:01&lt;/Install_1&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>&lt;Install_2&gt;070101&lt;/Install_2&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;Install_3&gt;01:01:01&lt;/Install_3&gt;</td>
</tr>
</tbody>
</table>

#### 3.2.2.2 ref(Reference)

The ref object is used to create a document reference to other psXML object. It must always specify a href property.

**ex:** `<ref href="........"/>`
3.2.3 Method or Function

In general, devices may have various functions from simple to complex. This function of device is referred to as method in this specification. Method may be composed of its name, arguments and returns similar to common programming languages.

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Primary Data Type, ID, Derived Data Type</td>
<td>name</td>
<td>String</td>
<td>Equivalent to parameters in programming languages.</td>
</tr>
<tr>
<td></td>
<td>default</td>
<td>Variable</td>
<td>Can specify default value depending on the data type.</td>
</tr>
</tbody>
</table>

Schema Definition:

```xml
<argument name="SilentMode" type="int" size="16" signed="false" default="0"/>
<argument name="RightNow" type="bool" default="true"/>
```

Usage:

```xml
<SilentMode>1</SilentMode>
<RightNow>0</RightNow>
```

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Primary Data Type, ID, Derived Data Type</td>
<td>name</td>
<td>String</td>
<td>Equivalent to returned parameters in programming languages.</td>
</tr>
</tbody>
</table>

Schema Definition:

```xml
<returns name="Result" type="bool"/>
```

Usage:

```xml
<Result>false</Result>
```

3.2.4 Derived Data Type

The derived data type uses the above mentioned data types to represent complex psXML objects such as device information, user information, energy consumption information and etc.
3.2.4.1 DeviceInfo

The DeviceInfo object is a derived data type to represent various device related information. The following figure shows the structure of the DeviceInfo object.

<table>
<thead>
<tr>
<th>Name</th>
<th>Basic Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DeviceID</td>
<td>ID</td>
<td>Device identifier</td>
</tr>
<tr>
<td>User</td>
<td>UserInfo</td>
<td>User who uses device</td>
</tr>
<tr>
<td>Position</td>
<td>PositionInfo</td>
<td>Where device is</td>
</tr>
<tr>
<td>DataOfInstallation</td>
<td>DateTime</td>
<td>Installation date and time of device</td>
</tr>
<tr>
<td>State</td>
<td>Enum</td>
<td>A set of device status</td>
</tr>
<tr>
<td>Energy</td>
<td>EnergyInfo</td>
<td>Energy usage of device</td>
</tr>
<tr>
<td>TurnOn</td>
<td>Methods</td>
<td>Device specific functions</td>
</tr>
</tbody>
</table>

```xml
<complexType name="DeviceInfo">
  <sequence>
    <element name="DeviceID" type="ID" Unique="true" Hierarchy="true"/>
    <element name="User" type="UserInfo"/>
    <element name="Position" type="PositionInfo"/>
    <element name="DataOfInstallation" type="dateTime" format="YYYYMMDD"/>
    <element name="State" type="DeviceState"/>
    <element name="Energy" type="EnergyInfo"/>
  </sequence>
</complexType>
```
3.2.4.2 UserInfo

The UserInfo object is a derived data type to represent user related information. The following figure shows the structure of the UserInfo object.

<table>
<thead>
<tr>
<th>Name</th>
<th>Basic Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UserName</td>
<td>str</td>
<td>User’s name</td>
</tr>
<tr>
<td>UserID</td>
<td>ID</td>
<td>User’s identifier</td>
</tr>
<tr>
<td>UserPass</td>
<td>str</td>
<td>User’s password</td>
</tr>
<tr>
<td>EnergyRate</td>
<td>real</td>
<td>Energy cost</td>
</tr>
<tr>
<td>possessedDevices</td>
<td>list</td>
<td>A set of devices which user owns</td>
</tr>
</tbody>
</table>
3.3 Event Model

The psXML event model defined in this specification complies with the DOM (Document Object Model) model of events in W3C recommendation and accordingly handles event occurrence and its processing. Event can be thought of as what occurred asynchronously associated with element in an psXML document and occur when the attribute of the element has wrong value, when error occurs in any element in an psXML document, when alarms should be generated, when the set of actions for offering various services progress and etc. According to DOM model of events in W3C recommendation, when an event occurs in the target element the observer element detects its occurrence and responds to an event. In general an event can be responded to at any element in the psXML document by causing an action, stopping the event and/or cancelling the default action for the event.

<table>
<thead>
<tr>
<th>element</th>
<th>Attribute Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>event</td>
<td>ID</td>
<td>Event identifier</td>
</tr>
<tr>
<td></td>
<td>Type</td>
<td>Event type</td>
</tr>
<tr>
<td></td>
<td>targetid</td>
<td>ID of object where the event occurred</td>
</tr>
<tr>
<td></td>
<td>[observer]</td>
<td>Object which detects the event. If this attribute is not specified, the default value is device gateway.</td>
</tr>
<tr>
<td></td>
<td>[handler]</td>
<td>Object which defines the action that should be performed if the observer detects the event. [User/Administrator/Device/DeviceGateway/DevicePortal]</td>
</tr>
<tr>
<td></td>
<td>[defaultAction]</td>
<td>Default action for the event that should be performed. [ignore/cancel/perform]</td>
</tr>
<tr>
<td></td>
<td>[propagate]</td>
<td>Specifying whether the event is allowed to continue on its path. [stop/continue]</td>
</tr>
</tbody>
</table>

In psXML, some objects defined in the previous sections can include event object to properly do their jobs. For example, the alarm event can notifies unexpectedly occurring events including power off and over...
power of the relevant administrators or users. psXML can also additionally define any other possible
events not defined in this specification. The message format of an event transferred in the Electrical IT
System is very similar to the response message format defined in the chapter 5 psXML – Networking.

### 3.3.1 Alarm Event

The alarm event is generated from device(or possible device gateway) and dispatched to the relevant
administrator or user. Since the alarm event may deal with the problem of great urgency, its message of
an alarm event should be composed of minimum number of necessary information.

<table>
<thead>
<tr>
<th>event:</th>
<th>Attribute</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>eventID</td>
<td></td>
<td>Event identifier</td>
</tr>
<tr>
<td>targetID</td>
<td>ID</td>
<td></td>
<td>Device identifier</td>
</tr>
<tr>
<td>observer</td>
<td>ID</td>
<td></td>
<td>Device gateway identifier</td>
</tr>
<tr>
<td>handler</td>
<td>List</td>
<td></td>
<td>User/Administrator</td>
</tr>
<tr>
<td>defaultAction</td>
<td>List</td>
<td></td>
<td>perform</td>
</tr>
<tr>
<td>propagate</td>
<td>List</td>
<td></td>
<td>continue</td>
</tr>
</tbody>
</table>

Examples of over power and power off events:

```xml
<event type="Alarm" id="e_20070810" targetid="monitor123", observer="GW123", handler="Administrator", defaultAction="perform", propagation="continue">
  <arguments>
    <argument name="state">OverPower</argument>
  </arguments>
</event>

<event type="Alarm" id="e_20070810" targetid="monitor123", observer="GW123", handler="Administrator", defaultAction="perform", propagation="continue">
  <arguments>
    <argument name="state">PowerOff</argument>
  </arguments>
</event>
```

An example of response message for over power event:

```
Response Message for Alarm Event

HTTP/1.1 200 OK
Content-Type : text/xml; charset = utf-8
Date: Thu, 04 Apr 2002 11:34:01 GMT
Content-Length : 587

```

```xml
<event type="Alarm" id="e_20070810" targetid="monitor123", observer="GW123", handler="Administrator", defaultAction="perform", propagation="continue">
  <arguments>
    <argument name="state">OverPower</argument>
  </arguments>
</event>
```

```xml
<event type="Alarm" id="e_20070810" targetid="monitor123", observer="GW123", handler="Administrator", defaultAction="perform", propagation="continue">
  <arguments>
    <argument name="state">PowerOff</argument>
  </arguments>
</event>
```
### 3.3.2 Error Event

The error event is generated to indicate an error when the response to requested message is not properly performed and is dispatched to the relevant administrator or user. It does not deal with an error from device. Its actual meanings are context dependent.

<table>
<thead>
<tr>
<th>event:Type</th>
<th>Attribute</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error</td>
<td>ID</td>
<td>eventID</td>
<td>Event identifier</td>
</tr>
<tr>
<td></td>
<td>targetID</td>
<td>ID</td>
<td>NULL</td>
</tr>
<tr>
<td></td>
<td>[observer]</td>
<td>ID</td>
<td>Object where the error occurred</td>
</tr>
<tr>
<td></td>
<td>[handler]</td>
<td>List</td>
<td>User/Administrator</td>
</tr>
<tr>
<td></td>
<td>[defaultAction]</td>
<td>List</td>
<td>perform</td>
</tr>
<tr>
<td></td>
<td>[propagate]</td>
<td>List</td>
<td>Continue</td>
</tr>
</tbody>
</table>

```xml
<event type="error" id="e_20070810" targetid="NULL", observer="CP123", handler="Administrator", defaultAction="perform", propagation="continue">
    <arguments>
        <argument name="method">DeviceView</argument>
        <argument name="code">400</argument>
    </arguments>
</event>
```
An example of response message for “Bad Request” error:

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset = utf-8
Date: Thu, 04 Apr 2002 11:34:01 GMT
Content-Length: 587
ETag: "0a7ccac50cbc11:1aad"
Server: localhost

<?xml version="1.0" encoding="utf-8">
<psXML version="1.0" type="event">
  <event type="error" id="e_20070810" targetid="NULL", observer="CP123", handler="Administrator", defaultAction="perform", propagation="continue">
    <arguments>
      <argument name="method">DeviceView</argument>
      <argument name="code">400</argument> <!-- Bad Request -->
    </arguments>
  </event>
</psXML>
```

The error code is given for cases in which the client seems to have errored or the server is aware that it has errored or is not capable of performing the request. The following table represents error codes that are a subset of HTTP error codes.

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Client Error</strong></td>
<td></td>
</tr>
<tr>
<td>400(Bad Request)</td>
<td>The request contains bad syntax or cannot be fulfilled.</td>
</tr>
<tr>
<td>401(Unauthorized)</td>
<td>The request was a legal request, but it contains wrong authentication information. Thus the server is refusing to respond to it.</td>
</tr>
<tr>
<td>402(Payment Required)</td>
<td>Some form of payment is necessary to get any response from the server.</td>
</tr>
<tr>
<td>403(Forbidden)</td>
<td>The request was a legal request, but the server is refusing to respond to it regardless of authentication.</td>
</tr>
<tr>
<td>404(Not Found)</td>
<td>The requested resource could not be found.</td>
</tr>
<tr>
<td>405(Method Not Allowed)</td>
<td>A request was made of a resource using a request method not supported by that resource.</td>
</tr>
<tr>
<td>406(Not Acceptable)</td>
<td>The URI specified by the client is not acceptable.</td>
</tr>
<tr>
<td>408(Request Timeout)</td>
<td>Client failed to continue the request.</td>
</tr>
<tr>
<td><strong>Server Error</strong></td>
<td></td>
</tr>
<tr>
<td>500(Internal Server Error)</td>
<td>A part of the server stopped or the error occurred in the configuration.</td>
</tr>
</tbody>
</table>
### 3.3.3 Device Handle (Control) Event

The device handle event is intended to control device or change device status. It can be used to send control commands including temperature setting, temperature change, and time setting to device. In the Electrical IT System, consumer portal receives the control request from the administrator or user via the control event and send it to device through device portal.

<table>
<thead>
<tr>
<th>event: type</th>
<th>Attribute</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Handle</td>
<td>ID</td>
<td>eventID</td>
<td>Event identifier</td>
</tr>
<tr>
<td></td>
<td>targetID</td>
<td>ID</td>
<td>Device identifier</td>
</tr>
<tr>
<td></td>
<td>[observer]</td>
<td>ID</td>
<td>Device gateway identifier</td>
</tr>
<tr>
<td></td>
<td>[handler]</td>
<td>ID</td>
<td>Administrator</td>
</tr>
<tr>
<td></td>
<td>[defaultAction]</td>
<td>List</td>
<td>perform</td>
</tr>
<tr>
<td></td>
<td>[propagate]</td>
<td>List</td>
<td>stop</td>
</tr>
</tbody>
</table>

The representative status values of device changed by the device handle event are temperature, time, price and etc. In psXML, any other possible control events to set or change device's status can be additionally defined.

An example of the control event for setting temperature:

```xml
<event type="handle" id="e_20070810" targetid="monitor123", observer="CP123", handler="Administrator", defaultAction="perform", propagation="stop">
  <arguments>
    <argument name="method">SetDegree</argument>
    <argument name="degree">40</argument>
  </arguments>
</event>
```

```xml
<event type="handle" id="e_20070810" targetid="monitor123", observer="CP123", handler="Administrator", defaultAction="perform", propagation="stop">
  <arguments>
    <argument name="method">ModifyDegree</argument>
    <argument name="degree">40</argument>
  </arguments>
</event>
```
### 3.3.4 Acknowledge Event

The acknowledgement event is used to send the response result to the requestor.

<table>
<thead>
<tr>
<th>event: type</th>
<th>Attribute</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Event_Ack</td>
<td>DeviceID</td>
<td>DeviceID</td>
<td>Device identifier</td>
</tr>
<tr>
<td></td>
<td>eventID</td>
<td>EventID</td>
<td>Event identifier</td>
</tr>
<tr>
<td>Action</td>
<td>List</td>
<td>Event response result</td>
<td></td>
</tr>
<tr>
<td></td>
<td>EventTime</td>
<td>DateTime</td>
<td>Event response duration (Format: default)</td>
</tr>
<tr>
<td>Result</td>
<td></td>
<td>Response result</td>
<td></td>
</tr>
</tbody>
</table>

An example of response message for changing temperature:

```
Response(Acknowledgement)
POST /path/to/some/DeviceInfo.cs/ HTTP/1.1
Host: localhost
Content-Type : application/x-www-form-urlencoded
Content-Length: 43
ETag: "0a7ccac50cbc11:1aad"
<?xml version="1.0" encoding="utf-8"?>
<psXML version="1.0" type="Event_Ack"
<DeviceID>monitor123</DeviceID>
<EventID>e_20070810</EventID>
```
<Action>Accept</Action>
<EventTime>2007-08-10 12:00:00</EventTime>
<Result>
<ModifyDegree>27</ModifyDegree>
<Difference>9</Difference>
</Result>
</psXML>

3.3.5 Remote Meter Reading Event
In general, devices send their status information to consumer portal through device portal in the previously configured period. The remote meter reading event allows the user or administrator to remotely request metering data. Consumer portal receives meter reading request from the user or administrator and send it to the meter through device portal.

<table>
<thead>
<tr>
<th>event:type</th>
<th>Type</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>reading</td>
<td>ID</td>
<td>eventID</td>
<td>Event identifier</td>
</tr>
<tr>
<td></td>
<td>targetID</td>
<td>ID</td>
<td>Device identifier</td>
</tr>
<tr>
<td></td>
<td>[observer]</td>
<td>ID</td>
<td>Device portal identifier</td>
</tr>
<tr>
<td></td>
<td>[handler]</td>
<td>ID</td>
<td>User/Administrator</td>
</tr>
<tr>
<td></td>
<td>[defaultAction]</td>
<td>List</td>
<td>perform</td>
</tr>
<tr>
<td></td>
<td>[propagate]</td>
<td>List</td>
<td>stop</td>
</tr>
</tbody>
</table>

<event type="reading" id="e_20070810" targetid="monitor123", observer="DP123", handler="Administrator", defaultAction="perform", propagation="stop">
<arguments>NULL</arguments>
</event>

3.3.6 Sample for Historical Data

3.3.6.1 Aperiodic Request
In case the user or administrator send an aperiodic request message to consumer portal, consumer portal first retrieve the requested information from the database and then send the request to device portal if those information is not available.
An example of the aperiodic request message to request information about monitor123 device and their current power consumption managed by DP123:

Request

POST /path/to/some/DeviceInfo.cs/ HTTP/1.1
Host: localhost
Content-Type: application/x-www-form-urlencoded
Content-Length: 43
ETag: "0a7ccac50cbc11:1aad"

```xml
<?xml version="1.0" encoding="utf-8">
<psXML version="1.0" type="event">
  <event type="reading" id="e_20070810" targetid="monitor123", observer="DP123", handler="Administrator", defaultAction="perform", propagation="stop">
    <arguments>NULL</arguments>
  </event>
</psXML>
```
An example of response message for the aperiodic request to get various information about monitor123 device that is sent to the user:

Response

HTTP/1.1 200 OK
Content-Type: text/xml; charset = utf-8
Date: Thu, 04 Apr 2002 11:34:01 GMT
Content-Length: 587
ETag: "0a7ccac50cbc11:1aad"
Server: localhost
<?xml version="1.0" encoding="utf-8">
<psXML version="1.0" type="DeviceInfo">
  <DeviceID>Monitor123</DeviceID>
  <User>
    <UserName>Hong Gil Dong</UserName>
    <UserID>HGD123</UserID>
    <UserPass>******</UserPass>
    <Address>Seoul</Address>
  </User>
  <Position>
    <Floor>1</Floor>
    <RoomNumber>A101</RoomNumber>
  </Position>
  <DateOfInstallation>20070710</DateOfInstallation>
  <State>TurnOff</State>
  <Energy>
    <DataOfMeasurement>20070711:120101</DataOfMeasurement>
    <MountOfUsing>100</MountOfUsing>
  </Energy>
</psXML>

3.3.6.2 Periodic Request

The following is a response message for the periodic request to send power consumption data every fifteen minutes to consumer portal.
Response

HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Date: Thu, 04 Apr 2002 11:34:01 GMT
Content-Length: 587
ETag: "0a7ccac50c0c11aadd"
Server: localhost

<?xml version="1.0" encoding="utf-8">
<psXML version="1.0" type="PeriodicData">
  <DeviceID>Monitor123</DeviceID>
  <Energy>
    <StartTime>20070711:120101</StartTime>
    <EndTime>20070711:120115</EndTime>
    <MountOfUsing>10</MountOfUsing>
    <MountOfUsing>11</MountOfUsing>
    <MountOfUsing>9</MountOfUsing>
    <MountOfUsing>11</MountOfUsing>
    <MountOfUsing>10</MountOfUsing>
    <MountOfUsing>15</MountOfUsing>
    <MountOfUsing>12</MountOfUsing>
    <MountOfUsing>1</MountOfUsing>
    <MountOfUsing>5</MountOfUsing>
    <MountOfUsing>12</MountOfUsing>
    <MountOfUsing>13</MountOfUsing>
    <MountOfUsing>9</MountOfUsing>
    <MountOfUsing>10</MountOfUsing>
    <MountOfUsing>10</MountOfUsing>
  </Energy>
</psXML>
4 psXML-Security

4.1 Overview

psXML provides security functions such as authentication, encryption, permission and user management. Authentication is verifying a user or client is who he says he is. Encryption is protecting psXML documents from hackers malicious or not. Permission is checking a user's permissions before granting access to psXML object. User management is dealing with user accounts and user access levels.

4.2 Authentication

Kerberos is a computer network authentication protocol and allows individuals to communicate over a non-secure network to prove their identity to one another in a secure manner. Typically, both the user and the server verify each other's identity using a trusted third party (the third authentication server). After authentication, the user and the server can encrypt and decrypt using the public cryptographic key. AES (Advanced Encryption Standard) encryption algorithm (Rijndael) is widely used, but ARIA algorithm developed by NSRI (National Security Research Institute) in Korea can be used. Kerberos uses the encryption key in the session or with the expiration time. Since there are no methods for the malicious hackers to know the next encryption key, Kerberos protocol messages are protected against eavesdropping and replay.

4.3 Encryption

4.3.1 HTTP

4.3.1.1 SSL

SSL (Secure Sockets Layer) is cryptographic protocol originally developed by Netscape that provide secure communications on the internet for such things as web browsing, e-mail, instant messaging and other data transfers. SSL protocol is typically used for authenticating the server and the client and allows applications to communicate across a network in a way designed to prevent eavesdropping, tampering and message forgery. SSL provides endpoint authentication and communications privacy over the internet using cryptography. In general, only the server is authenticated while the client remains unauthenticated meaning the client can be sure with whom it is securely communicating.

4.3.1.2 S-HTTP

S-HTTP (Secure Hypertext Transfer Protocol) is an alternative protocol for encrypting web communications carried over HTTP. While SSL is designed to provide secure communication between two computers, S-HTTP is designed to send individual messages securely. Typically, S-HTTP supports MKMM (Multiple Key Management Mechanisms) and a variety of cryptographic algorithms and provides
confidentiality, authentication, integrity and non-repudiation. S-HTTP cannot be used without specially licensed software.

4.3.1.3 S/MIME

S/MIME (Secure Multipurpose Internet Mail Extension) is a standard originally developed by RSA Data Security Inc. for public key encryption and signing of e-mail encapsulated in MIME. S/MIME provides authentication message integrity and non-repudiation and privacy and data security for electronic messaging applications.

4.3.2 WAP

4.3.2.1 WTLS (Wireless Transport Layer Security)

WTLS (Wireless Transport Layer Security) is a security protocol, part of the WAP (Wireless Application Protocol). WTLS is based on SSL/TLS and is optimized for low bandwidth mobile environments. WTLS provides confidentiality, integrity and authentication via secure communication channel. WTLS does not provide non-repudiation, but can deal with this by using WML (Wireless Markup Language) script.
5 psXML - Networking

5.1 Overview

In order to remotely monitor and control devices in the Electrical IT System, it is necessary to access information and services over network. In this specification, communications using HTTP and/or SOAP will be generally used. All network access will be achieved through the use of the request and response messages.

The request means that the user or administrator sends the message to consumer portal. The request message may be simply requesting for information or invoke special events or methods. The request message must contain the following information:

- Request method (type of request) : GET or POST
- Relative URI of resource requested
- HTTP Version
- Virtual host name

The response means that consumer portal send the message to the user or administrator. The response message may contain stored information from consumer portal’s database or processed information in consumer portal after receiving the raw information from device portal. Depending on the request message, the response message may be simply sending available information or returning results of special event or method invocation.

5.1.1 Get

This is the most common service used to request a representation of the specified resource. The data is included in the URL character string of URLEncoded format.

<table>
<thead>
<tr>
<th>Request</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET /path/to/some/DeviceInfo.cs/DeviceView?DeviceID=Monitor123 HTTP/1.1</td>
</tr>
<tr>
<td>Host: localhost</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Response</th>
</tr>
</thead>
</table>
HTTP/1.1 200 OK
Content-Type: text/xml; charset = utf-8
Date: Thu, 04 Apr 2002 11:34:01 GMT
Content-Length: 587
ETag: "0a7ccac50cbc11:1aad"
Server: localhost

<?xml version="1.0" encoding="utf-8">
<psXML version="1.0" type="DeviceInfo">
  <DeviceID>Monitor123</DeviceID>
  <User>
    <UserName>Hong Gil Dong</UserName>
    <UserID>HGD123</UserID>
    <UserPass>******</UserPass>
    <Address>Seoul</Address>
    <User>
      <Position>
        <Floor>1</Floor>
        <RoomNumber>A101</RoomNumber>
      </Position>
      <DateOfInstallation>20070710</DateOfInstallation>
      <State>TurnOff</State>
    <Energy>
      <DataOfMeasurement>20070711:120101</DataOfMeasurement>
      <MountOfUsing>100</MountOfUsing>
    </Energy>
  </User>
</psXML>

406 5.1.2 Post

Post submits data to be processed. Unlike Get, the data is included in the body of the request message.

Request
POST /path/to/some/DeviceInfo.cs/ HTTP/1.1
Host: localhost
Content-Type: application/x-www-form-urlencoded
Content-Length: 43
ETag: "0a7ccac50cbc11:1aad"
method = DeviceViewer
DeviceID = Monitor123

Response
HTTP/1.1 200 OK
Content-Type : text/xml; charset = utf-8
Date: Thu, 04 Apr 2002 11:34:01 GMT
Content-Length : 449
ETag: "0a7ccac50cbc11:1aad"
Server : localhost
<?xml version="1.0" encoding="utf-8">
<psXML version="1.0" type="DeviceInfo">
  <DeviceID>Monitor123</DeviceID>
  <User>
    <UserName>Hong Gil Dong</UserName>
    <UserID>HGD123</UserID>
    <UserPass>******</UserPass>
    <Address>Seoul</Address>
  </User>
  <Position>
    <Floor>1</Floor>
    <RoomNumber>A101</RoomNumber>
  </Position>
  <DateOfInstallation>20070710</DateOfInstallation>
  <State>TurnOff</State>
  <Energy>
    <DataOfMeasurement>20070711:120101</DataOfMeasurement>
    <MountOfUsing>100</MountOfUsing>
  </Energy>
</psXML>

5.2 Protocol

5.2.1 Consumer Portal to Client

5.2.1.1 HTTP(Hypertext Transfer Protocol)

HTTP is a communication protocol for the transfer of information on the intranet and the World Wide Web. HTTP is based on TCP, but can be implemented on top of any other protocols on the internet. HTTP uses request/response between a client and a server. A client is the user and the server is the web site. An HTTP client initiates a request, establishes a TCP connection to a particular port on a host and send request message to a server. An HTTP server listening on that port waits for the client to send a request message. Upon receiving the request, the server sends back a status line and a message of its own, the body of which is perhaps the requested file, an error message or some other information.
### 5.2.1.2 WAP (Wireless Application Protocol)

WAP is an open international standard for applications that use wireless communication by WAP forum.

- **WAP Architecture**

  WAP uses split connection scheme. When the wired and wireless network is used simultaneously, the wired and wireless network uses underlying protocol such as TCP/IP and HTTP and protocols suitable for mobile environments such as GSM and CDMA. Gateway is necessary to link dynamically split connection.

  WAP gateway converts encoded request message from mobile client to HTTP request message, converts WML or WMLScript to binary format and converts HTML contents to WML contents for linking with the internet.

![Figure. Typical WAP Architecture](image1)

![Figure. WAP Stack Architecture](image2)
5.2.1.3 SOAP (Simple Object Access Protocol)

SOAP is used to define the messaging framework for exchanging XML-based messages on the internet, normally using HTTP/HTTPS. It is required that the messaging framework is easy and simple to develop, independent on the platforms such as operating systems or programming languages or suitable for the distributed computing environments. SOAP meets the requirements of the messaging framework and also is a protocol for the most basic and minimum transfer of information.

1) Basic structure of SOAP message

SOAP message consists of envelope, header and body in the following.

1. Component of SOAP message
   - Envelope: the top element of the XML document representing the message
   - Header: the container of all the conditional attributes of the message
   - Body: the container for mandatory information intended for the ultimate recipient of the message

2) SOAP message transmission process

SOAP message is fundamentally one-way transmission from a sender to a receiver. Each node (Server, Client, and Intermediary) that receives SOAP message performs the following actions in the order listed below.

1. Current node that receives SOAP message determines all the possible roles.
2. Current node checks and processes header. In case that the value of the mustUnderstand attribute is true, if current node cannot understand header, send fault message and stop.
3. If current node is “ultimate SOAP receiver” in message, current node processes SOAP body.
4. If current node is “SOAP intermediary” and relay is necessary, current node relays SOAP message.

3) Protocol Binding
SOAP is a specification that defines the structure of the message transferred. In fact, SOAP message can be transmitted using various transfer protocols including HTTP, SMTP, FTP and MIME.

① SOAP over HTTP

SOAP message is transmitted using HTTP protocol.
- Request using POST : SOAP message is included in the HTTP Body
- Request using GET
- Response : SOAP message is included in the HTTP Response Body

② SOAP over WAP

SOAP message is transmitted using WAP protocol for mobile device

4) SOAP attributes

① "role" attribute
② "mustUnderstand" attribute
③ "relay" attribute

5) SOAP example

<table>
<thead>
<tr>
<th>Request</th>
</tr>
</thead>
<tbody>
<tr>
<td>POST /path/to/some/DeviceInfo.cs/ HTTP/1.1</td>
</tr>
<tr>
<td>Host: localhost</td>
</tr>
<tr>
<td>Content-Type: application/x-www-form-urlencoded</td>
</tr>
<tr>
<td>Content-Length: 262</td>
</tr>
<tr>
<td>ETag: &quot;0a7ccac50cbc11:1aad&quot;</td>
</tr>
<tr>
<td>SOAPAction: &quot;localhost/DeviceViewer&quot;</td>
</tr>
<tr>
<td>&lt;?xml version=&quot;1.0&quot; encoding=&quot;utf-8&quot;?&gt;</td>
</tr>
<tr>
<td>&lt;soap:Envelope xmlns:xsi=&quot;localhost/psXMLSchema-instance&quot; xmlns:xsd=&quot;localhost/psXML -Schema&quot; xmlns:soap=&quot;localhost/soap/envelope&quot;/&gt;</td>
</tr>
<tr>
<td><a href="">soap:body</a></td>
</tr>
<tr>
<td><a href="">Method:DeviceViewer</a></td>
</tr>
<tr>
<td>&lt;DeviceID&gt;Monitor123&lt;/DeviceID&gt;</td>
</tr>
<tr>
<td>&lt;/Method:DeviceViewer&gt;</td>
</tr>
<tr>
<td>&lt;/soap:body&gt;</td>
</tr>
<tr>
<td>&lt;/soap:Envelope&gt;</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Response</th>
</tr>
</thead>
</table>
HTTP/1.1 200 OK
Content-Type: text/xml; charset = utf-8
Date: Thu, 04 Apr 2002 11:34:01 GMT
Content-Length: 449
ETag: "0a7ccac50cbc11:1aad"
Server: localhost

<?xml version="1.0" encoding="utf-8">
<soap:Envelope xmlns:xsi="localhost/psXMLSchema-instance" xmlns:xsd="localhost/psXML-Schema"
xmlns:soap="localhost/soap/envelope/>
<soap:Body>
<DeviceViewerResponse xmlns="localhost">
<DeviceViewerResult type="DeviceInfo">
<DeviceID>Monitor123</DeviceID>
<User>
<UserName>Hong Gil Dong</UserName>
<UserID>HGD123</UserID>
<UserPass>******</UserPass>
<Address>Seoul</Address>
<User>
<Position>
<Floor>1</Floor>
<RoomNumber>A101</RoomNumber>
</Position>
<DateOfInstallation>20070710</DateOfInstallation>
<State>TurnOff</State>
<Energy>
<DataOfMeasurement>20070711:120101</DataOfMeasurement>
<MountOfUsing>100</MountOfUsing>
</Energy>
</DeviceViewerResult>
</DeviceViewerResponse>
</soap:Body>
</soap:Envelope>
A. Implementation of psXML Architecture

psXML can be viewed as the standard for information exchange within the Electrical IT System. In this appendix, the customer portal system (hereafter portal system) which implements the schematic architecture of the Electrical IT System defined in the chapter 2 is described.

A.1 Overview of the Portal System

The overall structure of the portal system is represented in the following figure. As previously described, the portal system consists of device, device gateway, device portal server, consumer portal server and client. Both device portal server in customer site and consumer portal server on behalf of ESP play a significant role in offering a variety of value-added electric services. Client provides GUI(Graphic User Interface) for the administrator and/or customers to make informed decisions about electric services.

Consumer portal server and client communicate using psXML over HTTP(Web). Device and device gateway communicates via legacy protocol including RS-232, RS-485, PLC and Wireless(ZigBee) thus enabling various information to be collected and control actions to be performed while in the future psXML itself can be directly used if device supports it. Device gateway and device portal server communicates using TCP/IP assuming that device gateway in the portal system is just a data relay without the capability of converting data from device to psXML format. Device portal server and consumer portal server communicates via psXML or SOAP over TCP/IP. In device portal server or consumer portal server, data is transferred from one module to the other and data is generally stored to and/or retrieved from database by using SQL query. If necessary or when the peculiar problems or events occur, components of the portal system can communicate each other without accessing to database.
**Figure. The overall structure of the portal system**

**Table The components of the Portal System**

<table>
<thead>
<tr>
<th>Component</th>
<th>Module</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Device</td>
<td>Object monitored and controlled</td>
<td></td>
</tr>
<tr>
<td>Device Gateway</td>
<td>Simple data relay between device and device portal server</td>
<td></td>
</tr>
<tr>
<td>Device Portal Server</td>
<td>DP Controller</td>
<td>Managing various actions in device portal server</td>
</tr>
<tr>
<td></td>
<td>Signal Receiver</td>
<td>Gathers data from device</td>
</tr>
<tr>
<td></td>
<td>Arbitter</td>
<td>Controls device</td>
</tr>
<tr>
<td></td>
<td>Signal Collector</td>
<td>Collects data from Signal Receiver</td>
</tr>
<tr>
<td></td>
<td>Data Extractor</td>
<td>Extract necessary data among collected data</td>
</tr>
<tr>
<td></td>
<td>Data Format Standardization</td>
<td>Standardize extracted data in compliance with psXML syntax</td>
</tr>
<tr>
<td></td>
<td>Query Engine</td>
<td>Writing query for requesting information to database</td>
</tr>
<tr>
<td></td>
<td>Database</td>
<td>Storing information about device</td>
</tr>
<tr>
<td>Consumer Portal Server</td>
<td>CP Controller</td>
<td>Managing various actions in consumer portal server</td>
</tr>
<tr>
<td></td>
<td>On-Demand Signal Collector</td>
<td>Sending device control command to arbiter and collecting information in real time</td>
</tr>
<tr>
<td></td>
<td>psXML Generator</td>
<td>Converting data extracted from database into psXML format</td>
</tr>
<tr>
<td></td>
<td>Query Engine</td>
<td>Writing query for requesting information to database</td>
</tr>
<tr>
<td></td>
<td>Database</td>
<td>Storing information about device and client</td>
</tr>
<tr>
<td></td>
<td>SOAP Server</td>
<td>Transferring psXML document to the user</td>
</tr>
<tr>
<td></td>
<td>HTTP/WAP</td>
<td>Communication protocol</td>
</tr>
</tbody>
</table>
A.2 Definition and Role of Each Component

The architecture of the portal system is represented in the following figure. Details for each component are described in the following.

Figure The architecture of the Portal System

A.2.1 Device

Device is an object monitored and controlled in the portal system and logically modeled by psXML. There are many different types of devices available and under development including SCP, meter, HVAC equipment, lighting fixtures and etc.

- Functions / Roles
  - Source of electric power information
  - Sending information to and receiving control commands from device gateway
  - Performing their functions according to control commands from device gateway
A.2.2 Device Gateway

Device gateway is a simple data relay for data communications between device and device portal server while in principle it should be a physical and/or logical component to aggregate information about and send control commands to devices. Device gateway may have the capability of converting legacy data from devices to psXML format and vice versa.

A.2.3 Device Portal Server

Device portal server is responsible for collecting data from device and sending control commands to device.

- Requirements
  - Device portal server should have component modules of the hierarchical structure.
  - Device portal server should be local system in the distributed network and manage a number of devices

- Functions / Roles
  - Collecting periodically data from device gateway
  - Standardizing data received in compliance with psXML syntax
  - Storing data to database
  - Sending periodically data to consumer portal server
  - Receiving request from consumer portal server and analyzing it
  - Transferring control commands from consumer portal server to device gateway

- Modules
  - DP Controller: Managing and controlling device portal server
  - Signal Receiver: Gathering data from device gateway.
  - Data Extractor: Analyzing bit-stream data and extracting necessary data
  - Data Format Standardization: Standardizing extracted data from data extractor in compliance with psXML syntax
  - Arbiter: Handling real time control command from consumer portal server
  - Query Engine: Requesting data to DBMS and receiving it
  - Database: Storing data about managed devices

A.2.3.1 DP Controller

DP controller is responsible for managing one individual device portal server. It analyzes on demand message requested from consumer portal server, determine whether that message is event or information
request to database in the server and then depending on the type of message directs corresponding
module to do the task. Although there is no request, it transfers periodically information collected from
device to consumer portal server.

**A.2.3.2 Arbiter**

Arbiter is responsible for requesting information through device gateway when the user requests
information in real time. The user may be the client or system administrator.

**A.2.3.3 Signal Receiver**

Signal receiver is responsible for gathering information from device gateway and sending it to data
extractor through communication interface and alarm interface. Also when the user requests information
in real time, it can request information to relevant device. Typically, collected information may have many
different formats depending on communication protocols.

**A.2.3.4 Data Extractor**

Data extractor is responsible for extracting necessary information including device ID from data collected
in signal receiver and sending it to data format standardization.

**A.2.3.5 Data Format Standardization**

Data format standardization is responsible for standardizing information from data extractor in a unifying
format and storing it to database. Unifying format means converting to object defined in psXML. Also it
transfers information collected in real time by the user's request to consumer portal server.

**A.2.3.6 Query Engine**

Query engine writes query for requesting information to database according to consumer portal server's
and/or DP controller's request and send requested information from database to consumer portal server
and/or DP controller.

**A.2.4 Consumer Portal Server**

Consumer portal server is responsible for providing services for the user. The user may be customer or
administrator. It includes additional admin module for overall system management.

- Requirements

  - Consumer portal server should have component modules of the hierarchical structure.
  - Consumer portal server should be operated in a centralized manner on behalf of ESP.
  - Consumer portal servers should be operated in a decentralized manner if necessary. In this
case, database synchronization must be considered.
Consumer portal server should communicate with the user via HTTP in terms of the compatibility and scalability.

Consumer portal server should support WAP for the user in mobile environments.

- Functions / Roles
  - Managing and storing the user’s information
  - Receiving request from and transferring corresponding information to the user
  - Converting requested information to psXML format
  - Sending periodic and aperiodic data request message to device portal server
  - Receiving periodic and aperiodic data from device portal server
  - Receiving device control command from the user and transferring it to device portal server
  - Standardizing received data in psXML format
  - Storing received and changed or upgraded data to database

- Modules
  - CP Controller: Managing and controlling consumer portal server
  - Query Engine: Requesting data to DBMS and receiving it
  - On-Demand Signal Collector: Transferring device control information
  - psXML Generator: Generating psXML document
  - SOAP Server: Transferring psXML document to the user

A.2.4.1 CP Controller
CP controller is responsible for managing and controlling consumer portal server. It analyzes request from the user and administrator and determine whether that request is event or data request in the server. In case of event, it transfers that request to on-demand signal collector and send event results to the user and administrator and event related data to database. In case of data request, it transfers that request to query engine and send requested data to the user and administrator.

A.2.4.2 Query Engine
Query engine writes query for requesting information to database according to CP controller’s request. If the requested information does not exist in database, it send immediately data request message to CP controller.

A.2.4.3 On-Demand Signal Collector
On-demand signal collector is responsible for transferring real time request to get information not stored in database about device to DP controller.
A.2.4.4 psXML Generator

psXML generator is responsible for converting data to psXML format to exchange standardized information irrespective of the client environments.

A.2.4.5 SOAP Server

SOAP server generates psXML based messages for providing services with the client and sends them to the user via HTTP or WAP. SOAP Server is composed of service application, SOAP request and SOAP response. Service application requests information to query engine and generate SOAP message by adding header information suitable for service in psXML document. SOAP request receives service request from the client and transfers it to service application. SOAP response transfers SOAP message to the client using HTTP or WAP.

A.2.4.6 Admin Module

Admin module is responsible for managing overall portal system and includes the following components.

- Requirements
  - Admin module should monitor device status in real time.
  - The administrator’s request should have higher priority than the user’s one.
  - Admin module should have simple and easy-to-use interface for managing system and database.
  - Admin module should provide security.

- Functions / Roles
  - Transferring device control command according to device status information
  - Managing database when adding or removing device
  - Managing services for the user
  - Retrieving information from databases in device portal server and/or consumer portal server and requesting information in real time

- Components and Their Roles
  - Device Monitor : Checking device status in real time
  - Device Manager : Controlling device in real time
  - Device Gateway Manager : Managing communication protocols between device and device gateway and controlling device portal server
  - Application Manager : Managing value-added services
  - Database Manager : Backing up database, generating, deleting and changing table and optimizing database
Emergency Controller: Analyzing the risk element of device and notifying it of the user and/or controlling device

System Management Interface: Administrator interface

A.2.5 Client

The client means customer who benefits from a variety of value-added services.

- Requirements
  - Client should provide easy-to-user interface suitable for various services.
  - Client should support various security modules.

- Functions / Roles
  - Requesting value-added service
  - Providing graphical information for the user
  - Transferring the user’s device control command to consumer portal server

A.2.5.1 SOAP Client

SOAP client transfers data request message to consumer portal server and send response message to the client. SOAP client is composed of client processor, SOAP request, SOAP response and service interface. Client processor generates SOAP message from the client’s request and extracts psXML document from received SOAP message. SOAP request transfers SOAP message to consumer portal server. SOAP response receives SOAP message and send it to the client. Service interface provides various services for the user and gets input from the user.