Revision History

<table>
<thead>
<tr>
<th>Rev.</th>
<th>Date</th>
<th>By</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.01</td>
<td>2-Dec-05</td>
<td>TJB</td>
<td>Initial Draft</td>
</tr>
<tr>
<td>0.90</td>
<td>1-Mar-06</td>
<td>TJB</td>
<td>Major revision of contents, for current Toolkit design.</td>
</tr>
<tr>
<td>0.91</td>
<td>10-Apr-06</td>
<td>TJB</td>
<td>Renamed from Orion_Data_Exchange.doc</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Revised per discussions with Orion team regarding data structure</td>
</tr>
<tr>
<td>0.92</td>
<td>19-Sept-06</td>
<td>CJE</td>
<td>Replaced “Nucleus” references with “Orion”.</td>
</tr>
<tr>
<td>1.00</td>
<td>20-Sep-06</td>
<td>TJB</td>
<td>Updated with latest design decisions.</td>
</tr>
<tr>
<td>1.01</td>
<td>27-Sep-06</td>
<td>TJB</td>
<td>Renamed the special six standard nodes</td>
</tr>
<tr>
<td>1.02</td>
<td>9-Dec-06</td>
<td>TJB</td>
<td>Added lists of implemented WS methods and service options</td>
</tr>
<tr>
<td>1.03</td>
<td>18-Dec-06</td>
<td>TJB</td>
<td>Removed iDeviceFirmwareVer, renamed iAssetIdTag to iAssetID</td>
</tr>
<tr>
<td>1.04</td>
<td>8-Mar-07</td>
<td>TJB</td>
<td>Adjusted some of the Device node requirements.</td>
</tr>
</tbody>
</table>

Table of Contents

1 INTRODUCTION ......................................................................................................................................... 3
  1.1 Definition of “Device” .................................................................................................................. 3

2 OVERVIEW OF WEB SERVICES INTERFACE IN ORION / PCD SYSTEMS .................................................. 3
  2.1 Orion / PCD System Context Diagram .......................................................................................... 4
  2.2 Use Cases: Orion Software to PCM Server ..................................................................................... 5
    2.2.1 Data Request and Response for Orion ...................................................................................... 5
    2.2.2 Publish / Subscribe Scenario for Orion .................................................................................... 6
  2.3 The BACnetWS+ WSDLs ..................................................................................................................... 7
  2.4 Attributes of a BACnetWS+ Node ..................................................................................................... 7
  2.5 Implemented BACnetWS+ Methods .................................................................................................. 7
  2.6 Implemented BACnetWS+ Service Options ...................................................................................... 8

3 STRUCTURE OF NODES FOR POWER CHAIN DEVICES ........................................................................ 9
  3.1 Structure of a General Power Chain Device .................................................................................. 9
  3.2 Structure of the Root Device Node ................................................................................................ 10
  3.3 Structure of a Network Device Node ............................................................................................. 10
  3.4 Hierarchy of Power Chain Device Nodes (Device tree) ................................................................... 10
  3.5 “Functional” nodes in a Power Chain Device tree .......................................................................... 10

4 BACNET PATH NAMING CONVENTION ............................................................................................. 11

5 USE CASES .......................................................................................................................................... 11
  5.1 Process to Learn all BACnetWS+ nodes of a PCD/PCM ................................................................. 11
1 Introduction

Two of the most important design aspects of the Power Chain Toolkit, as externally observed, are its Structure of Data and means of Data Communication. The purpose of this document is to describe that interface between a Toolkit-enabled Power Chain Device (PCD) and an Orion (i.e., PowerXpert Software) web services client. It will describe this interface in terms of:

- Overview of the transfer of PCD data by means of web services (using SOAP encoding).
- The terms of the Orion/BAcnet WSDL that is the contract between a PCD system and a web services client (like Orion).
- The structure of a single data point (“node”) in this framework, with important examples:
  - A simple String value
  - A user-writable String value
  - A Collection node
  - A metered data point, with a corresponding Alarm node
- The structure of nodes that will be provided by a Power Chain Device.
- The structure of nodes that will be provided by the root Device in a System.
- Important and illustrative Use Cases for Orion Data:
  - A normal getValue() exchange
  - A Publish/Subscribe scenario
  - The LearnNodes process for a PCD system, which builds both a device tree and a table of nodes for each device

What this document will NOT describe are the details of the data internal to the Power Chain Device or Microserver, though that information is complementary to what is described here. See the following documents for internal data details:

- PCD_CoreDataAPI.doc
- PCD_Database_And_Configuration_Services.doc
- PCD_Parameter_Data.doc
- PCD_MPL_EDS.doc

1.1 Definition of “Device”

The most overloaded term in this system is "Device". Consequently, it is very important to be clear about what is meant in the terms that use the word "Device". Please refer to section 1.3 Definitions, of the PCD_Toolkit_FRS.doc, for an explanation of the various forms of “Device”. This document will conform to the definitions in PCD_Toolkit_FRS.doc; in particular, it will use the terms

- PCenD: Power Chain enabled Device
- PCD: Power Chain Device
- PCM: Power Chain Microserver
- Device Subnetwork

2 Overview of Web Services Interface in Orion / PCD Systems

In the Orion / PowerChain Device (PCD) system, data is exchanged between the Orion software client and the PCD “server” primarily via web services. The choice of web services implementation was made intentionally broadly in the standards; we narrowed those choices to the most common implementation:

- Transport: TCP/IP
- Encoding: SOAP “literal” encoding (i.e., not RPC)
- Data types: Structures as described in the interface documents (IBACnetWSCore.cs, etc.)

There are many fine sources that describe web services, WSDLs, SOAP, XML, and TCP/IP; we will not recreate that effort here, but focus on the implementation specifics for the Orion / PCD systems.

The central reference for the data exchanged and the BACnet-WS and Eaton extension methods used to exchange that data is the document

- BACnetWSPlus.chm

A supplemental text which describes some of the rules for nodes is

- NodeRules.doc

Both of these documents are found in the toolkit’s Subversion database (at https://svn.eaton.com/svn/PCM-tk/trunk/documents/bacnet-ws ) as well as Orion’s VSS server (at \Chvippa01\archive\VSS\Nucleus ).
2.1 Orion / PCD System Context Diagram

The data exchanges between the server and client are illustrated in the following overview diagram:

A short description of these processes and components:

<table>
<thead>
<tr>
<th>Location</th>
<th>Module</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Common to both:</td>
<td>HTTP</td>
<td>HTML wrappers for the SOAP messages, sent using TCP transport over the Internet or Intranet network.</td>
</tr>
<tr>
<td></td>
<td>XML</td>
<td>Internal format of the SOAP messages</td>
</tr>
<tr>
<td>Orion Software</td>
<td>SOAP Client code</td>
<td>Built on the Microsoft .NET platform, which provides a SOAP client implementation</td>
</tr>
<tr>
<td></td>
<td>NetworkEntities code</td>
<td>Module of the Orion software which provides the web services interface and which constructs the data structures used by the rest of the Orion system.</td>
</tr>
<tr>
<td></td>
<td>Hierarchical Device Tree</td>
<td>A useful internal Orion data structure, which builds a tree of devices, starting with the PCM or PCD system server as the root, then showing the Subnetworks and Devices as children.</td>
</tr>
<tr>
<td></td>
<td>Table of Device Nodes</td>
<td>Each device in the system has a hashtable assigned to it; each table holds all the BACnetWS+ child nodes for this device</td>
</tr>
<tr>
<td>SOAP Server</td>
<td></td>
<td>A small web services host to receive the Published Data Updates sent by the PCD</td>
</tr>
<tr>
<td></td>
<td>Interface to Network</td>
<td>The interface which receives the Published Data and transfers it back into the Orion software.</td>
</tr>
<tr>
<td></td>
<td>Entities</td>
<td></td>
</tr>
<tr>
<td>PCD Toolkit Device</td>
<td>HTTP Server, with</td>
<td>The PCM or PCD will use a web server to host the SOAP messaging; this will usually be the Lighttpd HTTP server with the “soapsvr” module to manage the BACnetWS+</td>
</tr>
<tr>
<td></td>
<td>FastCGI and gSOAP</td>
<td></td>
</tr>
</tbody>
</table>
The functions which will translate the web service calls into Core Data Store calls in order to access the data.

The central process of the Toolkit services which manages the getting and setting of data in the system. Both device handlers and the other processes get or store their data in the CDS. This is explained in more detail in the PCD_CoreDataAPI.doc.

This is actually 3 entities, which provide information for filling out CDS nodes or BACnetWS+ attributes:
- The Electronic Data Sheet (EDS) describing the data and access methods for each device model
- The Master Parameter List (MPL) describing the most common device parameters and providing default values for their attributes
- The SQLite database library provides storage and lookup of persistent data.

The process that manages sending (“publishing”) changed or updated data to the hosts who have requested it (“subscribed”).

The functions for building and sending the SOAP packets with the published data, which take the form of web service client requests to be sent to a remote web server.

The lower level process(es) that communicate with attached devices (PCenDs). These commonly poll the devices and put the resulting data in the CDS.

External devices (PCenDs) attached to the PCD or PCM via some Device Subnetwork (INCOM, QCPort, Modbus/RTU, RS-232).

The first use cases to show are for normal traffic in the Orion system, between Orion monitoring software and a PCM server. The figure above may be referred to for these use cases.

Data Request and Response for Orion

This models a polled data request from the Orion Software, where it requests a specific data item. This normally only happens during the LearnNodes process, because afterwards Orion relies primarily on published data. It assumes that Orion already knows the path to this data item, and that the desired data already resides in the Core Data storage.

1. Orion’s NetworkEntities code makes a web service method getValue() call to its SOAP client, eg for the AssetID string for Trip Unit 6. The path to the data item (an argument for the call) looks like
   /ttyS01/03/iAssetID:Value
2. The SOAP client builds the request for the data item as an XML-encoded SOAP request message, and sends it to the PCM server. The target address is something like:
   http://166.99.9.128/BACnetWSPlus/publisher.cgi
3. The PCM Server receives the SOAP request message via its HTTP server and FastCGI support, and invokes the soapsvr process with its gSOAP handlers for the getValue() method.
4. The gSOAP handlers call the web service “implementation function” to supply the data item at its given path.
5. The web service implementation function determines the path into the Core Data for the requested data, and calls the Core Data API function to get the requested data item. The path to the sample item in the Core Data structure looks amazingly familiar:
   /ttyS01/03/iAssetID/Value
6. The web service implementation function then returns the requested data item to the gSOAP handler, which builds a SOAP response message.
7. The gSOAP response message is sent back to the requesting host.
8. The Orion system’s listening SOAP client process receives this SOAP response message, deserializes the Value string, and returns it to the calling Orion NetworkEntities code.
2.2.2 Publish / Subscribe Scenario for Orion
This models the usage of web services’ data Publish / Subscribe (Pub/Sub) methods for the Orion Software and a PCD. This is
the main means that Orion will use to obtain its data from the PCD.

Subscribe phase I - Initiation:
This models the start of a Subscription operations for a PCD.
1. The Orion client software chooses a PCD to Subscribe to, and starts a createSubscription() web service call for the PCD. (Note
that createSubscription() is an Eaton extension to the standard BACnet-WS methods.)
2. The SOAP client proxy on the Orion side builds the SOAP message and sends it to the targeted PCD server.
3. The PCD’s soapsvr module (running on the Lighttpd web server) receives the message, deserializes it, and sets the subscription
information in nodes of the toolkit’s “private” /Software tree; this will result in the wspublisher agent being notified of the
subscription.
4. The soapsvr module then sends an empty string response (to signal “success”) to the Orion client through its SOAP interface.
5. The wspublisher agent recognizes that the subscriber (Orion) is subscribing to receive all logged events, alarms, and trend data.
6. The wspublisher publishes all the trend data and/or alarms/events since the timestamp given in the createSubscription() call.

Subscribe phase II – Subscription Update:
This models Subscribing to specific real-time data in the PCD. This might be in response to an operator choosing to view a
particular set of the PCD data for a limited time (e.g., on a web view).
1. The Orion client software chooses data item(s) to be Subscribed to, and makes an updateSubscription() web service call to the
PCD. (Note that updateSubscription() is another Eaton extension to the standard BACnet-WS methods.)
2. The SOAP client proxy on the Orion side builds the SOAP message and sends it to the targeted PCD server.
3. The PCD’s soapsvr module (running on the Lighttpd web server) receives the message, deserializes it, and sets the real-time data
item information in nodes of the toolkit’s “private” /Software tree; this will result in the wspublisher agent being notified of the
change in the list of subscribed nodes (here, additions to the list).
4. The soapsvr gets the present value of each item to be subscribed to, noting any errors in the array of responses it is building. It
then sends the array of success or failure responses to the Orion client through its SOAP interface.
5. The wspublisher agent recognizes the change, and determines which data in the CDS is to be subscribed to, and registers to
receive updates on these points.

Publish phase:
This models a web services data Publish update for data which the Orion Software has already Subscribed to.
1. As a data item is updated in the Core Data area, the wspublisher agent is notified of the change and checks to see if any process
has subscribed to it. Finding that at least one Orion host has subscribed to this item, it builds the trend data object which will be
sent to the subscriber.
2. When ready, the wspublisher agent sends the data object to the gSOAP interface by calling the onUpdate() function. The
onUpdate() handler builds an onUpdate SOAP “request” message; note that this message takes the form of a client “request” to a
web services server.
3. The gSOAP onUpdate message is sent to the server at the URL provided by Orion when it initiated the subscription.
4. The Orion system’s listening SOAP server receives this SOAP “request” message, deserializes the message contents, and signals
the Orion client software.
5. The Orion Client then gets the data and processes it (usually in a callback function, which then has to signal the larger Orion
system of the data update).
2.3 The BACnetWS+ WSDLs

A WSDL (Web Services Descriptive Language) is the contract between a web services source (like a PCD) and a web services client (like Orion). This is an agreement that describes the messages to be sent, the methods used to send them, and the means to be used to send the messages. (Please refer to a good source on WSDLs for deep background on their construction and use.)

An industry-standard BACnet-WS WSDL does not actually exist, at this time. However, its presumed contents and form are set forth in the Appendix N supplement to the BACnet spec, which has been released as an ASHRAE standard (filename 20061006_135_2004_Addendum_C.pdf).

For the PCDs and Orion, the WSDL contract has been divided into 3-4 distinct areas, each with its own Interface description file. The following table lists these interface files that are used to define the contents of the overall BACnet-WS+ WSDL. All are under the namespace “Eaton.BACnetWSPlus”.

<table>
<thead>
<tr>
<th>Interface</th>
<th>Description</th>
<th>Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>IBACnetWSCore.cs</td>
<td>Incorporates all the methods described in Appendix N</td>
<td>getValue(), etc per BACnet-WS spec</td>
</tr>
<tr>
<td>IBACnetWSPlusPublisher.cs</td>
<td>Eaton extensions for Pub/Sub (the Subscription methods)</td>
<td>createSubscription(), updateSubscription(), cancelSubscription(), etc</td>
</tr>
<tr>
<td>IBACnetWSPlusTest.cs</td>
<td>Eaton extensions for Test/Internal-use methods</td>
<td>simulCreateNode(), simulSetValue(), simulReset(), etc</td>
</tr>
<tr>
<td>IBACnetWSPlusSubscriber.cs</td>
<td>The Eaton extension method for publishing data back to the Subscriber</td>
<td>onUpdate()</td>
</tr>
</tbody>
</table>

By agreement with the Orion software group, the PCM / PCD server implements most, but not all, of the BACnet-WS features; the Attribute differences are detailed in the file PCD_Parameter_Data.doc, and the following section describe which web service methods and service options are implemented in the toolkit code.

2.4 Attributes of a BACnetWS+ Node

The full set of possible Attributes for a BACnetWS+ node from a PCD or PCM is described in file PCD_Parameter_Data.doc; this set contains both standard BACnet-WS and Eaton extension Attributes.

2.5 Implemented BACnetWS+ Methods

The following methods are implemented in the Toolkit:

BACnet-WS standard methods implemented:
- getValue
- getValues
- getRelativeValues
- getArray
- getArrayRange
- getArraySize
- setValue
- setValues
- getDefaultLocale
- getSupportedLocales

BACnet-WS standard methods not implemented:
- getHistoryPeriodic
Eaton Pub/Sub web service methods implemented:

- createSubscription
- updateSubscriptionRealTimeNodes
- cancelSubscription
- acknowledgeAlarm

Eaton Test and internal-use web service methods implemented:

- CreateNode
- SetLeaf
- SetLeaves
- Reset
- RemoveNode
- getSubscribedNodes

### 2.6 Implemented BACnetWS+ Service Options

The following service options are implemented in the Toolkit:

**BACnet-WS standard service options implemented:** <PLANNED for v1.1>

- readback <PLANNED for v1.1>
- errorString <PLANNED for v1.1>
- errorPrefix <PLANNED for v1.1>
- locale <PLANNED for v1.1>
- writeSingleLocale <PLANNED for v1.1>
- canonical <PLANNED for v1.1>
- precision <PLANNED for v1.1>
- noEmptyArrays <PLANNED for v1.1>

**Eaton extension to service options:**

- persist – set to true if the setValue or setLeaf operation should make the item persistent in the CDS.

While, strictly speaking, the set of service options does not allow for extensions, it is safe to add this extension, because the implementation is still compliant with the BACnet-WS spec (e.g., for third-party BACnet-WS browsers); they simply do not know about this service option and therefore do not use it.
3 Structure of nodes for Power Chain Devices

It will be very useful for monitoring software to know what to expect for the structure of a device’s nodes. A Power Chain (toolkit-enabled) Device (PCD or PCenD) will be a branch of the BACnetWS+ tree starting with a node whose NodeType is one of the following:

- Device
- Network
- System
- Equipment

The following sections describe the characteristics of a general Device node, root Device node, network Device node, and of the Device hierarchy.

3.1 Structure of a general Power Chain Device

All of the Device NodeTypes (Device, Network, System, or Equipment), including the root node, must have the following child nodes (located directly as children of the device node, not in separate Collections):

- Configuration/Informational nodes:
  - VendorName (R/O String)
  - ModelName (R/O String – Name & Eaton Catalog #)

- Entity State nodes – statuses which vary dynamically and indicate the overall status of the device on several fronts (see PCD_SNMP_Guidelines.doc for fuller description of these)
  - sEntAdminState (Multistate: Unknown(1); Disabled(2); TurnedOff(3); Enabled(4))
  - sEntOperatingState (Multistate: Unknown(1); Disabled(2); TurnedOff(3); Enabled(4); UnderTest(5))
  - sEntReadinessState (Multistate: Unknown(1); TurnedOff(2); Idle(3); Active(4); Busy(5); Unavailable(6))
  - sEntAlarmState (Multistate: None(0); Unknown(1); Disarmed(2), CriticalAlarm(3); CautionaryAlarm(4); Unacknowledged(5))
  - sEntStandbyState (Multistate: Unknown(1); InParallel(2); HotStandby(3); OffLine(4); InService(5))

- User-writable nodes – may have default values, but should be available to the user to set to text useful in his/her environment:
  - iDisplayName (R/W String – maps to the device node’s DisplayName attribute, with default value of the ModelName if EDS doesn’t specify another value)
  - iAssetID (R/W String)
  - iLocation (R/W string)
  - iWhoToContact(R/W string)

In addition, the device node itself must implement the following attributes (which are otherwise optional):

- Description
- DisplayName (mapped from iDisplayName), as mentioned above
- .Quality is required for Device nodes (not Equipment, System, or Network) so it can be used to conveniently indicate when the entire Device has become unavailable.

The following attribute is required if the Device or Network’s node list can change:

- HasDynamicChildren

An important example of the use of HasDynamicChildren is for a Network with attached devices that may be added, removed, or lose communication with the parent device, and thus come and go from the Children node list. Software clients, both internal and external, can assume that they never need to rescan the child node list of a device which doesn’t have the HasDynamicChildren attribute; conversely, if a Device or Network node has this attribute, such clients should subscribe to or poll for changes in the list of Children nodes.

The following are optional configuration/informational nodes that should be implemented if the underlying information is available from the device or its EDS:

- HardwareVersion (R/O String)
- SoftwareVersion (R/O String, in Major.Minor.Path.Build form (eg, 1.2.0.34) to check for firmware upgrades)
- SerialNumber (R/O String)
- iFieldReplaceableUnit (Boolean)
- iMfgDate (R/O String)
3.2 Structure of the root Device node
The root node will usually have a NodeType = Device, though it may sometimes be “System”. Since it is a Device type node, it will contain the general Device nodes as shown in the last section, but in addition will have the following nodes found only in the root node.

A root node **must have** the following child nodes:
- All required device nodes shown in the previous section.
- /.sysinfo - Collection containing standard nodes as per the BACnet-WS spec:
  - .vendor-name
  - .model-name
  - .software-version
  - .standard-version
  - vendor-url (For UPnP; set to “http://www.eaton.com” or other appropriate value)
- /Software
  - OS, Users, etc as shown in PCD_Database_And_Configuration_Services.doc

The root node will have data Points as children as well; in the case of a PCD like the Power Expert Meter, there will be a great number of Points as children of the root.

3.3 Structure of a Network Device node
A Network Device node is a node for a Device Subnetwork or communication port (NodeType = Network). In addition to the general Device nodes as shown above, it must contain the following **required** nodes:
- iConnStyle (String – INCOM, Modbus/RTU, RS-232, QCPort, etc)
- iDeviceBaudRate (int32 – bits/second – may be writable)

3.4 Hierarchy of Power Chain Device nodes (Device tree)
The Device nodes of a PCM are arranged with a hierarchy that commonly looks something like this:
- Root Device – NodeType=Device -- “/”
  - Network – NodeType=Network -- /ttyS0
    - Address – NodeType=Device -- /ttyS0/03 (eg, DisplayName = Trip Unit 3)
    - Address – NodeType=Device -- /ttyS0/05 (eg, FP-5000)
  - Network – NodeType=Network -- /ttyS1
    - Address – NodeType=Device -- /ttyS1/01 (eg, PowerXpert)
      - Network – NodeType=Network -- /ttyS1/03/COM2
        - Address – NodeType=Device -- /ttyS1/03/COM2/01 (eg, Breaker)

Note that a device in the tree can have a device network and devices under that (eg, PowerXpert Meter).

The recommended hierarchy has a chain of device-type nodes (Device, Network, System, or Equipment), without intervening Collection nodes. All systems should follow this approach to avoid issues with device tree discovery.

3.5 “Functional” nodes in a Power Chain Device tree
Certain functional groups, like breakers and UPS output load groups, are given the NodeType of “Functional”. A Functional node is a collection of Points; it is similar to a Device node but does not have the requirement for the list of standard nodes.

Functional nodes are particularly useful when there are “N” identical or nearly identical instances of a function, as for one of 42 Breakers in a panel.

A Functional node **must** implement the following attribute:
- DisplayName
4 BACnet path naming convention
Per the BACnet-WS standard, the root’s name is “” (null string), though we usually treat it as “/”.
Most of the subsequent levels have fixed names as given above. The one variable area is at the Device and Network node levels. It is recommended that the following rules be employed there:
   1. Networks are given their common name (ttyS01, COM1, Port0, etc)
   2. The node name (BACnet path) of a Device should be its address on the Network, eg /ttyS0/01.
   3. If the Network only can connect to a single Device (e.g., ttyS0 is an RS-232 port connected to a UPS), then the Network node can be subsumed under the Device’s node information, since it will provide little value and normally have no real “owner” to update its contents.
   4. A GUID was previously used as a device’s node name, but this proved humanly awkward in practice.

5 Use Cases
A Use Case is provided to illustrate use of the PCD data.

5.1 Process to Learn all BACnetWS+ nodes of a PCD/PCM
This Use Case will describe the process for client software to parse through the BACnetWS+ data and construct
   • A Device Tree – a hierarchal tree of the Devices in the System, starting with the System node as the root
   • OrionWsNodes tables – each Device in the system will have its own hashtable or DataTable of the nodes contained within the device. The BACnet path is the key for each entry, and the value is a class or hashtable containing all the attributes and their values for the entry’s node.

The Device Tree is an artificial construct created from the BACnetWS+ data, but it is a very useful construct that is easy for users to understand and navigate. This is the approach used in Orion’s NetworkEntities code and in the PxTkImgMaker test tool.

1. Start by requesting the NodeType and Attributes of the root node of a PCD or PCM.
2. If you get a valid response, start the Device Tree and create its root node, and create an OrionWsNodes table for the root.
3. Request the values of all of its Attributes (screening out any ones you might not be interested in), and store them in the OrionWsNodes table.
4. If (while) there are any Children, for each, request its NodeType and Attributes
5. If any of these Children has the NodeType of a device (Device, Network, System, or Equipment), create a new node in the Device Tree for it, and an OrionWsNodes table, and begin to recursively request its OrionWsNodes data.
6. Otherwise, loop back to step 3, so that you recursively process all of the nodes and devices.