
# ISY
# Developer's Manual

**Web Services SDK and REST Interface**
**for INSTEON/UPB**
**Based on firmware 3.3.10**

TABLE OF CONTENTS

# Revision History

| Date/Firmware | Type | Change | Description |
|---|---|---|---|
| 2012/12/10<br>**3.3.6** | DOC | FIX | Authentication header |
| 2012/10/31<br>**3.3.4** | WSDL | MODIFY | Support for LEAK SENSOR 2852-222<br>Support for Open/Close Door Sensor 2843-222<br>1_fam.xml |
| 2012/09/04<br>**3.3.1** | WSDL | NEW | Added Device Linker Events |
| 2012/07/13 | WSDL | MODIFY | Added version to <security> element in <configuration> |
| 2012/06/26 | DOC | MODIFY | Added section 7 for Programs |
| 2012/06/26 | WSDL | FIX | Node Revised event is RV and not NR<br>Added more info for Program Status event |
| 2012/06/21 | WSDL | MODIFY | Added <baseDriver> to <configuration><br>Added CLIEMD (Climate Energy Saving Mode) as a control (insteon.xsd) |
| 2012/05/17 | WSDL | DOC | Subscribe using Web Services<br>Added <driver_timestamp> to <configuration> |
| 2012/05/15 | WSDL | MODIFY | Moved Zigbee objects to zigbee.xsd |
| 2012/04/28 | WSDL | MODIFY | Support for Support for 2475DA1<br>1_fam.xml |
| 2012/03/19<br>**3.2.1** | WSDL | NEW | Support for FanLinc, TempLinc, 2474DWH, 2477S |
| 2012/03/15<br>**3.2.0** | REST | FIX | PP53 … member->members |
| | WSDL | NEW | Err Control |
| 2012/01/05<br>**3.1.17** | WSDL | MODIFY | Added CC, CV, PPW, PF controls for ISY994<br>Removed CPW control<br>Most electricity events are now only handled in ISY994<br>Added 7_fam.xml (UDI)<br>Added 8_fam.xml (Brultech) |
| 2011/12/02<br>**3.1.14** | DOC | MODIFY | /rest/security moved to ISY-WS-SDK-ELK.docx/pdf |
| 2011/10/19<br>**3.1.12** | WSDL | NEW | /rest/zb/nodes/[euid]/ep … retrieves active endpoint list<br>*** ELK ***<br>21090 – ELK Module/Feature<br>elkobjs.xsd … defines ELK Objects/Classes/Events<br>udielkws1.wsdl … defines ELK Web Services<br>ISY-WS-SDK-ELK.docx/pdf … WSDK for ELK<br>5.19 … Events: ELK Control<br>6.16 REST: ELK REST<br>udiobjs.xsd:<br>o Addition of SupportedSecuritySystem and SecuritySystemType classes<br>o Inclusion of SecuritySystemType in the configuration element |
| | WSDL | MODIFY | 1_fam.xml: - SwitchLinc Relay and SwitchLinc Relay Timer were reported incorrectly |
| | LOG | NEW | ERROR_DEADLOCK_DETECTED |
| 2011/10/18<br>**3.1.10** | WSDL | ADDITION | Support for the following products in 1_fam.xml<br>*DEV_SCAT_REMOTE_LINC_2_KEYPAD_4*<br>*DEV_SCAT_REMOTE_LINC_2_SWITCH* |

| | | | |
|---|---|---|---|
| | | | *DEV_SCAT_REMOTE_LINC_2_KEYPAD_8* |
| 2011/08/11<br>**3.1.7** | Document | MODIFY | Base64 for admin:admin was wrong |
| | WSDL | NEW | -Added <family> to Node<br>-Definition for family/device category/device sub category<br>-- family.xsd … defines objects for family<br>-- cat.xml … defines high level device categories<br>-- 1_fam.xml (for INSTEON)<br>-- 3_fam.xml (for RCS) |
| | WSDL | NEW | -Added LogInfo objects and instances<br>-- log.xsd … defines system wide log objects<br>-- loginfo.xml .. system wide errors/messages<br>-- insterr.xml … INSTEON errors<br>-- upberr.xml … UPB errors<br>-- smtperr.xml … SMTP errors<br>-- netstat.xml … Network Interface errors |
| | WSDL | MODIFY | -Updated udiobjs.xsd to add the following with their respective namespaces<br>-- Point to LogInfo (in log.xsd)<br>-- Point to family.xsd |
| | DOC | UPDATE | -Added Irrigation Module |
| 2011/07/01<br>**3.1.5** | WSDL | NEW | -Added ZigbeeSignal element to ZigbeeNode. Includes RSSI (~Signal to Noise ratio) and LQI (Signal Level/Quality)<br>-Added RestResponse object |
| | REST<br>Zigbee | NEW | /rest/zb/scanNetwork<br>/rest/zb/ntable<br>/rest/zb/nodes/[euid]<br>/rest/zb/nodes/[euid]/ping<br>/rest/zb/nodes/[euid]/remove |
| 2011/06/20<br>**3.1.4** | WSDL | NEW | Added Irrigation Requirement & Water Deficit Yesterday to Climate Events |
| | WSDL | NEW | Zigbee node enhancements include Endpoints, DeviceId, ProfileId, and Clusters |
| 2011/05/23 –<br>**3.1.3** | DOC | NEW | Added UPB Device Types |
| | DOC | MODIFY | Structure: Moved INSTEON Cat/Subcat to Appendix A |
| | | | |
| 2011/04/17 –<br>**3.1.3** | REST | NEW | Zigbee Support for ISY994 Series |
| | WSDL | NEW | Zigbee Support for ISY994 Series |
| | EVENT | NEW | Zigbee Support for ISY994 Series |
| | EVENT | MODIFY | OpenADR Price Updated → OpenADR Status Update |
| | EVENT | DELETE | OpenADR Pending State |
| | WSDL | NEW | OpenADR State |
| | WSDL | NEW | udiobjs.xsd:SystemActor – user id in log/error log |

| | | | |
|---|---|---|---|
| | WSDL | NEW | Added *ADRPST* to control types (insteon.xsd, upb.xsd) which toggles auto inclusion in Open AutoDR events |
| | EVENT | NEW | Climate/WeatherBug Evapotranspiration |
| | MOD | NEW | New module: RCS Zigbee 22000 |
| | | | |
| 2011/03/18 - **3.1.1** | Nodes | NEW/UPB | pnode, sgid, tx, rx, qry, ctl, rsp tags for nodes |
| | REST | NEW | /rest/vars/definitions |
| | WSDL | CHANGE | Moved common objects to **udiobjs.xsd** Schema file. It's now imported with namespace ***udiobjects*** |
| | WSDL | CHANGE | Moved event objects to **udievnts.xsd** Schema file. It's now imported with namespace ***udievents*** |
| | WSDL | CHANGE | Moved INSTEON specific types/enumerations to **insteon.xsd** Schema file. It's now imported with namespace ***isy_insteon*** |
| | WSDL | CHANGE | Moved UPB specific types/enumerations to **upb.xsd** Schema file. It's now imported with namespace ***isy_upb*** |
| | WSDL | CHANGE | ControlType is now a **union** of CommonControlType, INSTEON_ControlType and UPB_ControlType |
| 2011/03/11 - **3.1.0** | **WSDL 3.0** | | |
| | SOAP | NEW | GetStartupTime – Returns a timestamp of when the ISY was last started |
| | SOAP | NEW | SendHeartbeat – Immediately sends a heartbeat message to all subscribers |
| | SOAP | NEW | SetVariable – Sets the variable to the given value |
| | SOAP | NEW | GetVariable – Returns the current value of the variable and the last time it was updated |
| | SOAP | NEW | GetVariables – Returns all the variables of the given type |
| | SOAP | NEW/UPB | RestoreNodesFromDevice – Reverse of Restore Device |
| | SOAP | NEW/UPB | GetNodeDeviceProps – Gets the device properties for a node |
| | SOAP | NEW/UPB | SetNodeDeviceProps – Sets the device properties for a node |
| | SOAP | MODIFIED | IsSubscribed – Returns an intResponse instead of DefaultResponse |
| | SOAP | MODIFIED | Unsubscribe – a new <flag> element |
| | EVENT | NEW | UD_TRIGGER_EVENT_VAR_STATUS ( _1 : 6 ) – Variable status |
| | EVENT | NEW | UD_TRIGGER_EVENT_VAR_INIT ( _1 : 7 ) – Variable was initialized |
| | EVENT | NEW | UD_NODE_ERROR_CLEARD_ACTION ( _3 : CE) – The node is no longer in error |
| | EVENT | NEW/UPB | UD_NODE_REVISED_ACTION ( _3 : RV ) – Node revised |
| | EVENT | NEW/UPB | UD_NODE_DEVICE_ID_CHANGED ( _3 : DI ) – The ID of the node changed |
| | EVENT | NEW/UPB | UD_NODE_DEVICE_PROPERTY_CHANGED ( _3 : DP) – The node device properties have changed |
| | EVENT | NEW/UPB | UD_PROGRESS_REPORT ( _7 ) – Progress report |
| | EVENT | NEW/UPB | UD_UPB_LINKER_EVENT ( _14 ) – UPB Linker Events |
| | EVENT | NEW/UPB | UD_UPB_DEVICE_ADDER_EVENT ( _15 ) – UPB Adder State |
| | EVENT | NEW | UD_ENERGY_MONITOR_ZIGBEE_STATUS ( _13 : 3 ) – |

| | | | |
|---|---|---|---|
| | | | Status of Zigbee Radio |
| | REST | NEW | Variable support |

# 1.0 Introduction

ISY is a sophisticated events based network platform which affords its clients unprecedented levels of integration and functionality. Now, with the introduction of WSDK, most of ISY functions are externalized as Web Services and defined in a well formed WSDL which can immediately be imported into an IDE of choice.

At a high level, ISY operates and may be communicated with in the following order:

1. Upon power up, ISY sends out broadcasts messages of its location to all the UPnP clients on the network

2. Interested clients may choose to:

    a. Search for a specific ISY (based on the device type it supports such as Insteon)
    b. Listen in for ISY generated announcements on the network
    c. Immediately start communicating with ISY using a predefined IP (static) address and port, if one is already known

3. Upon discovery of an ISY – regardless of the method chosen – communications with ISY takes place through Web Services/SOAP 1.2 calls:

    a. All requests need to have an HTTP Basic Authentication Header (Realm="/")
    b. Optionally ***subscribe*** to the ISY events from which time ISY continuously notifies the subscriber(s) of the changes in its state. Upon successful subscription, ISY publishes all its current states to the client so that the client and ISY are in synch at the moment of subscription. In this respect, then, the clients are started with the current state of ISY and are notified of all the changes as they occur and thus will never have to poll ISY

4. During application exit, the client must notify ISY that it wishes to terminate its session. This is achieved by issuing:

    a. Unsubscribing from ISY

5. During normal operations, the client *must* always respond back (immediately) with an Ack to ISY's Heartbeat events otherwise ISY assumes a client malfunction (the client didn't exit gracefully as outlined in step 4) and terminates the associated session

As mentioned before, ISY is event driven and thus every change in ISY is notified/published to all the ISY subscribed clients in real-time and almost immediately. In this respect, then, one could use the default ISY User Interface (a signed Java applet) to effect a change while using one's own client to view all the changes that are taking place (and vice versa).

# 2.0 Basic Concepts

## *2.1 Control*

A *Control* is the logical representation of either a state or a function that may be performed on a physical device (or a scene) linked to ISY. For example, "DON" is the name of the *Control* which instructs ISY to turn a "Deivce On" while "ST" is the name of the *Control* which holds that *state* of a device.

In essence, then, Control is what "captures" and "controls" changes in the states of physically linked devices or groups/scenes. Since Controls may be associated with states, thus, all ISY publications (publish) to all clients contain a Control parameter which identifies "what changed".

For example, a CLISP (Climate SetPoint) Control not only allows the client to effect a SetPoint change on a linked Thermostat but also, as soon as the change takes effect (or the state changes), ISY notifies all the clients of the change in "CLISP" and the current value thereto (see section 2.2: Action) if any.

The most important attributes of a Control are:

*A Name* – this is the Control's only meaningful unit of communications with ISY such as "CLISP", "DON", "DIM", etc.

*A Label* – this is an optional label that the developer/manufacturer may ascribe to a Control such as "SetPoint", "On", "Dim", "Fast On", etc.

*Action*s – this is a list of optional while permissible actions which may be performed on a Control such as "50" which, when applied to "DON", means turn the "Device On to 50%". Or, when "HEAT" is applied to "CLIMD" (Climate Mode) it means change the thermostat "Mode to Heat". For more details, see section 2.2: Action.

## 2.2 Action

An *Action* is the permissible "value" which may be applied to a *Control*. A Control may have a set of permissible Actions which are captured by a list.

When communicating a state change request to ISY, Action may be null. This said, however, when ISY publishes (to its subscribers) changes to a Control – and if the Control is associated with some state – then this attribute holds the "current value" of the state. For example, when issuing a "DON" to ISY the "ST" Control (which is associated with a state) is updated and, as such, ISY shall notify all the subscribed clients of a change in "ST" with Action being the current value of "ST" such as "50"%.

The most important attributes of an Action are:

*A Name* – this is the Action's only meaningful unit of communications with ISY. Depending on the Control, this attribute may take the form of a free text/object field the value of which is filled in by ISY upon publications of events.

*A Label* – this is an optional label that the developer/manufacturer may ascribe to an Action such as "Heat".

## 2.3 Node

A Node is a logical representation of a physical device linked to ISY. So, for instance, KeypadLinc's button A is a node and so are its buttons B, C, D through H.

In essence – and when put in the context of a Control and Action – the Node is the only missing piece which, when all put together, enables effecting the desired change on a physical device linked to ISY.

The most important attributes of Node are:

*An Address* – this is the address which ISY uses to communicate with the actual physical device such as 4 E 52 1

*A Name* – this is the user friendly name which can be changed by any ISY client

*States (device Variables)* – this is the list of all the Controls for a Node and their current associated Actions (values)

*A note on Insteon addresses:*

Since, as mentioned before, every button is also considered a device within ISY, thus, each button shall have its own address conforming to the following syntax: X X X B – where X is the actual Insteon address for the device in hex and B is the button group number.

For instance, a 6 button KeypadLinc with address 04 E8 52 will have the following nodes within ISY:
4 E8 52 1 – the main [loaded] button
4 E8 52 A – Button A
4 E8 52 B – Button B
4 E8 52 C – Button C
4 E8 52 D – Button D

## 2.4 Group/Scene

A Group is a specialization of Node with the added capability of aggregating associated/linked Nodes. Just like a Node, a Group may also be used to effect a change in ISY. The only difference is that issuing a state change on a Group results in ISY sending notifications on the states of all the Nodes within that Group/Scene (if there were any changes).

## 2.5 Putting it Together

By having a triplet {control, action, [node or group/scene]} it's quite easy to effect change on the physical devices which are linked/attached to ISY. For instance:

1. To turn on the light at address 7 B0 B2 to 60%, a simple service call of the type *UDIService* ("DON", "60", "7 B0 B2 1"), is all it takes.

2. To turn off the scene at address 52626, a simple method call of the type *UDIService*("DOF", null, "52626"), is all it takes.

## 2.6 ISY Messages and Web Services

All messages, Web Services, Parameters, Objects, and Events are captured in a WSDL file stored on ISY.

# 3.0 Discovering ISY and its Resources

ISY Can be found using UPnP Search method. ISY also advertises its presence on the network every 30 seconds.

## 3.1 Discovering ISY Using UPnP Search

Send the following UDP Packet to UPnP Multicast group of 239.255.255.250 and port 1900

```
M-SEARCH * HTTP/1.1
HOST:239.255.255.250:1900
MAN:"ssdp.discover"
MX:1
ST:urn:udi-com:device:X_Insteon_Lighting_Device:1
```

Note: X_Insteon_Lighting_Device is the UPnP Device Type for INSTEON ISY devices

ISY replies with:

```
HTTP/1.1 200 OK
CACHE-CONTROL:max-age=30
EXT:
LOCATION:http://192.168.0.208/desc
SERVER:UCoS, UPnP/1.0, UDI/1.0
ST:urn:udi-com:device:X_Insteon_Lighting_Device:1
USN:uuid:00:03:f4:03:0f:61::urn:udi-
com:device:X_Insteon_Lighting_Device:1
```

## 3.2 Listening For ISY Advertisements on the Network

As mentioned before, ISY advertises its existence on the network every 30 seconds. To receive these notification events, join the UPnP Multicast group at 239.255.255.250 and port 1900. ISY advertisements are as follows:

1. Root Device

```
NOTIFY * HTTP/1.1
HOST:239.255.255.250:1900
CACHE-CONTROL:max-age=30
LOCATION:http://192.168.0.208/desc
NT:upnp:rootdevice
NTS:ssdp:alive
```

```
SERVER:UCoS, UPnP/1.0, UDI/1.0
USN:uuid:00:03:f4:03:0f:61::upnp:rootdevice
```

2. Service

```
NOTIFY * HTTP/1.1
HOST:239.255.255.250:1900
CACHE-CONTROL:max-age=30
LOCATION:http://192.168.0.208/desc
NT:urn:udi-com:service:X_Insteon_Lighting_Service:1
NTS:ssdp:alive
SERVER:UCoS, UPnP/1.0, UDI/1.0
USN:uuid:00:03:f4:03:0f:61::urn:udi-
com:service:X_Insteon_Lighting_Service:1
```

3. Device

```
NOTIFY * HTTP/1.1
HOST:239.255.255.250:1900
CACHE-CONTROL:max-age=30
LOCATION:http://192.168.0.208/desc
NT:urn:udi-com:device:X_Insteon_Lighting_Device:1
NTS:ssdp:alive
SERVER:UCoS, UPnP/1.0, UDI/1.0
USN:uuid:00:03:f4:03:0f:61::urn:udi-
com:device:X_Insteon_Lighting_Device:1
```

## *3.3 Capturing ISY Resources*

Regardless of how ISY is discovered, the LOCATION header defines where other ISY resource URIs are located (UPnP Description file). Simply do an HTTP Get on the URL defined by the LOCATION header. The following is an example of the contents of: **http://192.168.0.208/desc**; The most important elements are in **bold**:

```
<?xml version="1.0" ?>
<root xmlns="urn:schemas-upnp-org:device-1-0">
<specVersion>
  <major>1</major>
  <minor>0</minor>
  </specVersion>
  <URLBase>http://192.168.0.208</URLBase>
  <device>
  <deviceType>urn:udi-com:device:X_Insteon_Lighting_Device:1</deviceType>
  <friendlyName>My Lighting</friendlyName>
  <manufacturer>Universal Devices Inc.</manufacturer>
  <manufacturerURL>http://www.universal-devices.com</manufacturerURL>
  <modelDescription>X_Insteon_Lighting_Device:1</modelDescription>
  <modelName>Insteon Web Control</modelName>
  <modelNumber>Insteon Web Control</modelNumber>
  <UDN>uuid:00:03:f4:03:0f:61</UDN>
  <UPC>uuid:00:03:f4:03:0f:61</UPC>
  <serviceList>
  <service>
  <serviceType>urn:udi-
com:service:X_Insteon_Lighting_Service:1</serviceType>
```

```
  <serviceId>urn:udi-com:serviceId:uuid:00:03:f4:03:0f:61</serviceId>
  <SCPDURL>/services.wsdl</SCPDURL>
   <controlURL>/services</controlURL>
   <eventSubURL>/eventing</eventSubURL>
  </service>
 … other services such as SEP
  </serviceList>
  <presentationURL>/</presentationURL>
  </device>
</root>
```

**URLBase:** is the absolute URL to ISY services. All the other URLs are relative to this URL

**UDN:** is the Unique Device Number which uniquely identifies ISY on the network

**SCPDURL**: is the location where the definition of ISY services are located (in WSDL).
**Note**: point your WebServices IDE to this URL to import all services. E.g. http://192.168.0.102:8080/services.wsdl

**controlURL**: is the URL to which all the Service requests are Posted

**eventSubURL**: is the URL to which clients subscribe and unsubscribe

## 3.4 ISY Configuration Resource

ISY Configuration Resource defines how ISY is presently configured. The most important feature of this resource is that it defines the permissible Controls/Actions which may be invoked in ISY. Here's the an example:

```
...
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
      -
      <deviceSpecs>
            <make>Universal Devices Inc.</make>
            <manufacturerURL>http://www.universal-
devices.com</manufacturerURL>
            <model>Insteon Web Controller</model>
            <icon>/web/udlogo.jpg</icon>
            <archive>/web/insteon.jar</archive>
            <chart>/web/chart.jar</chart>
            <queryOnInit>true</queryOnInit>
            <oneNodeAtATime>true</oneNodeAtATime>
      </deviceSpecs>
      <upnpSpecs>
            <upnpDevice>
                  <utype>X_Insteon_Lighting_Device</utype>
                  <version>1</version>
            </upnpDevice>
            <upnpService>
                  <utype>X_Insteon_Lighting_Service</utype>
                  <version>1</version>
            </upnpService>
      </upnpSpecs>
      <controls>
            <control>
                  <name>ST</name>
                  <label>Status</label>
                  <readOnly>true</readOnly>
                  <isQueryAble>true</isQueryAble>
                  <isNumeric>true</isNumeric>
                  <numericUnit>%</numericUnit>
            </control>
            <control>
                  <name>OL</name>
                  <label>On Level</label>
                  <readOnly>false</readOnly>
                  <isQueryAble>true</isQueryAble>
                  <isNumeric>true</isNumeric>
                  <numericUnit>%</numericUnit>
            </control>
            <control>
                  <name>RR</name>
                  <label>Ramp Rate</label>
                  <readOnly>false</readOnly>
                  <isQueryAble>true</isQueryAble>
                  <isNumeric>true</isNumeric>
                  <numericUnit>%</numericUnit>
```

```
</control>
<control>
      <name>DON</name>
      <label>On</label>
</control>
<control>
      <name>DFON</name>
      <label>Fast On</label>
</control>
<control>
      <name>DOF</name>
      <label>Off</label>
</control>
<control>
      <name>DFOF</name>
      <label>Fast Off</label>
</control>
<control>
      <name>BRT</name>
      <label>Brighten</label>
</control>
<control>
      <name>DIM</name>
      <label>Dim</label>
</control>
<control>
      <name>BMAN</name>
      <label>Fade Start</label>
</control>
<control>
      <name>SMAN</name>
      <label>Fade Stop</label>
</control>
<control>
      <name>BEEP</name>
      <label>Beep</label>
</control>
<control>
      <name>RESET</name>
      <label>Reset values</label>
</control>
<control>
      <name>CLISPH</name>
      <label>Heat Setpoint</label>
      <readOnly>false</readOnly>
      <isQueryAble>true</isQueryAble>
      <isNumeric>true</isNumeric>
      <numericUnit>F</numericUnit>
</control>
<control>
      <name>CLISPC</name>
      <label>Cool Setpoint</label>
      <readOnly>false</readOnly>
      <isQueryAble>true</isQueryAble>
      <isNumeric>true</isNumeric>
      <numericUnit>F</numericUnit>
```

```
</control>
<control>
      <name>CLIFS</name>
      <label>Fan State</label>
      <readOnly>false</readOnly>
      <isQueryAble>true</isQueryAble>
      <isNumeric>false</isNumeric>
      <actions>
            <action>
                  <name>7</name>
                  <label>On</label>
            </action>
            <action>
                  <name>8</name>
                  <label>Auto</label>
            </action>
      </actions>
</control>
<control>
      <name>CLIMD</name>
      <label>Thermostat Mode</label>
      <readOnly>false</readOnly>
      <isQueryAble>true</isQueryAble>
      <isNumeric>false</isNumeric>
      <actions>
            <action>
                  <name>0</name>
                  <label>Off</label>
            </action>
            <action>
                  <name>1</name>
                  <label>Heat</label>
            </action>
            <action>
                  <name>2</name>
                  <label>Cool</label>
            </action>
            <action>
                  <name>3</name>
                  <label>Auto</label>
            </action>
            <action>
                  <name>4</name>
                  <label>Fan</label>
            </action>
            <action>
                  <name>5</name>
                  <label>Program Auto</label>
            </action>
            <action>
                  <name>6</name>
                  <label>Program Heat</label>
            </action>
            <action>
                  <name>7</name>
                  <label>Program Cool</label>
```

```
            </action>
      </actions>
</control>
<control>
      <name>CLIHUM</name>
      <label>Humidity</label>
      <readOnly>true</readOnly>
      <isQueryAble>true</isQueryAble>
      <isNumeric>true</isNumeric>
      <numericUnit>%</numericUnit>
</control>
<control>
      <name>CLIHCS</name>
      <label>Heat/Cool State</label>
      <readOnly>true</readOnly>
      <isQueryAble>true</isQueryAble>
      <isNumeric>false</isNumeric>
      <actions>
            <action>
                  <name>0</name>
                  <label>Off</label>
            </action>
            <action>
                  <name>1</name>
                  <label>Heat On</label>
            </action>
            <action>
                  <name>2</name>
                  <label>Cool On</label>
            </action>
      </actions>
</control>
<control>
      <name>UOM</name>
      <label>Unit</label>
      <readOnly>true</readOnly>
      <isQueryAble>true</isQueryAble>
      <isNumeric>false</isNumeric>
      <actions>
            <action>
                  <name>1</name>
                  <label>Celsius</label>
            </action>
            <action>
                  <name>2</name>
                  <label>Fahrenheit</label>
            </action>
      </actions>
</control>
<control>
      <name>CPW</name>
      <label>Current Power Usage</label>
      <readOnly>true</readOnly>
      <isQueryAble>true</isQueryAble>
      <isNumeric>true</isNumeric>
      <numericUnit>W</numericUnit>
```

```
      </control>
      <control>
            <name>TPW</name>
            <label>Total Power Used</label>
            <readOnly>true</readOnly>
            <isQueryAble>true</isQueryAble>
            <isNumeric>true</isNumeric>
            <numericUnit>kWs</numericUnit>
      </control>
</controls>
<app>Insteon_UD994</app>
<app_version>2.8.11</app_version>
<platform>ISY-C-994</platform>
<build_timestamp>2011-01-20-01:09:20</build_timestamp>
<root>
      <id>00:03:f4:03:65:96</id>
      <name>ISY994</name>
</root>
<product>
      <id>1100</id>
      <desc>ISY 994i 1024</desc>
</product>
<features>
      <feature>
            <id>21010</id>
            <desc>Open Auto-DR</desc>
            <isInstalled>true</isInstalled>
            <isAvailable>true</isAvailable>
      </feature>
      <feature>
            <id>21011</id>
            <desc>Electricity Meter</desc>
            <isInstalled>true</isInstalled>
            <isAvailable>true</isAvailable>
      </feature>
      <feature>
            <id>21012</id>
            <desc>Gas Meter</desc>
            <isInstalled>false</isInstalled>
            <isAvailable>true</isAvailable>
      </feature>
      <feature>
            <id>21013</id>
            <desc>Water Meter</desc>
            <isInstalled>false</isInstalled>
            <isAvailable>false</isAvailable>
      </feature>
      <feature>
            <id>21020</id>
            <desc>Weather Information</desc>
            <isInstalled>true</isInstalled>
            <isAvailable>true</isAvailable>
      </feature>
      <feature>
            <id>21030</id>
            <desc>Network Modules</desc>
```

```
            <isInstalled>false</isInstalled>
            <isAvailable>false</isAvailable>
        </feature>
        <feature>
            <id>21040</id>
            <desc>Networking Module</desc>
            <isInstalled>true</isInstalled>
            <isAvailable>true</isAvailable>
        </feature>
        <feature>
            <id>21050</id>
            <desc>Utility Meter (Electricity)</desc>
            <isInstalled>false</isInstalled>
            <isAvailable>true</isAvailable>
        </feature>
        <feature>
            <id>21051</id>
            <desc>SmartMeter ESP</desc>
            <isInstalled>false</isInstalled>
            <isAvailable>false</isAvailable>
        </feature>
        <feature>
            <id>21060</id>
            <desc>A10/X10 for Insteon</desc>
            <isInstalled>false</isInstalled>
            <isAvailable>false</isAvailable>
        </feature>
        <feature>
            <id>21070</id>
            <desc>Portal Integration - Check-it.ca</desc>
            <isInstalled>true</isInstalled>
            <isAvailable>true</isAvailable>
        </feature>
        <feature>
            <id>21014</id>
            <desc>Current Cost Meter</desc>
            <isInstalled>false</isInstalled>
            <isAvailable>true</isAvailable>
        </feature>
        <feature>
            <id>21080</id>
            <desc>Broadband SEP Device</desc>
            <isInstalled>true</isInstalled>
            <isAvailable>true</isAvailable>
        </feature>
        <feature>
            <id>21071</id>
            <desc>Portal Integration - GreenNet.com</desc>
            <isInstalled>false</isInstalled>
            <isAvailable>true</isAvailable>
        </feature>
    </features>
    <triggers>true</triggers>
    <security>SSL</security>
    <isDefaultCert>false</isDefaultCert>
</configuration>
```

### 3.4.1 Modules (Features)

ISY allows for optional modules to be installed. The correct behavior of the client may depend on the installed modules such as Climate and A10/X10. Modules are identified by the **<feature>** element in the configuration resource (see section 3.4). As such, it's advised that the client code queries the modules to enable/disable functionality accordingly.

Currently available modules are listed herein under:

```
21010 - Open Auto-DR

21011 - Electricity Meter (Brultech)

21012 - Gas Meter

21013 - Water Meter

21014 - Current Cost Meter

21020 - Weather Information (WeatherBug)

21030 - Network Modules (NOT AVAILABLE)

21040 - Networking Module

21050 - Utility Meter (Electricity): Zigbee SEP

21051 - SmartMeter ESP

21060 - A10/X10 for Insteon

21070 - Portal Integration - Check-it.ca

21071 - Portal Integration - GreenNet.com

21080 - Broadband SEP Device

21090 – ELK Module

22000 – Zigbee RCS Module

23000 – Irrigation Module
```

## 3.5 ISY Nodes Configuration Resource

ISY Nodes Configuration Resource defines all the Nodes/Scenes as configured in ISY with their relationships.

There are 3 types of nodes in ISY:
**Node** … identifies and end or virtual device or button (<node>)
**Group** ... identifies a scene or a logical grouping between devices (<group>)
**Folder** … identifies a logical grouping of nodes and groups without regards to their relationships

**Note:**
1. A **Node** can belong to multiple **Groups** acting as Controller or Responder
2. A **Node** can belong only to ONE and only ONE **Folder** or another **Node**
3. A **Group** can belong only to ONE and only ONE **Folder**
4. A **Folder** can belong only to ONE and only ONE **Folder**

### 3.5.1 Types of Nodes/Parents

```
UD_HIERARCHY_NODE_TYPE_NOTSET 0 (unknown)
UD_HIERARCHY_NODE_TYPE_NODE   1
UD_HIERARCHY_NODE_TYPE_GROUP  2
UD_HIERARCHY_NODE_TYPE_FOLDER 3
```

### 3.5.2 Node (<node>)

This element defines a configured node in ISY. A node is anything that can be impacted upon or if it can impact some change in the environment. As such, a node could be a KeypadLinc's button or a Thermostat.

<**node** flag="0">
<**address**>The address of the node</**address**>
<**name**>Friendly name</name>
<**parent** type="see 3.5.1">the address of the parent</**parent**>
<**family**>optionally defines device's family; see below</**family**>
<**type**>device type; see below</**type**>
<**enabled**>"true"|"false"</enabled>
<**deviceClass**>1024</**deviceClass**>
<**wattage**>2000</**wattage**>
<**dcPeriod**>60</**dcPeriod**>
<**pnode**>The primary node address (see below)</**pnode**>
<**sgid**>Subgroup ID (see below)</**sgid**>
<**tx**>Controller end-point bitmask</**tx**>
<**rx**>Responder end-point bitmask</**rx**>

**<qry />**
**<ctl />**
**<rsp [type="<rsp-type>"] />**

</**node**>

**Terms:**

| | |
|---|---|
| *end-point* | a single feature on a device such as the load or a KPL button. |
| *subgroup* | A subset of one or more *end-points* for a device |
| *primary node* | The node designated as the overall representative of a device |

**e.g.** Sample nodes for a single device
<node><address>**0015**</address><pnode>**0015**</pnode><sgid>**1**</sgid>…
<node><address>0029</address><pnode>**0015**</pnode><sgid>**2**</sgid>…
<node><address>0042</address><pnode>**0015**</pnode><sgid>**3**</sgid>…

**<pnode>**

The address of the primary node for the device partially represented by this node. If this node is the primary node then <pnode> will equal <address>

A device may be represented by one or more nodes. One of these nodes is designated the *primary node* and is used to help group the set of nodes for a device.

*Note: UPB Only

**<sgid>**

The ID identifying the subgroup of end-points for the device.

Each node representing a subset of the end-points for a device is identified by a unique *subgroup ID*. This is similar in function to the last digit in an Insteon Node Address.

*Note: UPB Only

**<qry />**

If present then this node may be queried.

*Note: UPB Only

**<ctl />**

If present then this node may be used as a scene controller.

*Note: UPB Only

**<rsp [type="<rsp-type>"] />**

If present then this node may be used as a scene responder.

**type**

| | |
|---|---|
| U_DIM | Standard dimmer (**Default Value**) |
| U_S_DIMT | SAI Dimmer with Off Timer |
| U_RELAY | Relay Device (on/off only) |
| U_LED_6 | Button LEDS for 6-Button KPL |
| U_LED_8 | Button LEDS for 8-Button KPL |

*Note: UPB Only

**<tx>**

Bitmask indicating the controller endpoints represented by this node

*Note: UPB Only

**<rx>**

Bitmask indicating the responder endpoints represented by this node

*Note: UPB Only

**"flag" Attribute**

Defines the characteristics of the <node> as well as the <group> elements as follows (represented in decimal):

```
NODE_IS_INIT          0x01 //needs to be initialized
NODE_TO_SCAN          0x02 //needs to be scanned

//Node operations flags
NODE_IS_A_GROUP        0x04 //it's a group!
NODE_IS_ROOT    0x08 //it's the root group
NODE_IS_IN_ERR         0x10 //it's in error!

NODE_IS_NEW            0x20 //brand new node
NODE_TO_DELETE         0x40 //has to be deleted later
NODE_IS_DEVICE_ROOT    0x80 //root device such as KPL load
```

**<deviceClass> Element ***

Defines the class of device for energy management (as defined by SEP):

```
DC_HVAC                          0x0001
DC_STRIP_HEATER                  0x0002
```

```
DC_WATER_HEATER              0x0004
DC_POOL_PUMP                 0x0008
DC_SMART_APPLIANCE           0x0010
DC_IRRIGATION_PUMP           0x0020
DC_MANAGED_LOAD              0x0040
DC_SIMPLE                    0x0080
DC_EXTERIOR_LIGHTING         0x0100
DC_INTERIOR_LIGHTING         0x0200
DC_EV                        0x0400
DC_GENERATION_SYSTEM         0x0800
DC_WASHER                    0x1000
DC_DRYER                     0x2000
DC_OVEN                      0x4000
DC_FRIG                      0x8000
DC_ALL                       0xFFFF
```

*Available in ISY994 Series/EMS models only

**<dcPeriod> Element***

Defines the Duty Cycle period in minutes.

*Available in ISY994 Series/EMS models only

**<family> Element**

Defines a device/node family such as ZWave, RCS, INSTEON, etc. This element is useful for multi-protocol configurations including with security systems.

Family is defined in **family.xsd** with accompanying **[id]_fam.xml** which defines the categories/subcategories for each family id.

**<type> Element**

Defines the Category and Subcategories of underlying devices as follows:

*device category.device subcategory.version.reserved*

*For INSTEON Device Types, please checkout Appendix A*

### 3.5.3 Group/Scene (<group>)

Same as node but defines a scene with a list of members and their relationships to the scene.

<**group** flag="see 3.5.2 (Node)">
    <**address**>The address of the group</**address**>
    <**name**>Friendly name</**name**>
    <**fmtDevId**>Friendly name</**fmtDevId**>
    <**members**>
        <**link** type="relationship type">5 8A 37 1</**link**>
        <**link** type="relationship type">5 8A 37 3</**link**>
        …
    **</members>**
**</group>**

**<fmtDevId>**

A formatted device ID. In the case of UPB this contains the underlying UPB Link ID for the scene.
*Note: UPB Only

**<members> Element**

Is a list of members for the scene each one having different relationships defined by the **type** attribute of the **link** element.

**<link> Element**

Defines members of a group/scene.

**"type" Attribute** defines the role a node plays in relationship to other nodes in a scene:

NODE_IS_CONTROLLER 0x10 (decimal 16)

**Other values should be considered Responders.**

### 3.5.4 Folder (<folder>)

Same as node but identifies a folder.

# 4.0 Communicating with ISY

To successfully communicate with ISY, the following steps must be taken

1. **Find ISY** and retrieve its resources by parsing the contents of the LOCATION header (see section 3)
    a. Capture <controlURL> which should be the URL used for all the subsequent Web Services invocations
    b. Capture <eventSubURL> which should be the URL used for subscribing/unsubscribing from ISY
    c. Point your WebServices IDE to SCPDURL to import web services

2. **Authenticate – no longer needed as of release 2.7.5**

    Instead, you will use the **HTTP Basic Authorization** header to send the credentials to ISY with each request.

    The format of the Authorization header is:
    Authorization: Basic Base64(userid:password)

    Where, base64 converts the string representation of userid followed by ':' and followed by the password.

    As an example, the Authorization header for userid=admin , password= admin is:
    Base64(admin:admin)→ YWRtaW46YWRtaW4=

    Therefore the Authorization header shall be:

    **Authorization: Basic YWRtaW46YWRtaW4=**

3. **Subscribe**

```
POST /services HTTP/1.1
Host: 192.168.0.129:80
Authorization: Basic YWRtaW46YWRtaW4=
Content-Length: 193
Content-Type: text/xml; charset="utf-8"

<s:Envelope><s:Body><u:Subscribe xmlns:u="urn:udi-
com:service:X_Insteon_Lighting_Service:1"><reportURL>REUSE_SOCKET</reportU
RL><duration>infinite</duration></u:Subscribe></s:Body></s:Envelope>
```

    Note: you may provide a <reportURL> URL or you can keep this socket open (REUSE_SOCKET) to receive ISY events

## 4. Control Nodes/Scenes in ISY using UDIService

Simply provide the permissible values for <Control>, <Action>, and <Node>.

The <node> element is the address of the node/group to be impacted. The <flag> element **must be 4** if this is impacting a scene/group. Any other value is considered a node and not a group/scene.

**Note**: if you are using SOAP1.2 compliant IDE, **SOAPACTION** header is **not** required.

a. Turn Device Hall 2 On

```
POST /services HTTP/1.1
Host: 192.168.0.208:80
Content-Length: 210
Authorization: Basic YWRtaW46YWRtaW4=
Content-Type: text/xml; charset="utf-8"
SOAPACTION:"urn:udi-
com:service:X_Insteon_Lighting_Service:1#UDIService"


<s:Envelope><s:Body><u:UDIService xmlns:u="urn:udi-
com:service:X_Insteon_Lighting_Service:1"><control>DON</control><actio
n></action><flag>65531</flag><node>7 B0 A5
1</node></u:UDIService></s:Body></s:Envelope>
```

b. Turn Device Hall 2 On to 46%

```
POST /services HTTP/1.1
Host: 192.168.0.208:80
Content-Length: 213
Authorization: Basic YWRtaW46YWRtaW4=
Content-Type: text/xml; charset="utf-8"
SOAPACTION:"urn:udi-
com:service:X_Insteon_Lighting_Service:1#UDIService"


<s:Envelope><s:Body><u:UDIService xmlns:u="urn:udi-
com:service:X_Insteon_Lighting_Service:1"><control>DON</control><actio
n>117</action><flag>65531</flag><node>7 B0 A5
1</node></u:UDIService></s:Body></s:Envelope>
```

c. Turn Master Scene On Immediately (Fast On)

```
POST /services HTTP/1.1
Host: 192.168.0.208:80
Content-Length: 203
Authorization: Basic YWRtaW46YWRtaW4=
Content-Type: text/xml; charset="utf-8"
SOAPACTION:"urn:udi-
com:service:X_Insteon_Lighting_Service:1#UDIService"


<s:Envelope><s:Body><u:UDIService xmlns:u="urn:udi-
com:service:X_Insteon_Lighting_Service:1"><control>DFON</control><acti
```

```
on></action><flag>4</flag><node>34612</node></u:UDIService></s:Body></
s:Envelope>
```

d. Increment Thermostat Setpoint

```
POST /services HTTP/1.1
Host: 192.168.0.208:80
Content-Length: 209
Authorization: Basic YWRtaW46YWRtaW4=
Content-Type: text/xml; charset="utf-8"
SOAPACTION:"urn:udi-
com:service:X_Insteon_Lighting_Service:1#UDIService"


<s:Envelope><s:Body><u:UDIService xmlns:u="urn:udi-
com:service:X_Insteon_Lighting_Service:1"><control>BRT</control><actio
n></action><flag>65531</flag><node>0 0 11
1</node></u:UDIService></s:Body></s:Envelope>
```

e. Start Fading (Up) Ceiling 1

```
POST /services HTTP/1.1
Host: 192.168.0.208:80
Content-Length: 212
Authorization: Basic YWRtaW46YWRtaW4=
Content-Type: text/xml; charset="utf-8"
SOAPACTION:"urn:udi-
com:service:X_Insteon_Lighting_Service:1#UDIService"


<s:Envelope><s:Body><u:UDIService xmlns:u="urn:udi-
com:service:X_Insteon_Lighting_Service:1"><control>BMAN</control><acti
on>1</action><flag>65531</flag><node>7 A5 95
1</node></u:UDIService></s:Body></s:Envelope>
```

For Fading Down, use <action>0</action>

f. Stop Fading Ceiling 1

```
POST /services HTTP/1.1
Host: 192.168.0.208:80
Content-Length: 211
Authorization: Basic YWRtaW46YWRtaW4=
Content-Type: text/xml; charset="utf-8"
SOAPACTION:"urn:udi-
com:service:X_Insteon_Lighting_Service:1#UDIService"


<s:Envelope><s:Body><u:UDIService xmlns:u="urn:udi-
com:service:X_Insteon_Lighting_Service:1"><control>SMAN</control><acti
on></action><flag>65531</flag><node>7 A5 95
1</node></u:UDIService></s:Body></s:Envelope>
```

**5. Process Events/Notifications (see section 5)**

ISY Events are of the form:

```
<?xml version="1.0"?><e:propertyset xmlns:e="urn:schemas-upnp-org:event-1-
0"><e:property><ST>0</ST></e:property><e:property><node>9 89 8B
5</node><eventInfo>Event Specific elements</eventInfo></e:property>
</e:propertyset>
```

Where the first <property> element contains the Control which was changed, in this case ST (status) and the Action for this Control, in this case 0. The <node> element defines the node which was impacted.

# 5.0 Events

Events are very important feature of ISY and are published to subscribers. There can be a maximum of 10 simultaneous subscribers subscribing to ISY and all will get event notifications in real time. The following is a list of event definitions:

## *5.1 Device Status (control = Device Property)*

**action = Property value**

**Node = The address of the device**

## *5.2 Heartbeat (control = "_0")*

**action = Duration in seconds**

node = null
eventInfo = null

## *5.3 Trigger Events (control = "_1")*

**action = "0" →Event Status**

node = null
eventInfo =
<**eventInfo**>
<**id**>Hex representation of program id</**id**>
<**X/**> … X=**on** if enabled, and **off** if disabled
<**Y**/> … Y=**rr** if run at reboot and **nr** if not run at reboot
<**r**> last run time in YYMMDD HH:MM:SS</**r>**
**<f**> last finish time in YYMMDD HH:MM:SS</**f**>
<**s>** status* **</s>**
</**var**>
</**eventInfo**>

*Status is a bitwise OR of **RUN**_X and ST_X:

| | | |
|---|---|---|
| **RUN**_IDLE | = | 0x01 |
| **RUN**_THEN | = | 0x02 |
| **RUN**_ELSE | = | 0x03 |
| | | |
| **ST**_UNKNOWN | = | 0x10 |
| **ST**_TRUE | = | 0x20 |
| **ST**_FALSE | = | 0x30 |
| **ST**_NOT_LOADED | = | 0xF0 |

**action = "1" →Get Status (notifies subscribers to refresh)**

node = null
eventInfo = null

**action = "2" →Key Changed**

node = key
eventInfo = null

**action = "3" →Info String**

node = key
eventInfo = text

**action = "4" →IR Learn Mode**

node = null
eventInfo = null

**action = "5" → Schedule (schedule status changed)**

node = key
eventInfo = null

**action = "6" → Variable Status (status of variable changed)**

node = key
<**eventInfo**>
<**var** id="<var-id>" type ="<var-type>">
<**val**>value</**val**>
<**ts**>YYYYMDD HH:MM:SS</**ts**>
</**var**>
</**eventInfo**>

**action = "7" → Variable Initialized (initial value of a variable was set)**

node = key
<**eventInfo**>
<**var** id="<var-id>" type ="<var-type>">
<**init**>value</**init**>
</**var**>
</**eventInfo**>

## 5.4 Driver Specific Events (control = "_2")

Depends on the underlying protocol driver

## *5.5 Node Changed/Updated (control = "_3")*

**action = "NN" →Node Renamed**

node = the address of the node
<**eventInfo>**
<**newName**>name</**newName**>
**</eventInfo>**

**action = "NR" → Node Removed**

node = the address of the node removed
eventInfo = null

### action = "ND" → Node Added

node = the address of the node added
<**eventInfo**>
    <**nodeName**>name</**nodeName**>
    <**nodeType**>see 3.5.2</**nodeType**>
</**eventInfo**>

### action = "MV" →Node Moved (into a scene)

node = the address of the **scene** to which the node was moved to
<**eventInfo**>
    <**movedNode**>the address of the moved node</**movedNode**>
    <**linkType**>controller/responder</**linkType**> (see 3.5.3)
</**eventInfo**>

### action = "CL" →Link Changed (in a scene)

Not supported

### action = "RG" →Removed From Group (scene)

node = the address of the group/scene
<**eventInfo**>
    <**removedNode**>the address of the removed node</**removedNode**>
</**eventInfo**>

### action = "EN" →Enabled

node = the address of the node that was enabled/disabled
<**eventInfo**>
    <**enabled**>"true"|"false"</**enabled**>
</**eventInfo**>

### action = "PC" →Parent Changed

node = the address of the node whose parent changed
<**eventInfo**>
    <**node**>the address of the node</**node**>
    <**nodeType**>see 3.5.1</**nodeType**>
    <**parent**>the address of tne new parent (NULL if none)</**parent**>
    <**parentType**>see 3.5.1</**parentType**>
</**eventInfo**>

### action = "PI" →Power Info Changed

node = the address of the node whose power info changed
**<eventInfo>**
**<deviceClass>**see 3.5.1**</deviceClass>**
**<wattage>**see 3.5.1**</wattage>**
**<dcPeriod>**see 3.5.1**</dcPeriod>**
**</eventInfo>**

### action = "DI" →Device ID Changed

Not implemented.

### action = "DP" →Device Property Changed

UPB only

### action = "GN" →Group Renamed

node = the address of the group
<**eventInfo>**
<**newName**>name</**newName**>
**</eventInfo>**

### action = "GR" → Group Removed

node = the address of the group removed
eventInfo = null

### action = "GD" → Group Added

node = the address of the node added
<**eventInfo>**
<**groupName**>name</**groupName**>
<**groupType**>see 3.5.2</**groupType**>
</**eventInfo**>

### action = "FN" →Folder Renamed

node = the address of the folder
<**eventInfo>**
<**newName**>name</**newName**>
**</eventInfo>**

**action = "FR" → Folder Removed**

node = the address of the group removed
eventInfo = null

**action = "FD" → Folder Added**

node = the address of the folder added
eventInfo=null

**action = "NE" → Node Error (Comm. Errors)**

node = the address of the node with communications errors

**action = "CE" → Clear Node Error (Comm. Errors Cleared)**

node = the address of the node with communications errors

**action = "SN" → Discovering Nodes (Linking)**

node = null

**action = "SC" → Node Discovery Complete**

node = null

**action = "WR" → Network Renamed**

node = the new name for network

**action = "WH" → Pending Device Operation**

node = the address of the node for which there's pending operations

**action = "WD" → Programming Device**

node = the address of the node to which programming/write operations are being carried out

**action = "RV" → Node Revised (UPB)**

Indicates the node may have been drastically changed therefore the UI should throw away any information about the node and reconstruct/repopulate it with the new information

node = the address of the node to which programming/write operations are being carried out
<**eventInfo**>standard node structure</**eventInfo**>

## *5.6 System Configuration Updated (control = "_4")*

**action = "0" → Time Changed**

**action = "1" → Time Configuration Changed**

**action = "2" → NTP Settings Updated**

**action = "3" → Notifications Settings Updated**

**action = "4" → NTP Communications Error**

**action = "5" → Batch Mode Updated**

node = null
<**eventInfo**>
<**status**>"1"|"0"</**status**>
</**eventInfo**>

**action = "6" → Battery Mode Programming Updated**

node = null
<**eventInfo**>
<**status**>"1"|"0"</**status**>
</**eventInfo**>

## *5.7 System Status Updated (control = "_5")*

node = null

**action = "0" → Not Busy**

**action = "1" → Busy**

**action = "2" → Idle**

**action = "3" → Safe Mode**

## *5.8 Internet Access Status (control = "_6")*

**action = "0" → Disabled**

**action = "1" → Enabled**

node = null
<**eventInfo**>external URL</**eventInfo**>

**action = "2" → Failed**

## *5.9 Progress Report (control = "_7")*

node = null
eventInfo = text

**action = "1" → Update**

**action = "2.1" → Device Adder Info (UPB Only)**

**action = "2.2" → Device Adder Warn (UPB Only)**

**action = "2.3" → Device Adder Error (UPB Only)**

## 5.10 Security System Event (control = "_8")

node = null
eventInfo = null

**action = "0" → Disconnected**

**action = "1" → Connected**

**action = "DA" → Disarmed**

**action = "AW" → Armed Away**

**action = "AS" → Armed Stay**

**action = "ASI" → Armed Stay Instant**

**action = "AN" → Armed Night**

**action = "ANI" → Armed Night Instant**

**action = "AV" → Armed Vacation**

## 5.11 System Alert Event (control = "_9")

Not implemented and should be ignored

## 5.12 OpenADR and Flex Your Power Events (control = "_10")

**Only applicable to ISY994 Z Series.*

**action = "1" → Open ADR Error**

node = null
eventInfo = null

**action = "2" → Open ADR Status Update**

node = null
Namespace: **udiobjects (udiobjs.xsd)**
<**eventInfo**>
udiobjects:OpenADRState
</**eventInfo**>

### action = "5" → Flex Your Power Error

node = null
eventInfo = null

### action = "6" → Flex Your Power Status Updated

node = null
<**eventInfo**>
    <**active**>whether or event has been issued</**active**>
</**eventInfo**>

## *5.13 Climate Events (control = "_11")*

*Requires WeatherBug Module

### action = "0" → Error

node = null
eventInfo = null

### action = "1" → Temperature

node = null
<**eventInfo**>
    <**value**>The value for this event</**value**>
    <**unit**>The unit of measure for this value</**unit**>
</**eventInfo**>

### action = "2" → Temperature High

node = null
<**eventInfo**>
    <**value**>The value for this event</**value**>
    <**unit**>The unit of measure for this value</**unit**>
</**eventInfo**>

### action = "3" → Temperature Low

node = null
<**eventInfo**>
    <**value**>The value for this event</**value**>
    <**unit**>The unit of measure for this value</**unit**>
</**eventInfo**>

### action = "4" → Feels Like

node = null
<**eventInfo**>
    <**value**>The value for this event</**value**>
    <**unit**>The unit of measure for this value</**unit**>
</**eventInfo**>

### action = "5" → Temperature Rate

node = null
<**eventInfo**>
    <**value**>The value for this event</**value**>
    <**unit**>The unit of measure for this value</**unit**>
</**eventInfo**>

### action = "6" → Humidity

node = null
<**eventInfo**>
    <**value**>The value for this event</**value**>
    <**unit**>The unit of measure for this value</**unit**>
</**eventInfo**>

### action = "7" → Humidity Rate

node = null
<**eventInfo**>
    <**value**>The value for this event</**value**>
    <**unit**>The unit of measure for this value</**unit**>
</**eventInfo**>

### action = "8" → Pressure

node = null
<**eventInfo**>
    <**value**>The value for this event</**value**>
    <**unit**>The unit of measure for this value</**unit**>
</**eventInfo**>

### action = "9" → Pressure Rate

node = null
<**eventInfo**>
    <**value**>The value for this event</**value**>
    <**unit**>The unit of measure for this value</**unit**>
</**eventInfo**>

### action = "10" → Dew Point

node = null
<**eventInfo**>
    <**value**>The value for this event</**value**>
    <**unit**>The unit of measure for this value</**unit**>
</**eventInfo**>

### action = "11" → Wind Speed

node = null
<**eventInfo**>
    <**value**>The value for this event</**value**>
    <**unit**>The unit of measure for this value</**unit**>
</**eventInfo**>

### action = "12" → Average Wind Speed

node = null
<**eventInfo**>
    <**value**>The value for this event</**value**>
    <**unit**>The unit of measure for this value</**unit**>
</**eventInfo**>

### action = "13" → Wind Direction

node = null
<**eventInfo**>
    <**value**>The value for this event</**value**>
    <**unit**>The unit of measure for this value</**unit**>
</**eventInfo**>

### action = "14" → Average Wind Direction

node = null
<**eventInfo**>
    <**value**>The value for this event</**value**>
    <**unit**>The unit of measure for this value</**unit**>
</**eventInfo**>

### action = "15" → Gust Wind Speed

node = null
<**eventInfo**>
    <**value**>The value for this event</**value**>
    <**unit**>The unit of measure for this value</**unit**>
</**eventInfo**>

### action = "16" → Gust Wind Direction

node = null
<**eventInfo**>
    <**value**>The value for this event</**value**>
    <**unit**>The unit of measure for this value</**unit**>
</**eventInfo**>

### action = "17" → Rain Today

node = null
<**eventInfo**>
    <**value**>The value for this event</**value**>
    <**unit**>The unit of measure for this value</**unit**>
</**eventInfo**>

### action = "18" → Ambient Light

node = null
<**eventInfo**>
    <**value**>The value for this event</**value**>
    <**unit**>The unit of measure for this value</**unit**>
</**eventInfo**>

### action = "19" → Ambient Light Rate

node = null
<**eventInfo**>
    <**value**>The value for this event</**value**>
    <**unit**>The unit of measure for this value</**unit**>
</**eventInfo**>

### action = "20" → Rain Rate

node = null
<**eventInfo**>
    <**value**>The value for this event</**value**>
    <**unit**>The unit of measure for this value</**unit**>
</**eventInfo**>

### action = "21" → Max Rain Rate

node = null
<**eventInfo**>
    <**value**>The value for this event</**value**>
    <**unit**>The unit of measure for this value</**unit**>
</**eventInfo**>

### action = "22" → Evapotranspiration

node = null
<**eventInfo**>
    <**value**>The value for this event</**value**>
    <**unit**>The unit of measure for this value</**unit**>
</**eventInfo**>

### action = "23" → Irrigation Requirement

node = null
<**eventInfo**>
    <**value**>The value for this event</**value**>
    <**unit**>The unit of measure for this value</**unit**>
</**eventInfo**>

### action = "24" → Water Deficit Yesterday

node = null
<**eventInfo**>
    <**value**>(Signed) The value for this event</**value**>
    <**unit**>The unit of measure for this value</**unit**>
</**eventInfo**>

## *5.14 AMI/SEP Events (control = "_12")*

Only applicable to ISY994 Series!
See OR-WS-SDK-Manual-Energy Management.pdf.

If you do not have this manual, please send an email to support@universal-devices.com.

## *5.15 External Energy Monitoring Events (control = "_13")*

ISY994 Series currently support Brultech ECM1240 Energy Monitors.
**Note**: In ISY994 Series, all these events are incorporated into node events and no longer valid

### action = "1" → Number of Channels

node = null
<**eventInfo**>
    <**numChannels**>The number of monitored channels</**numChannels**>
</**eventInfo**>

### action = "2" → Channel Report

node = null
<**eventInfo**>
    <**channel** num="channel num" sampling=" ">
        <power unit="unit">Instantaneous</power>
        <total unit="unit">Total accumulative</total>
        <voltage unit="unit">voltage</voltage>
        <current unit="unit">current</current>
        <polarized unit="unit">polarized power</polarized>
    </**channel**>
</**eventInfo**>

### action = "3" → Zigbee Status

**Only applicable to ISY994 Z Series.*

node = null
<**eventInfo**>
    "Down " |
    "Establishing PAN" |
    "Established"
</**eventInfo**>

Down = Zigbee Network Down
Establishing PAN = Trying to Establish Network
Established = Network Established and Operational

**action = "7" → Raw Packet**

node = null
<**eventInfo**>
[![CDATA]
Raw binary packet directly from Brultech]
</**eventInfo**>

## *5.16 UPB Linker Events (control = "_14")*

Only applicable to UPB enabled units.

**action = "1" → Device Status**

Status info for a device being linked

**action = "2" → Pending Stop Find**

Device Adder Stop Find Pending

**action = "3" → Pending Cancel Device Adder**

Device Adder Cancel Device Adder Pending

## *5.17 UPB Device Adder State (control = "_15")*

## *5.18 UPB Device Status Events (control = "_16")*

Only applicable to UPB enabled units.

**action = "1" → Device Signal Report**

**action = "2" → Device Signal Report Removed**

## 5.17 Gas Meter Events (control = "_17")

Only applicable to ISY994 Series.

### action = "1" → Status

node = null
<**eventInfo**>
<**total**>The actual meter value</**total**>
<**lastReadTS**>Timestamp of the last read</**lastReadTS**>
</**eventInfo**>

### action = "2" → Error

node = null
eventInfo = null

## 5.18 Zigbee Events (control = "_18")

Only applicable to ISY994 Series.
Namespace: **udiobjects (udiobjs.xsd)**

### action = "1" → Status

node = null
<**eventInfo**>
Zigbee.xsd:ZigbeeStatus
</**eventInfo**>

## 5.19 ELK Events (control = "_19")

Only applicable when ELK Module is installed.
Consult ISY-WS-SDK-ELK

## *5.20 Device Linker Events (control = "_20")*

Device Linking events are all defined in **udievnts.xsd**.

### action = "1" → Status

node = null
<**eventInfo**>DeviceLinkerEventInfo</**eventInfo**>

### action = "2" → Cleared

Device linking list is now cleared
node = null
eventInfo = null

# 6.0 REST Interface

REST is an easy to use URL based command set which allows the developer to communicate and control ISY.

Except for uploading configuration files, all REST commands use HTTP GET method.

## *6.1 Batch Commands*

### /rest/batch

Returns the Batch mode:

```
<batch><status>[0|1]</status></batch>
```

### /rest/batch/on

Turns on Batch mode. Does not write changes to device. Only internal configuration files are updated

### /rest/batch/Off

Turns off Batch mode. Writes all pending changes to devices and no longer buffers changes

### /rest/batteryPoweredWrites

Returns the status of Battery Powered device operations

```
<batteryPoweredWrites>

      <status>[0|1]</status>

</batteryPoweredWrites>
```

### /rest/batteryPoweredWrites/on

Writes all pending changes to battery powered devices when Batch mode is off

### /rest/batteryPoweredWrites/off

Does not write changes to battery powered devices when batch is off

## 6.2 Configuration

For schema, please review the WSDL.

### /rest/config

Returns the configuration of the system with permissible commands

### /rest/sys

Returns system configuration

### /rest/time

Returns system time

### /rest/network

Returns network configuration

### /rest/subscriptions

Returns the state of subscriptions

## 6.3 Nodes

For schema, please review the WSDL.

### /rest/nodes

Returns nodes, scenes, types, and their status

### /rest/nodes/devices

Returns only devices and their status but no scenes

### /rest/nodes/scenes

Returns only scenes but no devices

### /rest/nodes/<node-id>

Returns all the attributes & property values for a specific node

### /rest/nodes/<node-id>?members=true|false

Works on a scene only. Using members=true includes all the scene members in the result

## 6.4 X10

**/rest/X10/<Housecode[Unitcode]>/<X10 command>**
UnitCode and X10 command are both optional

## 6.5 Properties

**/rest/nodes/<node-id>/<property>**
Returns the specific property value for a given node id

**/rest/nodes/<node-id>/set/<property>/<value>**
Insteon - OL/128 – Set On-Level to 50%
UPB - OL/50 – Set On-Level to 50%

**/rest/nodes/<node-id>/write**
Writes all pending changes to the device

**/rest/nodes/<node-id>/cmd/<command_name>/<param1>/<param2>/.../<param5>**
eg:
/rest/nodes/<node-id>/cmd/DOF - turn off a device or a scene
Insteon - /rest/nodes/<node-id>/cmd/DON/128 - turn on a scene to 50%
UPB - /rest/nodes/<node-id>/cmd/DON/50 - turn on a scene to 50%

## 6.6 Status

**/rest/status**
Returns the status for all the nodes

**/rest/status/<node-id>**
Returns the status for the given node

## 6.7 Query

**/rest/query**
Queries all the nodes

**/rest/query/<node-id>**
Queries the given node

## 6.8 Programs

### /rest/programs/<pgm-id>/<pgm-cmd>
e.g. /rest/program/0032/runThen -- Runs a command for a single program

### /rest/programs/<pgm-id>
Returns single program, or folder with folders/programs within it

### /rest/programs/<pgm-id>?folderContents=false
Returns single program or folder

### /rest/programs/<pgm-id>?subfolders=true
Returns single program, or folder with folders/programs within it recursively

### /rest/programs
Returns all the programs in the root folder e.g. same as /rest/programs/<root-pgm-id>

### /rest/programs?folderContents=false
Returns root folder only (same as /rest/programs/<root-pgm-id>?folderContents=false)

### /rest/programs?subfolders=true
Returns all programs & folders (same as /rest/programs/<root-pgm-id>?subfolders=true)

**Defaults**:
folderContents=true, subfolders=false

**pgm-cmd**:
run|runThen|runElse|stop|enable|disable|enableRunAtStartup|
disableRunAtStartup
'runIf' is supported as well, but 'run' should be used instead.

## 6.9 Modules

### /rest/electricity
**Only applicable to 994 Z Series.*

Returns electricity module info and specifically Energy Monitor, Open ADR and Flex Your Power status

**/rest/climate** –
Returns climate module info

**/rest/networking/resources**
Returns the networking resources configuration

**/rest/networking/resources/<resource_id>**
Calls and executes net resource

**/rest/networking/wol**
Returns the networking Wake On LAN configuration

**/rest/networking/wol/<wol_id>**
Calls and executes the WOL resource

## *6.10 Security*

Please see section 6.16 or consult ISY-WS-SDK-ELK documentation for complete integration of ELK security features with ISY.

## *6.11 Energy Management AMI/Smart Grid/SEP*

Please see the Energy Management section.

## *6.12 Gas*

*Only available on 992

**/rest/gmeter**
Returns the status of the gas meter

**/rest/gmeter/log**
Returns gas meter log

**/rest/gmeter?reset=true**
Clears all gas meter log entries

## 6.13 Logs

### /rest/log

Returns system/event log

### /rest/log?reset=true

Clears all system log entries

### /rest/log/error

Return error log

### /rest/log/error?reset=true

Clears all error log entries

## 6.14 Variables

### /rest/vars/init/<var-type>/<var-id>/<value>

Sets the initial value of variable at ISY startup
var-type:
1 – Integer
2 – State

### /rest/vars/set/<var-type>/<var-id>/<value>

Sets a variable given by var-id
var-type:
1 – Integer
2 – State

### /rest/vars/get/<var-type>/<var-id>

Retrieves a variable given by var-id
var-type:
1 – Integer
2 – State

### /rest/vars/get/<var-type>

Retrieves all variables of the given type
var-type:
1 – Integer
2 – State

**Schema (*udiobjects:Variable & udiobjects:Variables*):**
<**var** id="<var-id>" type ="<var-type>">
<**val**>value</**val**>
<**init**>init-value</**init**>
<**ts**>YYYYMDD HH:MM:SS</**ts**>
</**var**>

**ts** is the last changed timestamp of the current value

### /rest/vars/definitions/<var-type>

Retrieves all variable definitions of the given type
var-type:
1 – Integer
2 – State

**Schema (*udiobjects:VarDefinition & udiobjects:VarDefinitions*):**
<**CList type**="VAR_INT">
<**e id**="<var-id>" **name**="<var-name>" />
…
<**e id**="<var-idN>" **name**="<var-nameN>" />
</**CList**>

| | |
|---|---|
| **type** | The low level data type (integer) |
| **var-id** | The variable id |
| **var-name** | The variable name |

## *6.15 Zigbee*

**Only applicable to ISY994 Z Series.*

### /rest/zb

Retrieves the status of Zigbee Network including Joined nodes if any
Namespace: **udizb (zigbee.xsd)**
Returns udiobjects:ZigbeeStatus

### /rest/zb/scanNetwork

Sends a broadcast to all nodes already in the PAN so that they would announce themselves again. This is a good diagnostics tool for orphaned nodes
Namespace: **udizb (zigbee.xsd)**
Returns udiobjects.RestResponse

### /rest/zb/ntable

Retrieves the neighbor table for the COO. This is a good diagnostics tool for retrieving LQI for each node in the PAN
Namespace: **udizb (zigbee.xsd)**
Returns udiobjects.RestResponse

### /rest/zb/nodes

Retrieves all the nodes which have joined the PAN
Namespace: **udizb (zigbee.xsd)**
Returns udiobjects:ZigbeeNodes

### /rest/zb/nodes/[euid]

Retrieves information for the node with the given euid (joined only)
Namespace: **udizb (zigbee.xsd)**
Returns udiobjects:ZigbeeNode

### /rest/zb/nodes/[euid]/ping

Pings the node with the given euid (joined only)
Namespace: **udizb (zigbee.xsd)**
Returns udiobjects:ZigbeeNode if successful, udiobjects.RestResponse otherwise

### /rest/zb/nodes/[euid]/remove

Removes the node with the given euid (joined only) from the PAN
Namespace: **udizb (zigbee.xsd)**

Returns udiobjects.RestResponse

### /rest/zb/nodes/[euid]/ep

Retrieves the active endpoints for a node
Namespace: **udizb (zigbee.xsd)**
Returns udiobjects.RestResponse

## *6.16 ELK*

**Only applicable when ELK Module is installed.*

### /rest/elk/…

Please refer to ISY-WS-SDK-ELK .

# 7.0 Programs

You can setup and configure ISY programs using either SOAP or REST commands.

1. You can use the REST commands to retrieve programs or their status (see section 6.8)
2. You can update programs by POSTing a well-defined (see below for example) XML to /program/upload/ID?key=KEY
    a) Where ID is the 4 digit hexadecimal representation of the program ID you retrieved in #1
    b) KEY is used to let ISY know if there are concurrent changes (see below for more details).

***Warning***: please note that if you upload bad programs or bad keys you might render ISY inoperable

## *7.1 Program IDs*

1. It is the clients responsibility (i.e. your responsibility) to manage program IDs for the full set of programs and folders
2. Every program has a unique ID from 1-1024
3. When a program is deleted its program ID is reused
4. Folders are programs and have a program ID (note: These are different from the folders used for organizing your Insteon devices on the main device page)
5. All programs reference a parent folder by program ID
6. The root folder has a parent of 0
7. The root folder may have any program ID, there is no guarantee it will be 1

## *7.2 Key*

1. All changes to programs are scoped to the entire set of programs by a Key. In other words, if you change one thing it's like you've changed everything
2. When saving/deleting programs you do the following:
    a. Create a new key, using "D2DCommand" "setKey". You must create the new key string: <timestamp>.<random number>
    b. Save all new and changed folders and programs one at a time
    c. Save the program: "/program/upload/<4_digit_hexadecimal_program_id>?key=<current key>"
    d. Enable or Disable the program using "D2DCommand" "**enable**"/"**disable**"
    e. Remove all deleted folders and programs, using "D2DCommand" "delete"

f. Create another new key as in step 1
g. Inform all other clients of the change by sending "D2DCommand" "broadcast". When the admin console receives one of these messages, it reloads the programs

**Note**: Programs may be implicitly deleted using save, for example, You delete program id=7 and then create a new program assigning it the newly available id=7, when it comes time to "Save Changes", you just save program id=7 and the other will implicitly be deleted.

## *7.3 Details*

To send commands to programs you must first get all the programs. This gives you:
- A key value
- The content of all programs, allowing you to map between program name and program ID

The key-value ensures the client is using a current version of the programs. Whenever programs are added/changed/deleted, the key value changes, and thus all clients must get all the programs again.

To get all programs, issue the u:GetAllD2D command or you could use the REST equivalent (see section 6.8):

The XML you get back contains all the programs:

```
<key>key-value</key>
<triggers>
        <d2d>
                <trigger><id>1</id>
                <name>Living Room Program</name>
                ...
                </trigger>
        </d2d>
        <d2d>
                <trigger><id>3</id>
                <name>Kitchen Off</name>
                ... </trigger>
        </d2d> ...
</triggers>
```

To run a command for a program, use the u:D2DCommand, with the following content:
(Note: you do not need to specify the <mask> tag)

```
<u:D2DCommand xmlns:u="urn:udi-com:service:X_Insteon_Lighting_Service:1">
        <id>pgm-id</id>
        <key>key-value</key>
        <CDATA><cmd>cmd-tag</cmd></CDATA>
</u:D2DCommand>
```

You can send the following commands:

```
<enable />
<disable />
<runif />
<runthen />
<runelse />
<runAtReboot />
<notRunAtReboot />
<stop />
```

eg. The following runs the 'Then path of program 'Kitchen Off' using the key-value of 914CEA.6967A0 returned from GetAllD2D

```
<u:D2DCommand xmlns:u="urn:udi-com:service:X_Insteon_Lighting_Service:1">
        <id>3</id>
        <key>914CEA.6967A0</key>
        <CDATA><cmd><runthen /></cmd></CDATA>
</u:D2DCommand>
```

## *7.3 Examples*

**Setting the Key - Sets the current key**
Use the <setKey> subcommand of <D2DCommand>
<key> - The current key
<setKey> - The new key

e.g.

```
<s:Envelope><s:Body><u:D2DCommand xmlns:u="urn:udi-com:service:X_Insteon_Lighting_Service:1">
<key>FFA019C6.7C7E</key>
<CDATA><cmd><setKey>FFA0A130.7C7E</setKey></cmd></CDATA>
</u:D2DCommand></s:Body></s:Envelope>
```

**Deleting a program**
Use the <delete /> subcommand of <D2DCommand>
<id> - The program ID (not in hexadecimal)
<key> - The current key

e.g.
<s:Envelope><s:Body><u:D2DCommand xmlns:u="urn:udi-com:service:X_Insteon_Lighting_Service:1">
<id>10</id>
<key>FFA0A130.7C7E</key>
<CDATA><cmd><delete /></cmd></CDATA>
</u:D2DCommand></s:Body></s:Envelope>

**Enabling a program**
Use the <enable /> subcommand of <D2DCommand>
<id> - The program ID (not in hexadecimal)
<key> - The current key

e.g.
<s:Envelope><s:Body><u:D2DCommand xmlns:u="urn:udi-com:service:X_Insteon_Lighting_Service:1">
<id>10</id>
<key>FFA0A130.7C7E</key>
<CDATA><cmd><enable /></cmd></CDATA>
</u:D2DCommand></s:Body></s:Envelope>

**Saving a program** (used for both creating and changing a program)
Use /program/upload/<4_digit_hexadecimal_program_id>?key=<current key>

e.g. This saves program id=10 (hexadecimal 000A):
https://isy.url/program/upload/000A?key=FFABF47E.7C7E

# 8.0 Logs

You can retrieve logs by using REST commands (see section 6.13)
/rest/log – retrieves system log
/rest/log/error – retrieves error log

All logs are tab delimited with an new line (\n) at the end of each line.

## 8.1 System Log (/rest/log)

System log has the following format:
**Node** – the address of the node to which the log belongs
**Control** – the control that was impacted and which caused the log entry
**Action** – the value of the control
**Time** – in NTP format with epoch of 36524 (see section 7.3)
**UID*** – the user or task which initiated the event
{
*SYSTEM_USER*=0 | *SYSTEM_DRIVER_USER*=1 | *WEB_USER*=2, *SCHEDULER_USER*=3 | *D2D_USER*=4, *ELK_USER*=5 | *SEP_DEVICE_UMETER_USER*=6 | *SEP_DEVICE_UPRICE_USER* | *SEP_DEVICE_UMSG_USER* | *SEP_DEVICE_UDR_USER* | *GAS_METER_USER*
}
**Log Type –** the type of entry … for a list of errors/types **see section 7.4**

To clear System Log, use **/rest/log/reset=true**.
* *udiobjs.xsd:SystemActor*

## 8.2 Error Log (/rest/log/error)

Error log has the following format:
**Time** – in NTP format with epoch of 36524 (see section 7.3)
**UID*** – the user or task which initiated the event
{
*SYSTEM_USER*=0 | *SYSTEM_DRIVER_USER*=1 | *WEB_USER*=2, *SCHEDULER_USER*=3 | *D2D_USER*=4, *ELK_USER*=5 | *SEP_DEVICE_UMETER_USER*=6 | *SEP_DEVICE_UPRICE_USER* | *SEP_DEVICE_UMSG_USER* | *SEP_DEVICE_UDR_USER* | *GAS_METER_USER*
}
**Log Type –** the type of entry … for a list of errors/types **see section 7.4**
**Error Message –** free form text

To clear Error Log, use **/rest/log/error/reset=true**.
* *udiobjs.xsd:SystemActor*

## 8.3 Converting NTP Formatted Time

For efficiency, Time in the log is an unsigned integer (4 Bytes) formatted using NTP with the following parameters:

EPOCH_OFFSET = 36524 (01/01/200)
SEC_IN_DAY=86400
EPOCH_YEAR = 2000
EPOCH_DAY = 1
EPOCH_MONTH = 1
YEAR_STARTS_WITH_DAY=1

There are very many code libraries that support conversion from NTP/UTC time to a Time structure. This said, the following code snippet should get you started

```
        /**
         * Returns a <code>DateTime</code> object from an NTP
         * representation of date/time
         * @param cv - the NTP representation of date/time
         * @param bt - the <code>DateTime</code> object which is
         * returned (may be empty)
         * @param epochOffset - the offset to be used from epoch ;
         * USE EPOCH_OFFSET
         * @return a <code>DateTime</code> object converted from NTP
         */
        public static DateTime FromNTP( long cv, DateTime bt, long
epochOffset )
        {

           int w;
           long x = epochOffset*SEC_IN_DAY;
           long  tv = ( cv - x );
           long tmp = 0;

           bt.dow = (int)( ( ( cv / SEC_IN_DAY ) + 1 ) % 7 );

           for ( w = EPOCH_YEAR; tmp <= tv; w++ )
           {
              tmp += DAYS_IN_YEAR( w ) * SEC_IN_DAY;
           }

           w--;

           tmp -= DAYS_IN_YEAR( w ) * SEC_IN_DAY;


           bt.year = w;
           tv -= tmp; /* Now we are ready for days */

           bt.day_of_year = (int)( tv / SEC_IN_DAY ) ;

           //+ YEAR_STARTS_WITH_DAY ) %
                        //   ( IsLeap( bt.year ) ? 366 : 365 );
           tv = tv % SEC_IN_DAY;

           bt.hour = (int) tv / 3600;
           bt.min = (int)( tv / 60 ) % 60;
           bt.sec =(int) tv % 60;
```

```
    bt = FixMonthDay( bt );
    if (YEAR_STARTS_WITH_DAY == 1)
      bt.day_of_year++;
    return bt;
}

 /**
  * Fixes the day/month combination
  * @param bt - a <code>DateTime</code> object
  * @return the fixed <code>DateTime</code> object
  */
 public static DateTime FixMonthDay( DateTime bt )
 {
    bt.month = 0;
    bt.day = 0;
    long dn = bt.day_of_year;

    if ( IsLeap( bt.year ) )
    {
       if ( dn == 59 )
       {
          bt.month = 2;
          bt.day = 29;
          return bt;
       }
       else if ( dn > 59 )
       {
          dn--;
       }
    }

    /* Now we find the month */
    for ( bt.month = 1; bt.month < 12 && monthdays[bt.month + 1]
      <= dn; bt.month++ )
      ;



    bt.day = (int) dn - monthdays[bt.month] + 1;
      /* Month starts with 1 not 0 */
    return bt;
}

 /**
  * Returns whether or not the year is a leap year
  * @param year - the year
  * @return - true if leap year, false otherwise
  */
 public static boolean IsLeap( int year )
{
    return ( ( year % 4 ) == 0 );
}
```

## *8.4 Log/Error Types*

All Log related objects are defined in:
Schema: **log.xsd** and **udiobjs.xsd (***uo:LogInfo***)**; namespace=**ulog**
Instances:

1. **loginfo.xml** : defines all log information and messages
2. **insterr.xml** : sub definition for INSTEON error messages
3. **upberr.xml**: sub definition for UPB error messages
4. **smtperr.xml**: sub definition for SMTP error messages
5. **netstat.xml**: sub definition for Network Layer error messages

# Appendix A – INSTEON Device Categories/Subcategories

## *A1. Device Categories*

All system wide and high level device categories are defined in:
Schema: **family.xsd**; namespace=**ufamily**
Instance: **cat.xml**

## *A2. Device Sub-Categories*

Device sub-categories are defined in the respective family instances:
Schema: **family.xsd**; namespace=**ufamily**
Instance: **[familyId]_fam.xml**
e.g. INSTEON Family ID=1 → **1_fam.xml**

# Appendix B – UPB Device Types

Device types are stored in the type field of a node and are in the form <MID>.<PID>.<firmware_level>, where the MID is the manufacturer ID, the PID is a product ID within a MID.

## *B1. PCS Device Types*

PCS Device Types currently supported:
WS1D,WS1E,WS1N,WMC6,DTC6,WMC8,DTC8,AM1,FMR1,KPC6,KPLR6,KPLR8,KPLD6,KPLD8,KPC8,LM1,LM2,FMD2,OCM,ICM2,DBM,ICM,SPR,TPR

PIDs for PCS (MID == 1)

```
WS1D     1   // Wall Switch - 1 Channel - Dimmer
WS1N     2   // Wall Switch - 1 Channel - Non-Dimmer
WMC6     3   // Wall Mount Controller - 6 Button
WMC8     4   // Wall Mount Controller - 8 Button
CRM      5   // Controlled Receptacle Module
OCM      6   // Output Control Module - 2 Channel
LCM1     7   // Load Control Module - 1 Channel
LCM2     8   // Load Control Module - 2 Channel
LM1      9   // Lamp Module - 1 Channel
LM2     10   // Lamp Module - 2 Channel
ICM2    11   // Input Control Module - 2 Channel

DTC6    13   // Desktop Controller - 6 Button
DTC8    14   // Desktop Controller - 8 Button
AM1     15   // Appliance Module - 1 Channel
CLC1    16   // Commercial Lighting Controller
CFC     17   // Commercial Fixture Controller
RRS     18   // Phase Repeater Slave
RRM     19   // Phase Repeater Master
MSC     20   // Motion Sensor Controller
HFC     21   // HID Fixture Controller
RNM     22   // Repeater Master Chip
RNS     23   // Repeater Slave Chip
WS1E    24   // Wall Switch - 1 Channel - Electronic low voltage Dimmer
LSM     25   // Load Shedding Module
TEC     26   // Timed-Event Controller
RFC     27   // Radio-Frequency Input Controller
RNC     28   // Repeater Network Controller
XUB     29   // X10-to-UPB Bridge

VHC     32   // Vacuum Handle Controller
VPC     33   // Vacuum Power Controller
VIM     34   // Vacuum Interrupt Module
VPM     35   // Vacuum Pan Module
DBM     36   // Doorbell Module
TCM     37   // Telephone Control Module

SPR     40   // Split-Phase Repeater
```

```
PIM     41   // Powerline Interface Module

DIAG1   49   // Diagnostics 1
DIAG2   50   // Diagnostics 2
DIAG3   51   // Diagnostics 3
DIAG4   52   // Diagnostics 4
TX4     53   // Test Transmitter 4
TM4     54   // Test Module 4
UTM     55   // UPB Test Module
ATM     56   // UPB Alpha Test Module
RIM     57   // Repeater Powerline Interface Module

FMD2    60   // Inline Fixture Dimmer Module - 2 Channel
FRM     61   // Inline Fixture Relay Module
WS2D    62   // Wall Switch Dimmer - Bar Graph
KPLD6   63   // Controller
SCM     64   // Shade Control Module
KPC6    65   // Controller
KPC8    66   // Controller

KPLD8   69   // 8-Button Keypad Light Dimmer
```