Universal Devices, Inc. Wiki

ISY Developers:API:REST Interface

ISY_Developers : API : REST Interface

REST Interface

Notes

- All calls use the HTTP GET method (Except for uploading configuration files)
- Not all features are implemented in the Rest API, for now, some operations are only available via the SOAP API

Return Values / Codes

- All calls will return the requested data (if there is any) or Boolean status for the command

Successful :

```
<RestResponse succeeded="true">
    <status>200</status>
</RestResponse>
```

Error :

```
<RestResponse succeeded="false">
   <status>404</status>
</RestResponse>
```

Configuration

/rest/config

returns the configuration of the system with permissible commands

See also Example config output

/rest/sys

Returns system configuration

Example :

```
<SystemOptions>
 <MailTo/>
 <HTMLRole>3</HTMLRole>
 <CompactEmail>false</CompactEmail>
 <QueryOnInit>true</QueryOnInit>
 <PCatchUp>true</PCatchUp>
 <PGracePeriod>900</PGracePeriod>
 <WaitBusyReading>true</WaitBusyReading>
 <NTPHost>0.north-america.pool.ntp.org</NTPHost>
 <NTPActive>true</NTPActive>
 <NTPEnabled>true</NTPEnabled>
 <NTPInterval>43200</NTPInterval>
</SystemOptions>
```

/rest/time

Returns system time

Example :

```
<DT>
 <NTP>3578531154</NTP>
 <TMZOffset>-28800</TMZOffset>
 <DST>true</DST>
 <Lat>37.766701</Lat>
 <Long>122.416702</Long>
 <Sunrise>3578536351</Sunrise>
 <Sunset>3578588453</Sunset>
 <IsMilitary>true</IsMilitary>
</DT>
```

/rest/network

Returns network configuration

Example :

```
<NetworkConfig>
 <Interface isDHCP="true">
  <ip>10.1.2.36</ip>
 </Interface>
 <WebServer>
  <httpPort>80</httpPort>
  <httpsPort>443</httpsPort>
 </WebServer>
</NetworkConfig>
```

/rest/subscriptions

Returns the state of subscriptions

Example :

```
<Subscriptions>
 <Sub isExpired="yes" isPortal="no" sid="-1" sock="-1"
isReusingSocket="no" isConnecting="no"/>
 <Sub isExpired="no" isPortal="no" sid="40" sock="29"
isReusingSocket="yes" isConnecting="no"/>
 <Sub isExpired="yes" isPortal="no" sid="-1" sock="-1"
isReusingSocket="no" isConnecting="no"/>
  ....
</Subscriptions>
```

Nodes

/rest/nodes

returns nodes, scenes, types, and their status

/rest/nodes/scenes

returns scenes only

/rest/nodes/<node>

returns all the attributes & property values for a specific node

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

this only works on a scene. using members=true includes all the scene members in the result

/rest/nodes/<node>/enable

Enables a device

Returns: Success or Error status

/rest/nodes/<node>/disable

Disable a device

Returns: Success or Error status

/rest/nodes/<node>/notes

Returns the notes for the node

Schema (Notes):

```
<NodeProperties>
  <location>Free form text</location>
  <description>Free form text</description>
  <isLoad></isLoad>
  <spoken>The spoken name</spoken>
</NodeProperties>
```

Schema (Node):

```
<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 />
</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

Flags:

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

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

Schema (Scene):

```
<group flag="(see Node flags)">
  <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>
```

Node Type (Hierarchy) Flags:

NODE_TYPE_NOTSET 0 (unknown)
NODE_TYPE_NODE 1
NODE_TYPE_GROUP 2
NODE_TYPE_FOLDER 3

Schema (Folder):

```
<folder flag="(see Node flags)">
  <address>The address of the Folder</address>
  <name>Friendly name</name>
</folder>
```

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

- Node/Scene/Folder/Program name are not unique
- Scene Folder addresses are unique only with in their group type

Properties

/rest/nodes/<node>/<property>
returns the specific property value for a given node id

Example :

```
<properties>
    <property id="ST" value="0" formatted="Off" uom="%/on/
off"/>
</properties>
```

/rest/nodes/<node>/set/<property>/<value>
set a value such as OL/250

Commands

/rest/nodes/<node>/cmd/<command_name>/<param1>/<param2>/.../<param5>

eg:

/rest/nodes/<node>/cmd/DOF
turn off a device or a scene

/rest/nodes/<node>/cmd/DON/
turn on a device

- Insteon - /rest/nodes/<node-id>/cmd/DON/128 - turn on a scene to 50% (valid parameters = 0 - 255)
- UPB - /rest/nodes/<node-id>/cmd/DON/50 - turn on a scene to 50% (valid parameters = 0 - 100)

/rest/nodes/<node>/cmd/DFON

turn on a device fast

/rest/nodes/<node>/cmd/DFOF
turn off a device fast

/rest/nodes/<node>/cmd/BRT
increase brightness of a device by ~3%

/rest/nodes/<node>/cmd/DIM
decrease brightness of a device by ~3%

/rest/nodes/<node>/cmd/BMAN
begin manual dimming

/rest/nodes/<node>/cmd/SMAN
stop manual dimming

Fanlinc : 0 - 0 : Off 1% - 49% : Slow 50% - 99% : Medium 100% : Fast

eg:

/rest/nodes/<fan node>/cmd/DON/49
turn the fan to slow

Status

/rest/status
returns the status for all the nodes

Example

```
<nodes>
 <node id="15 4B 53 1">
  <property id="ST" value="0" formatted="Off" uom="%/on/
off"/>
 </node>
 <node id="16 38 C0 1">
  <property id="ST" value="0" formatted="Off" uom="%/on/
off"/>
 </node>
 <node id="16 3C 43 1">
  <property id="ST" value="0" formatted="Off" uom="%/on/
off"/>
 </node>
  ....
```

```
</nodes>
```

/rest/status/<node>

returns the status for the given node

Example

```
<properties>
   <property id="ST" value="0" formatted="Off" uom="%/on/
off"/>
</properties>
```

Query

/rest/query

queries all the nodes
Returns: Success or Error status

/rest/query/<node>

queries the given node
Returns: Success or Error status

X10

/rest/X10/<Housecode>/<X10>

UnitCode and X10 command are both optional

List of X10 commands

Code	Function	Description	One Way	Two Way
13	All units off	Switch off all devices with the house code indicated in the message	X	
5	All lights on	Switches on all lighting devices (with the ability to control brightness)	X	
1	All lights off	Switches off all lighting devices	X	
3	On	Switches on a device	X	
11	Off	Switches off a device	X	
15	Dim	Reduces the light intensity	X	
7	Bright	Increases the light intensity	X	
9	Extended code	Extension code		X
14	Hail request	Requests a response from the device(s) with the house code indicated in the message		X
6	Hail acknowledge	Response to the previous command		X
12	Pre-set dim	Allows the selection of two predefined levels of light intensity		X
8	Status is on	Response to the Status Request indicating that the device is switched on		X
2	Status is off	Response indicating that the device is switched off		X
10	Status request	Request requiring the status of a device		X

Please note this mapping is not equal to the actual X10 binary codes.

Programs

/rest/programs/<pgm>/<cmd>

Runs a command for a single program

Returns: Success or Error status

/rest/programs/<pgm>
returns single program, or folder with folders/programs within it

/rest/programs/<pgm>?folderContents=false
returns single program or folder

/rest/programs/<pgm>?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>

/rest/programs?folderContents=false
returns root folder only (same as /rest/programs/<root>?folderContents=false)

/rest/programs?subfolders=true
returns all programs & folders (same as /rest/programs/<root>?subfolders=true)

Defaults:

- folderContents=true
- subfolders=false

Example:

/rest/programs/0032/runThen

pgm-cmd

- run|runThen|runElse|stop|enable|disable|enableRunAtStartup|disableRunAtStartup

'runIf' is supported as well, but 'run' should be used instead

Schema :

```
 <program id="0053" parentId="0001" status="true"
folder="true">
  <name>Am Dim</name>
```

```
  <lastRunTime></lastRunTime>
  <lastFinishTime></lastFinishTime>
 </program>
```

Logs

/rest/log
Returns system/event log

/rest/log?reset=true
Clears all system log entries
Returns: Success or Error status

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

/rest/log/error
Return error log

/rest/log/error?reset=true
Clears all error log entries
Returns: Success or Error status

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

Variables

For the following command related to variables, <var-type> =

1 = Integer
2 = State

/rest/vars/init/<var-type>/<var-id>/<value>
Sets the initial value of variable at ISY startup

/rest/vars/set/<var-type>/<var-id>/<value>
Sets a variable given by var-id

/rest/vars/get/<var-type>/<var-id>
Retrieves a variable given by var-id

/rest/vars/get/<var-type>
Retrieves all variables of the given type

Schema :

```
<var id="<var-id>" type ="<var-type>">
    <val>value</val>
    <init>init-value</init>
    <ts>YYYYMDD HH:MM:SS</ts>
</var>
```

/rest/vars/definitions/<var-type>
Retrieves all variable definitions of the given type

Schema :

```
<CList type="VAR_INT">
    <e id="<var-id>" name="<var-name>" />
    …
```

```
    <e id="<var-idN>" name="<var-nameN>" />
 </CList>
```

Subscriptions (Web Sockets)

Starting with firmware version 4.2.3, you can now use REST to subscribe to ISY using Web Sockets.

Web Sockets are persistent HTTP/S connections that can be used in HTML5 and Javascript.

For more information on Web Sockets: RFC6455 (http://datatracker.ietf.org/doc/rfc6455/?include_text=1)

The process is quite simple: 1. Use a Web Socket request to /rest/subscribe 2. Ensure that the following headers are included a. Authorization. Refer to section 4.2 for more details. b. Sec-WebSocket-Protocol: ISYSUB c. Sec-WebSocket-Version: 13 d. Origin: com.universal-devices.websockets.isy 3. The initial return is the subscription response which includes the subscription id 4. ISY will continue publishing events to the client as long as the socket remains open. Refer to section 5 in ISY WS-SDK Manual for more details on events.

Here is an example HTML page that implements a websocket connection to ISY, displaying the received events in the browser window. This page must either be hosted directly on the ISY user web space to function, or you must specially configure a product like Apache to proxy requests to your ISY, and exempt the path where your html file is hosted. For an example of this configuration, see here

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>WebSocket ISY Client</title>
  <meta charset="UTF-8" />
  <script>
    "use strict";
    // Initialize everything when the window finishes
loading
    window.addEventListener("load", function(event) {
      var status = document.getElementById("status");
      var url = document.getElementById("url");
      var open = document.getElementById("open");
      var close = document.getElementById("close");
      var message = document.getElementById("message");
      var socket;

      status.textContent = "Not Connected";
      var isy_host = window.location.host;
      var x = location.protocol;
```

```
      if (x == "https:") {
        var ws_host = "wss://" + isy_host
      } else {
          var ws_host = "ws://" + isy_host
      }
      url.value = ws_host + "/rest/subscribe";
      close.disabled = true;

      // Create a new connection when the Connect button is
clicked
      open.addEventListener("click", function(event) {
        open.disabled = true;
        socket = new WebSocket(url.value, "ISYSUB");

        socket.addEventListener("open", function(event) {
          close.disabled = false;
          status.textContent = "Connected";
                  console.log("Connected");
        });

        // Display messages received from the server
        socket.addEventListener("message", function(event) {
                  document.getElementById("log").innerText =
event.data + String.fromCharCode(13) +
document.getElementById("log").innerText;
        });

        // Display any errors that occur
        socket.addEventListener("error", function(event) {
          message.textContent = "Error: " + event;
                  console.log("Error: " + event)
        });

        socket.addEventListener("close", function(event) {
          open.disabled = false;
          status.textContent = "Not Connected";
                  console.log("Not Connected");
        });
      });

      // Close the connection when the Disconnect button is
clicked
      close.addEventListener("click", function(event) {
        close.disabled = true;
        message.textContent = "";
        socket.close();
```

```
      });

    });
  </script>
</head>
<body>
  Status: <span id="status"></span><br />
  <input id="url" type="hidden" />
  <input id="open" type="button" value="Connect" />
  <input id="close" type="button" value="Disconnect" /><br /
>
  <span id="message"></span>
  <p id="log"></p>
</body>
</html>
```

Modules

Electricity

/rest/electricity

returns electricity module info

Climate

/rest/climate

returns climate module info

Schema (example):

```
<?xml version="1.0" encoding="UTF-8"?>
<climate enabled="true" locationId="BRUCB" rss="http://
api.wxbug.net/getLiveWeatherRSS.aspx?
ACode=AA2&stationId=BRUCB&unittype=0">
   <Temperature>67.1 F</Temperature>
   <Temperature_High>81 F</Temperature_High>
   <Temperature_Low>60 F</Temperature_Low>
   <Feels_Like>67 F</Feels_Like>
   <Temperature_Rate>2.6 F/h</Temperature_Rate>
   <Humidity>29 %</Humidity>
   <Humidity_Rate>-11 %/h</Humidity_Rate>
   <Pressure>29.74 inches</Pressure>
   <Pressure_Rate>0 inches/h</Pressure_Rate>
   <Dew_Point>34 F</Dew_Point>
```

```
   <Wind_Speed>1 mph</Wind_Speed>
   <Wind_Average_Speed>0 mph</Wind_Average_Speed>
   <Wind_Direction>ESE</Wind_Direction>
   <Wind_Average_Direction>ESE</Wind_Average_Direction>
   <Gust_Speed>9 mph</Gust_Speed>
   <Gust_Direction>SSW</Gust_Direction>
   <Rain_Today>0 inches</Rain_Today>
   <Light>0 %</Light>
   <Light_Rate>0 %/h</Light_Rate>
   <Rain_Rate>0 inches/h</Rain_Rate>
   <Max_Rain_Rate>0 inches/h</Max_Rain_Rate>
   <Evapotranspiration>0.0637 inches/day</
Evapotranspiration>
   <Irrigation_Requirement>9.8963 inches</
Irrigation_Requirement>
   <Water_Deficit_Yesterday>0.0637 inches</
Water_Deficit_Yesterday>
   <Elevation>935 </Elevation>
</climate>
```

Networking

/rest/networking/resources

Returns the networking resources configuration / resource_id list

Schema (example):

```
<NetConfig>
 <NetRule>
  <name>Camera Start</name>
  <id>40</id>
  <isModified>false</isModified>
  <ControlInfo>
   <mode>URL-Encoded</mode>
   <protocol>http</protocol>
   <host>10.1.1.43</host>
   <port>80</port>
   <timeout>500</timeout>
   <method>GET</method>
   <encodeURLs>true</encodeURLs>
  </ControlInfo>
 </NetRule>
  ....
</NetConfig>
```

/rest/networking/resources/<resource_id>

Calls and executes net resource
Returns: Success or Error status

/rest/networking/wol

Returns the networking Wake On LAN configuration / wol_id list

Schema (example):

```
<NetConfig>
 <NetRule>
  <name>NAS</name>
  <id>1</id>
  <isModified>false</isModified>
  <mac>00-01-55-12-2E-AC</mac>
  <subnet>10.1.1.255</subnet>
 </NetRule>
 ....
</NetConfig>
```

/rest/networking/wol/<wol_id>

Calls and executes the WOL resource
Returns: Success or Error status

/rest/security

returns security module info

The following have been disabled for security reasons

- /rest/security/<code>/arm/stay
- /rest/security/<code>/arm/away
- /rest/security/<code>/disarm

Misc

/rest/locations

returns the floorplan

Schema :

```
<UDFloorPlan>
 <locations>
  <location ISIMPORTEDFROMMODEL="false">
   <preferredBounds WIDTH="134" HEIGHT="208" Y="2" X="10"/>
   <preferredLocation Y="15" X="334"/>
   <address>0.45816479823871636</address>
   <name>dinn</name>
   <nodes>
    <node ISLABELONLY="false" ISCONTROLLER="true"
TYPE="single">
     <preferredBounds WIDTH="150" HEIGHT="37" Y="0" X="0"/>
     <preferredLocation Y="42" X="18"/>
     <deviceUUID>uuid:00:21:b9:01:0c:e7</deviceUUID>
     <name>Dinning Room Light</name>
     <address>16 38 C0 1</address>
    </node>
    ....
   </nodes>
  </location>
  ...
</locations>
</UDFloorPlan>
```

Zigbee

Only applicable to ISY994 Z Series.

/rest/zb
: Retrieves the status of Zigbee Network including Joined nodes if any

/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

/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

/rest/zb/info
: Retrieves Zigbee radio information such as EUID and firmware

/rest/zb/reset
Soft resets the Zigbee radio

/rest/zb/restoreDefaults
Factory resets the Zigbee radio

/rest/zb/nodes
Retrieves all the nodes which have joined the PAN

/rest/zb/nodes/[euid]
Retrieves information for the node with the given euid (joined only)

/rest/zb/nodes/[euid]/ping
Pings the node with the given euid (joined only)

/rest/zb/nodes/[euid]/remove
Removes the node with the given euid (joined only) from the PAN

/rest/zb/nodes/[euid]/ep
Retrieves the active endpoints for a node

Z-Wave

Only applicable to ISY994 Series with the Z-Wave Dongle (http://wiki.universal-devices.com/index.php?title=Z-Wave:_Ordering/Assembly_Instructions).

/rest/zwave/node/include?[power=true|false]&[*nwi*=true|false]
Add a device into the Z-Wave network.

power : True=High Power, False=Normal Power (default)
nwi : True=Network wide inclusion, False=Standard Inclusion (default)

/rest/zwave/node/exclude
Remove a device into the Z-Wave network.

/rest/zwave/node/cancel
Cancel include/exclude/replication.

/rest/zwave/sendPrimary
As Primary Controller, replicate to another controller and make it the new

Primary Controller.

/rest/zwave/learnMode

Go into Z-Wave learn mode (to replicate, be added/removed from Z-Wave network, etc.).

/rest/zwave/sync?[id=<nodeAddress>]|[uid=<zwaveUnitId>]

Synchronize ISY with info on Z-Wave dongle for the given node.

id : The ISY node address
uid : The Z-Wave unit ID of the device
Specify either id or uid. If neither is specified then sync all new & deleted nodes.

/rest/zwave/sync/full?[id=<nodeAddress>]|[uid=<zwaveUnitId>]

Synchronize ISY with info on Z-Wave dongle for the given node.

id : The ISY node address
uid : The Z-Wave unit ID of the device
Specify either id or uid. If neither is specified then sync all nodes.

/rest/zwave/factoryReset/dongle?[force=true|false]

Factory reset the Z-Wave dongle.

force : True=Force factory reset, False=Factory reset if dongle is not part of existing Z-Wave network.

/rest/zwave/set/antenna?[0|1]

Sets active antenna; switches between internal and external.

0 - Internal Antenna
1 - External Antenna

Version 4.3.26+ and 5.0.6+ only

/rest/zwave/node/<nodeAddress>/config/query/<parameterNumber>

Query zwave device parameter
/rest/zwave/node/ZW003_1/config/query/2
returns something like:
<config paramNum="2" size="1" value="80"/>

/rest/zwave/node/<nodeAddress>/config/set/<parameterNumber>/<value>/<size>

Set zwave device parameter
The parameter size can be either 1,2, or 4 bytes
/rest/zwave/node/ZW003_1/config/set/1/77/1
/rest/zwave/node/ZW003_1/config/set/2/0xFFFFFFFF/4

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

Batch Commands

/rest/batch

Returns the Batch mode:

Example:

```
<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

Energy Monitoring

REST Interface for ECM

/rest/nodes/<nodeId>/stopRT

Stops Real Time reporting for an ECM

/rest/nodes/<nodeId>/startRT

Starts Real Time reporting for an ECM

/rest/nodes/<nodeId>/reset

Resets accumulated values for an ECM

/rest/nodes/<nodeId>/cfg

Retrieves all the configuration information for the given ECM the XML for which is as follows:

```
<EMonConfig>
 <ct1 type="167" range="4"/>
 <ct2 type="167" range="4"/>
 <pt type="131" range="6"/>
 <rtInterval>real time interval (sec) </rtInterval>
```

```
  <dlInterval>data logger interval (sec) </dlInterval>
  <firmware>1026</firmware>
  <id>unique id for the unit</id>
  <serial>serial number for the unit</serial>
  <aux constant="151" options="62">
   <trim1>0</trim1>
   <trim2>0</trim2>
   <trim3>0</trim3>
   <trim4>0</trim4>
   <trim5>0</trim5>
   <trim6>0</trim6>
  </aux>
  <k1 h="142" l="141"/>
  <k2 h="142" l="141"/>
  <kv0>212</kv0>
  <option>0</option>
  <trigger>200</trigger>
 </EMonConfig>
```

/rest/nodes/<nodeId>/setCT?type=<T>&range=<R>

Sets CT configurations for a given channel (identified by the node). Only channels 1 and 2 are supported.

T = Types as defined by ECM (100 | 167)
R = Range as defined by ECM (3 | 4 | 5)

/rest/nodes/<nodeId>/setPT?type=<T>&range=<R>

Sets PT configurations on the given ECM.

T = Types as defined by ECM (?)
R = Range as defined by ECM (?)

/rest/nodes/<nodeId>/setInt?value=<V>

Sets Real Time interval (in seconds) on the given ECM.

V = The interval in seconds

/rest/nodes/<nodeId>/setTrig?value=<T>

Sets the Trigger value (in watts) on the given ECM.

T = The trigger value in Watts

/rest/nodes/<nodeId>/setAux?value=<O>

Sets the Aux options on the given ECM.

O = A byte bitmap as follows (Gain is X2): Bit 0 = Aux 1 Gain Bit 1 = Aux 2 Gain Bit 2 = Aux 3 Gain Bit 4 = Aux 4 Gain Bit 5 = Aux 5 Gain Bit 5 = Aux 5 Is used for Counting when set Bit 6 = Aux 5 is DC bipoloar Bit 7 = ? Please note that these options are retrieved in the configuration in the option attribute of the aux element: <aux constant="151" options="62">

/rest/nodes/<nodeId>/toggPol

Toggles the polarity for the given channel (node) on the given ECM.

REST Interface for GreenEye Monitor

Prefix: /rest/nodes/<nodeId>/cmd

/rest/nodes/<nodeId>/stopRT

Stops Real Time reporting for an ECM

/rest/nodes/<nodeId>/startRT

Starts Real Time reporting for an ECM

/rest/nodes/<nodeId>/reset?type=<T>

Resets certain counters based on the type.

1: Reset Pulse Counter 1
2: Reset Pulse Counter 2
3: Reset Pulse Counter 3
4: Reset Pulse Counter 4
5: Reset All Pulse Counters
6: Reset All Counters
7: Reset All Seconds Counters
8: Reset Seconds Counter for the Node in <nodeId>

/rest/nodes/<nodeId>/cfg

Not implemented.

/rest/nodes/<nodeId>/setCT?type=<T>&range=<R>

Not Implemented

/rest/nodes/<nodeId>/setPT?type=<T>&range=<R>

Sets PT configurations on the given ECM.

T = Types as defined by ECM (?)
R = Range as defined by ECM (?)

/rest/nodes/<nodeId>/setInt?value=<V>

Sets Real Time interval (in seconds) on the given ECM.

V = The interval in seconds

/rest/nodes/<nodeId>/setTrig?value=<T>

Not Implemented

/rest/nodes/<nodeId>/setAux?value=<O>

Not Implemented

/rest/nodes/<nodeId>/toggPol

Not Implemented

REST Interface for EM3

/rest/nodes/<nodeId>/stopRT

Stops Real Time reporting for an EM3

/rest/nodes/<nodeId>/startRT

Starts Real Time reporting for an EM3

/rest/nodes/<nodeId>/reset?type=*

Resets various parameters in EM3.

1 = Reset accumulated values
2 = Reset configuration parameters
4 = Reset Zigbee
8 = Reset Pulse Count

can be a bitwise OR of the following:

*

/rest/nodes/<nodeId>/cfg

Retrieves all the configuration information for the given EM3 the XML for which is as follows:

```
    <EM3Config debug="Boolean" realtime="Boolean"
tempUnit="F|C" kyzMode="Boolean">
        <powerReportInterval>integer (seconds)</
powerReportInterval>
        <vaReportInterval>integer (seconds)</
vaReportInterval>
        <tempReportInterval>integer (seconds)</
tempReportInterval>
        <pulseReportInterval>integer (seconds)</
pulseReportInterval>
        <powerStorageInterval>integer (seconds)</
powerStorageInterval>
        <pulseStorageInterval>integer (seconds)</
pulseStorageInterval>
    </EM3Config>
```

/rest/nodes/<nodeId>/setCT?type=<T>&range=<R>

Currently not supported.

/rest/nodes/<nodeId>/setPT?type=<T>&range=<R>

Currently not supported.
Universal Devices, Inc.

/rest/nodes/<nodeId>/setInt?type=<T>&value=<V>

Sets Real Time interval (in seconds) for a various parameters, defined in T, on the given EM3.

T: could be any of the following:

1 = Power reporting interval
2 = Voltage/Current/Powerfactor (va) reporting interval
3 = Temperature reporting interval
4 = Pulse reporting interval
5 = Energy storage interval
6 = Pulse storage interval

V: The interval in seconds

/rest/nodes/<nodeId>/setTrig?value=<T>

Sets the Trigger value (in watts) on the given EM3.

T = The trigger value in Watts

/rest/nodes/<nodeId>/setOption?type=<OT>&value=<O>

Sets operating parameters for the EM3.

OT = is an option type which could be any of the following:

1 = Temp unit

Where O could be:

0=F

4=C

2 = Debug

Where O could be:

0 = Debug Off

1 = Debug On

3 = KYZ Mode

Where O could be:

0 = KYZ Mode Off

1 = KYZ Mode On

Please note that these options are retrieved in the configuration in the option attribute of the aux element:

<aux constant="151" options="62">

Portal Integration

/rest/whoami

This interface returns uniquely identifying attributes of the given ISY to which the Proxy Server is connected:

Schema :

```
    <iam>
        <partner>A string representing the partner ID</
partner>
        <type>A string representing the type of ISY</type>
        <product>ISY’s product ID which represents a model
number</product>
        <id>Hexadecimal MAC address</id>
        <SID>
            Subscription ID if any  this element will not be
there if
            ISY has no active subscription to the Portal.
Otherwise,
            the value is always: uuid:0
        </SID>
```

```
        <electricity>
            <providerId>
                A String representing the Utility provider
ID if any
            </providerId>
            <enrollmentGroup>
                An Integer representing the Utility
enrollment group if any
            </enrollmentGroup>
         </electricity>
    </iam>
```

Example:

```
        <iam>
            <partner>Portal-X-Partner</partner>
            <type>ISYv30</type>
            <product>1110</product>
            <id>0021b9030405</id>
            <SID>uuid:0</SID>
            <electricity>
                <providerId>SDGE</providerId>
                <enrollmentGroup>0</enrollmentGroup>
            </electricity>
        </iam>
```

ELK Integration

Area Commands

/rest/elk/areas/query

Queries all areas and changes to states are published through event infrastructure (see section 3.2).

/rest/elk/area/<areaId>/get/status

Retrieves the status of the given area.

areaId : Defined in elkobjs.xsd:uelk:AreaIDType
Response : elkobjs.xsd:uelk:AreaResponseType

/rest/elk/area/<areaId>/cmd/bypass?code=<accessCode>

Bypasses all violated burglar alarms in the area given in areaId

areaId : Defined in elkobjs.xsd:uelk:AreaIDType
accessCode : Optional; Defined by elkobjs.xsd:uelk:AccessCode

/rest/elk/area/<areaId>/cmd/unbypass?code=<accessCode>

Unbypasses all burglar alarms in the area given in areaId

areaId : Defined in elkobjs.xsd:uelk:AreaIDType
accessCode : Optional; Defined by elkobjs.xsd:uelk:AccessCode

/rest/elk/area/<areaId>/cmd/display?
text=<eText>&beep=<boolean>&offTimerSeconds=<seconds>

Displays text given in eText to all keypad in the given area.
Optionally beep and turn off after offTimerSeconds.

areaId : Defined in elkobjs.xsd:uelk:AreaIDType
eText : URL escaped text. 2 Lines with 16 characters per line max. Use "^" to separate lines if less than 16 characters.
e.g. text=”Hello^World”
beep : Optional; True then beep when displaying the message
seconds : Optional; Number of seconds to display the message

/rest/elk/area/<areaId>/cmd/display?
id=<notifyId>&beep=<boolean>&offTimerSeconds=<seconds>

Displays formatted text to all keypad in the given area given a custom content ID.
This uses the same custom content entries as are used for email notifications.
Optionally beep and turn off after offTimerSeconds.

areaId : Defined in elkobjs.xsd:uelk:AreaIDType
notifyId : The id of the customized content entry
beep : Optional; True then beep when displaying the message
seconds : Optional; Number of seconds to display the message

/rest/elk/area/<areaId>/cmd/arm?
armType=<elkArmType>&code=<accessCode>

areaId : Defined in elkobjs.xsd:uelk:AreaIDType
elkArmType : Defined by elkobjs.xsd:uelk:ArmType
accessCode : Optional; Defined by elkobjs.xsd:uelk:AccessCode

/rest/elk/area/<areaId>/cmd/disarm?code=<accessCode>

areaId : Defined in elkobjs.xsd:uelk:AreaIDType
accessCode : Optional; Defined by elkobjs.xsd:uelk:AccessCode

Zone Commands

/rest/elk/zones/query

Queries all zones and changes to states are published through event

infrastructure (see section 3.3).

/rest/elk/zones/<zoneId>/query/voltage

Queries the zone given by zoneId and changes to states are published through event infrastructure (see section 3.3).
To query voltage for all zones, specify a zoneId of 0 (zero).

zoneId : Defined in elkobjs.xsd:uelk:ZoneIDType

/rest/elk/zone/<zoneId>/query/temperature

Queries the thermostats for the given zone and changes to states are published through event infrastructure (see section 3.3).
To query temperature for all zones, specify a zoneId of 0 (zero).

zoneId : Defined in elkobjs.xsd:uelk:ZoneIDType

/rest/elk/zone/<zoneId>/cmd/trigger/open

Momentarily triggers a zone to the physical state of Open.
An error will occur if the zone is defined as normally open, or is currently open.

zoneId Defined in elkobjs.xsd:uelk:ZoneIDType

/rest/elk/zone/<zoneId>/cmd/toggle/bypass?code=<accessCode>

Toggles bypass for the zone given in zoneId

zoneId : Defined in elkobjs.xsd:uelk:ZoneIDType
accessCode : Optional; Defined by elkobjs.xsd:uelk:AccessCode

General Commands

/rest/elk/query/all

Gets the status for all areas, zones, outputs, counters, etc. and publishes the state changes through event infrastructure (see section 3).

/rest/elk/get/status

Returns the status of all of zones, areas, keypads, outputs, etc. configured in ELK.

Response : elkobjs.xsd:uelk:ELKAllStatus

/rest/elk/get/topology

Returns the representation of all the zones, areas, keypads, outputs, etc. configured in ELK.

Response : elkobjs.xsd:uelk:Topology

/rest/elk/refresh/topology

Causes ISY to recreate/refresh the topology. Returns the representation of

all the zones, areas, keypads, outputs, etc. configured in ELK.
Response : elkobjs.xsd:uelk:Topology

System Commands

/rest/elk/system/get/status
Returns the enabled and connected status.
Response : elkobjs.xsd:uelk:SystemResponseType

Keypad Commands

/rest/elk/keypad/<kpId>/cmd/press/funcKey/<fkId>
Simulates pressing the fkId button on keypad given in kId
kpId : Keypad number (1-8)
fkId : Defined in elkobjs.xsd:uelk:FunctionKeyType

/rest/elk/keypad/<kpId>/query/temperature
Queries the status of temperature sensor on a keypad and changes to states are published through event infrastructure : To query temperature for all keypads, specify a kpId of 0 (zero).

/rest/elk/keypad/<kpId>/get/status
Retrieves the status of a keypad.
To get status for all keypads, specify a kpId of 0 (zero).
kpId : Keypad number (1-8)
Response : elkobjs.xsd:uelk:KeypadResponseType
Note: See Area Commands for displaying text on keypads

Output Commands

/rest/elk/outputs/query
Queries all outputs and changes to states are published through event infrastructure

/rest/elk/output/<outputId>/get/status
Returns the status of the given output
To get status for all outputs, specify an outputId of 0 (zero).
outputId : elkobjs.xsd:uelk:OutputIDType
Response : elkobjs.xsd:uelk:OutputResponseType

/rest/elk/output/<outputId>/cmd/on?offTimerSeconds=<seconds>
Turns On an output defined by outputId. Optional offTimerSeconds can be defined such that the output is turned back off after the amount of seconds

given in <seconds>.
outputId : elkobjs.xsd:uelk:OutputIDType
seconds : Optional; Interval after which the relay turns off

/rest/elk/output/<outputId>/cmd/off
Turns Off an output defined by outputId
outputId : elkobjs.xsd:uelk:OutputIDType

Audio Commands

Audio commands allow communications with A/V equipment that have been configured and attached to ELK.

/rest/elk/audio/zone/<audioZone>/source/<audioSource>/cmd/<audioCommand>?value=<audioValue>
Sends the command given in audioCommand to zone given in audioZone.
audioZone : Audio Zone number (1-18)
audioSource : Audio Source number (1-12)
audioCommand : Defined by elkobjs.xsd:uelk:AudioCommandType
audioValue : 3-digit decimal Number, currently only used for Volume

Voice Announcement Commands

Voice Announcement commands cause predefined words or phrases to be spoken by the ELK security system.

/rest/elk/speak/word/<wordId>
Instructs ELK to speak the word given by wordId.
wordId : Defined by elkobjs.xsd:uelk:VoiceWordType

/rest/elk/speak/phrase/<phraseId>
Instructs ELK to speak the phrase given by phraseId.
phraseId : Defined by elkobjs.xsd:uelk:VoicePhraseType

Thermostat Commands

Thermostat commands impact a thermostat .

/rest/elk/tstat/<tstat_id>/query
Instructs ELK query the thermostat given by tstat_id and state changes are published as events (see section 3.7).
tstat_id : Defined by elkobjs.xsd:uelk:ThermostatIDType

/rest/elk/tstat/<tstat_id>/get/status

Retrieves the status of the given thermostat.
tstat_id : Defined by elkobjs.xsd:uelk:ThermostatIDType
Response : elkobjs.xsd:uelk:ThermostatResponseType

/rest/elk/tstat/<tstat_id>/cmd/<cmd_id>?value=value
Instructs ELK to apply the value using command defined by cmd_id to thermostat defined by tstat_id.
tstat_id Defined by elkobjs.xsd:uelk:ThermostatIDType
cmd_id : Defined by elkobjs.xsd:uelk:ThermostatCommandType
value : Depends on the command:
Temperatures (such as setpoints): 1-99
Mode: elkobjs.xsd:uelk:ThermostatModeState
Fan: elkobjs.xsd:uelk:ThermostatFanStat

OpenADR (VEN)

/rest/oadr
receive full Payload

/rest/oadr/status
receive the connection status to VTN

/rest/oadr/<event-id>/optin
opt in to an event

/rest/oadr/<event-id>/optout
opt out to an event

/rest/oadr/clear
clear all events

/rest/oadr/party/register
Register to VTN

/rest/oadr/party/unregister
Cancel Registration to VTN

/rest/oadr/party/queryreg
Query Registration

/rest/oadr/party/reregister?type=val
Re-Registration
Where type is *refresh* | *renew*
refresh = uses the existing Registration ID

renew = does not use the existing Registration ID

/rest/oadr/report/register?type=val

Register report Meta Data

Where type is **usage** | **status** | **history** | **all** [default]

/rest/oadr/report/status

Get status of all reports (oadrobjs:OADRReportsStatus)

/OpenADR2/Simple/EiEvent

For 2.0a Push mode

/OpenADR2/Simple/2.0b/[Service]

For 2.0b Push mode, ISY endpoint is:

Where service is:

EiEvent - Event services
EiReport - Reporting services
EiOpt - Scheduling services

Billing

These REST URLs are used to provide utility billing information when used in conjunction with Smart Meters (ZS Series).

/rest/billing/today[/reset]

returns the billing information for today (WSDL/billobjs.xsd::Billing)

/rest/billing/yesterday[/reset]

returns the billing information for yesterday (WSDL/billobjs.xsd::Billing)

/rest/billing/cycle[/reset]

returns the billing information for the current cycle (WSDL/billobjs.xsd::Billing)

/rest/billing/month/year

returns the billing information for the given month/year cycle (WSDL/billobjs.xsd::Billing)

/rest/billing/report/today

returns the hourly billing information for today (WSDL/billobjs.xsd::BillingReport)

/rest/billing/report/yesterday

returns the hourly billing information for yesterday (WSDL/billobjs.xsd::BillingReport)

/rest/billing/report/month/year

returns the daily billing information for the given month/year cycle (WSDL/billobjs.xsd::BillingReport)

Related Pages

- Networking
- Quick Start Guide
- User Guide
- How-To Guide
- Example Java REST Requester Aplication
- Advanced Configuration Guide
- Frequently Asked Questions

ISY-99i Series INSTEON:
