# SimpleHomeNet

Simplifying the modern home

# EZSrve XML API
# Reference Document

CIL-5010API-DD
Version 2.0
September 28, 2009

Compacta International, Ltd.
dba SimpleHomeNet
30011 Ivy Glenn Drive • Suite 123
Laguna Niguel, CA 92677
Phone 949.429.3303 • Fax 949.429.8053

# Contents

# 1. Overview

This document describes the data structures and the methods used in the EZSrve that implement a complete home/building automation client/server architecture. EZSrve uses xml files to represent its data structures, and xml functions to perform all the required operations on those data structures. EZSrve acts on the physical world and interacts with its clients via the xml structures and functions. The description of the data structures follows, and the xml Application Programming Interface (API) is detailed subsequently. The API also includes numerous functions that help manage the EZSrve and allow direct access to the devices.

# 2. Devices.xml

This file maintains all information needed to control and monitor devices. As such, it contains the instantiation of device objects with all their attributes represented as nodes in xml. The main interaction between the EZSrve and clients is via API functions that manipulate this file. The format of this file is as follows:

**File Template**

```
<?xml version="1.0"?>
<Devices>
   <Device Rec="" Name="" ID="" DevCat="" MSB="" PollInt="" Pic="">
<Clusters>
   <Cluster CID="" Attribute_1="" Attribute_n="" />
</Clusters>
<Links>
   <Link Rec=""  ID=""  Grp="" Cntrlr="" LD="" SCR=""/>
</Links>
</Device>
<Scenes>
   <Scene SCR="1" Name=" "/>
   <Scene SCR="2" Name=" "/>
</Scenes>
</Devices>
```

| Nodes/Tags | Description |
|---|---|
| <Device> | Holds the top level attributes of the device record |
| Rec | The device record number – value: 0-9999 |
| Name | String of up to 16 characters with the device name. |
| DevCat | Device/Category code – value: 0x0000-0xFFFF |
| MSB | If Insteon, this is the size of the links db – 0 if small, 1 if large. |
| PollInt | Polling interval for main cluster – value: 0-99 (in 5 sec. int.) |
| Pic | String of up to 16 characters, usually the name of a picture. |
| <Clusters> | Holds the Cluster nodes |
| <Cluster> | Holds the attributes of a cluster |
| CID | ID of the cluster – value: 1-255 |
| <Links> | Holds the link records |
| <Link> | Holds the attributes of a link record |
| Rec | Link record number – value: 0-9999 |
| ID | ID of the device affected by this link. Form: 01.FE.3E |
| Grp | Group number – value 1-254 |
| Cntrlr | Flag to indicate type of link – 0 if Responder, 1 if Controller |
| LD | Link data of the form 01-FE-1D |
| SCR | Scene record number this link is a member of. Values: 0-255 |
| <Scenes> | Holds the scene records |
| <Scene> | Holds the attributes for a scene record |
| SCR | Record number – Value: 0-255 |
| Name | String of up to 16 characters with the Scene name. |
| Pic | String of up to 16 characters, usually the name of a picture. |

**Method Examples**

Retrieve entire file:

```
<Command Name = "Read" File = "Devices"/>
```

Retrieve device record through record number:

```
<Command Name = "Read" File ="Devices">
   <Device Rec =”01”/>
</Command>
```

Retrieve device record through device name:

```
<Command Name = "Read" File ="Devices">
```

```
    <Device Name =”Dimmer 1“/>
  </Command>
```

Retrieve device record through device ID:

```
  <Command Name = "Read" File ="Devices">
    <Device ID=”06.00.71“/>
  </Command>
```

Retrieve complete device links table (by device name shown):

```
  <Command Name = "Read" File ="Devices">
    <Device Name=Dimmer 1“/>
        <Links />
  </Command>
```

Retrieve specific link record by device ID:

```
  <Command Name = "Read" File ="Devices">
    <Device ID =”06.00.71“/>
        <Link Rec=”25”/>
  </Command>
```

Retrieve all clusters through Device ID:

```
  <Command Name=”Read” File =” Devices”/>
    <Device ID=”06.00.71“>
        <Clusters />
    </Device>
  </Command>
```

Retrieve a specific cluster through Device ID:

```
  <Command Name=”Read” File =”Devices”/>
    <Device ID=”06.00.71“>
        <Cluster CID=”250”/>
    </Device>
  </Command>
```

Retrieve all clusters through Device ID:

```
  <Command Name=”Read” File =”Devices”/>
    <Device ID=”06.00.71“>
        <Clusters />
    </Device>
  </Command>
```

Retrieve complete device scenes table:

```
  <Command Name = "Read" File =" Devices ">
    <Scenes />
  </Command>
```

Add a complete file:

```
  <Command Name="Write" File="Devices">
    <valid device file/>
  </Command>
```

Add a device by providing all the details:

```
  <Command Name="Write" File="Devices">
    <Device Name="Dimmer1" ID="06.00.91" DevCat="0x0107" PollInt="1" Pic=”Dimmer.jpeg”/>
  </Command>
```

Add a device by providing Name and ID:

```
  <Command Name="Write" File="Devices">
    <Device Name="Dimmer2" ID="06.00.71" />
  </Command>
```

Add a device by providing just the name (Push Button Method):

```
  <Command Name="Write" File="Devices">
    <Device Name="Dimmer3" />
  </Command>
```

*Please note in the above format that if the device name already exists, the application deletes the device from Devices.xml. This is because there is no difference in the XML format to delete a device from the file, and to add new device.*

In all the above 3 methods providing PollInt and Pic attribute is optional

Modify device name:

```
  <Command Name="Write" File="Devices">
    <Device Rec=”1” Name="New Name">
    </Device>
  </Command>
```

Modify device ID (replace a device):

```
<Command Name="Write" File="Devices">
  <Device Rec=”1” ID="New ID">
  </Device>
</Command>
```

*After this operation, EZSrve automatically attempts to load all the parameters from the new device.*

Modify device Poll Int:

```
<Command Name="Write" File="Devices">
  <Device Rec=”1” PollInt="5">
  </Device>
</Command>
```

Modify device Pic:

```
<Command Name="Write" File="Devices">
  <Device Rec=”1” Pic="NewImage.jpeg">
  </Device>
</Command>
```

Modify device links:

```
<Command Name="Write" File="Devices">
  <Device Name="Dimmer">
    <Links>
      <Link Rec="1" ID="06.00.5D" Grp="01" Cntrlr="1"
        LD="FF-FF-00" SCR=”1”/>
      <Link Rec="2" ID="06.00.5E" Grp="01" Cntrlr="1"
        LD="FF-FF-00" SCR=”2”/>
    </Links>
  </Device>
</Command>
```

Modify device scenes:

```
<Command Name="Write" File="Devices">
  <Scenes>
    <Scene SCR=”01” Name="Default_01"/>
    <Scene SCR=”02” Name="Scene_01"/>
  </Scenes>
</Command>
```

Modify device clusters:

```
<Command Name="Write" File="Devices">
  <Device Name="EZServe">
    <Clusters>
      <Cluster CID="1" PLM="0D.FD.02" Rev="63"/>
      <Cluster CID="2" Lat="40.714630" Lon="-74.005806"
  TimeZone="EST"/>
      <Cluster CID="3" IP="192.168.1.118" NetMask="255.255.255.0"
        GateWay="192.168.1.0"/>
    </Clusters>
  </Device>
</Command>
```

Modify device Names:

```
<Command Name="Write" File="Devices">
  <Device Name="lamp2">
    <Names N1="harshaOne" N2="harshaTwo" N3="harshaThree"  N4="Name4"
N5="Name5" N6="Name6" N7="Name7"  N8="Name8" N9="Name9" N10="Name10"
N11="Name11"  N12="Name12" N13="Name13" N14="Name14" N15="Name15"
  N16="Name16" />
  </Device>
</Command>
```

Delete device links:

```
<Command Name="Write" File="Devices">
  <Device Name=”dimmer”>
    <Links/>
  </Device>
</Command>
```

Delete device clusters:

```
<Command Name="Write" File="Devices">
  <Device Name=”dimmer”>
    <Clusters/>
```

```
    </Device>
  </Command>
```

Delete device Names:

```
  <Command Name="Write" File="Devices">
    <Device Name=”dimmer”>
        <Names/>
    </Device>
  </Command>
```

Delete a device by name:

```
  <Command Name="Write" File="Devices">
    <Device Name=”dimmer”/>
  </Command>
```

## 3. Areas.xml

This file maintains the logical grouping of devices as specified by the user. An example follows:

**File Template**

```
  <?xml version="1.0"?>
  <Areas>
    <Area Rec=”” Name="" Pic=””>
        <Devices>
            <Device Name=""/>
        </Devices>
    </Area>
  </Areas>
```

| Nodes/Tags | Description |
|---|---|
| <Area> | Holds the top level attributes of the area record |
| Rec | The device record number – value: 0-999 |
| Name | String of up to 16 characters with the area name. |
| Pic | String of up to 16 characters, usually the name of a picture. |
| <Devices> | Holds the device records contained in the area. |
| <Device> | Holds the attributes of a device contained in an area. |
| Name | Name of the device contained in the area |

**Method Examples**

Retrieve an entire file:

```
  <Command Name="Read" File="Areas"/>
```

Retrieve a specific Area by name:

```
  <Command Name="Read" File="Areas">
    <Area Name="Living Room" >
        <Devices/>
    </Area>
  </Command>
```

Retrieve a specific Area by record number:

```
  <Command Name="Read" File="Areas">
    <Area Rec="1">
        <Devices/>
    </Area>
  </Command>
```

Clear the entire file (delete all area records):

```
  <Command Name="Write" File="Areas">
```

Add a complete file:

```
  <Command Name="Write" File="Areas">
    <Areas>
        <Area Rec="1" Name="house">
            <Devices>
                <Device Name="Dimmer"/>
                <Device Name="Sprinkler"/>
                <Device Name="Icon Lamp Linc"/>
            </Devices>
        </Area>
        <Area Rec="2" Name="House">
```

```
            <Devices>
                <Device Name="Dimmer"/>
            </Devices>
        </Area>
    </Areas>
</Command>
```

Add an area with several devices:

```
<Command Name="Write" File="Areas">
    <Area Name=”New Living Room”>
        <Devices>
            <Device Name=”Garage door”/>
            <Device Name=”Garage light”/>
        <Devices/>
    </Area>
</Command>
```

Modify the device list of an area:

```
<Command Name="Write" File="Areas">
    <Area Name=”Living Room”/>
        <Devices>
            <Device Name=”Hallway”/>
            <Device Name=”Gateway”/>
        </Devices>
    </Area>
</Command>
```

Modify an area name

```
<Command Name="Write" File="Areas">
    <Area Rec=”10” Name=”newname”/>
</Command>
```

Delete device list of an area

```
<Command Name="Write" File="Areas">
    <Area Name=”Living Room”/>
        <Devices/>
</Command>
```

Delete an area

```
Command Name="Write" File="Areas">
    <Area Name=”Living Room”/>
</Command>
```

## 4. Actions.xml

Actions are events triggered by the EZSrve on up to 6 devices in response to up to 6 specified time and/or device status conditions. The system allows for 255 actions that are stored in the Actions.xml database. Conditions may be AND’ed or OR’ed, however the system will not handle more than 1 level of nesting.

Each of the specified effects can be either immediate, or delayed by a preset time. Time conditions include an absolute time and Sunrise or Sunset plus or minus (+/-) an offset of HH:MM:SS, or triggered at a set interval (e.g. event triggers every 10 minutes), with the first trigger occurring on system startup or when the event is first added. The day interval can be every day, specified days of the week (Sun-Sat), specified days known as “holidays”, every other day or every even or odd day.

Actions may be simple or complex. A complex one may allow for up to 6 set conditions, with each condition an AND / OR. For example, IF (A AND B) OR (C AND D) OR E THEN Y. A simple action could just define an effect happening at a preset time of the week such as IF A THEN Y. The format of the Actions file is explained below.

**File Template**

```
<?xml version="1.0"?>
<Actions>
    <Action Rec="" Name="" Active="">
        <Conditions>
            <Condition Time="09:00" Days=""/>
        </Conditions>
        <Effects>
            <Effect Device="Pool Pump" DevAct="1300" Time="14:00:00"/>
        </Effects>
```

```
    </Action>
  </Actions>
```

| **Nodes/Tags** | **Description** |
|---|---|
| <Action> | Top level of an action record containing the global attributes of the action record |
| Rec | The action record number – value: 0-9999 |
| Name | String of up to 16 characters with the name of the Action |
| Active | Indicates whether the Action is active (enabled) |
| <Conditions> | Holds the Conditions nodes |
| <Condition> | Holds the attributes of a Condition |
| Logic | "OR" or "AND" in combination with a preceding condition. Invalid for the first condition in the list. |
| Device | A given device if a device event match is to occur. |
| AttrName | The attribute of the device to match |
| AttrVal | The value of the attribute to match |
| AtTime | The time to delay the effect |
| StartTime | A starting time for repeating conditions (when using intervals) |
| Days | The days pattern |
| Active | Indicates whether the condition is enabled. |
| <Effects> | Holds the Effects nodes |
| <Effect> | Holds the attributes of Effect |
| Device | The device to be affected |
| AttrName | The name of the device attribute to change |
| AttrVal | The value to set the attribute to |
| Time | A delay to apply to the effect |

**Method Examples**

Retrieve entire file:

```
<Command Name="Read" File="Actions"/>
```

Retrieve a specific Action by record #:

```
<Command Name="Read" File="Actions">
    <Action Rec="01"/>
</Command>
```

Retrieve a specific Action by name:

```
<Command Name="Read" File="Actions">
    <Action Name="Outside Timer"/>
</Command>
```

Retrieve a specific Action Condition by name:

```
<Command Name="Read" File="Actions">
    <Action Name="Outside Timer"/>
    <Conditions/>
</Command>
```

Retrieve a specific Action Effect by name:

```
<Command Name="Read" File="Actions">
    <Action Name="Outside Timer"/>
    <Effects/>
</Command>
```

Clear the file:

```
<Command Name="Write" File="Actions"/>
```

Add complete file:

```
<Command Name="Write" File="Actions">
    <Actions>
        <Action Rec="0" Name="action1" Active="1">
            <Conditions>
                <Condition Logic="OR" AtTime="Absolute"
    StartTime="05:00:00" Days="SMT" Active="1"  />
            </Conditions>
            <Effects>
                <Effect Device="keypadlinc" AttrName="Status"
    AttrVal="0x00" />
            </Effects>
        </Action>
    </Actions>
</Command>
```

Add an action:

```
<Command Name="Write" File="Actions">
```

```
    <Action Rec="0" Name="action1" Active="1">
        <Conditions>
            <Condition Logic="OR" AtTime="Absolute" StartTime="05:00:00"
    Days="SMT" Active="1"  />
        </Conditions>
        <Effects>
            <Effect Device="keypadlinc" AttrName="Status" AttrVal="0x00" />
        </Effects>
    </Action>
  </Command>
```

Modify the name of an action:

```
  <Command Name="Write" File="Actions">
    <Action Rec="1" Name="New action name">
    </Action>
  </Command>
```

Modify a condition:

```
  <Command Name="Write" File="Actions">
    <Action Name="Pool Pump Timer">
        <Conditions>
        <Condition Logic="OR" AtTime="Absolute" StartTime="05:00:00"                    Days="SMT"
  Active="1"  />
        </Conditions>
    </Action>
  </Command>
```

Modify an effect:

```
  <Command Name="Write" File="Actions">
    <Action Name="Pool Pump Timer">
        <Effects>
            <Effect Device="keypadlinc" AttrName="Status" AttrVal="0x00" />
        </Effects>
    </Action>
  </Command>
```

Delete a condition:

```
  <Command Name="Write" File="Actions">
    <Action Name="Pool Pump Timer">
        <Conditions/>
    </Action>
  </Command>
```

Delete an effect:

```
  <Command Name="Write" File="Actions">
    <Action Name="Pool Pump Timer">
    <Effects/>
    </Action>
  </Command>
```

Delete an action:

```
  <Command Name="Write" File="Actions">
    <Action Name= “harsha” />
  </Command>
```

## 5. Holidays.xml

This file maintains the list of holidays. These are days when the timer function will ignore its timer setting (won’t run.) Its format follows:

**File Template**

```
  <?xml version="1.0"?>
  <Holidays>
    <Holiday Rec="1" Date="03Mar08"/>
    <Holiday Rec="2" Date="10Mar08"/>
    <Holiday Rec="3" Date="25Dec08"/>
  </Holidays>
```

| Nodes/Tags | Description |
|---|---|
| <Holiday> | Holds the attributes of a holiday record |
| Rec | The action record number – value: 0-9999 |

| | |
|---|---|
| Date | String representing date formatted as: 10Mar09 |
| Method Examples | |
| Retrieve entire file:<br><Command Name="Read" File="Holidays"/> | |
| Clear out the file:<br><Command Name = "Write" File="Holidays"/> | |
| Add holidays:<br><Command Name = "Write" File="Holidays"><br><Holidays><br><Holiday Rec="1" Date="03Mar08"/><br><Holiday Rec="2" Date="26Jan09"/><br><Holiday Rec="3" Date="10Jan10"/><br></Holidays><br></Command> | |

## 6. DevClusters.xml

This is an internal EZSrve file that describes the methods applicable to each cluster attribute. It does, therefore, become the means by which EZSrve distinguishes the physical devices. The format of this file is not shown in this document as a client application would not normally be concerned with reading or writing this file.

## 7. Internal Functions

EZSrve performs multiple functions that are facilitated by a concise API. These functions include management of the devices, their applicable functions such as automatic Actions, as well as many other internal functions.

## 8. XML API

| ***8.1 Read – Retrieve an XML file or one of its nodes from the EZSrve*** | |
|---|---|
| *Prototype* | <Command Name="Read" File="filename" /> |
| **Parameter** | **Description** |
| *Filename* | The name of the file to be retrieved excluding the .xml extension<br>In the most general case, the entire file is retrieved.<br>A given node or one of its sub nodes can be retrieved by specifying it for matching. |
| *Response* | <Response Name="Read" File="filename" Status="Success"><br><file contents/><br></Response><br>Or a failure message if malformed xml or other error |
| | |
| ***8.2 Write – Write an XML file or one of its nodes to the EZSrve*** | |
| *Prototype* | <Command Name="Write" File="Devices"><br><Device Rec="1" ID="01.34.e4"/><br></Command> |
| **Parameter** | **Description** |
| *Filename* | The name of the file to be written excluding the .xml extension<br>In the most general case, the entire file is written or cleared (if empty nodes).<br>A given node or one of its sub nodes can be written by specifying it for matching. |
| *Response* | <Response Name="Write" File="filename" Status="In Progress"><br></Response><br>followed by:<br><Response Name="Write" File="filename" Status="Success"><br>or a failure message if malformed xml or other error |
| | |
| ***8.3 ReadCluster – Read Attribute(s) in a Cluster*** | |
| *Prototype* | <Command Name="ReadCluster" Device="Device Name"><br><Cluster CID="#" Attribute1="Read" Attibute2="Read" AttributeN="Read"/><br></Command> |
| **Parameter** | **Description** |
| *Filename* | Cluster with the attribute(s) value pairs to be Read The function will handle from 1 to all the attributes in the cluster that are set to "Read" |
| *Response* | <Response Name="ReadCluster" Device="Device Name" Status="Success"><br><Cluster CID="#" Attribute1="Value"/> |

| | |
|---|---|
| | </Response><br>The above message will contain the desired attributes in the cluster |
| | |
| ***8.4 WriteCluster – Write Attribute(s) in a Cluster*** | |
| *Prototype* | <Command Name=”WriteCluster” DevID=”xx.xx.xx”><br><Cluster CID=”#” Attribute1=”Value” Attibute2=”Value” AttributeN=”Value”/><br></Command> |
| **Parameter** | **Description** |
| *Filename* | Cluster with the attribute(s) value pairs to be written. The function will handle from 1 to all the attributes in the cluster. |
| *Response* | <Response Name=”WriteCluster” DevID=”xx.xx.xx” Status”Success”><br><Cluster CID=”#” Attribute1=”Value”/><br></Response><br>The above message is repeated for each affected Attribute<br>Or a failure message if not successful |
| *Example* | 1. Changing parameters in the EZSrve itself<br>a) Set a new latitude/longitude location:<br><Command Name="WriteCluster" DevID="03.45.f6"><br><Cluster CID="2" Lat="33.532029" Lon="-117.702148" TimeZone="PST"/><br></Command><br>b) Changing to static IP:<br><Command Name="WriteCluster" DevID="03.45.f6"><br><Cluster CID="3" MAC="EZSrveMAC" DHCP="0" IP="192.168.1.118" Port="80" Gateway="192.168.1.254" NetMask="255.255.255.0" NTP="EZSrveNTP"/><br></Command><br><br>2. Actuating an INSTEON dimmer device<br>a) Adjust the brightness (ON) level:<br><Command Name="WriteCluster" DevID="07.4F.6B"/><br><Cluster CID=”0” Status="0x3B"/><br></Command><br>b) Modifying the dimmer presets<br><Command Name="WriteCluster" DevID="07.4F.6B"><br><Cluster CID=”1” Level=”0x40” Rate=”0x1F” ResumeLevel=”0xE0”/><br></Command><br><br>3. Controlling and configuring the Irrigation controller<br>a) Turn valve #3 On<br><Command Name="WriteCluster" DevID="05.2B.96"/><br><Cluster CID=”0” Status="0x82" Config=”0x08”/><br></Command><br>b) Changing the Valve 1 timer value<br><Command Name="WriteCluster" DevID="05.2B.96"/><br><Cluster CID="250" DV1="0x10" DV2="0x20" DV3="0x30" DV4="0x40" DV5="0x50" DV6="0x60" DV7="0x70" DV8="0x80" /><br></Command><br><br>4. Controlling an X10 device<br>a) Turn the device ON<br><Command Name="WriteCluster" DevID="A05"/><br><Cluster CID=”0” Status="0x02"/><br></Command><br>b) Increase brightness<br><Command Name="WriteCluster" DevID="A05"/><br><Cluster CID=”0” Status="0x05"/><br></Command><br>c) Request device status (not supported by all X10 devices)<br><Command Name="WriteCluster" DevID="A05"/><br><Cluster CID=”0” Status="0x0f"/><br></Command> |
| *Notes* | For any cluster, the <Cluster> node specifies from 1 to several attribute value pairs. Only those attributes are affected.<br>The function can be used for 1 cluster only. |

| | |
|---|---|
| | Only attributes in a node beyond the identifier are updated. There can be as many attributes as the node contains in the original file. No new attributes will be created. |

# 9. Device Management

| ***9.1 Link – Create a link between two or more devices with an optional scene name, and update the XML database*** | |
|---|---|
| *Prototype* | <Command Name="LinkDevs" Scene="scenename"><br><Cntrlrs><br><Cntrlr ID="01.34.56" Grp="1"/><br><Cntrlr ID="89.37.D4" Grp="3"/><br></Cntrlrs><br><Rspndrs><br><Rspndr ID="01.54.D4" LD="FF-1F-00"/><br><Rspndr ID="89.D7.84" LD="FF-1F-00"/><br><Rspndr ID="4F.5C.32" LD="FF-1F-00"/><br></Rspndrs><br></Command> |
| **Parameter** | **Description** |
| *Filename* | Controllers and responders as desired. Link data is specified for responders. |
| *Returns* | <Response Name="LinkDevs" Scene="scenename" Status="In Progress"><br></Response><br>Followed by:<br><Response Name="LinkDevs" Scene="scenename" Status="Success"><br></Response><br>Or a failure indication. |
| *Example* | 1. Create link between controller and responder devices and assign the provided scene name<br><Command Name="LinkDevs" Scene="scenename"><br><Cntrlrs><br><Cntrlr ID="01.34.56" Grp="1"/><br></Cntrlrs><br><Rspndrs><br><Rspndr ID="01.54.D4" LD="FF-1F-00"/><br><Rspndr ID="4F.5C.32" LD="FF-1F-00"/><br></Rspndrs><br></Command><br><br>2. Create link between controller and responder devices and assign the default scene name as "Scene-xx"<br><Command Name="LinkDevs" ><br><Cntrlrs><br><Cntrlr ID="01.34.56" Grp="1"/><br></Cntrlrs><br><Rspndrs><br><Rspndr ID="01.54.D4" LD="FF-1F-00"/><br><Rspndr ID="4F.5C.32" LD="FF-1F-00"/><br></Rspndrs><br></Command><br><br>3. Add controller and create link with all existing responders of this scene<br><Command Name="LinkDevs" Scene="scenename"><br><Cntrlrs><br><Cntrlr ID="01.34.56" Grp="1"/><br></Cntrlrs><br></Command><br><br>4. Add responder and create link with all existing controllers of this scene<br><Command Name="LinkDevs" Scene="scenename"><br><Rspndrs><br><Rspndr ID="01.54.D4" LD="FF-1F-00"/><br></Rspndrs><br></Command><br><br>5. Add controller and responder and create link with all existing controller and responders of this scene respectively |

| | |
|---|---|
| | <Command Name=”LinkDevs” Scene=”scenename”><br><Cntrlrs><br><Cntrlr ID=”01.34.56” Grp=”1”/><br></Cntrlrs><br><Rspndrs><br><Rspndr ID=”4F.5C.32” LD=”FF-1F-00”/><br></Rspndrs><br></Command> |
| *Notes* | The link can be between any devices, including the EZSrve. If any device is not already in the EZSrve database, and the other device is not the EZSrve, then the device is first added to EZServe database.<br>API doesn’t takes care of detecting duplicate Scene name while creating new scene<br>ID and LD are case sensitive |

| | |
|---|---|
| ***9.2 UnLinkDevs – Remove link between two or more devices and update the XML database (Case 1: Delete a device)*** | |
| *Prototype* | <Command Name=”UnLinkDevs” ID=”04.B6.22”/> |
| **Parameter** | **Description** |
| *Filename* | ID: ID of the device which user wants to delete from the EZServe database |
| *Returns* | <Response Name=” UnLinkDevs” Status=” Success”><br></Response> |
| *Example* | Delete the 04.B6.22 device from the EZServe database<br><Command Name=”UnLinkDevs” ID=”04.B6.22”/> |
| *Notes* | The link can be between any 2 or more devices, including the EZSrve. The scene name, if given, can be the only parameter to delete all the associated links. |

| | |
|---|---|
| ***9.3 UnLinkDevs – Remove link between two or more devices and update the XML database (Case 2: Delete a scene)*** | |
| *Prototype* | <Command Name=”UnLinkDevs” Scene=”scenename”/> |
| **Parameter** | **Description** |
| *Filename* | Scene: scene name which user wants to delete from the EZServe database |
| *Returns* | <Response Name=” UnLinkDevs” Status=”Success”/> |
| *Example* | <Command Name=”UnLinkDevs” Scene=”scenename”/><br>Deletes the scene from the EZServe database |
| *Notes* | Delete all associated links in the scene |

| | |
|---|---|
| ***9.4 UnLinkDevs – Remove link between two or more devices and update the XML database (Case 3: Partially delete a scene)*** | |
| *Prototype* | <Command Name=”UnLinkDevs” Scene=”scenename”><br><Cntrlrs><br><Cntrlr ID=”01.34.56” Grp=”1”/><br><Cntrlr ID=”89.37.D4” Grp=”3”/><br></Cntrlrs><br><Rspndrs><br><Rspndr ID=”01.54.D4”/><br><Rspndr ID=”89.D7.84”/><br><Rspndr ID=”4F.5C.32”/><br></Rspndrs><br></Command> |
| **Parameter** | **Description** |
| *Filename* | |
| *Returns* | <Response Name=” UnLinkDevs” Status=”Success”></Response> |
| *Example* | 1. Delete the controller and its associated links with all responders from this scene<br><Command Name=”UnLinkDevs” Scene=”scenename”><br><Cntrlrs><br><Cntrlr ID=”01.34.56” Grp=”1”/><br></Cntrlrs><br></Command><br><br>2. Delete the responder and its associated links with all controllers from this scene<br><Command Name=”UnLinkDevs” Scene=”scenename”><br><Rspndrs> |

| | |
|---|---|
| | <Rspndr ID=”4F.5C.32”/><br></Rspndrs><br></Command><br><br>3. Delete the controller and its associated links with all responders from this scene and also deletes the responder and its associated links with all controllers from this scene<br><Command Name=”UnLinkDevs” Scene=”scenename”><br><Cntrlrs><br><Cntrlr ID=”01.34.56” Grp=”1”/><br></Cntrlrs><br><Rspndrs><br><Rspndr ID=”4F.5C.32”/><br></Rspndrs><br></Command><br><br>4. Delete the link from both controller and responder devices. Both controller and responder information is mandatory here<br><Command Name=”UnLinkDevs><br><Cntrlrs><br><Cntrlr ID=”01.34.56” Grp=”1”/><br></Cntrlrs><br><Rspndrs><br><Rspndr ID=”4F.5C.32”/><br></Rspndrs><br></Command> |
| *Notes* | Unlink between provided controller and responder. Scene name is optional here<br>ID is case sensitive. |
| | |
| ***9.5 WritetoDev – Write links to a device to match the XML database*** | |
| *Prototype* | <Command Name=”WritetoDev” ID=”01.35.60”/> |
| **Parameter** | **Description** |
| *Filename* | ID: ID of the device |
| *Returns* | <Response Name=”WritetoDev” Status=”Success”/> |
| *Example* | |
| *Notes* | Develop return error codes |
| | |
| ReadfromDev – Read links from a device into the XML database | |
| *Prototype* | <Command Name=”ReadfromDev” ID=”01.35.60”/> |
| **Parameter** | **Description** |
| *Filename* | ID: ID of the device |
| *Returns* | <Response Name=”WritetoDev” Status=”Success”/> |
| *Example* | |
| *Notes* | |
| | |

## 10. Network Management

| | |
|---|---|
| ***10.1 WritetoNet - Synchronizes Insteon Network to existing XML*** | |
| *Prototype* | <Command Name=”WriteToNet”/> |
| **Parameter** | **Description** |
| *Filename* | |
| *Returns* | <Response Name=”WriteToNet” Status=”Success”/> |
| *Example* | |
| *Notes* | |
| | |
| ReadfromNet - Scans the existing Insteon network | |
| *Prototype* | <Command Name=”ReadFromNet”/> |
| **Parameter** | **Description** |
| *Filename* | |
| *Returns* | <Response Name=”ReadFromNet” Status=”Success”/> |
| *Example* | |
| *Notes* | Beginning with devices in the EZSrve database, scan each device database and write in XML file. |
| | |

| *10.2 ClusterResponse – Asynchronously reports a cluster when one or more of its attributes changes* | |
|---|---|
| *Prototype* | <Response Command="ClusterResponse" Device="Dimmer"><br><Cluster CID="0" Status="0xFF" /><br></Response> |
| **Parameter** | **Description** |
| *Filename* | |
| *Returns* | |
| *Example* | |
| *Notes* | This is the primary mechanism for devices to report their status to a client. By grouping various device attributes into clusters, it is possible to maintain a real time representation of device status independent of the protocol in use. |
| | |

## 11. Administrative Commands

| *11.1 Cancel – Aborts the running operation* | |
|---|---|
| *Prototype* | <Command Name=”Cancel” CmdName=”Write”/> |
| **Parameter** | **Description** |
| *Filename* | CmdName: Command name of the operation that is currently running |
| *Returns* | <Response Name="Write" File="Devices" Status="Process terminated by the user"/> |
| *Example* | |
| *Notes* | . |
| | |

| *11.2 PLMRaw – Enables XML messages for ALL messages sent and received to/from PLM* | |
|---|---|
| *Prototype* | To activate (enable) messages:<br><Command Name=”PLMRaw” Mode=”On”/><br><br>Or to deactivate (disable) messages:<br><Command Name=”PLMRaw” Mode=”Off”/> |
| **Parameter** | **Description** |
| *Filename* | |
| *Returns* | <Response Name=”PLMRaw” Status=”Success”/><br>Messages from PLM subsequently will take the form:<br><Response Name="PLMRaw"><br><Byte1="0x50"/><br><Byte2="0x05"/><br><Byte3="0x95"/><br><Byte4="0x9F"/><br><Byte5="0x05"/><br><Byte6="0xE9"/><br><Byte7="0x38"/><br><Byte8="0x2B"/><br><Byte9="0x44"/><br><Byte10="0xB5"/><br></Response> |
| *Example* | See prototypes |
| *Notes* | 1. Mode is case sensitive<br>2. Byte1 in the response always indicates command number |
| | |

| *11.3 SetPasswd - Sets a password and enables or disables security* | |
|---|---|
| *Prototype* | <Command Name="SetPasswd" Password="Simplehomenet"/> |
| **Parameter** | **Description** |
| *Filename* | Password: The desired password |
| Returns | |
| Notes | |
| | |

| *11.4 GetRevision – Get EZSrve firmware revision* | |
|---|---|
| *Prototype* | <Command Name="GetRevision"/> |
| **Parameter** | **Description** |
| *Filename* | |
| *Returns* | <Response Name="GetRevision" Rev=”2.00.48”><br></Response> |

| | |
|---|---|
| *Notes* | |
| | |

### *11.5 GetClock - Requests the time currently set in the EZSrve internal clock*

| | |
|---|---|
| *Prototype* | <Command Name="GetClock"/> |
| **Parameter** | **Description** |
| *Filename* | |
| *Returns* | <Response Name="GetClock" Ststus=”Success” Time="2/09/06/2009 11:52:47" NTP="1" DST="1"/><br>Time is formatted as A/DD/MM/YYYY where:<br>A = Day of Week (0-7), where 0 = Sunday<br>DD = Date (01-31)<br>MM = Month (01-12) where 01 is January<br>YYYY = Year<br><br>NTP = If “1” use NTP time, if “0” use manual time<br>DST = If “1” DST is in effect, if “0” DST is not in effect |
| *Example* | |
| *Notes* | |

### *11.6 SetClock - Sets a time value in the EZSrve internal clock*

| | |
|---|---|
| *Prototype* | <Command Name="SetClock" Time="2/10/22/2007 22:05:00" NTP=”1” DST=”0”/> |
| **Parameter** | **Description** |
| *Filename* | Time is formatted as A/DD/MM/YYYY where:<br>A = Day of Week (0-7), where 0 = Sunday<br>DD = Date (01-31)<br>MM = Month (01-12) where 01 is January<br>YYYY = Year<br><br>NTP = If “1” use NTP time, if “0” use manual time<br>DST = If “1” DST is in effect, if “0” DST is not in effect |
| *Returns* | |
| *Example* | |
| *Notes* | |

### *11.7 GetSunriseSunset -*

| | |
|---|---|
| *Prototype* | <Command Name="GetSunriseSunset"/> |
| **Parameter** | **Description** |
| *Filename* | |
| *Returns* | <Response Name=”GetSunriseSunset” Status=”Success” Sunrise=”05:45” Sunset=”19:48” ><br></Response> |
| *Example* | |
| *Notes* | |

### *11.8 ClearFlash – Clears all system data*

| | |
|---|---|
| *Prototype* | <Command Name="ClearFlash"/> |
| **Parameter** | **Description** |
| *Filename* | |
| *Returns* | <Response Name=" ClearFlash " Status="Success"> |
| *Example* | |
| *Notes* | |

### *11.9 Restart – Reboots the EZSrve*

| | |
|---|---|
| *Prototype* | <Command>Reset</Command> |
| **Parameter** | **Description** |
| *Filename* | |
| *Returns* | <Response Name="Restart" Status="Success"> |
| *Example* | |
| *Notes* | |

# 12. High-Level Functions

| *12.1 SendInsteon – Sends standard or extended Insteon message to a specific device* | |
|---|---|
| *Prototype* | <Command Name=”SendInsteon” ID=”05.E4.49”><br><CommandDetail Cmd1=”0x11” Cmd2=”0xFF”/><br></Command><br>Or to activate/deactivate a Group:<br><Command Name=”SendInsteon” ID=”9”><br><CommandDetail Cmd1=”0x11” Cmd2=”0x3”/><br></Command><br>Or to send an extended message:<br><Command Name=”SendInsteon” ID=”05.E4.49”><br><CommandDetail Cmd1=”0x11” Cmd2=”0xFF” Data=”0x01,0x02,0x03,0x04,0x05,0x06,0x07,0x08,0x4F”/><br></Command> |
| **Parameter** | **Description** |
| *Filename* | |
| *Returns* | |
| *Example* | See prototypes |
| *Notes* | |

| *12.2 SendX10 - Send a complete X10 Command* | |
|---|---|
| *Prototype* | <Command Name=”SendX10” House=”A” Unit=”2” Cmd=”On”/> |
| **Parameter** | **Description** |
| *Filename* | Uses the English representations for House codes (A-M), Units (1-16) and Commands (On, Off, Dim, etc.)<br>X10_Commands: "All-Lights-OFF", “Status=OFF","ON", "Pre-Set:Bright",<br>"All-Lights-ON","Hail-Acknowledge","Bright",<br>"Status=ON","Extended-Code","Status-Request","OFF",<br>"Pre-Set:Dim","All-Units-OFF","Hail-Request","Dim",<br>"Extended-Data(analog)"<br>House_Codes: "A","B","C","D","E","F","G","H","I","J","K","L","M","N","O","P"<br>Unit_Codes:<br>"1","2","3","4","5","6","7","8","9","10","11","12",13","14","15","16" |
| *Returns* | |
| *Example* | |
| *Notes* | |

| *12.3 Peek – Read a block of 1-8 locations from a device’s memory* | |
|---|---|
| *Prototype* | <Command Name=”Peek” ID=”03.45.78” Address=”0x03ff” Bytes=”8”/> |
| **Parameter** | **Description** |
| *Filename* | address: Memory location to start reading, in hex.<br>Bytes: number of bytes to read in decimal |
| *Returns* | <Response Name=”Peek” ID=”03.45.78” Data=”00,01,02,03,04,FF,3F,2C” Status=”Success”/> |
| *Example* | |
| *Notes* | Returns a string with each byte read as 2 digits in hex, first byte read on the left. |

| *12.4 Poke – Write to a block of 1-8 locations in a device’s memory* | |
|---|---|
| *Prototype* | <Command Name=”Poke” ID=”02.34.56” Address=”0x3ff” Data=”00,01,02,03,04,05,06,07” /> |
| **Parameter** | **Description** |
| *Filename* | |
| *Returns* | <Response Name=”Poke” ID=”02.34.56” Status=”Success”/> |
| *Example* | |
| *Notes* | |

## 13. Actions Reference Table

| Condition Options | | | | |
|---|---|---|---|---|
| **Selected Item** | **Selected Option** | **Mandatory Values** | **Ignored Values** | **Description** |
| Logic | AND | - | | AND: This condition is logically ANDed with next condition |
| Logic | OR | - | | OR: This condition is logically ORed with next condition |
| Device | A Device listed in the list of options | Attribute: Select in the list of options<br><br>Attribute Value: Select in the list of options | - | During condition evaluation, Attribute Value is matched with respective device CID attribute value |
| At Time | AnyTime | - | Start Time<br>Offset Time | Time is not considered during condition evaluation |
| | Absolute | Start Time: *hh:mm:ss* | Offset Time: | Start Time is considered during condition evaluation |
| | Interval | Start Time: *hh:mm:ss*<br><br>Offset Time: hh:mm:ss | | Start Time is considered during first condition true evaluation and after that (Start Time + Offset Time) is considered during preceding condition evaluations |
| | Window | Start Time: *hh:mm:ss*<br><br>Offset Time: hh:mm:ss | | Time between Start Time and (Start Time + Offset Time) is considered during condition evaluation |
| | Sunset | - | Start Time<br>Offset Time | Sunset Time is considered during condition evaluation |
| | Sunset + | Offset Time: *hh:mm:ss* | Start Time | (Sunset Time + Offset Time) is considered during condition evaluation |
| | Sunset - | Offset Time: *hh:mm:ss* | Start Time | (Sunset Time - Offset Time) is considered during condition evaluation |
| | Sunrise | | Start Time<br>Offset Time | Sunrise Time is considered during condition evaluation |
| | Sunrise + | Offset Time: *hh:mm:ss* | Start Time | (Sunrise Time + Offset Time) is considered during condition evaluation |
| | Sunrise - | Offset Time: *hh:mm:ss* | Start Time | (Sunrise Time - Offset Time) is considered during condition evaluation |
| | SunsetToSunrise | - | Start Time<br>Offset Time | Time between Sunset and Sunrise is considered during condition evaluation |
| | SunriseToSunset | - | Start Time<br>Offset Time | Time between Sunrise and Sunset is considered during condition evaluation |
| | Random Time | Start Time: *mm:ss* | Start Time: *hh*<br>Offset Time | Random time is calculated between present EZServe time and (EZServe Time + Start Time). This random time is considered during condition evaluation |
| Days | Sun \|\| Mon \|\| Tue \|\| Wed \|\| Thu \|\| Fri \|\| Sat \|\| all above | - | Hol, Even, Odd and Every | Selected days option are logically ORed during condition evaluation |
| Days | Hol \|\| Even \|\| Odd \|\| Every | - | Sun, Mon, Tue, Wed, Thu, Fri and Sat | Selected days option are logically ORed during condition evaluation |

**Condition Evaluation: (assuming Condition is Active)**
To evaluate a condition as TRUE, all the selected condition options are ANDed together
i.e. Device Attribute value(if selected) && At Time && Days

**Conditions Evaluation: (assuming all selected Conditions are Active)**

To execute Action, all the Active Condition results are logically ANDed or ORed together, depending on the respective Condition Logic selection

Ex: 1. Condition 1 (AND),
Condition 2 (AND),
Condition 3 (OR),
Condition 4 (OR),
Condition 5 (AND)

Result =
[ {(Condition 5 AND Condition 4) OR (Condition 3 OR Condition 2)} AND Condition 1 ]

Ex: 2. Condition 1 (AND),
Condition 2 (AND),
Condition 3 (OR),
Condition 4 (OR),

Result =
[(Condition 4 OR Condition 3) OR (Condition 2 AND Condition 1)]

Ex: 3. Condition 1 (AND),
Condition 2 (OR),
Condition 3 (AND),

Result = [(Condition 3 AND Condition 2) OR Condition 1]

Ex: 4. Condition 1 (OR),
Condition 2 (AND),

Result = [Condition 2 AND Condition 1]

Ex: 5. Condition 1 (AND),

Result = [Condition 1 **OR** 01]