

## EZSrv API 2.0 Examples

<b>On Initialization – Client gets all relevant information in xml files</b>	
Read “Devices.xml”	<Command Name=”Read” File=”Devices”/>
Read “Areas.xml”	<Command Name=”Read” File=”Areas”/> EZSrv responds with: <?xml version=”1.0”?> <Response Name=”Read” File=”Areas” Status=”Success”> <Areas> <Area Rec=”2” Name=”Bedroom”> <Devices/> </Area> <Area Rec=”1” Name=”Garage”> <Devices/> </Area> <Area Rec=”3” Name=”Master Bedroom” > <Devices/> </Area> <Area Rec=”4” Name=”Front Office” > <Device Name=”Dimmer1”/> <Device Name=”Default_01”/> </Area> </Areas> </Response>
Read “Holidays.xml”	<Command Name=”Read” File=”Holidays”/>
Read “User.xml”	<Command Name=”Read” File=”User”/> *1
Read “Actions.xml”	<Command Name=”Read” File=”Actions”/> *1
Read “User.xml”	<Command Name=”Read” File=”User”/>
Get clock/calendar	<Command Name=”GetClock”/> Response is: <?xml version=”1.0”?> <Response> GetClock <Parameter1>01/02/2007 17:48:28</Parameter1> </Response>
Get sunrise/sunset times	<Command Name=”GetSunriseSunset”/> Response is: <?xml version=”1.0”?> <Response>GetSunriseSunSet <sunrise>12:00</sunrise> <sunset>22:20</sunset> </Response>
Get additional hardware and network information	Items such as PLM ID, MAC, location, etc., are contained in “Devices.xml” under specific clusters of EZServe.
<b>Area Management (add, delete, modify area, or add/remove device to/from area)</b>	
Add an area	Modify local copy of “Areas.xml”, then write the entire file

## EZSrv API 2.0 Examples

	<p>back to the EZSrv with the command:</p> <pre>&lt;Command Name="Write" File="Areas"&gt; &lt;Areas file contents here&gt; &lt;/Command&gt;</pre> <p>Then, verify operation by reading the file back:</p> <pre>&lt;Command Name="Read" File="Areas"/&gt;</pre>
Delete an area	<p>Modify local copy of "Areas.xml", then write the entire file back to the EZSrv with the command:</p> <pre>&lt;Command Name="Write" File="Areas"&gt; &lt;Areas file contents here&gt; &lt;/Command&gt;</pre> <p>Then, verify operation by reading the file back:</p> <pre>&lt;Command Name="Read" File="Areas"/&gt;</pre>
Add a device to an area	<p>Modify local copy of "Areas.xml", then write the entire file back to the EZSrv with the command:</p> <pre>&lt;Command Name="Write" File="Areas"&gt; &lt;Areas file contents here&gt; &lt;/Command&gt;</pre> <p>Then, verify operation by reading the file back:</p> <pre>&lt;Command Name="Read" File="Areas"/&gt;</pre>
Remove a device from an area	<p>Modify local copy of "Areas.xml", then write the entire file back to the EZSrv with the command:</p> <pre>&lt;Command Name="Write" File="Areas"&gt; &lt;Areas file contents here&gt; &lt;/Command&gt;</pre> <p>Then, verify operation by reading the file back:</p> <pre>&lt;Command Name="Read" File="Areas"/&gt;</pre>
<b>Clock, Calendar and Holidays Management</b>	
Set the clock/calendar	<pre>&lt;Command Name="SetClock" Time="2/10/22/2009 22:05:00"/&gt;</pre> <p>Note that the first number is the day of the week. In this example, we are setting Tuesday, October 22 2009.</p>
Change the list of holidays	<p>Same as the examples above for Areas. First change the local copy of "Holidays.xml", then write to the EZSrv with:</p> <pre>&lt;Command Name="Write" File="Holidays"&gt; &lt;Holidays&gt; &lt;Holiday Rec"1" Date="01Jan09"/&gt; &lt;/Holidays&gt; &lt;/Command&gt;</pre> <p>Then re-read the file from the EZSrv with:</p> <pre>&lt;Command Name="Read" File="Holidays"/&gt;</pre>
<b>Device Management (Add, Delete, Modify a device)</b>	
Add a device	<p>The behavior of the device addition will vary, depending on how many of the possible 3 attributes are sent in the command.</p>

## EZSrve API 2.0 Examples

a) When passing all 3 attributes like thus:

```
<Command Name="Write" File="Devices">  
<Device Name="New Device" ID="06.00.71"  
  DevCat="0x400"/>  
</Command>
```

The device record will be added without concern whether the device is physically present or not. Any links in the device will need to be read later.

b) If passing only the name and the ID like thus:

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

EZsrve will first discover the device. If successful, it will then read the links from the device and create the record. The client gets messages as the addition progresses.

c) If passing only the name, like so:

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

EZSrve will put itself in linking mode and will wait for a device to join it. The client must instruct the user to put the device of interest in linking mode, and then follow the progress messages from EZSrve. If successful, EZSrve reads the device links and creates the device record. If the name passed already exists in the database, EZSrve responds with an error "AlreadyExists".

d) If passing the record number, the name and the ID where the device name is already in the database:

```
<Command Name="Write" File="Devices">  
<Device Rec="3" Name="Known Device"  
  ID="06.00.71"/>  
</Command>
```

EZsrve will change the device ID in the existing record but will not attempt to read the device links. This is useful for replacing devices.

e) If passing only the record number like thus:

```
<Command Name="Write" File="Devices">  
<Device Rec="10"/>  
</Command>
```

EZsrve will delete the device record.

**Please note that if a device record exists with a device of this name, the device will be deleted!!**

**Once the operation completes, it is best to re-read the entire "Devices.xml" file to synchronize the client**

## EZSrv API 2.0 Examples

	<b>application.</b>
Delete a device	See case e) above
Replace a Device	See case d) above
<p><b>Device Operations (turn on/off, dim/brighten, get status, etc.)</b>  Please refer to the specific device class file to understand each device object. All control/monitoring of devices is done by getting and setting device “attributes” that are contained in device “clusters”.</p>	
To set a device attribute	<p>a) The command below will turn on a dimmer fully on, since its “Status” attribute represents the “On” level:</p> <pre>&lt;Command Name="WriteCluster" Device="Dimmer 1"&gt;   &lt;Cluster CID="0" Status="0xff"/&gt; &lt;/Command&gt;</pre> <p>EZSrv will respond as follows:</p> <pre>&lt;?xml version="1.0"?&gt; &lt;Response Command="ClusterResponse"   Device="Dimmer1"/&gt; &lt;Cluster CID="0" Status="0xFF" /&gt; &lt;/Response&gt;</pre> <p>b) To turn on valve 1 in an EZFlora, send:</p> <pre>&lt;Command Name="WriteCluster" Device="EZFlora1"&gt;   &lt;Cluster CID="0" Status="0x80"/&gt; &lt;/Command&gt;</pre> <p>And the EZSrv will respond with:</p> <pre>&lt;?xml version="1.0"?&gt; &lt;Response Command="ClusterResponse"   Device="Dimmer1"/&gt; &lt;Cluster CID="0" Status="0xFF" /&gt; &lt;/Response&gt;</pre>
To get a device attribute	
<p><b>Managing Scenes</b></p>	
To get the scene information	<p>The scene information is found in Devices.xml and must be gathered from two places as follows:  Each link record (responder or controller), if part of a scene, has an attribute SCR="number", where “number” is the scene number that the EZSrv maintains. This number is an index into the Scenes names list. An example of link records for a device that is both a controller and a responder in scenes is shown below.</p> <pre>&lt;Links&gt;   &lt;Link Rec="1" ID="06.00.5D" Grp="01" Cntrl="1" LD="FF-FF-00" SCR="2"/&gt;   &lt;Link Rec="2" ID="04.30.2E" Grp="02" Cntrl="0" LD="FF-FF-00" SCR="5"/&gt; &lt;/Links&gt;</pre>

## EZSrv API 2.0 Examples

	<p>The names of the scenes are listed under the main node &lt;Scenes&gt; at the end of Devices.xml, and look like this:</p> <pre>&lt;Scenes&gt; &lt;Scene SCR="1" Name="Movie Time"/&gt; &lt;Scene SCR="2" Name="Party"/&gt; &lt;Scene SCR="3" Name="Quiet Dining"/&gt; &lt;/Scenes&gt;</pre> <p>The client parses Devices.xml to get a scene record that looks like the one below. This can be used to populate any menus or forms to view or edit scenes individually.</p> <pre>&lt;Scene SCR="1" Name="Movie Time"&gt; &lt;Cntlrs&gt; &lt;Cntlrl ID="01.34.56" Grp="1"/&gt; &lt;Cntlrl ID="89.37.D4" Grp="3"/&gt; &lt;/Cntlrs&gt; &lt;Rspndrs&gt; &lt;Rspndr ID="01.54.D4" LD="FF-1F-00"/&gt; &lt;Rspndr ID="89.D7.84" LD="FF-1F-00"/&gt; &lt;Rspndr ID="4F.5C.32" LD="FF-1F-00"/&gt; &lt;/Rspndrs&gt; &lt;/Scene&gt;</pre>
<p>To create a scene with any number of controllers and responders (scene members)</p>	<pre>&lt;Command Name="LinkDevs" Scene="scenename"&gt; &lt;Cntlrs&gt; &lt;Cntlrl ID="01.34.56" Grp="1"/&gt; &lt;/Cntlrs&gt; &lt;Rspndrs&gt; &lt;Rspndr ID="01.54.D4" LD="FF-1F-00"/&gt; &lt;Rspndr ID="4F.5C.32" LD="FF-1F-00"/&gt; &lt;/Rspndrs&gt; &lt;/Command&gt;</pre>
<p>To delete a scene permanently</p>	<p>The client issues the following command</p> <pre>&lt;Command Name="UnLinkDevs" Scene="scenename"/&gt;</pre> <p>EZSrv goes ahead and physically removes the links in the various devices. The client should re-read Devices.xml after the confirmation for the scene deletion arrives from the EZSrv.</p>
<p><b>Specifics of the INSTEON thermostat class</b></p>	
<p>To get the temperature</p>	<p>This requires a “dummy” write which has no effect on the thermostat, other than to return the temperature value:</p> <pre>&lt;Command Name="WriteCluster" Device="Upstairs Thermostat"&gt; &lt;Cluster CID="0" Temp="0"/&gt; &lt;/Command&gt;</pre> <p>EZSrv will respond with:</p> <pre>&lt;?xml version="1.0"?&gt;</pre>

## EZSrv API 2.0 Examples

	<pre>&lt;Response Command="ClusterResponse"   Device="Upstairs Thermostat"/&gt; &lt;Cluster CID="0" Temp="0x9C" CoolSetPoint="0x98"   HeatSetPoint="0x9F" Mode="0xA0"/&gt; &lt;/Response&gt;</pre> <p><b>Temperature values in hex are 2X the actual value.</b></p>																		
<p>To set the mode</p>	<p>Each mode corresponds to a number as follows:</p> <table border="1" data-bbox="571 451 1393 793"> <tr><td>0x04</td><td>Heat mode – heat setpoint displays</td></tr> <tr><td>0x05</td><td>Cool mode – cool setpoint displays</td></tr> <tr><td>0x06</td><td>Auto mode – both setpoints displayed</td></tr> <tr><td>0x07</td><td>Turn fan on – mode is not affected</td></tr> <tr><td>0x08</td><td>Turn fan to auto – mode is not affected</td></tr> <tr><td>0x09</td><td>Turn everything off</td></tr> <tr><td>0x0a</td><td>Program heat mode</td></tr> <tr><td>0x0b</td><td>Program Cool mode</td></tr> <tr><td>0x0c</td><td>Program Auto mode</td></tr> </table> <p>To set the mode to any of the above (e.g. auto mode), pass the desired value as follows:</p> <pre>&lt;Command Name="WriteCluster" Device="Upstairs   Thermostat"&gt; &lt;Cluster CID="0" Mode="0x6"/&gt; &lt;/Command&gt;</pre> <p>EZSrv will respond with:</p> <pre>&lt;?xml version="1.0"?&gt; &lt;Response Command="ClusterResponse"   Device="Upstairs Thermostat"/&gt; &lt;Cluster CID="0" Temp="0x9C" CoolSetPoint="0x98"   HeatSetPoint="0x9D" Mode="0x6"/&gt; &lt;/Response&gt;</pre> <p><b>Temperature values in hex are 2X the actual value.</b></p>	0x04	Heat mode – heat setpoint displays	0x05	Cool mode – cool setpoint displays	0x06	Auto mode – both setpoints displayed	0x07	Turn fan on – mode is not affected	0x08	Turn fan to auto – mode is not affected	0x09	Turn everything off	0x0a	Program heat mode	0x0b	Program Cool mode	0x0c	Program Auto mode
0x04	Heat mode – heat setpoint displays																		
0x05	Cool mode – cool setpoint displays																		
0x06	Auto mode – both setpoints displayed																		
0x07	Turn fan on – mode is not affected																		
0x08	Turn fan to auto – mode is not affected																		
0x09	Turn everything off																		
0x0a	Program heat mode																		
0x0b	Program Cool mode																		
0x0c	Program Auto mode																		
<p>To set the cooling setpoint</p>	<p>To set the cooling setpoint, pass a value as follows:</p> <pre>&lt;Command Name="WriteCluster" Device="Upstairs   Thermostat"&gt; &lt;Cluster CID="0" CoolSetPoint="0x98"/&gt; &lt;/Command&gt;</pre> <p>EZSrv will respond with:</p> <pre>&lt;?xml version="1.0"?&gt; &lt;Response Command="ClusterResponse"   Device="Upstairs Thermostat"/&gt; &lt;Cluster CID="0" Temp="0x9C" CoolSetPoint="0x98"   HeatSetPoint="0x9D" Mode="0x6"/&gt; &lt;/Response&gt;</pre> <p><b>Temperature values in hex are 2X the actual value.</b></p>																		
<p>To set the heating setpoint</p>	<p>To set the cooling setpoint, pass a value as follows:</p> <pre>&lt;Command Name="WriteCluster" Device="Upstairs   Thermostat"&gt;</pre>																		

## EZSrv API 2.0 Examples

	<pre>&lt;Cluster CID="0" HeatSetPoint="0x9F"/&gt; &lt;/Command&gt; EZSrv will respond with: &lt;?xml version="1.0"?&gt; &lt;Response Command="ClusterResponse"   Device="Upstairs Thermostat"/&gt; &lt;Cluster CID="0" Temp="0x9C" CoolSetPoint="0x98"   HeatSetPoint="0x9F" Mode="0x6"/&gt; &lt;/Response&gt;</pre> <p><b>Temperature values in hex are 2X the actual value.</b></p>	
<b>Specifics of the INSTEON Irrigation (EZFlora) Class</b>		
Status byte explained	<p>Bits in this byte are interpreted as follows:</p> <ul style="list-style-type: none"> <li>Bits 0:2 are the valve number -1</li> <li>Bits 3:4 are the program number -1</li> <li>Bit 5 is set to start a program</li> <li>Bit 6 is set to use valve 8 for auxiliary pump control</li> <li>Bit 7 is set to turn on a valve</li> </ul>	
To control a valve or program sequence	<p>To control a valve or program, write a value from the table below into Status as follows:</p> <pre>&lt;Command Name="WriteCluster" Device="EZFlora1"&gt; &lt;Cluster CID="0" HeatSetPoint="0x9F"/&gt; &lt;/Command&gt;</pre> <p>EZSrv will respond with:</p> <pre>&lt;?xml version="1.0"?&gt; &lt;Response Command="ClusterResponse"   Device="Upstairs Thermostat"/&gt; &lt;Cluster CID="0" Temp="0x9C" CoolSetPoint="0x98"   HeatSetPoint="0x9F" Mode="0x6"/&gt; &lt;/Response&gt;</pre> <p>To turn the valve and program off, simply write 0x00 into the status byte.</p>	
	0x80	Start valve 1
	0x81	Start valve 2
	0x82	Start valve 3
	0x83	Start valve 4
	0x84	Start valve 5
	0x85	Start valve 6
	0x86	Start valve 7
	0x87	Start valve 8
	0x20	Start program 1
	0x28	Start program 2
	0x30	Start program 3
	0x38	Start program 4
Timer banks explained	Timers for "manual" valve running are designated "DV1"	

## EZSrv API 2.0 Examples

	<p>through “DV8”.</p> <p>Timers for “program 1” valve running are designated “P1V1” through “P1V8”</p> <p>Timers for “program 2” valve running are designated “P2V1” through “P2V8”</p> <p>Timers for “program 3” valve running are designated “P3V1” through “P3V8”</p> <p>Timers for “program 4” valve running are designated “P4V1” through “P4V8”</p>
To read a timer bank	<p>Read a specific timer bank by referencing the cluster as follows:</p> <pre>&lt;Command Name="Read" Device="EZFlora1"&gt;   &lt;Cluster CID="255"/&gt; &lt;/Command&gt;</pre> <p>EZSrv will respond with:</p> <pre>&lt;?xml version="1.0"?&gt; &lt;Response Command="ClusterResponse"   Device="EZFlora1"/&gt; &lt;Cluster CID="254" DV1="0x10" DV2="0x0"   DV3="0x12" DV4="0x20" DV5="0x10" DV6="0x30"   DV7="0x10" DV8="0x10"/&gt; &lt;/Response&gt;</pre>
To set a timer bank	<p>To control a valve or program, write a value from the table below into Status as follows:</p> <pre>&lt;Command Name="WriteCluster" Device="EZFlora1"&gt;   &lt;Cluster CID="254" DV1="0x10" DV2="0x0"   DV3="0x12" DV4="0x20" DV5="0x10" DV6="0x30"   DV7="0x10" DV8="0x10"/&gt; &lt;/Command&gt;</pre> <p>EZSrv will respond with:</p> <pre>&lt;?xml version="1.0"?&gt; &lt;Response Command="WriteCluster" Status="Success"/&gt; &lt;/Response&gt;</pre>
<b>Specifics of the INSTEON EZIO Class</b>	
<b>Specifics of the INSTEON X10RF</b>	
<b>Specifics of the INSTEON EZUIRT</b>	
<b>Specifics of the INSTEON EZSnsRF</b>	

## EZSrve API 2.0 Examples
