EZSrve API 2.0 Examples

On Initialization – Client gets all relevant information in xml files	
Read “Devices.xml”	<Command Name=”Read” File=”Devices”/>
Read “Areas.xml”	<Command Name=”Read” File=”Areas”/> EZSrve 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.
Area Management (add, delete, modify area, or add/remove device to/from area)	
Add an area	Modify local copy of “Areas.xml”, then write the entire file

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

	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**

	application.
Delete a device	See case e) above
Replace a Device	See case d) above
Device Operations (turn on/off, dim/brighten, get status, etc.) 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".	
To set a device attribute	a) The command below will turn on a dimmer fully on, since its "Status" attribute represents the "On" level: <Command Name="WriteCluster" Device="Dimmer 1"> <Cluster CID="0" Status="0xff"/> </Command> EZSrve will respond as follows: <?xml version="1.0"?> <Response Command="ClusterResponse" Device="Dimmer1"/> <Cluster CID="0" Status="0xFF" /> </Response> b) To turn on valve 1 in an EZFlora, send: <Command Name="WriteCluster" Device="EZFlora1"> <Cluster CID="0" Status="0x80"/> </Command> And the EZSrve will respond with: <?xml version="1.0"?> <Response Command="ClusterResponse" Device="Dimmer1"/> <Cluster CID="0" Status="0xFF" /> </Response>
To get a device attribute	
Managing Scenes	
To get the scene information	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 EZSrve 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. <Links> <Link Rec="1" ID="06.00.5D" Grp="01" Cntrl="1" LD="FF-FF-00" SCR="2"/> <Link Rec="2" ID="04.30.2E" Grp="02" Cntrl="0" LD="FF-FF-00" SCR="5"/> </Links>

	The names of the scenes are listed under the main node <Scenes> at the end of Devices.xml, and look like this: <Scenes> <Scene SCR=”1” Name=”Movie Time"/> <Scene SCR=”2” Name=”Party"/> <Scene SCR=”3” Name=”Quiet Dining"/> </Scenes> 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. <Scene SCR=”1” Name=”Movie Time> <Cntrlrs> <Cntrlr ID=”01.34.56” Grp=”1”/> <Cntrlr ID=”89.37.D4” Grp=”3”/> </Cntrlrs> <Rspndrs> <Rspndr ID=”01.54.D4” LD=”FF-1F-00”/> <Rspndr ID=”89.D7.84” LD=”FF-1F-00”/> <Rspndr ID=”4F.5C.32” LD=”FF-1F-00”/> </Rspndrs> </Scene>
To create a scene with any number of controllers and responders (scene members)	<Command Name=”LinkDevs” Scene=”scenename”> <Cntrlrs> <Cntrlr ID=”01.34.56” Grp=”1”/> </Cntrlrs> <Rspndrs> <Rspndr ID=”01.54.D4” LD=”FF-1F-00”/> <Rspndr ID=”4F.5C.32” LD=”FF-1F-00”/> </Rspndrs> </Command>
To delete a scene permanently	The client issues the following command <Command Name=”UnLinkDevs” Scene=”scenename”/> EZServe 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 EZServe.
Specifics of the INSTEON thermostat class	
To get the temperature	This requires a “dummy” write which has no effect on the thermostat, other than to return the temperature value: <Command Name=”WriteCluster” Device=”Upstairs Thermostat”> <Cluster CID=”0” Temp=”0”/> </Command> EZSrve will respond with: <?xml version="1.0"?>

<table>
<tr><td></td><td>
<Response Command="ClusterResponse"
Device="Upstairs Thermostat"/>
<Cluster CID="0" Temp="0x9C" CoolSetPoint=”0x98”
HeatSetPoint=”0x9F” Mode=”0xA0”/>
</Response>
Temperature values in hex are 2X the actual value.
</td></tr>
<tr><td>To set the mode</td><td>
Each mode corresponds to a number as follows:
<table>
<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>
To set the mode to any of the above (e.g. auto mode), pass the desired value as follows:
<Command Name=”WriteCluster” Device=”Upstairs Thermostat”>
<Cluster CID=”0” Mode=”0x6”/>
</Command>
EZSrve will respond with:
<?xml version="1.0"?>
<Response Command="ClusterResponse"
Device="Upstairs Thermostat"/>
<Cluster CID="0" Temp="0x9C" CoolSetPoint=”0x98”
HeatSetPoint=”0x9D” Mode=”0x6”/>
</Response>
Temperature values in hex are 2X the actual value.
</td></tr>
<tr><td>To set the cooling setpoint</td><td>
To set the cooling setpoint, pass a value as follows:
<Command Name=”WriteCluster” Device=”Upstairs Thermostat”>
<Cluster CID=”0” CoolSetPoint=”0x98”/>
</Command>
EZSrve will respond with:
<?xml version="1.0"?>
<Response Command="ClusterResponse"
Device="Upstairs Thermostat"/>
<Cluster CID="0" Temp="0x9C" CoolSetPoint=”0x98”
HeatSetPoint=”0x9D” Mode=”0x6”/>
</Response>
Temperature values in hex are 2X the actual value.
</td></tr>
<tr><td>To set the heating setpoint</td><td>
To set the cooling setpoint, pass a value as follows:
<Command Name=”WriteCluster” Device=”Upstairs Thermostat”>
</td></tr>
</table>

<table>
<tr><td></td><td><Cluster CID=”0” HeatSetPoint=”0x9F”/>
</Command>
EZSrve will respond with:
<?xml version="1.0"?>
<Response Command="ClusterResponse"
Device="Upstairs Thermostat"/>
<Cluster CID="0" Temp="0x9C" CoolSetPoint=”0x98”
HeatSetPoint=”0x9F” Mode=”0x6”/>
</Response>
Temperature values in hex are 2X the actual value.</td></tr>
<tr><td></td><td></td></tr>
<tr><td></td><td></td></tr>
<tr><td colspan="2">Specifics of the INSTEON Irrigation (EZFlora) Class</td></tr>
<tr><td>Status byte explained</td><td>Bits in this byte are interpreted as follows:
Bits 0:2 are the valve number -1
Bits 3:4 are the program number -1
Bit 5 is set to start a program
Bit 6 is set to use valve 8 for auxiliary pump control
Bit 7 is set to turn on a valve</td></tr>
<tr><td>To control a valve or program sequence</td><td>To control a valve or program, write a value from the table below into Status as follows:
<Command Name=”WriteCluster” Device=”EZFlora1”>
<Cluster CID=”0” HeatSetPoint=”0x9F”/>
</Command>
EZSrve will respond with:
<?xml version="1.0"?>
<Response Command="ClusterResponse"
Device="Upstairs Thermostat"/>
<Cluster CID="0" Temp="0x9C" CoolSetPoint=”0x98”
HeatSetPoint=”0x9F” Mode=”0x6”/>
</Response>
To turn the valve and program off, simply write 0x00 into the status byte.
<table>
<tr><td>0x80</td><td>Start valve 1</td></tr>
<tr><td>0x81</td><td>Start valve 2</td></tr>
<tr><td>0x82</td><td>Start valve 3</td></tr>
<tr><td>0x83</td><td>Start valve 4</td></tr>
<tr><td>0x84</td><td>Start valve 5</td></tr>
<tr><td>0x85</td><td>Start valve 6</td></tr>
<tr><td>0x86</td><td>Start valve 7</td></tr>
<tr><td>0x87</td><td>Start valve 8</td></tr>
<tr><td>0x20</td><td>Start program 1</td></tr>
<tr><td>0x28</td><td>Start program 2</td></tr>
<tr><td>0x30</td><td>Start program 3</td></tr>
<tr><td>0x38</td><td>Start program 4</td></tr>
</table></td></tr>
<tr><td>Timer banks explained</td><td>Timers for “manual” valve running are designated “DV1”</td></tr>
</table>

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

EZSrve API 2.0 Examples
