Table of Contents

Controller Constants and Variables

Maximum Zones	100
Maximum Scenes	100
Maximum Zones in Scene	100
Maximum Name Length	20 characters, plus null char
Maximum Socket Connections	7
Power Level	1-100
Ramp Rate	1-100%

SetZoneProperties

Description:	• Sets the designated property(s) of the designated Top Dog device/zone(s). PowerLevel and Ramp Rate can be set for each dimmer and switches are on or off. This command can be used if the user wants to immediately turn on or off a Dimmer or Switch. • PowerLevel is level from 1= off to 100 = fully on. • Ramp Rate is a percent from 1 to 100%, giving the rate of power to be applied when turning on a light/switch zone. • Power is a Boolean controlling power on or off for either a device or switch, True:On, False:Off. • If AppContextId is sent in with the command, it will be sent back to the application. • NOTE: ReportZonePropertiesChanged broadcast will be sent from the controller on any change to any zone. Also, ReportZonePropertiesChanged will be sent as the power is changed on the top dog device.	
Parameters:	"ID"	json_integer unique identifier to refer to this packet
	"Service"	"SetZoneProperties"
	"ZID"	json_integer Which zone to act upon - json_integer: Zone ID (ZID) – (0-99)
	"PropertyList"	json_object Key / Value pairs for each property to be set Allowable Properties: "Name": json_string (20 chars max) "PowerLevel": json_integer (1-100) "RampRate": json_integer (1-100) "Power": json_boolean (true/false)
	"AppContextId"	Json_integer – sent in by app and echoed to the app, if received

Examples:
 	
Command:
	This example sets Zone 1 to power level 50 with a ramp rate of 75% and the name of the zone will now be "Kitchen". If zone was on, the power level of the top dog device will now be set to 50.
	{
	"ID": 12345,
	"Service": "SetZoneProperties",
	"ZID": 1,
	"PropertyList": {"Name": "Kitchen",
	"PowerLevel": 50,
	"RampRate":75
	}
	}	

Command:	This example sets Zones 1 to power level 75 and names all three of these zones to be "Upstairs Hallway".	

	Command: { “ID”: 12345, “Service”: “SetZoneProperties”, “ZID”: 1, “PropertyList”: {"Name": "Upstairs Hallway", “PowerLevel”: 75 } }
Command:	This example sets Zones 5 to “on”, the level will be currently stored power level and names the zone "Living Room Lamp". Command: { “ID”: 12345, “Service”: “SetZoneProperties”, “ZID”: 5, “PropertyList”: {"Name": " Living Room Lamp ", “Power”: true } }
Response:	{ “ID”: 12345, “Service”: “SetZoneProperties”, “Status”: “Success” }
Response:	{ “ID”: 12345, “Service”: “SetZoneProperties”, “Status”: “error – ‘Zone 3’ unavailable” }

ReportZoneProperties

Description:	• Query the properties of the Top Dog device in the specified Zone. • Only one Zone is allowed to be queried at a time	
Parameters:	“ID”	json_integer - unique identifier to refer to this packet
	“Service”	“ReportZoneProperty”
	“ZID”	json_integer - which zone we are requesting - Zone ID (ZID) – (0-99)
Response Parameters:		**See SetZoneProperties for property list**
Examples: **Command:**	This is asking for Zone 1 properties. { “ID”: 12345, “Service”: “ReportZoneProperties”, “ZID”: 1 }	
Response:	This is the response to query for Zone 1 properties. The property list for zone 1 shows that the zone has been named “Kitchen”, and is at full power and was ramped at 50%. { “ID”: 12345, “Service”: “ReportZoneProperties”, “ZID”: 1, “PropertyList”: {	

	"Name": "Kitchen", "DeviceType": "Dimmer" "PowerLevel": 100, "RampRate": 50, "Power": true }, "Status": "Success" }

Response:	{
	"ID": 12345,
	"Service": "ReportZoneProperties",
	"Status": "error – Zone 1 unavailable"
	}

SetSceneProperties

Description:	• This sets up a scene, which is a group of zones with the action associated with the device. • The scene can be controlled independently on or off. • This command can set a new scene or change an existing scene, depending on the scene ID sent in. If the scene ID exists in the controller, the properties will be changed to the parameters. • The App has to create a scene with the CreateScene command to get a new SID. • The positive response will be broadcast as a ScenePropertiesChanged when a new scene has been created at the controller. If another App user wants to know what the scene contains, the App should send a "ReportSceneProperties" with the scene id. • If no errors are found in the command, this will clear out the scene if it already exists. And fill it in with the data of this command.	
Parameters:	"ID"	json_integer - unique identifier to refer to this packet
	"Service"	"SetSceneProperties"
	"SID"	json_integer - Which scene to act upon :Scene ID (SID) – (0-99)
	"PropertyList"	json_object Key / Value pairs for each property to be set Allowable Properties
	"Name"	json_string - The name of this scene – 20 chars max – must have at least 1 char
	"ZoneList"	json_array Key / Value pairs for each property to be set No Duplicate Zone IDS are allowed Allowable Properties: [" ZID": json_integer (0-99) "Lvl": json_integer (1-100) "St": json_boolean "RR": json_integer (1-100)]
	"TriggerTime"	json_integer – time_t
	"Frequency"	json_integer (0-7) 0 – none 1 - once 2 - every Week
	"TriggerType"	json_integer (0-2) 0 – Regular time

		1 – Sunrise 2 - Sunset
	"DayBits"	json_integer (00-0x3f) Bit 0 → 1: Sunday, Bit 1 → 2: Monday Bit 2 → 4: Tuesday, Bit 3 → 8:Wednesday, Bit 4 → 16: Thursday Bit 5 → 32: Friday Bit 6 → 64: Saturday
	"Delta"	json_integer – time in minutes before or after trigger time – -120 -120 minutes
	"Skip"	Json_boolean – set to true to skip the next trigger
	"AppContextId"	Json_integer – optional App Id sent in and echoed back

Examples:
 	
Command:
	This example sets scene 1 to include a Zone List consisting of Zone 1 to power level 50 with a ramp rate:50 and Zone 45 to power level 85 with a ramp rate of 25 and zone 5 is a switch and it is turned on.
	{
	"ID": 12345,
	"Service": "SetSceneProperties",
	"SID": 1,
	
	"Property List":
	{
	"Name": "Dinner Party",
	"ZoneList":
	[
	{"ZID":1
	"Lvl": 50
	},
	{"ZID":45
	"Lvl": 85
	},
	{" ZID":5,
	"Power": True}
	],
	"TriggerTime": 1422526715,
	"Frequency": "Once",
	"TriggerType ": "Regular time",
	"Delta": 50,
	}
	}	

Response:	This response will be broadcast so all users of the controller can see that a scene has been created at the controller.
	{
	"ID": 12345,
	"Service": "SetSceneProperties",
	"SID": 1
	"Status": "Success",
	// property list of the command sent – see above
	}	

Response:	{ "ID": 12345,	

	“Service”: “SetSceneProperties”, “Status”: “error – ‘Zone 45’ does not exist” }

ReportSceneProperties

Description:	• This query command will return the scene property for one scene.	
Parameters:	“ID”	json_integer unique identifier to refer to this packet
	“Service”	“ReportSceneProperties”
	“SID”	json_integer - Which scene requested - Scene ID (SID) – (0-99)
Response Parameters:	“PropertyList”:	json_object Key / Value pairs for each property to be set Allowable Properties:
	“Running”	Json_boolean – indicates true if scene is executing
		SEE SetSetProperties for properties

Examples:
 	
Command:
	This example queries scene 1.
	{
	“ID”: 12345,
	“Service”: “ReportSceneProperties”,
	“SID”: 1
	}	

Response:	This response returns that scene 1 is named Dinner Party and has 2 zones with power levels and ramp rates.
	{
	“ID”: 12345,
	“Service”: “ReportSceneProperties”,
	“SID”: 1,
	“PropertyList”:
	{
	“Name”: “Dinner Party”,
	“ZoneList”:
	[
	{“ZID”:1165
	“Lvl”: 50
	}
	{“ZID”:45
	“Lvl”: 85
	}
	]
	“TriggerTime”: 8829292,
	“frequency”:0,
	“Trigger Type”:1,
	“DayBits”:1,
	“Delta”:50
	}
	“Status”: “Success”
	}	

Response:	{
	“ID”: 12345,
	“Service”: “ReportSceneProperties”,
	“Status”: “Scene 1 does not exist”
	}	

ListScenes

Description:	• This query command will send back the entire scene list in the controller. Use ReportSceneProperties to get each individual scene’s property.	
Parameters:	“ID”	json_integer - unique identifier to refer to this packet
	“Service”	“ListScenes”
Response Parameters:	“SceneList” – an array of scene ids	

Examples:
 	
Command:
	This example asks for a list of all the scenes stored in the controller.
	{
	“ID”: 12345,
	“Service”: “ListScenes”,
	}	

Response:	This is the response from the ListScenes query. It shows that the controller has 3 scenes stored, with Ids 2, 45 and 82.
	{
	“ID”: 12345,
	“Service”: “ListScenes”,
	“SceneList”:[“SID”:2, “SID”:45,”SID”:82]
	“Status”: “Success”
	}	

ListZones

Description:	• Query a list of all Zones known to the system	
Parameters:	“ID”	json_integer - unique identifier to refer to this packet
	“Service”	“ListZones”

Examples:
 	
Command:
	This is the command that is sent to the controller to query the zone known.
	{
	“ID”: 12345,
	“Service”: “ListZones”
	}	

Response:	The response will contain the following elements: “ZoneList”: a json_array that contains json_objects describing each of the zones. These json_objects contain the following elements: • “ZID”: a json_integer representing the Zone ID of the zone.	
Example:	This response shows that we have 3 zones known to the controller. It gives the zone ID. { “ID”: 12345, “Service”: “ListZones”, “ZoneList”: [{“ZID”:55 }, {“ZID”:7},{“ZID”:88}]	

	“Status”: “Success” }

ZonePropertiesChanged

Description:	• This broadcast response is used by the controller to report changes to any connected Top Dog Device. Note that the ID field of any asynchronous report will always be 0. Therefore, no packet sent TO the controller should use the ID: 0. • Reports any property changes to the property values for the specified Zone(s) of the top dog device. • The controller does not expect any response from the App for this response. • See “SetZoneProperties” command for a description of the parameters that can be sent.
Examples:	na

Response:	This broadcast response tells the APP that zone 4012 has gone to full power.
	{
	  “ID”: 0,
	  “Service”: “ZonePropertiesChanged”,
	  “ZID”: 4012,
	  “PropertyList”: { “PowerLevel”: 100},
	  “Status”: “Success”
	}

Response:	This broadcast response tells the APP that zone 55 has gone to power level of 35 with a ramp rate of 25.
	
	{
	  “ID”: 0,
	  “Service”: “ZonePropertiesChanged”,
	  “ZID”: 55,
	  “PropertyList”:
	   {
	     “PowerLevel”: 35,
	     “RampRate”: 25
	   },
	  “Status”: “Success”
	}

ScenePropertiesChanged

Description:	This command is a broadcast response from the controller. It informs the App that a scene has changed.	
Parameters	See SetSceneProperties for description of properties	
Examples:		

Broadcast Response:
Broadcast Response:
	{
	  “ID”: 0,
	  “Service”: “ScenePropertiesChanged”,
	  “SID”: 1,
	  “PropertyList”:{ “Running”:true}	

	}	

	{
	"ID": 0,
	"Service": "ScenePropertiesChanged",
	"SID": 1,
	"PropertyList":{ "Running":true}
	}	

ZoneAdded

Description:	• Broadcast command that tells the App that a new zone has been detected	
Parameters:	"ID"	json_integer - unique identifier to refer to this packet
	"Service"	"ZoneAdded"
Examples: **Command:**	na	
Response:	• "ZID": a json_integer representing the Zone ID of the zone. (0-99)	

Example:	{
	"ID": 0,
	"Service": "ZoneAdded",
	"ZID":85
	"Status": "Success"
	}	

DeleteZone

Description:	• a command to tell the LCM to delete a zone • The zone will be unusable after that	
Parameters:	"ID"	json_integer - unique identifier to refer to this packet
	"Service"	"DeleteZone"
Examples: **Command:**	• "ZID": a json_integer representing the Zone ID of the zone. (0-99) • "AppContextId": App Id sent and echoed back, optional parameter	

	{
	"ID": json_integer,
	"Service": "DeleteZone",
	"ZID": json_integer
	}	

Response:		

Example:	{
	"ID": 1,
	"Service": "DeleteZone",
	"ZID":3,
	"Status": "Success"
	}	

ZoneDeleted

Description:	• Broadcast command that tells the APP that a zone has been deleted	
Parameters:	"ID"	json_integer - unique identifier to refer to this packet
	"Service"	"ZoneDeleted"
Examples: **Command:**	na	
Response:	• "ZID": a json_integer representing the Zone ID of the zone. (0-99) • "AppContextId": App Id sent and echoed back, optional parameter	

Example:	{
	"ID": 0,
	"Service": "ZoneDeleted",
	"ZID":85
	"Status": "Success"
	}	

CreateScene

Description:	• This command will return the scene ID of the scene created, if there is room in the scene array • This will return an error if the scene can't be created (because the maximum number of scenes will be exceeded). • On success, this will also broadcast a SceneCreated with the Scene ID of the created scene.	
Parameters:	"ID"	json_integer unique identifier to refer to this packet
	"Service"	"CreateScene"
Command Parameters:	"PropertyList":	
	"Name"	json_string The name of this scene – 20 chars max
	"ZoneList"	json_array of Zone Properties (TBD) Key / Value pairs for each property to be set Allowable Properties: "ZID": json_integer – (0-99) "Lvl": json_integer – (1-100)
	"TriggerTime""	json_integer
	"Frequency"	json_integer
	"TriggerType"	json_integer
	"DayBits"	json_integer
	"Delta"	json_integer
	"AppContextId"	json_integer (not part of property list)

Examples:
 	
Command:
	This queries the controller for the next scene that is open.
	{
	"ID": 12345,
	"Service": "CreateScene",
	}	

Response:	{
	"ID": 12345,
	"Service": "CreateScene",
	"Status": "Can't Create Scene"
	}	

Response:	This is the response.
	{
	"ID": 12345,
	"Service": "SceneCreated",
	"SID":9,
	"Status": "Success"
	}	

RunScene

Description:	• This command will allow the user of the App to start the execution of a scene that has been set up previously. • When the App sends this command with a scene number, the controller will add the scene to the controllers RF queue and transmit the commands to the top dog devices as soon as possible: send the levels to the top dog switches with the zone in the scene using a Ramp command. • A Broadcast "ScenePropertiesChanged", with the SID and Property:"Running": true, will be transmitted as the scene executes. And a Broadcast "ScenePropertiesChanged", with the SID and Property:"Running": false, will be transmitted when the scene completes.	
Parameters:	"ID"	json_integer - unique identifier to refer to this packet
	"Service"	"RunScene"
	"SID"	json_integer – Which scene to act upon: Scene ID (SID)
	"AppContextId"	Json_integer – App Id sent in and echoed back

Examples:
 	
Command:
	This example sets Scene 1 to execute.
	{
	"ID": 12345,
	"Service": "RunScene",
	"SID": 1
	}	

Response:	{
	"ID": 12345,
	"Service": "RunScene",
	"Status": "Success"
	}	

Response:	{
	"ID": 12345,
	"Service": "RunScene",
	"Status": "Scene 5 does not exist"
	}	

DeleteScene

Description:	This command will remove the scene permanently from the controller memory. The response is broadcast so all App users are informed.	
Parameters:	"ID"	json_integer unique identifier to refer to this packet
	"Service"	"DeleteScene"
	"SID"	json_integer - Which scene to act upon - Scene ID (SID)
	"AppContextId"	Json_integer – App Id sent in and echoed back

Examples:
 	
Command:
	This example removes scene 1.
	{
	"ID": 12345,
	"Service": "DeleteScene",
	"SID": 1
	}	

Broadcast Response:	{
	"ID": 12345,
	"Service": "SceneDeleted",
	"SID": 1
	"Status": "Success"
	}	

SetSystemProperties

Description:	This command is used to set system properties.	
Parameters:	"ID":	json_integer - unique identifier to refer to this packet
	"Service":	"SetSystemProperties"
	"PropertyList":	
		"AddALight" – json_boolean - Set true to allow the LCM to recognize a new zone being added "TimeZone" – json_integer - An offset from GMT, represented with seconds "EffectiveTimeZone" – json_integer - An offset from GMT, represented with seconds that takes into account daylight saving time "DaylightSavingTime" – json_boolean - True if the area uses daylight saving time "LocationInfo" – json_string - String describing the value that was used to get the location. . "Location" – json_object - Latitude and Longitude degrees, minutes, and seconds for the LCM to use when querying for sunrise sunset information - Allowable Properties: - "Lat": json_object ○ "Deg": json_integer ○ "Min": json_integer

		o “Sec”: json_integer - “Long”: json_object o "Deg": json_integer o “Min”: json_integer o “Sec”: json_integer “Configured” – json_boolean - Set true to indicate that the LCM has been configured. Default value of false on power up

Examples:
 	
Command:
	This command will enable the add a light property
	{
	  “ID”: 12345,
	  “Service”: “SetSystemProperties”,
	  “PropertyList”:
	  {
	    “AddALight”: true,
	
	  }
	}	

Examples:
 	
Command:
	This command will disable the add a light property
	{
	  “ID”: 12345,
	  “Service”: “SetSystemProperties”,
	  “PropertyList”:
	  {
	    “AddALight”: false
	  }
	}	

Response:	{
	  “ID”: 12345,
	  “Service”: “SetSystemProperties”,
	  “Status”: “Success”
	}	

ReportSystemProperties

Description:	This command is used to get the system properties.	
Parameters:	“ID”	json_integer - unique identifier to refer to this packet
	“Service”	“ReportSystemProperties”
	“PropertyList”	See “SetSystemProperties” for parameter list

Examples:
 	
Command
	This command will get the system properties
	{
	  “ID”: 12345,
	  “Service”: “ReportSystemProperties”
	}	

Response	{
	  “ID”: 12345,
	  “Service”: “ReportSystemProperties”,
	  “PropertyList”:{
	    “AddALight”: true
	  }
	  “Status”: “Success”
	}	

SystemPropertiesChanged

Description:	This command is a broadcast response used to inform all connected apps that a system property has changed.
Parameters	See SetSystemProperties for description of properties

Broadcast Response:	{
	  "ID": 0,
	  "Service": "SystemPropertiesChanged",
	  "PropertyList":{
	    "AddALight":true
	  }
	  "Status": "Success"
	}

TriggerRampCommand

Description:	This command will simulate a switch change.	
Parameters:	"ID"	json_integer - unique identifier to refer to this packet
	"Service"	"TriggerRampCommand"
	"BuildingID"	json_integer – Building ID for the ramp command (0-255)
	"HouseID"	json_integer – House ID for the ramp command (0-255)
	"GroupID"	json_integer – Group ID for the ramp command (0-65535)
	"PowerLevel"	json_integer – Power level between (0-100)
	"DeviceType"	json_integer – Device type of the switch 65 = Dimmer 66 = Binary Switch 67 = Fan Controller

Examples: Command	{
	  "ID": 12345,
	  "Service": "TriggerRampCommand"
	  "BuildingID": 1
	  "HouseID": 2
	  "GroupID": 3
	  "PowerLevel": 100
	  "DeviceType": 65
	}	

Response:	{
	  "ID": 12345
	  "Service": "TriggerRampCommand"
	  "Status": "Success"
	}	

TriggerRampAllCommand

Description:	This command will turn on/off/set a level for all lights	
Parameters:	"ID"	json_integer - unique identifier to refer to this packet
	"Service"	"TriggerRampAllCommand"
	"BuildingID"	json_integer – Building ID for the ramp command (0-255)
	"PowerLevel"	json_integer – Power level between (0-100)

Examples: Command	{
	  "ID": 12345,
	  "Service": "TriggerRampAllCommand"
	  "BuildingID": 1
	  "PowerLevel": 100
	}	

Response:	{
	 “ID”: 12345
	 “Service”: “TriggerRampAllCommand”
	 “Status”: “Success”
	 }

Scenarios

Power Up

- On power up, the controller will read data structures from memory, restoring house ID, time zone, zip code, zones, and scenes.
- The controller will read time from the time server.
- The controller will read sunset and sunrise times from the weather server at `weather.vantagecontrols.com` and set a timer to read these every day, setting sunrise and sunset for today and tomorrow.
- The App should send a ListZones command
    - The App should then iterate through all of the zones, sending a ReportZoneProperties.
- The App should send a ListScenes command
    - The App should then iterate through each scene, sending a ReportSceneProperties command

System Set Up from ground zero

- The APP should send a “SetSystemProperties” with the property AddALight=true. The APP should then prompt the user to press any switch.
    - The LCM will then receive a ramp command from that top dog device and use that house ID to set the global house ID.
    - The LCM will send a “ZoneAdded” with the new Zone ID to the APP.
- The user can now identify all of the zones in the house
    - On user command, the APP will send a “SetSystemProperties” command with the property AddALight=true.
    - The APP will then prompt the user to press the next switch they want to identify.
    - When they press the switch, the LCM will receive a RAMP command and determine if the house ID matches the initial one identified. If so, a “ZoneAdded” with the next Zone ID slot is returned to the APP.
    - The APP can then send a “SetZoneProperties” with a unique name for the zone and power level, if wanted for that switch.
- Once all zones are set up, The App can send a List Scenes command
    - It will then iterate through each scene, setting up scenes by sending a ListSceneProperties command

Add a Light once system is running

- The App knows the current state of the LCM, i.e.: all lights are known
- The user physically adds a new light to their home. The lights must be bound to the house ID by the installer by following the directions provided by with the device.
- The user will need to add a light from the APP.
- When the LCM detects a ramp command from the device (the user physically presses the light switch), a ZoneAdded response will be broadcast, with the new zone ID as a parameter, as long as there is space in the zone array.

- The app should send a ReportZoneProperties to that zone, for default information. Default values are as follows: Name: zone XX (where XX is zone ID), Power Level: value set based on command from the switch, State: value set based on command from the switch, Ramp Rate:50

Light Value Changed

- This scenario will occur when the user physically changes the value of the light by hitting the switch.
- The LCM will receive a ‘Ramp’ command from the light with the new values physically set by the user.
- The LCM will broadcast a ReportZonePropertiesChanged, with parameters set to the values received by the device, usually Power Level (1-100) and state (true/false).

Create a scene:

- API will send “CreateScene” with all scene properties filled in, but no Scene ID (SID)
    - LCM will verify have room for another scene, send error if not
    - LCM will verify that all of the parameters are OK, send error if not
    - LCM will generate a scene with new Scene ID
        - Send back “CreateScene”, Status:Success
        - Broadcast “SceneCreated”, “SID”:x (where x is an integer scene id)

Delete a scene:

- API will send “DeleteScene” with SID
    - LCM will verify scene id, send error if SID doesn’t exist
    - LCM will delete the scene from the Scene array
    - Send back “DeleteScene”, Status:Success
    - Broadcast “SceneDeleted”, “SID”:x (where x is an integer scene id)

Change a scene property:

- API will send “SetSceneProperties” with SID and changed property
    - LCM will verify scene id, send error if SID doesn’t exist
    - LCM will verify property, send error if property invalid
    - LCM will change the scene property in the Scene array
    - Send back “SetSceneProperties”, Status:Success
    - Broadcast “ScenePropertiesChanged”, “SID”:x , property changed

Scene is executing:

- When a scene starts executing by the LCM, the LCM will broadcast “ScenePropertiesChanged” with SID and “Runnning”:True
- When the scene is complete, the LCM will broadcast “ScenePropertiesChanged with SID and “Runnning”:False.

Power Up / Down Rules

- Receive async rf ramp command from top dog device, the LCM performs the following:

```
▪   if addALightMode
     {
       if we have no zones in our zoneArray
       {
          set our houseID
       }
       if the rf packet groupID doesn't match any group id in our zoneArray
       {
          find the first available slot in our zone array for this new zone
          fill in zoneArray slot with defaults
          fill in zoneArray slot with rf packet info
          json broadcast ZoneAdded
          set addALightMode false
          json broadcast SystemPropertiesChanged for addALightMode property
      }
      else
      {
          this is a ramp command for someone in our list, call:
          HandleTargetValue(groupID, targetValue)
      }
     }
     else
     {    // addALightMode == false
      if packet house id matches our house
      {
        if the group id is in our zoneArray
        {
            HandleTargetValue(groupID, targetValue)
        }
      }
    }
```

```
//------------------------------------------------------------------------------------

  HandleTargetValue(groupID, targetValue)
  {
     find zoneArray slot matching groupID
      if targetValue not zero
     {
       if zone state property was false
       {
          set zone state property to true
          tag announcePropertyBitmask for state property
       }
       if zone level property doesn't match targetValue
       {
          set zone level property
         tag announcePropertyBitmask for level property
       }
     }
     else
     {  // targetLevel == 0
       if zone state was true
      {
         set zone state property to false
          tag announcePropertyBitmask for state property
       }
     }
    if announcePropertyBitmask
    {
       build PropertyList based off of announcePropertyBitmask
       broadcast zonePropertiesChanged with created PropertyList
```

```
                    }
                  }
```

- Receive SetZoneProperties from App
    - If sent Power state:true
        - Set power state in LCM to true
        - Send ramp command to top dog device with power level setting stored in LCM
        - send ZoneChanged to App
    - If sent Power state:false
        - Set power state in LCM to false
        - Send ramp command to top dog device with power level = 0
        - Power Level setting in the LCM does not change
        - If power was true, then send ZoneChanged to App
    - If Power Level > 0
        - Local LCM power setting is set to level sent in by App
        - If device is currently On, (power state = true)
            - the ramp command with new power level is sent to the top dog device
            - send ZoneChanged to App
        - If device is currently Off, (power state = false)
            - Nothing is sent to the top dog device
    - If Power level == 0
        - error