EventWatcher

@akbooer, December 2014

Overview:

The EventWatcher plugin does these things:

- watches selected device categories and logs key variable changes
- functions as a local 'Alternate Event Server' for Vera notifications
- generates reports and plots of current device variable configurations and values
- lists scene configurations including schedules, triggers, actioned devices, and Lua code
- graphic plots of CPU usage and memory resources

The whole point is to address the needs of those who want:

- access to events locally and independently of Mios.com
- much simpler and less cluttered logging
- simple reporting of Vera configuration and device status

EventWatcher is NOT a tool for detailed analysis of the Vera log, but contains higher level, and briefer, information which may be useful for analysing the operation of Vera scenes and devices. It's also NOT a replacement for dataMine, since it doesn't provide any analysis tools for long-term archived data.

EventWatcher responds to a variety of HTTP requests returning dynamic server pages for different report formats. All WATCH-ed variables and Vera notification EVENTS are stored in a memory buffer (holding, by default, the last 1000 events) available for analysis, and also, optionally, written out to a weekly file (ideally stored on an external storage device like a USB or NAS.)

Manual Installation (ignore if downloading from Mios App Store):

There are three groups of files to be installed in different places:

- **XML and Lua files** - unzip and load into Vera using the APPS > Develop Apps > Luup files > Browse > Upload menu
- **Key and Certificate files** (required for the Alternate Event Server functionality) should go into a top-level Vera directory (you will have to create) - the default location is
`/eventWatcher/`.
(Use ssh or some such utility to log directly into Vera to do this)
- **Icon file** - goes to `/www/cmh/skins/default/icons/`.

Once these files are loaded, use the APPS > Develop Apps > Create device menu
to set the following parameters:

- **Device type**: urn:akbooer-com:device:EventWatcher:1
- **Upnp Device Filename**: D_EventWatcher.xml
- optionally, at this time, set the Description to your required name (eg. Event-Watcher)

No other parameters need to be set.
Press the 'Create device' button and wait for Vera to restart.
Once it has, do another restart.
After this, you should see an EventWatcher device.

Further configuration may be done using the Advanced tab on the device UI. Device variables there include:

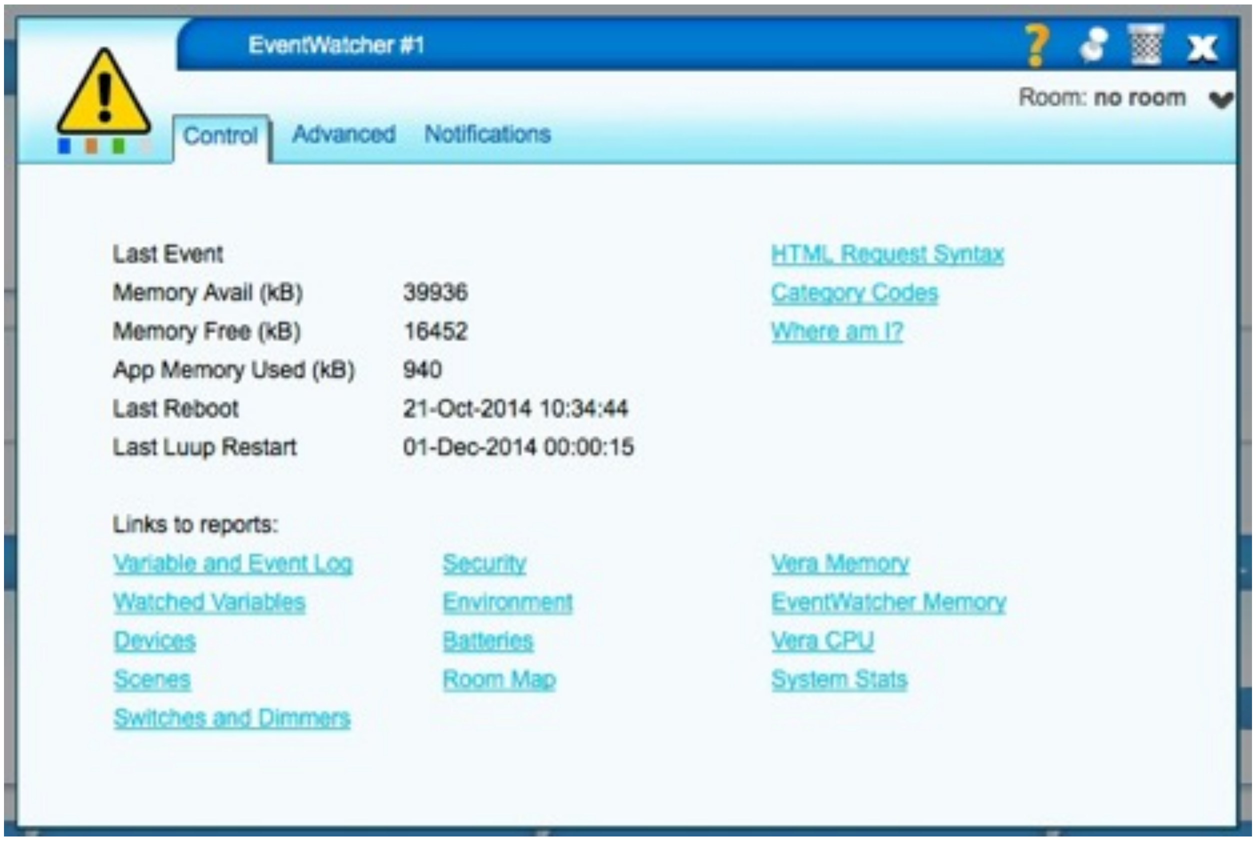
- **Version** - a read-only parameter showing the current EventWatcher code version
- **ServerKeyFile** - defaults to `/eventWatcher/EventWatcher.key`
- **ServerCertificateFile** - defaults to `/eventWatcher/EventWatcher.crt`
- **Syslog** - set to IP address and Port number of syslog server, if required. Syntax of the form `172.16.33.123:514` where 514 is the port number.
- **LogDirectory** - location for weekly log file output, initially blank,
see 'Logging to a file on USB or NAS' (below) for details on this
- **WatchCategories** - a list of the device category types to watch,
defaults to `XYSM`
- **CacheSize** - number of in-memory events to store,
defaults to 1000 (min 200, max 2000)
- **Debug** - turns on debug logging if set to anything other than "0"
- **ExtraVariablesFile** - a list of additional (perhaps non-categorised) device variables to watch (see 'Watching Other Variables' below)

Changes to these variables will require a SAVE, which restarts Luup.

Basic use:

Out-of-the-box, EventWatcher watches device categories for lights and switches, scene controllers, security sensors and power meters (whole house energy). This can be extended to temperature, humidity, light levels, and other generic sensors. You can view the 'log' of this 'dynamic' data (essentially any change of watched device status) through any web browser.

The easiest way to access EventWatcher reports and graphs is through the links provided on the Control tab of the Device panel in Vera's UI5 interface. (This probably doesn't format correctly in UI7 yet, sorry.)



Last Event	
Memory Avail (kB)	39936
Memory Free (kB)	16452
App Memory Used (kB)	940
Last Reboot	21-Oct-2014 10:34:44
Last Luup Restart	01-Dec-2014 00:00:15

HTML Request Syntax
Category Codes
Where am I?

Links to reports:

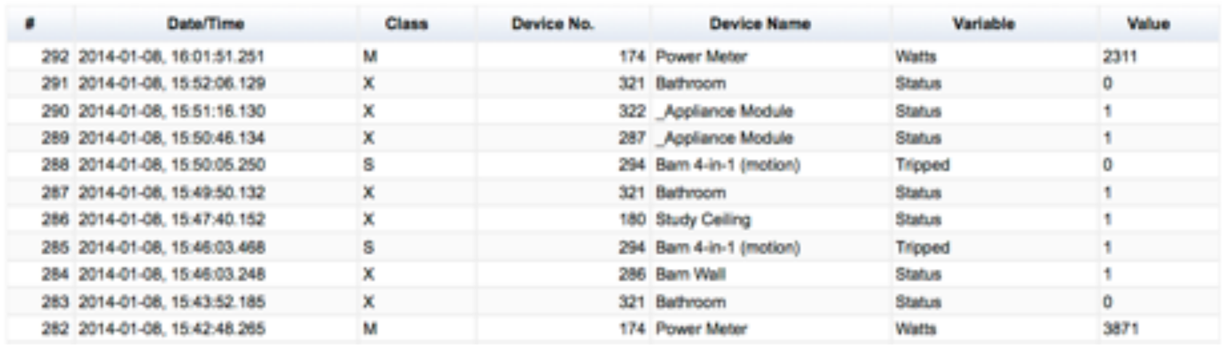
Variable and Event Log	Security	Vera Memory
Watched Variables	Environment	EventWatcher Memory
Devices	Batteries	Vera CPU
Scenes	Room Map	System Stats
Switches and Dimmers		

The 'Watched Variables' report itself contains hyperlinks to graphs of each variable, showing the data recorded in EventWatcher's in-memory event buffer.

HTTP Requests:

You can view and of the EventWatcher reports or graphs through a web browser by a URL request of the form:

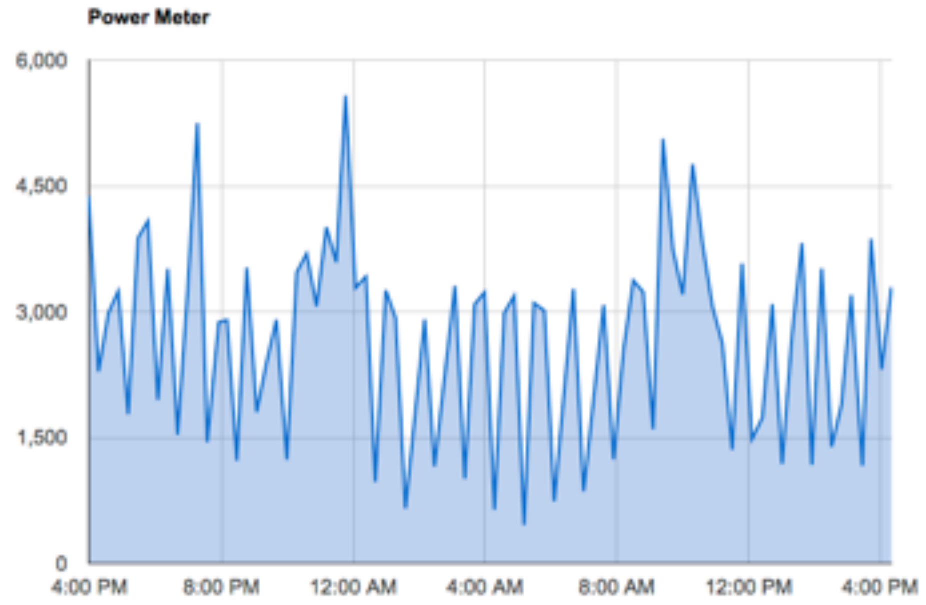
```
http://[your Vera IP]:3480/data_request?id=lr_EventWatcher&report=log
```



#	Date/Time	Class	Device No.	Device Name	Variable	Value
292	2014-01-08, 16:01:51.251	M	174	Power Meter	Watts	2311
291	2014-01-08, 15:52:06.129	X	321	Bathroom	Status	0
290	2014-01-08, 15:51:16.130	X	322	_Appliance Module	Status	1
289	2014-01-08, 15:50:46.134	X	287	_Appliance Module	Status	1
288	2014-01-08, 15:50:05.250	S	294	Barn 4-in-1 (motion)	Tripped	0
287	2014-01-08, 15:49:50.132	X	321	Bathroom	Status	1
286	2014-01-08, 15:47:40.152	X	180	Study Ceiling	Status	1
285	2014-01-08, 15:46:03.468	S	294	Barn 4-in-1 (motion)	Tripped	1
284	2014-01-08, 15:46:03.248	X	286	Barn Wall	Status	1
283	2014-01-08, 15:43:52.185	X	321	Bathroom	Status	0
282	2014-01-08, 15:42:48.265	M	174	Power Meter	Watts	3871

The table can be scrolled and rows can be sorted by different column categories simply by clicking on the column headings. Individual device variables can be plotted over time:

```
http://[your-Vera-IP]:3480/data_request?id=lr_EventWatcher&plot=174
```

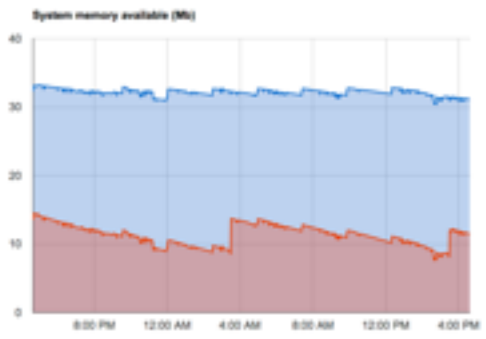


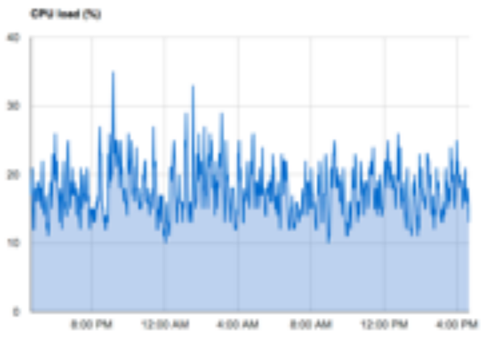
Recall that the internal buffer length is, by default, 1000, so the duration of the plotted variable depends on what other events are also being recorded.

EventWatcher can also report on memory and cpu statistics:

```
http://[your-Vera-IP]:3480/data_request?id=lr_EventWatcher&rep=memory
```

```
http://[your-Vera-IP]:3480/data_request?id=lr_EventWatcher&rep=cpu
```

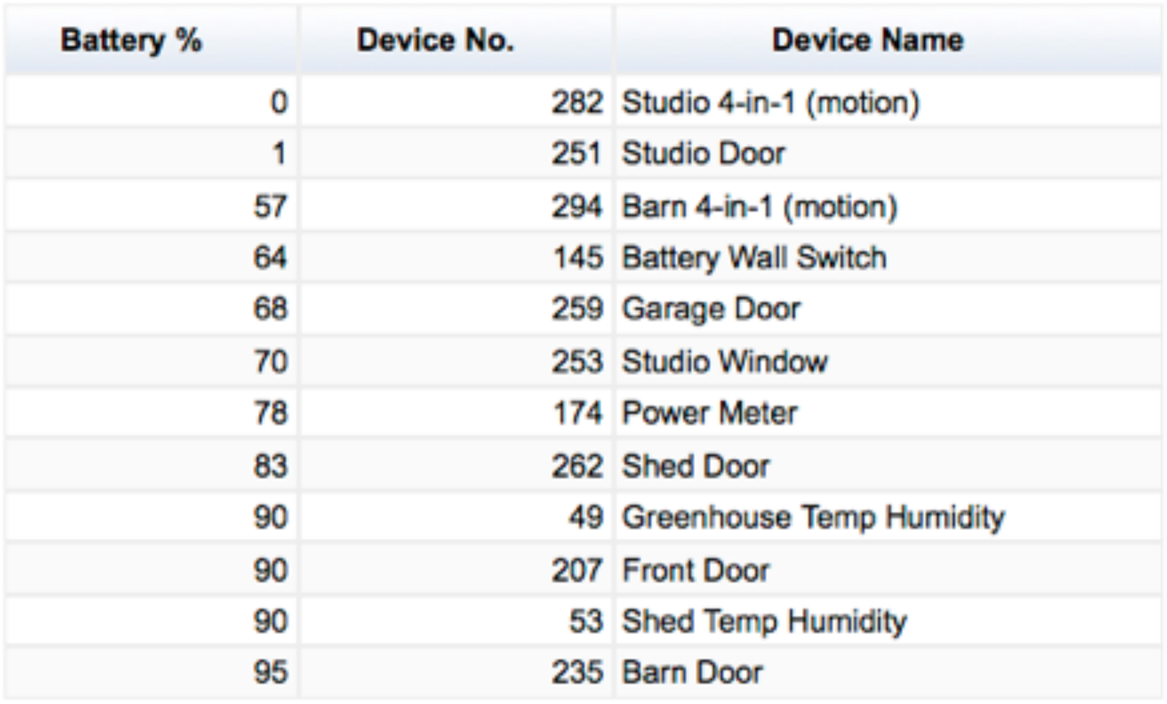




The blue area in the memory plot shows ‘cached’ memory, the red is ‘free’, the total being the ‘available’ memory. If the free memory falls towards zero then the Luup process is liable to restart itself.

A report for battery-powered devices is available:

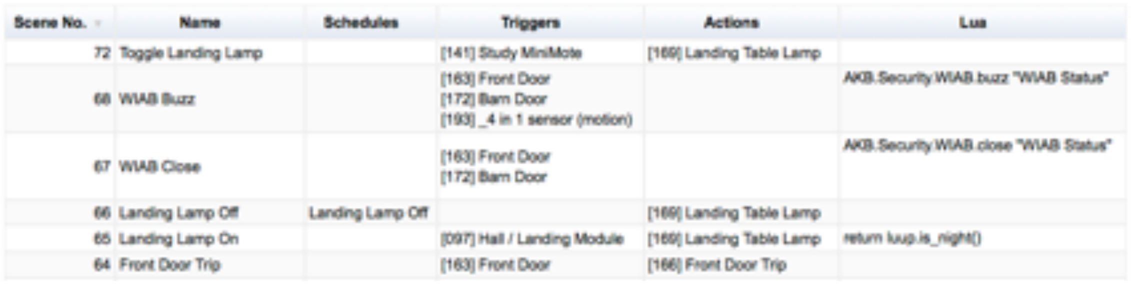
```
http://[your-Vera-IP]:3480/data_request?id=lr_EventWatcher&rep=battery
```



Battery %	Device No.	Device Name
0	282	Studio 4-in-1 (motion)
1	251	Studio Door
57	294	Barn 4-in-1 (motion)
64	145	Battery Wall Switch
68	259	Garage Door
70	253	Studio Window
78	174	Power Meter
83	262	Shed Door
90	49	Greenhouse Temp Humidity
90	207	Front Door
90	53	Shed Temp Humidity
95	235	Barn Door

An analysis of all scenes:

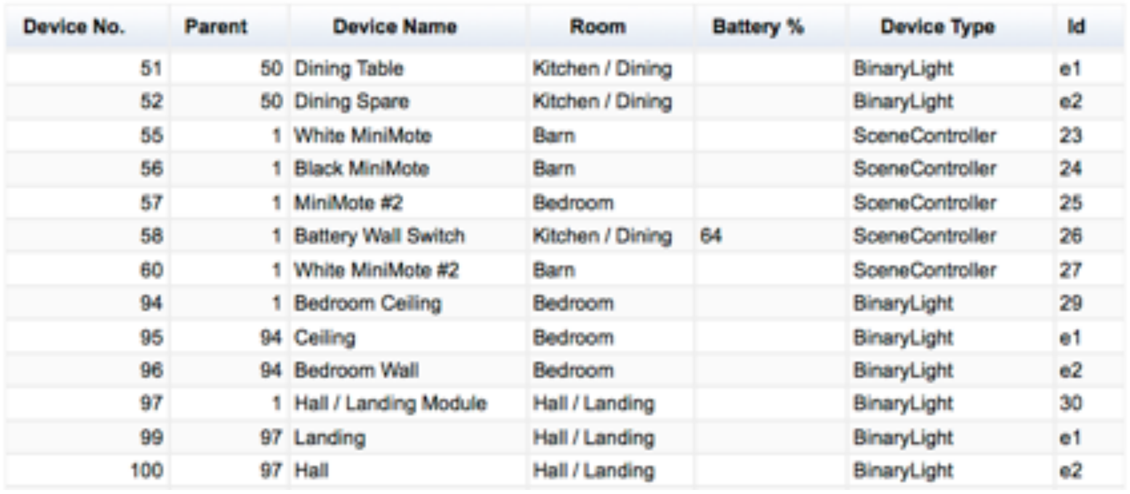
```
http://[your-Vera-IP]:3480/data_request?id=lr_EventWatcher&rep=scenes
```



Scene No.	Name	Schedules	Triggers	Actions	Lua
72	Toggle Landing Lamp		[141] Study MiniMote	[169] Landing Table Lamp	
68	WIAB Buzz		[163] Front Door [172] Barn Door [193] _4 in 1 sensor (motion)		AKB.Security.WIAB.buzz "WIAB Status"
67	WIAB Close		[163] Front Door [172] Barn Door		AKB.Security.WIAB.close "WIAB Status"
66	Landing Lamp Off	Landing Lamp Off		[169] Landing Table Lamp	
65	Landing Lamp On		[097] Hall / Landing Module	[169] Landing Table Lamp	return luup.is_night()
64	Front Door Trip		[163] Front Door	[166] Front Door Trip	

and devices:

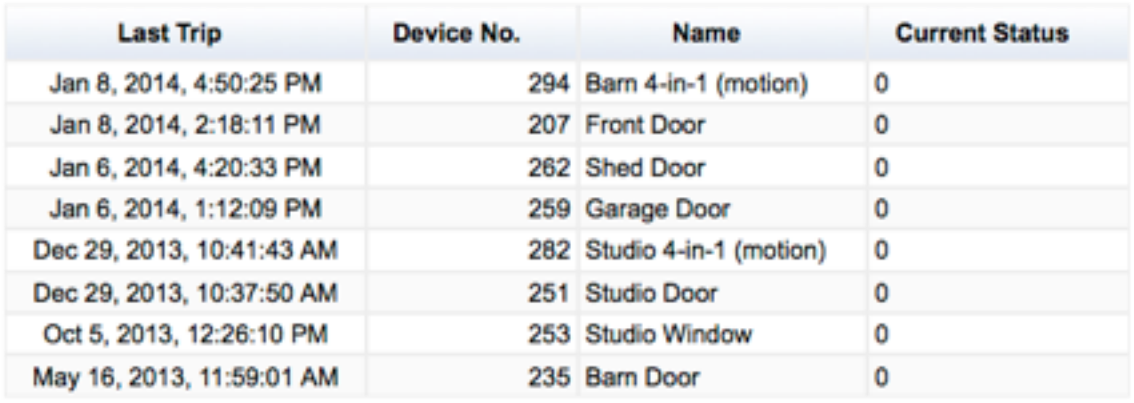
```
http://[your Vera IP]:3480/data_request?id=lr_EventWatcher&rep=devices
```



Device No.	Parent	Device Name	Room	Battery %	Device Type	Id
51	50	Dining Table	Kitchen / Dining		BinaryLight	e1
52	50	Dining Spare	Kitchen / Dining		BinaryLight	e2
55	1	White MiniMote	Barn		SceneController	23
56	1	Black MiniMote	Barn		SceneController	24
57	1	MiniMote #2	Bedroom		SceneController	25
58	1	Battery Wall Switch	Kitchen / Dining	64	SceneController	26
60	1	White MiniMote #2	Barn		SceneController	27
94	1	Bedroom Ceiling	Bedroom		BinaryLight	29
95	94	Ceiling	Bedroom		BinaryLight	e1
96	94	Bedroom Wall	Bedroom		BinaryLight	e2
97	1	Hall / Landing Module	Hall / Landing		BinaryLight	30
99	97	Landing	Hall / Landing		BinaryLight	e1
100	97	Hall	Hall / Landing		BinaryLight	e2

or subsets of devices, like security sensors:

```
http://[your-Vera-IP]:3480/data_request?id=lr_EventWatcher&rep=security
```



Last Trip	Device No.	Name	Current Status
Jan 8, 2014, 4:50:25 PM	294	Barn 4-in-1 (motion)	0
Jan 8, 2014, 2:18:11 PM	207	Front Door	0
Jan 6, 2014, 4:20:33 PM	262	Shed Door	0
Jan 6, 2014, 1:12:09 PM	259	Garage Door	0
Dec 29, 2013, 10:41:43 AM	282	Studio 4-in-1 (motion)	0
Dec 29, 2013, 10:37:50 AM	251	Studio Door	0
Oct 5, 2013, 12:26:10 PM	253	Studio Window	0
May 16, 2013, 11:59:01 AM	235	Barn Door	0

<u>Events:</u>

The functionality of an Alternate Event Server is described here:

http://wiki.micasaverde.com/index.php/AlternateEventServer

and elsewhere, but all that's need (just once) is to browse the URL

```
http://[your-Vera-IP]:3480/data_request?id=variableset&
Variable=AltEventServer&Value=127.0.0.1
```

and you should receive the status OK.

EventWatcher then records each event along with the other watched variables.

Watched Variables:

To make it simple to specify which variables to watch, EventWatcher uses the Luup device categorisation listed here:

`http://wiki.micasaverde.com/index.php/Luup_UPNP_Files#Device_Categories`

The numeric values have been mapped to a single character symbol, and a variable `WatchCategories` containing a string of these letters specifies which categories to watch. The category number, symbol, service and variable names are defined in the table below:

```
 1. "F",                                                                        --  DEVICE_CATEGORY_INTERFACE
 2. "X", "urn:upnp-org:serviceId:SwitchPower1",              "Status"            --  DEVICE_CATEGORY_DIMMABLE_LIGHT
 3. "X", "urn:upnp-org:serviceId:SwitchPower1",              "Status"            --  DEVICE_CATEGORY_SWITCH
 4. "S", "urn:micasaverde-com:serviceId:SecuritySensor1",    "Tripped"           --  DEVICE_CATEGORY_SECURITY_SENSOR
 5. "K",                                                                        --  DEVICE_CATEGORY_HVAC
 6. "C",                                                                        --  DEVICE_CATEGORY_CAMERA
 7. "D",                                                                        --  DEVICE_CATEGORY_DOOR_LOCK
 8. "W",                                                                        --  DEVICE_CATEGORY_WINDOW_COV
 9. "R",                                                                        --  DEVICE_CATEGORY_REMOTE_CONTROL
10. "I",                                                                        --  DEVICE_CATEGORY_IR_TX
11. "O",                                                                        --  DEVICE_CATEGORY_GENERIC_IO
12. "G", "urn:micasaverde-com:serviceId:GenericSensor1",     "CurrentLevel"      --  DEVICE_CATEGORY_GENERIC_SENSOR
13. "B",                                                                        --  DEVICE_CATEGORY_SERIAL_PORT
14. "Y", "urn:micasaverde-com:serviceId:SceneController1",   "sl_SceneActivated" --  DEVICE_CATEGORY_SCENE_CONTROLLER
15. "V",                                                                        --  DEVICE_CATEGORY_AV
16. "H", "urn:micasaverde-com:serviceId:HumiditySensor1",    "CurrentLevel"      --  DEVICE_CATEGORY_HUMIDITY
17. "T", "urn:upnp-org:serviceId:TemperatureSensor1",        "CurrentTemperature" -- DEVICE_CATEGORY_TEMPERATURE
18. "L", "urn:micasaverde-com:serviceId:LightSensor1",       "CurrentLevel"      --  DEVICE_CATEGORY_LIGHT_SENSOR
19. "Z",                                                                        --  DEVICE_CATEGORY_ZWAVE_INT
20. "J",                                                                        --  DEVICE_CATEGORY_INSTEON_INT
21. "M", "urn:micasaverde-com:serviceId:EnergyMetering1",    "Watts"             --  DEVICE_CATEGORY_POWER_METER
22. "A",                                                                        --  DEVICE_CATEGORY_ALARM_PANEL
23. "P",                                                                        --  DEVICE_CATEGORY_ALARM_PARTITION
```

You'll see that some services/variables are missing - I'm open to suggestions as to how to populate these items.

To keep the log entries down, PING sensors are excluded from the watched security devices. The default `WatchCategories` string is `XYSM`, which watches switches and dimmers, scene controllers, security devices, and energy meters.

<u>HTTP Requests (in a bit more detail):</u>

HTTP requests are of the general form:

```
http://[your-Vera-IP]:3480/data_request?id=lr_EventWatcher&report=XXX
```

An ‘empty’ HTTP request of the form:

```
http://[your-Vera-IP]:3480/data_request?id=lr_EventWatcher
```

will generate a ‘help’ text describing all the parameters available:

```
PARAMETERS: names and values can be abbreviated

  [actions]
                plot = plot specific device [number]
               event = create a user entry in the event log [string]
               scene = show specific scene [number]
              report = report types [log/devices/events/scenes/
security/environment/battery/batteries/system/cpu/memory/watch]

  [options]
              height = HTML output height [number]
               width = HTML output width [number]

EXAMPLE: &report=devices&width=1000
```

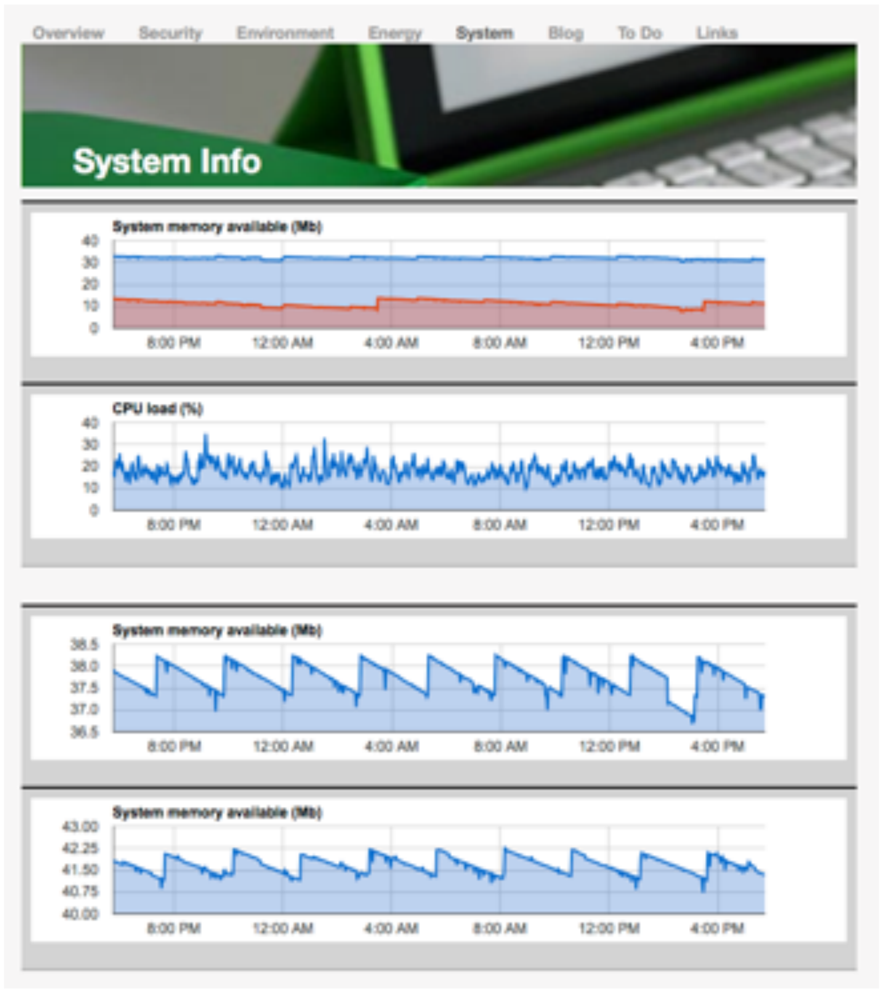
XXX can be one of the following (which may be truncated but should be unique):

- **`log`** - log of monitored events and watched variables
- **`events`** - log of events only
- **`devices`** - all devices (physical and virtual) on the system
- **`scenes`** - all scenes, their triggers, schedules, actions, and any attached Lua code
- **`security`** - current status and last tripped time for security devices
- **`environment`** - environmental sensors including temperature, humidity, light, and other generic types
- **`batteries`** or **`battery`** - current level of all battery powered devices
- **`cpu`** - graphical plot of CPU usage (5 minute averages)
- **`memory`** - graphical plot of memory usage (5 minute averages)
- **`plot=NNN`** - plots the major variable of the specified device number (if it is being watched)
- **`scene=NNN`** - plots a tree structure representing a specific scene (additional information appears as ‘tool tip’ text when cursor is over a specific item)
- **`event=text`** - record a user entry in the event log. Note that the text must be URL-escaped (ie. spaces replaced by %20, etc.)

Additionally, options may be added to size the report page, for example:

```
http://[your-Vera-IP]:3480/data_request?
id=lr_eventWatcher&report=scenes&width=1200&height=750
```

These requests may also be embedded into other web pages, so, for example, I have a web page which shows the recent CPU and Memory statistics of my three VeraLites.



All of the report types, and plots, are available over the MiOS secure servers with the following syntax (under UI5 only):

```
http://fwd7.mios.com/username/password/vera-serial-number/data_request?
id=lr_EventWatcher&report=XXX
```

you may need to change `fwd7` to your own appropriate server number.

Writing events to the log file

An HTTP request of the form:

```
http://[your-Vera-IP]:3480/data_request?id=lr_EventWatcher&event=YourMessageHere
```

will get written to the event log with a tag of 'User' and the message. Note that the message has to be URL encoded (no spaces, for example). You can do this from any machine which has access to Vera, including through the MiOS secure servers. If you're doing this from code on the Vera you log the event to, then you can use the 'local' URL 127.0.0.1. Here's a snippet of Lua code which will deal with the URL encoding:

```
local function myEvent (text)
  local url = require "socket.url"
  luup.inet.wget ("http://127.0.0.1:3480/data_request?id=lr_EventWatcher&event=" ..
url.escape(text))
end
```

So if you invoke it like this:

```
myEvent "hello world"
```

...then you get an entry in the event log like this:

```
2014-01-09 10:41:23.797 E   0 , User = hello world
```

The &report=treemap option

The &report=treemap option provides a graphical overview of all your devices, showing which ones are being watched and which not. It presents an overview of all your devices grouped into rooms: blue/grey ones are not being watched, yellow/gold ones are.

You can click on any room to zoom in on the devices there and a tooltip appears when you mouse over any device with a bit more information. You get back to the top level by right-clicking (or control-click on a Mac.)

Logging to a file on USB or NAS

Logging to a physical file (rather than just the internal memory buffer) is controlled by a device variable `LogDirectory` - if it's blank then no file is written; if it's non-blank it's the path to an **existing directory** where the log file is written, surviving restarts, one file per week.

The file name is `NNNN.txt`, where NNN is the Unix epoch week number - I'm writing this in week 2296 which starts on a Thursday. A bit bizarre, but easy to compute and, in fact, compatible with the way that dataMine arranges its own file system.

Typical entries are as follows:

```
2014-01-06 20:02:51.144 M 174 Power Meter, Watts = 2541
2014-01-06 20:05:59.134 X 346 Chandelier, Status = 0
2014-01-06 20:12:51.128 X 190 Dining Table, Status = 0
2014-01-06 20:20:35.235 S 294 Barn 4-in-1 (motion), Tripped = 1
2014-01-06 20:20:51.146 M 174 Power Meter, Watts = 3495
```

The file location could be anywhere accessible by Vera: internal 'disk' (not recommended, not much space); attached USB; NAS (but for this you will require CIFS installed - a good tutorial available on the forum.) EventWatcher doesn't mount any volumes, or create directories, itself, so the path has to be there, and accessible, already.

Assuming you have dataMine running and writing to a NAS mounted on /dataMine/, then setting `LogDirectory` to '/dataMine/EventWatcher/', after having created that directory manually on the NAS, will result in a file there: `2296.txt` (this week, `2297.txt` next week, etc...)

File size will depend, of course, on what you choose to watch and whether you also choose to monitor notification events, but currently I'm estimating about 500 lines per day for my largest system, which would mean ~200kB per file per week, or about 10Mb per year. Very manageable, and MUCH easier than trawling a full Vera log file.

Logging to syslog server

As well a logging the events to a physical file, EventWatcher can send them to a syslog server using UDP datagrams. If you don't know what this is, then you don't need it.

One device variables to set:

```
Syslog        xxx.xx.xx.xxx:yyy
```

…where xxx is the IP address and yyy the Port number.   If non-blank (and valid) then the syslog server is sent each event.

Logging Other Variables

In addition to the usual category-based method of defining which devices to watch, an additional feature allows ANY serviceId/variable to be watched.

The device variable parameter ExtraVariablesFile defines the location of a file with the description of one variable per line in the format <devNo>.<serviceId>.<Variable>. This file is read at startup and the requested variables added to the watch list.

So, for example, my ExtraVariablesFile is set to /www/EWextras.conf,
and that file contains:

```
    044.urn:akbooer-com:serviceId:EventWatcher1.AppMemoryUsed
    044.urn:akbooer-com:serviceId:EventWatcher1.CpuLoad05
```