Application Note: Log Files

Rev 1.0 March 2018

Understanding Log Files

Log files help us debug and understand issues with Wattmon. Log files are located in the /logs/ folder. There are various types of logs: System logs, alert logs, and CSV logs. Each of these is described in the following sections.

The System Log

The system log documents system restarts and other potentially relevant information such as script errors or warnings. The currently active system log is always called log.txt and historic log files have the format `logYYYYMMDD.txt` and `logYYYYMMDD-X.txt` if multiple log rotations have been initated for a particular day. The latest WattmonOS versions run a log rotate at midnight every to make it possible to manager potentially large log files remotely when debugging issues.

Each entry in a log file contains 1 log message, starting with a human-readable date and time followed by the alert level (Error, Warning) in most cases.

Detecting Reboots

There are several reasons why a Wattmon may reboot: Physical power cyel, software initated restart, watchdog timeout, or access violation. In order to determine the cause, search through the log for the following lines:

```
Wed Mar 14 10:36:00 2018: Warning: RCON=8000040 (or RCON=40 on PRO)
Wed Mar 14 10:36:00 2018: Warning: Software initiated reset
Wed Mar 14 10:36:00 2018: System booted
```

The first line gives you the RCON register (in the chip) value and the second line gives the human-readable interpretation of that value. In the above example the user most likely pressed the reset button in the WattmonOS web interface.

```
Wed Mar 14 10:40:03 2018: Warning: RCON=C8000003 (or RCON=3 on PRO)
Wed Mar 14 10:40:03 2018: Warning: Normal power up
Wed Mar 14 10:40:03 2018: System booted
```

In the above example, power was physically removed and restored to the Wattmon such as what would be typical with a power cut.

```
Fri Mar 09 10:57:59 2018: Warning: RCON=C8000010 (or RCON=10 on PRO)
Fri Mar 09 10:57:59 2018: Warning: Reset after watchdog timeout
Fri Mar 09 10:57:59 2018: System booted
```

In the above example, a code error or memory corruption caused the chip to get stuck in a loop forcing a restart. A firmware update with patch for the issue may be required if you see this regularly.

When a software exception occurs, usually the address of the issue is also included int he log file for debugging purposes as wit the following example:

```
Fri Mar 09 11:01:19 2018: Exception at 9D079F68
Fri Mar 09 11:01:19 2018: bus error (load/store)
Fri Mar 09 11:01:20 2018: Warning: RCON=40
Fri Mar 09 11:01:20 2018: Warning: Software initiated reset
```

Establishing Connectivity

Wattmon connectivity happens through both LAN and cellular dongles. Preference is given to a cellular dongle. Depending on the log level you will see more or less messages pertaining to connection status, especially while the dongle is connecting. Setting the log level to Verbose will result in most PPP connection messages to be logged, allowing for more detailed debugging.

A typical connection for a 3G dongle would appear as follows. See comments inline:

```
Sat Mar 10 16:11:49 2018: Notice: Init PPP module
 Sat Mar 10 16:12:01 2018: Notice: PPP connecting
 Sat Mar 10 16:12:11 2018: Notice: ppp:Error in connect - memFree=15904
 Sat Mar 10 16:12:11 2018: Notice: ppp: [out] LCP REQ
```

^ This initiates the LCP protocol. If this line is present, the dongle is properly recognized and the USB CDC channel is configured.

```
Sat Mar 10 16:12:11 2018: Notice: ppp: [in] LCP REQ
Sat Mar 10 16:12:11 2018: Notice: Opt 8=7 2
Sat Mar 10 16:12:11 2018: Notice: Opt 7=2 6
Sat Mar 10 16:12:11 2018: Notice: Opt 2=0 0
Sat Mar 10 16:12:11 2018: Notice: Opt 1=5 DC
Sat Mar 10 16:12:11 2018: Notice: Opt 5=0 0
Sat Mar 10 16:12:11 2018: Notice: Opt 3=C2 23
Sat Mar 10 16:12:11 2018: Notice: BAD 1=5 DC
Sat Mar 10 16:12:11 2018: Notice: BAD 5=0 0
Sat Mar 10 16:12:11 2018: Notice: BAD 3=C2 23
Sat Mar 10 16:12:11 2018: Notice: ppp: [out] LCP REQ REJ
Sat Mar 10 16:12:11 2018: Notice: ppp: [in] LCP ACK
Sat Mar 10 16:12:11 2018: Notice: ppp: LCP layer UP
Sat Mar 10 16:12:11 2018: Notice: ppp: [in] LCP REQ
Sat Mar 10 16:12:11 2018: Notice: Opt 8=7 2
Sat Mar 10 16:12:11 2018: Notice: Opt 7=2 6
Sat Mar 10 16:12:11 2018: Notice: Opt 2=0 0
Sat Mar 10 16:12:11 2018: Notice: BAD 7=2 6
Sat Mar 10 16:12:11 2018: Notice: ppp: [out] LCP REQ REJ
Sat Mar 10 16:12:11 2018: Notice: ppp: [in] LCP REQ
Sat Mar 10 16:12:11 2018: Notice: Opt 8=2 6
Sat Mar 10 16:12:11 2018: Notice: Opt 2=0 0
Sat Mar 10 16:12:12 2018: Notice: ppp: [out] LCP REQ ACK
```

^ The above sequence takes care of LCP protocol negotiation. We do not support CHAP and hence it goes through two cycles, until PAP (Password Authentication Protocol) is activated

```
Sat Mar 10 16:12:12 2018: Notice: ppp: [in] IPCP REQ [seq=1] [len=4]
[type=187]
Sat Mar 10 16:12:12 2018: Notice: ppp: [out] IPCP ACK [2.0.8.10]
[seq=1]
Sat Mar 10 16:12:16 2018: Notice: ppp: [out] IPCP REQ [0.0.0.0]
Sat Mar 10 16:12:16 2018: Notice: ppp: [in] IPCP NAK
Sat Mar 10 16:12:16 2018: Notice: ppp: [in] IPCP NAK IP=[2.0.8.10]
```

```
             Sat Mar 10 16:12:18 2018: Notice: ppp: [in] IPCP REQ
[seq=2] [len=4] [type=223]
Sat Mar 10 16:12:18 2018: Notice: ppp: [out] IPCP ACK [2.0.8.10]
[seq=2]
Sat Mar 10 16:12:20 2018: Notice: ppp: [out] IPCP REQ IP [2.0.8.10]
Sat Mar 10 16:12:21 2018: Notice: ppp: [in] IPCP NAK
Sat Mar 10 16:12:21 2018: Notice: ppp: [in] IPCP NAK IP=[2.0.8.10]
Sat Mar 10 16:12:24 2018: Notice: ppp: [in] IPCP REQ [seq=3] [len=4]
[type=3]
Sat Mar 10 16:12:24 2018: Notice: ppp: [out] IPCP ACK [2.0.8.10]
[seq=3]
Sat Mar 10 16:12:25 2018: Notice: ppp: [out] IPCP REQ IP [2.0.8.10]
Sat Mar 10 16:12:27 2018: Notice: ppp: [in] IPCP NAK
Sat Mar 10 16:12:27 2018: Notice: ppp: [in] IPCP NAK IP=[2.0.8.10]
Sat Mar 10 16:12:30 2018: Notice: ppp: [in] IPCP REQ [seq=4] [len=4]
[type=6]
Sat Mar 10 16:12:30 2018: Notice: ppp: [out] IPCP ACK [2.0.8.10]
[seq=4]
Sat Mar 10 16:12:31 2018: Notice: ppp: [out] IPCP REQ IP [2.0.8.10]
Sat Mar 10 16:12:31 2018: Notice: ppp: [in] IPCP NAK
Sat Mar 10 16:12:31 2018: Notice: ppp: [in] IPCP NAK IP=[10.17.101.47]
Sat Mar 10 16:12:31 2018: Notice: ppp: [out] IPCP REQ IP
[10.17.101.47]
Sat Mar 10 16:12:31 2018: Notice: ppp: [in] IPCP ACK
Sat Mar 10 16:12:31 2018: Notice: ppp: [in] IP = 10.17.101.47  [seq=5]
Sat Mar 10 16:12:31 2018: Notice: ppp: [in] DNS = 218.248.112.72
[seq=5]
Sat Mar 10 16:12:31 2018: Notice: ppp: IPCP layer UP
```

The final part of the negotiation happens at the IPCP layer while an IP address is obtained. The temporary ip of 2.0.8.10 is just a dummy IP address and it will keep requesting a new address until it obtains a valid one, in this case 10.17.101.47 at which point the IP Layer is fully established and the default route is set to the new interface.

Detecting Connectivity Issues

Assuming a connection is properly established as in the example above, at some point the provide may decide to terminate the connection without warning or the signal may drop. At this point Wattmon won't be able to connect to remote sockets, and you will see a pattern as below:

```
Thu Mar 08 11:56:50 2018: Warning on ip.cgi line 378 : Socket open
timed out!
Thu Mar 08 11:58:02 2018: Warning on ip.cgi line 378 : Socket open
timed out!
Thu Mar 08 11:59:13 2018: Warning on ip.cgi line 378 : Socket open
timed out!
Thu Mar 08 12:00:22 2018: Warning on ip.cgi line 378 : Socket open
timed out!
Thu Mar 08 12:01:49 2018: Warning on ip.cgi line 378 : Socket open
timed out!
Thu Mar 08 12:01:49 2018: Notice on ip.cgi line 612 : Cellular link
seems to have died, restarting USB power
```

It will try reconnecting several times and eventually will restart the power of the dongle. Unless there is a real issue where the tower signal is not present, this will allow for a new IP address to be obtained and bring up the connection again.

In some cases, the dongle actually terminates the connection in which case Wattmon will immediately try to re-negotiate a new IP address.

```
Fri Mar 09 10:35:26 2018: Notice: ppp: [in] LCP TERM
Fri Mar 09 10:35:26 2018: Notice: usb: Device detached
```

Miscellaneous Issues

When a non-recovable script errors it will show the type of error in the script and the line number as shown below:

```
Fri Mar 09 09:37:16 2018: Error on cronsec.cgi line 238 : Missing } in
include file
Fri Mar 09 09:37:16 2018: [uphp] Force recompile cronsec.cgc
```

The Force recompile line indicates that this script file (cronsec.cgi in this case) will be recompiled the next time it is run – this will resolve any issues of a corrupted comipled file.

The Alert Log

Wattmon actions can be used to create special alerts that are pushed to the Wattmon ems portal and can then be forwarded via sms/email to specific user accounts. These alerts are in the file /logs/alerts.txt.

Each alert resides on one line, as in the example below:

```
Tue Mar 13 16:09:58 2018: WARN|System restarted
```

There are 3 alert types define: INFO, WARN and ERROR. The | after the type separates the message (in the above example it’s System restarted) and the alert type. You can also add messages to the alert log programmatically using the log() function.

CSV Logs

Wattmon has a *Data Collection* module that lets you specify which roles and global variables are logged on a per-minute basis. The default is to log entries in a log file per day. This is sorted into sub-folders in the logs folder, by year and month. So the log file for July 12th 2017 would be stored in the following location:

```
/logs/2017/07/20170712_0.csv
```

The filename is formatted thus:
YYYYMMDD_X.csv
The X (0 in this case) is the first data collection index – it is possible but not recommended to use multiple data collection files to log different types of data.