Read AP7940 PDU from script

The AP7940 from American Power Conversion (https://www.apc.com) is an older unit with limited ability to automatically read the values. I came up with a set of Perl scripts which will do this.

It is definitely a hack. The first thing you must do is set up an sftp server and tell the AP7940 to write the log files to that location (defined as $logdir in this script).

This is done asynchronously. The AP7940 ftp's the log files hourly. *processPDULogs* is then set up as a cron job which runs once an hour. It looks for the log file, processes it, then deletes it (so the next sftp upload can overwrite).

NOTE: processPDULogs must have read/write access to $logdir and, obviously, so must the sftp server.

The second script, *getPDU*, is used to read the file. When called with the name of the PDU and the key to be read, returns the value (useful for Zabbix monitoring, http://zabbix.com)

processPDULogs

```
#! /usr/bin/perl

# Script reads data log produced by APC AP7940 which if ftp'd to
$logDir
# it finds the last entry read (from status file) and scrolls down
through
# log file until it finds the next entry.
# then summarizes the entries (only happens once an hour) and stores
the value
# in the out file.

# Copyright 2015, Rod, Daily Data, Inc.
# Free to use, modify, change, anything you want
# You do NOT have to give attribution, but it would be
# appreciated.
# Also, if you make changes and want to give them back, just send them
to
# us. Visit the http://www.dailydata.net and select Contact Us

my $logDir = '/home/apcpdu'; # location where log is ftp'd to by AP7940
my @PDUNames = ('pdu1','pdu2'); # names of pdu's to look for
my $logFileExtension = 'log'; # extension expected for log file name
my $statusExtension = 'status'; # extension used for recording last
record processed
my $outputFileExtension = 'out'; # extension used for most recent
summary
my $voltage = 208; # could not figure out how to get voltage from pdu,
so hard coded
my $pf = 1; # ditto power factor
```

```
foreach my $thisLog ( @PDUNames ) {
   my $statusFile = $logDir . '/' . $thisLog . '.' . $statusExtension;
   my $logFile = $logDir . '/' . $thisLog . '.' . $logFileExtension;
   my $outFile = $logDir . '/' . $thisLog . '.' . $outputFileExtension;
   my @log;
   my $status = 0;
#   print "Processing log from $thisLog\n";
   if ( -f $statusFile ) {
#      print "\tReading $statusFile\n";
      open STATUS, "<$statusFile";
      $status = <STATUS>;
      close STATUS;
   }
   if ( -f $logFile ) {
#      print "\tReading  $logFile\n";
      open LOG,"<$logFile" or die "Could not read $logFile: $!\n";
      my $log = join( '', <LOG> );
      @log = split( "\n", $log );
      close LOG;
   } else {
      next;
   }
   chomp @log;
#   print "\tTrying to find header\n";
   my $line = 0;
   while ( $line < @log ) {
      last if $log[$line] =~ m/Date\s+Time\s+I\s+IMax\s+IMin/;
      $line++;
   };
#   print "\tFinding first line to process\n";
   if ( $status ) {
      while ( $line < @log && $log[$line] ne $status ) {
         $line++;
      }
   }
   $line++;
   # we should now be at the first entry AFTER the last one read
#   print "First line to process would be line $line\n$log[$line]\n";
   my $max = 0;
   my $min = 1000;
   my $sum = 0;
   my $count = 0;
   my $datetime; # last date and time read
   while ( $line < @log ) {
      my ($date,$time,$current,$currentMax,$currentMin) = split( /\s+/,
$log[$line]);
      $max = $currentMax if $currentMax > $max;
      $min = $currentMin if $currentMin < $min;
      $count++;
```

```
      $sum += $current;
      $datetime = "$date $time";
      $line++;
   }
   next unless $count;
   open LOG,">$outFile" or die "Could not write to $outFile: $!\n";
   print LOG "timestamp\t$datetime\n";
   print LOG "average\t" . $sum/$count . "\n";
   print LOG "max\t$max\n";
   print LOG "min\t$min\n";
   print LOG "count\t$count\n";
   print LOG "voltage\t$voltage\n";
   print LOG "pf\t$pf\n";
   close LOG;
   `chmod 666 $outFile`;
   open STATUS,">$statusFile" or die "Could not write to $statusFile:
$!\n";
   print STATUS $log[scalar(@log)-1];
   close STATUS;
   `mv $logFile $logFile.old`;
}

1;
```

getPDU

```
#! /usr/bin/perl -w

# script to get pdu data from summary file created by
# processPDULogs. Simply reads file pduname.out in $outputDir
# with pduname being the value of the first parameter
# returns the entry from the tab delimited file in the form
# key value
# where key is passed in as the second value

# Copyright 2015, Rod, Daily Data, Inc.
# Free to use, modify, change, anything you want
# You do NOT have to give attribution, but it would be
# appreciated.
# Also, if you make changes and want to give them back, just send them
to
# us. Visit the http://www.dailydata.net and fill out the contact form

my $pduName = shift;
my $key = shift;

my $outputExtension = 'out';
my $outputDir = '/home/apcpdu';

# retrieve one value (based on $key) from $datafile
```

```
sub getValue {
   my $datafile = shift;
   my $key = shift;
   return -1 unless $key;
   # open data file and get line with key in it
   # maybe faster using grep???
   open DATA, "<$datafile" or die "No data to read: $!\n";
   my @lines = grep { /^$key\t/ } <DATA>;
   close DATA;
   # clean up line and return the value (second field of tab delim
line)
   chomp( $lines[0] );
   ($key,$value) = split ("\t", $lines[0] );
   return $value;
}

die "Invalid PDU name" unless $pduName;
my $filename = "$outputDir/$pduName.$outputExtension";
die "No Datafile" unless -e $filename;

die "No key given" unless $key;

print getValue( $filename, $key );

1;
```

From:
https://kb.unixservertech.com/ - **Unix Server Tech Knowledge Base**

Permanent link:
https://kb.unixservertech.com/other/apc/network_read

Last update: **2016/11/06 01:08**