ChromeOS System Report

ChromeOS does not have a simple way to record a system's configuration. The following will get the information similar to Belarc Advisor (https://www.belarc.com/products/belarc-advisor).

The resulting mhtml file is fairly useless, but can be parsed using the included script. Additionally, we can save it as a pdf, meaning it is viewable on most systems.

1. Open Chrome web browser
2. Enter URL as chrome://system
3. Click “Expand All”
4. Right click in page (use two fingers if no mouse)
5. Select “Save As”
6. Choose Webpage, Single File (default)
7. name the file your_nameChrome.mhtml, so in my case, it would be RodChrome.mhtml
8. Save in your downloads folder
9. Right click again
10. Choose Print (it will hang for a minute or two)
11. Choose Save as PDF
12. Save as your_nameChrome.pdf (ie, RodChrome.pdf)

You now have two large files in your downloads folder, and mhtml file and a PDF. The following Perl script parses the mhtml file, using ====CATEGORY==== as headers, with the contents afterwards. I have written it to NOT include the logs. Use the command:

```
tr -d '\r' < filename | ./processChrome.pl > filename.txt
```

processChrome.pl

```
#! /usr/bin/env perl

use strict;
use warnings;

# tr -d '\r' < filename | ./processChrome.pl


my %lookFor = (
   'div-bios-info-value' => 'bios_info',
   'div-blkid-value' => 'blkid',
   'div-cpuinfo-value' => 'cpuinfo',
   'div-disk-usage-value' => 'disk_usage',
   'div-ifconfig-value' => 'ifconfig',
   'div-lsblk-value' => 'lsblk',
   'div-meminfo-value' => 'meminfo',
   'div-network-devices-value' => 'network_devices',
   'div-os-release-NAME-value' => 'os-release NAME',
   'div-os-release-VERSION-value' => 'os-release VERSION',
   'div-uname-value' => 'uname',
```

```
   'div-vpd-2-0-value' => 'vpd_2.0'
);

sub getLine {
   my @lines;
   while ( my $thisLine = <> ) {
      chomp $thisLine;
      if ( substr( $thisLine, -1 ) eq '=' ) {
         #die "$thisLine\n";
         chop $thisLine; # remove the equals sign
         push @lines, $thisLine;
      } else {
         push @lines, $thisLine;
         #print "returning\n$thisLine\n";
         return join( '', @lines ) . "\n";
      }
   }
}


my %found;
my $line;
my $count = 100;
my $category = '';
while ( $line = &getLine() ) {
   #print "$line\n";
   #die unless $count--;
   $line =~ s/=3D/=/gi;
   # Ok, if we have a category already, and have found a </div>, clear
the category
   $category = '' if $category && $line =~ m"</div>";
   if ( $line =~ m/^.*<div.*id="(div-[^"]+-value)">(.*)$/ ) {
      my $key = $1;
      $line = $2;
      #print "$key\n$line\n"; die;
      # this is one of the ones we want, so start using it
      if ( $lookFor{$key} ) {
         $category = $lookFor{$key};
         #print "$key\n$line\n"; die;
         $found{$category} = $line . "\n";
      }
   } elsif ( $category ) {
      $found{$category} .= $line;
   }
}

foreach my $key ( sort keys %found ) {
   print "\n====$key====\n" . $found{$key} . "\n";
}
```

```
1;
```