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.

Note: The mhtml output uses Windows line endings, so if you're running this on a Unix machine, you'll want to run it through tr to get convert it to Unix line endings.

Use the command:

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

processChrome.pl

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

use strict;
use warnings;

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

# anything matching these keys will be include, and all else
# will be excluded. If empty, will include everything not
# matching $ignoreKeys. In this case, we've commented out
# everything, so it will output everything not ignored
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'
);

# anything matching this regex will be ignored
# this matches anything with log, profile or ui-heirarchy in
# the key. NOTE: match is case insensitive
my $ignoreRegex = qr'.*(log)|(profile)|(ui-hierarchy).*'i;


# get a single line. This is more difficult since the output
# takes very long lines and separates them into 80 column blocks
# the indicator for this continuation is an equals sign (=)
# at the end of the line. Thus, we continue to read lines from
# STDIN so long as the current line ends with an equal sign
# and we concatinate all of the lines together
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";
      }
   }
}

# makes a category name from the div name
sub makeCategory {
   my $initValue = shift;
   $initValue =~ m/div-(.*)-value/;
   $initValue = $1;
   $initValue =~ s/-/ /g;
   return $initValue;
}


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>";
   # check for a line like <div id="div-something-value"> and begin
processing if found
   # $matches[0] has the key name (ie, something), and $matches[1] has
any subsequent
   # information, generally the beginning of the actual block
   if ( my @matches = ($line =~ m/^.*<div.*id="(div-[^"]+-
value)">(.*)$/ ) ) {
#      print "Found div $matches[0]\n";
      # skip if $ignoreRegex defined has matches. Note, this is case
insensitive
#      die "Checking $matches[0]\nagainst\n$ignoreRegex\n";
      next if $ignoreRegex && $matches[0] =~ m/.*($ignoreRegex).*/;
      # if %lookFor empty, or not empty and has matching key, inlcude
it
      if ( ! %lookFor || $lookFor{$matches[0]} ) {
         # Create the category
         $category = defined( $lookFor{$matches[0]} ) ?
$lookFor{$matches[0]} : &makeCategory( $matches[0] ) ;
         # include any information beginning the div
         $found{$category} = $matches[1] . "\n";
      }
   } elsif ( $category ) {
      # we're in the middle of a div, so just add the line.
      $found{$category} .= $line;
   }
}

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

1;
```