

```

#! /usr/local/bin/perl -w
#
#####
# Script Name.....: SEE BELOW
# Version/Date.....: SEE BELOW
# Author.....: Noel Milton Vega (Rensselaer Technology Group, LTD for NetApp).
# Maintained By...: Noel Milton Veg
# October 12, 2009
#####
my $progVer = "0.9.9";
our $scriptName = "filerStatusCheck";
#####
# PERL2EXE Excludes (Must come BEFORE any 'use' directives.)      #
#####
#perl2exe_exclude "Digest/Perl/MD5.pm"
#####
use strict;
use warnings;
use Getopt::Std;
use Time::Local;
use Digest::MD5 qw(md5 md5_hex md5_base64);
#####

#####
# PROGRAM "PIRACY PROTECTION" & "TRIAL VERSION" HARDCODE SETTINGS.      #
#                                                                           #
# $hostID_Hardcode = LINUX: 'hostid' command output (root# hostid).      #
# $hostID_Hardcode = WIN32: 'C:' Volume Serial number (c:\> dir c:).      #
#                                                                           #
# $maxDate_TrialHardcode = yyyy/mm/dd (e.g.: 2008/02/19)                #
# $maxDate_TrialHardcode = "-1" # NOT a trial version (i.e. Purchased).  #
#####
my $hostId_Hardcode      = "a8c0067e";
my $maxDate_TrialHardcode = "-1";
my $maxDate_TrialHardcode = "2008/11/16";
my $licenseStatus       = "PERMANENT/1-SERVER BOUND";
($maxDate_TrialHardcode ne "-1") && ($licenseStatus = "TRIAL: EXPIRES ON ${maxDate_TrialHardcode}");
licenseCheck();
#####

#####
# Get and process arguments passed to this program.
#####
getopts('hg:m:l:');
our ($opt_g, $opt_h, $opt_m, $opt_l);
( defined ${opt_h} ) && Usage(); # Put this check before the rest.
(! defined ${opt_g} ) && Usage();
(! defined ${opt_m} ) && Usage();
(! defined ${opt_l} ) && Usage();
#
($opt_g =~ m/^-/) && Usage();
($opt_m =~ m/^-/) && Usage();

```

```

($opt_l =~ m/^-/) && Usage();
($#ARGV != "-1") && Usage();
#
my $dfmGroup = $opt_g;
my $mailHost = $opt_m;
my $lagHWM = $opt_l;
#####

#####
# Define an array of ONTAP commands to be run.      #
#####
my @ontapCmds =
("sysconfig -a;",
 "vol status;",
 "vol status -f;",
 "fcadmin device_map;",
 "snapmirror status;",
 "aggr status;",
 "vol status -s;",
 "disk show -n;",
 "storage show disk -p;",
 "vif status;",
 "options autosupport;",
);
#####

#####
# Define Variables                                #
#####
my (@filers, $result);
my (@htmlTable, @htmlTableCellData);
my $htmlRow = 0;
my $htmlCol;
#
#my $logDir;
#($^O =~ m/linux/i) && ($logDir = "/tmp");
#($^O =~ m/MSWin32/i) && ($logDir = "C:/Temp");
#unless (-d $logDir) {mkdir($logDir, 0755) or die "Can't mkdir(): $logDir\n"};
#####

#####
# Retrieve the list of Storage Arrays known to DFM. #
#####
open (PIPE, "dfm group list -q -t host -m -r \"\$dfmGroup\" |") or die "Cannot get Filer list from '$dfmGroup': $!\n";
while (defined ($_ = <PIPE>))
{
$_ =~ s/^\s+//; # Remove leading white spaces.
#####
# "die" with message if we cannot get a      #
# of filers.                                #

```

```

#####
($_ =~ m/^(There are no hosts\./i) && die "Cannot get Filer list: $_";
($_ =~ m/^(Error: There is no object named '\${dfmGroup}'\./i) && die "Cannot get Filer list: $_";
($_ =~ m/^(The \${dfmGroup} group has no members\./i) && die "Cannot get Filer list: $_";
#####
next if ($_ !~ /\d+\s+[\^s]/); chomp $_;
push @filers, (split(/\s+/, $_))[1]; # Array Host names.
}
close (PIPE);

#####
# Convert all filer names to lowercase; #
# Then ensure there are no duplicates due #
# to potentially nested DFM group Names. #
#####
my (@tmpArray, %tmpHash);
foreach my $filer (@filers)
{
    push(@tmpArray, "\L$filer");
}
@filers = grep (!$tmpHash{$_}++, @tmpArray);
undef @tmpArray; undef %tmpHash;
#####

#####
# TEST ONLY
#####
our @filers; # TEST ONLY
do "./FILER_RAWDATA.d/filerArray.pl"; # TEST ONLY
#print map ("$_ \n", @filers); # TEST ONLY
#exit 0; # TEST ONLY
#####

#####
# PERL does not provide a simple method for #
# for creating multi-dimensional arrays. And #
# we need a 2D-Array to store the contents of #
# the HTML table output we are building. So we #
# manually build this 2D array structure here. #
# #
# The HTML table will have one ROW for each #
# filer (plus +1 extra for the table heading #
# ROW), and each row will have a COLUMN for #
# every ONTAP status condition we are reporting #
# on (note: we do not include the filerName #
# column in this count). Thus: #
# #
# Num HTML table rows = $#filers + 1 #
# Num HTML table cols = Set as necessary #
# #

```

```

# We run through loop1 "$#filer + 1" number of #
# times, and with each iteration, we "push" #
# another array reference onto the end of the #
# "@htmlTableCellData" (making it an array of #
# arrays-references); and through loop2 #
# "$nbrColumns" number of times to "grow" the #
# array that this new reference points to, so #
# that it ends up containing "$nbrColumns" #
# number of elements. #
#####
# You MUST grow this as you add more status columns.
my $nbrColumns = 13;
#####
for my $row (0 .. $#filers + 1)
{
  push @htmlTableCellData, ["foo"];
  for my $col (0 .. $nbrColumns)
  {
    $htmlTableCellData[$row][$col] = "<font color=#FF0000>" . "NoData($row,$col)" . "</font>";
    ($col == 0 && $row != 0) && ($htmlTableCellData[$row][$col] = "<font color=#00FF00>" . "$filers[$row - 1]" . "</font>");
    ($col == 0 && $row == 0) && ($htmlTableCellData[$row][$col] = "<font color=#FFFFFF>" . "Filer / ONTAP Cmd" . "</font>");
  }
}
#####

#####
# #
#####
print "\n#####\n";
print "# Connecting to storage controllers and retrieving/analyzing RAW data... #";
print "\n#####\n";

foreach my $filer (@filers)
{
  print "  ${filer}:\t\t";
  #####
  # Ensure we can connect to the filer and #
  # get output. If we cannot, simply skip #
  # that filer and let the data in the #
  # HTML table ROW for that filer indicate #
  # "NoData" (as initialized above). #
  #####
  $result = `dfm run cmd -t 60 $filer version`;
  #next if ($result !~ m/^Stdout:\sNetApp\sRelease\s/smi);
  $result = `dfm run cmd -t 60 $filer "@ontapCmds[0,1,2,3]"`;
  $result = $result . `dfm run cmd -t 60 $filer "@ontapCmds[4,5,6,7]"`;
  $result = $result . `dfm run cmd -t 60 $filer "@ontapCmds[8,9,10]"`;
  #
  $result = `cat ./FILER_RAWDATA.d/$filer`; # TEST ONLY (AND COMMENT OUT 5 LINES ABOVE).
  print "[Data Received]";
}

```

```

#####
# Reposition the HTML table cell pointer #
# to the start of the next row, col 1.  #
#####
$htmlRow++; $htmlCol = 1;
#####
checkFiler($filer, $result, $htmlRow, $htmlCol);
print "[Data Analyzed]\n";
}

print "#####\n";
#####

#####
sub checkFiler {

my $filer = shift;
my $result = shift;
my $htmlRow = shift;
my $htmlCol = shift;

#Stdout:          Model Name:          FAS3050

#####
# PLATFORM MODEL #
#####
$htmlTableCellData[0][$htmlCol] = "<font color=#FFFFFF>" . "MODEL" . "</font>";
# ===== #
# If we don't get inside the next 'if' section, then #
# the data for this cell will remain as "NoData". #
# ===== #
$_ = grep {/^Stdout:\s+Model Name:\s+\w/i} (split(/\n/, $result));
if ($_ != 0)
{
my $model = (grep {/^Stdout:\s+Model Name:\s+\w/i} (split(/\n/, $result)))[0];
$model = (split(/\s+/, $model))[3];
$htmlTableCellData[$htmlRow][$htmlCol] = "<font color=#00FF00>" . ${model} . "</font>";
}
$htmlCol++;

#####
# ONTAP VERSION #
#####
$htmlTableCellData[0][$htmlCol] = "<font color=#FFFFFF>" . "vONTAP" . "</font>";
# ===== #
# If we don't get inside the next 'if' section, then #
# the data for this cell will remain as "NoData". #
# ===== #
$_ = grep {/^Stdout:\s+NetApp Release (\d+\.)+\d+:\s/i} (split(/\n/, $result));
if ($_ != 0)
{

```



```

}
@shelfFWentries = grep (!$tmpHash{$_}++, @shelfFWentries); # Basically "sort -u" array.
@shelfModuleTypes = grep (!$tmpHash{$_}++, @shelfModuleTypes); # Basically "sort -u" array.
#####

#####
# The "if" test below checks for Case1 and Case2. #
# The "else / for" loop below checks for Case3. #
# #
# Case1: A shelf is missing a module where it shouldn't #
# necessarily have to. Shelf-1 in this example #
# has two AT-FCX module; so Shelf-2 should #
# be able to have the full complement of two #
# AT-FCX modules too. Shelf-3 has only one #
# module of type AT-FC, so there is no reason #
# to think that Shelf-4 should or could have #
# two modules of type AT-FC. Some older systems #
# work this way, so we consider it correct. #
# Shelf 1: AT-FCX FIRMWARE REV. AT-FCX A: 36 AT-FCX B: 36 #
# Shelf 2: AT-FCX FIRMWARE REV. AT-FCX A: 36 (missing) #
# Shelf 3: AT-FC FIRMWARE REV: 0208 #
# Shelf 4: AT-FC FIRMWARE REV: 0208 #
# #
# Case2: Identical modules types on different shelves run #
# different F/W revisions (even though the F/W #
# levels are the same within a shelf). #
# Shelf 1: AT-FCX FIRMWARE REV. AT-FCX A: 36 AT-FCX B: 36 #
# Shelf 2: AT-FCX FIRMWARE REV. AT-FCX A: 37 AT-FCX B: 37 #
# #
# Case3: Identical modules types on the same shelf run #
# different F/W revisions. #
# Shelf 1: ESH4 FIRMWARE REV. ESH A: 30 AT-FCX B: 31 #
#####
if ($#shelfFWentries != $#shelfModuleTypes) {
    $htmlTableCellData[$htmlRow][$htmlCol] = "<font color=#FF0000>" . "Missing Shelf or Inconsistent F/W" . "</font>";
    $htmlTableCellData[$htmlRow][0] = "<font color=#FF0000>" . "$filers[$htmlRow - 1]" . "</font>";
}
else {
    for $_ (@shelfFWentries)
    {
        my $fwRev1 = $_; $fwRev1 =~ s/[^:]+\s+(\d+).*/$1/;
        my $fwRev2 = $_; $fwRev2 =~ s/.*\s+(\d+).*/$1/;
        if ($fwRev2 != $fwRev1)
        {
            $htmlTableCellData[$htmlRow][$htmlCol] = "<font color=#FF0000>" . "Inconsistent F/W" . "</font>";
            $htmlTableCellData[$htmlRow][0] = "<font color=#FF0000>" . "$filers[$htmlRow - 1]" . "</font>";
            last;
        }
    }
}
}
$htmlCol++;

```

```

#####
# ONTAP Command: "aggr status" #
# - Raise Flag if any aggregate is not online. #
#####
$htmlTableCellData[0][$htmlCol] = "<font color=#FFFFFF>" . "AGGR status" . "</font>";

# ===== #
# If we don't get inside the next 'if' section, then #
# the data for this cell will remain as "NoData". #
# ===== #
$_ = grep {/^Stdout:\s+.*\s+raid[_0-9][^,]*,\s+aggr\s+\/} (split(/\n/, $result));
if ($_ != 0)
{
    $htmlTableCellData[$htmlRow][$htmlCol] = "<font color=#00FF00>" . "AGGRs online" . "</font>";
    for $_ (grep {/^Stdout:\s+.*\s+raid[_0-9][^,]*,\s+aggr\s+\/} (split(/\n/, $result)))
    {
        if ($_ !~ m/\s+online\s+raid[_0-9][^,]*,\s+aggr/i)
        {
            $htmlTableCellData[$htmlRow][$htmlCol] = "<font color=#FF0000>" .
                "AGGR(s) offline" .
                "</font>";
            $htmlTableCellData[$htmlRow][0] = "<font color=#FF0000>" . "$filers[$htmlRow - 1]" . "</font>";
            last;
        }
    }
}
$htmlCol++;

#####
# ONTAP Command: "vol status" #
# - Raise Flag if any volume is not online. #
#####
$htmlTableCellData[0][$htmlCol] = "<font color=#FFFFFF>" . "VOL status" . "</font>";

# ===== #
# If we don't get inside the next 'if' section, then #
# the data for this cell will remain as "NoData". #
# ===== #
$_ = grep {/^Stdout:\s+.*\s+raid[_0-9][^,]*,\s+(trad|flex)\s+\/} (split(/\n/, $result));
if ($_ != 0)
{
    $htmlTableCellData[$htmlRow][$htmlCol] = "<font color=#00FF00>" . "VOLs online" . "</font>";
    for $_ (grep {/^Stdout:\s+.*\s+raid[_0-9][^,]*,\s+(trad|flex)\s+\/} (split(/\n/, $result)))
    {
        if ($_ !~ m/\s+online\s+raid[_0-9][^,]*,\s+(trad|flex)/i)
        {
            $htmlTableCellData[$htmlRow][$htmlCol] = "<font color=#FF0000>" .
                "VOL(s) offline" .
                "</font>";
            $htmlTableCellData[$htmlRow][0] = "<font color=#FF0000>" . "$filers[$htmlRow - 1]" . "</font>";
            last;
        }
    }
}

```

```

    }
}
$htmlCol++;

#####
# ONTAP Command: "vol status -f" #
# - Raise Flag if any output is *not* literally: #
# "Stdout: Broken disks (empty)" #
#####
$htmlTableCellData[0][$htmlCol] = "<font color=#FFFFFF>" . "Broken Disks" . "</font>";

# ===== #
# If we don't get inside the next 'if' section, then #
# the data for this cell will remain as "NoData". #
# ===== #
($_) = (grep {/^Stdout:\s+Broken\s+disks\s*/} (split(/\n/, $result)));
if (defined $_)
{
    $htmlTableCellData[$htmlRow][$htmlCol] = "<font color=#00FF00>" . "No Broken Disks" . "</font>";
    if ($_ !~ /^Stdout:\s+Broken\s+disks\s+(\empty)\s*/)
    {
        $htmlTableCellData[$htmlRow][$htmlCol] = "<font color=#FF0000>" . "Broken Disks" . "</font>";
        $htmlTableCellData[$htmlRow][0] = "<font color=#FF0000>" . "$filers[$htmlRow - 1]" . "</font>";
    }
}
$htmlCol++;

#####
# ONTAP Command: "fcadmin device_map" #
# - Raise Flag if any line has string "xxx" or "byp".#
#####
$htmlTableCellData[0][$htmlCol] = "<font color=#FFFFFF>" . "Missing/Bypassed Disks" . "</font>";

# ===== #
# If we don't get inside the next 'if' section, then #
# the data for this cell will remain as "NoData". #
# ===== #
$_ = grep {/^Stdout:\s+Shelf\s+\d+:(\s+\d+)+/} (split(/\n/, $result));
if ($_ != 0)
{
    $htmlTableCellData[$htmlRow][$htmlCol] = "<font color=#00FF00>" . "NONE" . "</font>";
    for $_ (grep {/^Stdout:\s+Shelf\s+\d+:(\s+\d+)+/} (split(/\n/, $result)))
    {
        if ($_ =~ m/^(.*(XXX|BYP).*)$/i)
        {
            $htmlTableCellData[$htmlRow][$htmlCol] = "<font color=#FF0000>" . "YES" . "</font>";
            $htmlTableCellData[$htmlRow][0] = "<font color=#FF0000>" . "$filers[$htmlRow - 1]" . "</font>";
            last;
        }
    }
}
}
}

```

```

$htmlCol++;

#####
# ONTAP Command: "vol status -s" #
# - Output the number of spare disks found. #
# # #
# Stdout: Spare disks (empty) #
# Stdout: Spare disks #
# Stdout: Spare disks for block or... [omitted] ... #
# Stdout: Pool1 spare disks (empty) #
#####
$htmlTableCellData[0][$htmlCol] = "<font color=#FFFFFF>" . "# Spare Disks" . "</font>";

# ===== #
# If we don't get inside the next 'if' section, then #
# the data for this cell will remain as "NoData". #
# ===== #
$_ = grep {/^Stdout:\s+(\w\s)?Spare\s+Disks\s*/i} (split(/\n/, $result));
if ($_ != 0)
{
    $htmlTableCellData[$htmlRow][$htmlCol] = "<font color=#FF0000>" . "0" . "</font>";
    $_ = grep {/^Stdout:\s+spare\s+\w+\.\w+\s+.*\s+FC:/} (split(/\n/, $result));
    $htmlTableCellData[$htmlRow][$htmlCol] = "<font color=#00FF00>" . "$_" . "</font>";
    ($_ == 0) && ($htmlTableCellData[$htmlRow][0] = "<font color=#FF0000>" . "$filers[$htmlRow - 1]" . "</font>");
}
$htmlCol++;

#####
# ONTAP Command: "disk show -n" #
# - Raise Flag if output is *not* literally: #
# "disk show: No disks match option -n.", or "N/A" #
# if the "disk show" command doesn't exist (i.e. #
# hardware disk ownership). #
#####
$htmlTableCellData[0][$htmlCol] = "<font color=#FFFFFF>" . "Disk Ownership" . "</font>";

# ===== #
# If we don't get inside the next 'if' section, then #
# the data for this cell will remain as "NoData". #
# ===== #
$_ = grep {(/^Stdout:\s+DISK\s+OWNER\s+POOL\s+SERIAL\s+NUMBER\s*$/ |
            /^Stderr: disk show: No disks match option -n\.$/ |
            /^Stderr: disk: Did not recognize option \"show\".\.$/)} (split(/\n/, $result));
if ($_ != 0)
{
    for $_ (grep {(/^Stdout:\s+DISK\s+OWNER\s+POOL\s+SERIAL\s+NUMBER\s*$/ |
                /^Stderr: disk show: No disks match option -n\.$/ |
                /^Stderr: disk: Did not recognize option \"show\".\.$/)} (split(/\n/, $result)))
    {
        if ($_ eq "Stderr: disk show: No disks match option -n.")
        {

```

```

$htmlTableCellData[$htmlRow][$htmlCol] = "<font color=#00FF00>" . "All Owned" . "</font>";
last;
}
elseif ($_ eq "Stderr: disk: Did not recognize option \"show\".")
{
$htmlTableCellData[$htmlRow][$htmlCol] = "<font color=#00FF00>" . "N/A: H/W Owned" . "</font>";
last;
}
elseif ($_ =~ m/^Stdout:\s+DISK\s+OWNER\s+POOL\s+SERIAL\s+NUMBER\s*$/i)
{
$htmlTableCellData[$htmlRow][$htmlCol] = "<font color=#FF0000>" . "Unowned Disks" . "</font>";
$htmlTableCellData[$htmlRow][0] = "<font color=#FF0000>" . "$filers[$htmlRow - 1]" . "</font>";
last;
}
}
}
$htmlCol++;

```

```

#####
# ONTAP Command: "storage show disk -p" #
# - Raise Flag if any disk is not attached to an A #
# *and* a B port (i.e. not mulitpathed) -OR- it #
# is but it's A and B ports are attached to the #
# same IO board number or, for on-board port 0, to #
# the same ASIC. #
# #
# CABLED RIGHT EXAMPLES: #
# PRIMARY PORT SECONDARY PORT SHELF BAY #
# ----- #
# 0a.16 A 0[c|d|e|f|g|h].16 B 1 #
# 0b.16 A 0[c|d|e|f|g|h].16 B 1 #
# 0c.16 A 0[a|b|e|f|g|h].16 B 1 #
# 0d.16 A 0[a|b|e|f|g|h].16 B 1 #
# 0a.16 A 2a.16 B 1 0 #
# 1b.16 A 2a.16 B 1 0 #
# #
# NOT CABLED RIGHT EXAMPLES: #
# PRIMARY PORT SECONDARY PORT SHELF BAY #
# ----- #
# 1a.16 A 1c.16 B 1 0 #
# 0a.16 A 0b.16 B 1 0 #
# 0b.16 A 0a.16 B 1 0 #
# 0c.16 A 0d.16 B 1 0 #
# 0d.16 A 0c.16 B 1 0 #
#####

```

```

$htmlTableCellData[0][$htmlCol] = "<font color=#FFFFFF>" . "MPHA Status" . "</font>";

```

```

# ===== #
# If we don't get inside the next 'if' section, then #
# the data for this cell will remain as "NoData". #
# ===== #
$_ = grep {(/^Stdout:\s+(\w+\.\d+\s+[AB]\s+)+\d+\s+\d+\s*$/)} (split(/\n/, $result));

```

```

if ($_ != 0)
{
$htmlTableCellData[$htmlRow][$htmlCol] = "<font color=#00FF00>" . "MPHA OK" . "</font>";
for $_ (grep {(/^Stdout:\s+(\w+\.\d+\s+[AB]\s+)+\d+\s+\d+\s*$)/} (split(/\n/, $result)))
{
#####
# If an A and a B side doesn't exist, then FAIL. #
#####
if ($_ !~ m/^Stdout:\s+(\w+\.\d+\s+A\s+(\w+\.\d+\s+B\s+\d+\s+\d+\s*$)/ &&
    $_ !~ m/^Stdout:\s+(\w+\.\d+\s+B\s+(\w+\.\d+\s+A\s+\d+\s+\d+\s*$)/)
{
# YELLOW since some models can't be MPHA'ed; Or customers chose not to do this.
$htmlTableCellData[$htmlRow][$htmlCol] = "<font color=#FFFF00>" . "MPHA NotOk" . "</font>";
$htmlTableCellData[$htmlRow][0] = "<font color=#FFFF00>" . "$filers[$htmlRow - 1]" . "</font>";
last;
}

#####
# An A and B side exists. FAIL if any of the following isn't true. #
#####
my $ioBoardNbr1 = (split (/s+/, $_))[1]; $ioBoardNbr1 =~ s/(\d+).*/$1/;
my $ioBoardNbr2 = (split (/s+/, $_))[3]; $ioBoardNbr2 =~ s/(\d+).*/$1/;
my $ioPort1 = (split (/s+/, $_))[1]; $ioPort1 =~ s/(\w+).*/$1/;
my $ioPort2 = (split (/s+/, $_))[3]; $ioPort2 =~ s/(\w+).*/$1/;
#####
next if ($ioPort1 eq "0a" && $ioPort2 =~ m/0[cdefgh]/);
next if ($ioPort1 eq "0b" && $ioPort2 =~ m/0[cdefgh]/);
next if ($ioPort1 eq "0c" && $ioPort2 =~ m/0[abefgh]/);
next if ($ioPort1 eq "0d" && $ioPort2 =~ m/0[abefgh]/);
next if ($ioPort1 eq "0e" && $ioPort2 =~ m/0[abcdgh]/);
next if ($ioPort1 eq "0f" && $ioPort2 =~ m/0[abcdgh]/);
next if ($ioPort1 eq "0g" && $ioPort2 =~ m/0[abcdef]/);
next if ($ioPort1 eq "0h" && $ioPort2 =~ m/0[abcdef]/);
next if ($ioBoardNbr1 != $ioBoardNbr2); #Note this "next if" must come last.
#####
$htmlTableCellData[$htmlRow][$htmlCol] = "<font color=#FF0000>" . "MPHA NotOk" . "</font>";
$htmlTableCellData[$htmlRow][0] = "<font color=#FF0000>" . "$filers[$htmlRow - 1]" . "</font>";
last;
}
}
$htmlCol++;

#####
# ONTAP Command: "snapmirror status" #
# - Raise Flag if snapmirror is off, or if any #
# sm relationship LAG exceeds 1 hour. #
#####
$htmlTableCellData[0][$htmlCol] = "<font color=#FFFFFF>" . "SM Status" . "</font>";

# ===== #
# If we don't get inside the next 'if' section, then #

```

```

# the data for this cell will remain as "NoData".    #
# ===== #
$_ = grep {(/^Stdout:\s+([\^:]+\s+){2}.*\d+:\d+:\d+\s+\S+.*$|^Stdout:\s+Snapmirror\s+is\s+\S+\.$/i)}
      (split(/\n/, $result));
if ($_ != 0)
{
  $htmlTableCellData[$htmlRow][$htmlCol] = "<font color=#00FF00>" . "OK: LAGs under ${lagHWM}hrs" . "</font>";
  for $_ (grep {(/^Stdout:\s+([\^:]+\s+){2}.*\d+:\d+:\d+\s+\S+.*$|^Stdout:\s+Snapmirror\s+is\s+\S+\.$/i)}
            (split(/\n/, $result)))
  {
    #####
    # Snapmirror is off.
    #####
    if ($_ =~ /^Stdout: Snapmirror is off\.$/i)
    {
      # YELLOW FLAG
      $htmlTableCellData[$htmlRow][$htmlCol] = "<font color=#FFFF00>" . "OFF" . "</font>";
      $htmlTableCellData[$htmlRow][0] = "<font color=#FFFF00>" . "$filers[$htmlRow - 1]" . "</font>";
      last;
    }

    #####
    # Snapmirror is on.
    # Source      Destination State      Lag      Status
    # filer1:VOL1 filer3:VOL1 Snapmirrored 00:11:10 Idle
    # filer3:VOL1 filer1:VOL1 Source      00:11:10 Idle
    #####
    next if ($_ =~ /^Stdout: Snapmirror is on\.$/i || $_ =~ /^Stdout: Source\s+Destination\s+.*$/i);

    my $xferStatus = (split(/\s+/, $_))[5];
    my $lagHours = (split(/\s+/, $_))[4]; $lagHours =~ s/:.*//;
    my $pattern = (split(/\s+/, $_))[3];
    if ($pattern !~ /^Snapmirrored$/i && $pattern !~ /^Source$/i)
    {
      # RED FLAG
      $htmlTableCellData[$htmlRow][$htmlCol] = "<font color=#FF0000>" . "Check Links" . "</font>";
      $htmlTableCellData[$htmlRow][0] = "<font color=#FF0000>" . "$filers[$htmlRow - 1]" . "</font>";
      last;
    }
    elsif ($xferStatus ne "Transferring" && $lagHours >= $lagHWM)
    {
      # RED FLAG
      $htmlTableCellData[$htmlRow][$htmlCol] = "<font color=#FF0000>" . "NotOK: LAG(s) exceed ${lagHWM}hrs" . "</font>";
      $htmlTableCellData[$htmlRow][0] = "<font color=#FF0000>" . "$filers[$htmlRow - 1]" . "</font>";
      last;
    }
  }
  else
  {
  }
}
$htmlCol++;

```

```

#####
# ONTAP Command: "vif status" #
# - Raise Flag if any vif is not in the "Up" state, #
# that is, does not begin with: #
# "Stdout: VIF Status Up " #
#####
$htmlTableCellData[0][$htmlCol] = "<font color=#FFFFFF>" . "VIF Status" . "</font>";

# ===== #
# If we don't get inside the next 'if' section, then #
# the data for this cell will remain as "NoData". #
# ===== #
$_ = grep {(/^Stdout:\s+VIF Status\s+(Up|Down|Broken)\s+/i |
/^Stdout:\s+No\s+configured\s+vifs\s+present\s*/i)} (split(/\n/, $result));
if ($_ != 0)
{
# ===== #
# If we don't get inside this if section, then #
# the data for this cell will remain as "NoData". #
# ===== #
$htmlTableCellData[$htmlRow][$htmlCol] = "<font color=#00FF00>" . "ONLINE" . "</font>";
for $_ (grep {(/^Stdout:\s+VIF Status\s+(Up|Down|Broken)\s+/i |
/^Stdout:\s+No\s+configured\s+vifs\s+present\s*/i)} (split(/\n/, $result)))
{
if ($_ =~ /^Stdout:\s+No\s+configured\s+vifs\s+present\s*/i)
{
$htmlTableCellData[$htmlRow][$htmlCol] = "<font color=#00FF00>" . "No vifs configured" . "</font>";
last;
}
next if ($_ =~ m/^Stdout:\s+VIF Status\s+Up\s+/i);
$htmlTableCellData[$htmlRow][$htmlCol] = "<font color=#FF0000>" . "OFFLINE" . "</font>";
$htmlTableCellData[$htmlRow][0] = "<font color=#FF0000>" . "$filers[$htmlRow - 1]" . "</font>";
}
}
$htmlCol++;

#####
# ONTAP Command: "options autosupport" #
# - Raise Flag if any of the following isn't true: #
# #
# options autosupport.enable on #
# options autosupport.support.enable on #
# options autosupport.mailhost mail.ml.com #
#####
$htmlTableCellData[0][$htmlCol] = "<font color=#FFFFFF>" . "ASUP options" . "</font>";

# ===== #
# If we don't get inside the next 'if' section, then #
# the data for this cell will remain as "NoData". #
# ===== #

```

```

$_ = grep {/^Stdout:\s+autosupport\./i} (split(/\n/, $result));
if ($mailHost eq "NULL")
{
    $htmlTableCellData[$htmlRow][$htmlCol] = "<font color=#00FF00>" . "USER IGNORED" . "</font>";
}
elseif ($_ != 0)
{
    $htmlTableCellData[$htmlRow][$htmlCol] = "<font color=#00FF00>" . "CORRECT" . "</font>";
    if ($result !~ (m/^Stdout:\s+autosupport\.enable\s+on/smi) ||
        $result !~ (m/^Stdout:\s+autosupport\.support\.enable\s+on/smi) ||
        $result !~ (m/^Stdout:\s+autosupport\.mailhost\s+${mailHost}/smi))
    {
        $htmlTableCellData[$htmlRow][$htmlCol] = "<font color=#FF0000>" . "MISCONFIGURED" . "</font>";
        $htmlTableCellData[$htmlRow][0] = "<font color=#FF0000>" . "$filers[$htmlRow - 1]" . "</font>";
    }
}
$htmlCol++;
}
#####

#####
# Generate the HTML table output. #
#####
push @htmlTable,
("<html>
<head>
<title>NetApp Storage Controller status grid</title>

<STYLE TYPE=\"text/css\">
<!--
TD {font-family: Verdana; font-size: 8pt;}
BODY {color:black; background-color:white; font-family:courier;}
--->
</STYLE>

</head>
<body>\n\n");

push @htmlTable, ("<b><u>NetApp Storage Controller status grid</u></b><br><br>\n");

push @htmlTable, ("<font size=\"1\" face=\"Verdana\">");

push @htmlTable, ("<b><u>Column definitions and table Key</u></b>:<br>\n");
push @htmlTable, ("<b>[FILER NAME]</b>: Storage Controller Name.<br>\n");
push @htmlTable, ("<b>[MODEL]</b>: Storage Controller Model Type.<br>\n");
push @htmlTable, ("<b>[vONTAP]</b>: ONTAP Version.<br>\n");
push @htmlTable, ("<b>[Shelf F/W Status]</b>: Red indicates different F/W Rev. levels exist across like shelf modules; or DUAL shelf module
(s) are potentially missing.<br>\n");
push @htmlTable, ("<b>[AGGR Status]</b>: Red indicates that at least one AGGREGATE is NOT online.<br>\n");
push @htmlTable, ("<b>[VOL Status]</b>: Red indicates that at least one VOLUME is NOT online.<br>\n");

```

```

push @htmlTable, ("[Broken Disks]: Red indicates that at least one disk is broken.<br>\n");
push @htmlTable, ("[Missing/Bypassed Disks]: Red indicates that at least one disk is missing or is bypassed.<br>\n");
push @htmlTable, ("[# Spare Status]: Number of spare disks. Red indicates there are no spare disks available.<br>\n");
push @htmlTable, ("[Disk Ownership]: Red indicates Software ownership -AND- at least one disk is UN-OWNED.<br>\n");
push @htmlTable, ("[MPHA Status]: Optimal cabling check. YELLOW = Single pathed; RED = MultiPathed w/ A & B on same off-board IO card; RED = Same ON-Board ASIC used for A & B sides (i.e. 0a/0b, 0c/0d, 0e/0f, 0g/0h).<br>\n");
push @htmlTable, ("[SM Status]: Yellow = SnapMirror is OFF; RED = A relationship is not in the SnapMirrored state; RED = A relationship exceeds the user specified LAG High-Water-Mark.<br>\n");
push @htmlTable, ("[VIF Status]: Red indicates that VIFs are in use -AND- at least one VIF is NOT in the \"Up\" state.<br>\n");
push @htmlTable, ("[ASUP options]: Red indicates that ASUP options are mis-configured to send ASUP emails.<br>\n");
push @htmlTable, ("NOTE: RED/YELLOW statuses can be caused by the condition existing on the filer -AND/OR- on its partner. Please check both filers in a cluster.<br>\n");
push @htmlTable, ("</font> <br>\n");

push @htmlTable,
  ("




```

```
print "#                                END HTML OUTPUT                                #\n";
print "#####\n\n";
```

```
#####
```

```
sub Usage
{
    print "\n#####\n";
    print "Usage: $scriptName -h |" . "\n" .
        "\t-g \"DFMgroup\"" . "\n" .
        "\t-m \"mailhostName | NULL\"" . "\n" .
        "\t-l \"smLagHWM\"" . "\n\n";

    print "\t-h Displays usage and version information. (v${progVer})\n";
    print "\t-g Specify the DFM GroupName or GroupID to operate on.\n";
    print "\t-m Specify mailhost that should be checked for in ONTAP ASUP options, or 'NULL' to ignore it.\n";
    print "\t-l Integer that specifies the largest acceptable SnapMirror LAG, in hours (1, 2, 3, etc).\n";

    print "\nImportant Note(s):\n";
    print "-----\n";
    print " - Double quotes (\") must enclose the value substituted for DFMgroup.\n\n";
    print " - When specifying a nested DFM group, then DFMgroup must be the\n";
    print "   fully qualified name for it. For example: \"GROUP1/GROUP2\".\n\n";
    print " - DFM must have privileges set to login to filers. An easy way to\n";
    print "   test this, is to run the following command for every filer:\n\n";
    print "   root# dfm run cmd -t 30 filerName version\n\n";
    print "   A correct setup will return the ONTAP version.\n";
    print "#####\n\n";
    exit 0;
}
```

```
#####
```

```
sub licenseCheck
{
    #####
    # Check for LICENSE VIOLATION. #
    #####
    my $hostId;
    if ($^O =~ m/MSWin32/i)
    {
        $hostId = (split(/\n/, `dir c:`))[1];
        $hostId =~ s/^\.*\s([\^s]+)$/$1/;
        chomp ($hostId);
    }
    elsif ($^O =~ m/linux/i)
    {
        $hostId = `hostid`;
        chomp ($hostId);
    }
    else
    {
        print "\nLicense violation for ACTIVE STORAGE MANAGEMENT ENHANCEMENT '$scriptName' software:\n";
    }
}
```

```

print " Expected key.....: [N/A]\n";
print " Calculated key....: [N/A]\n";
print " O/S Type.....: [0] - UNSUPPORTED O/S!\n\n";
exit 1;
}

if ($hostId !~ m/^\${hostId_Hardcode}$/i)
{
my $hostId_Hardcode_MD5 = md5_hex($hostId_Hardcode);
my $hostId_MD5          = md5_hex($hostId);
print "\nLicense violation for ACTIVE STORAGE MANAGEMENT ENHANCEMENT '${scriptName}' software:\n";
print " Expected key.....: [0]\n";
print " Calculated key....: [0]\n";
print " O/S Type.....: [0]\n\n";
exit 1;
}

#####
# Check for TRIAL EXPIRATION. #
#####
my $tmpVar = ${maxDate_TrialHardcode}; $tmpVar =~ s/\//g;
my ($sec,$min,$hour,$mday,$mon,$year,$yday,$isdst) = localtime(time); $mon += 1; $year += 1900;
if ((${maxDate_TrialHardcode} ne "-1") &&
    (${year}.${mon}.${mday} > ${tmpVar}))
{
print "\nLicense violation for ACTIVE STORAGE MANAGEMENT ENHANCEMENT '${scriptName}' software:\n";
print " O/S Type.....: [0]\n";
print " TRIAL EXPIRED ON.....: ${maxDate_TrialHardcode}\n\n";
exit 1;
}
#####
}

```