GreenScreen Services

Monitoring

|  |  |
| --- | --- |
| Author: | kevin.snook.eu |
| Date: | 27/07/2021 |
| Version: | 1.0 |

Document Control

Change Control

| Date | Version | Author | Comments |
| --- | --- | --- | --- |
| 21st June 2022 | 1.0 | kevin.snook | Initial Document |

Contacts

| Name | Location | Author | Email |
| --- | --- | --- | --- |
| Kevin Snook |  | kevin.snook | kevin.snook@outlook.com |

Table of Contents

[1 Introduction and Design 6](#_Toc114657288)

[1.1 Introduction 6](#_Toc114657289)

[1.2 Design 6](#_Toc114657290)

[1.3 Environment GreenScreen 6](#_Toc114657291)

[1.3 Component GreenScreen 7](#_Toc114657292)

[2 Configuration 8](#_Toc114657293)

[2.1 Scheduled Task 8](#_Toc114657294)

[2.2 IIS 11](#_Toc114657295)

[2.3 GreenScreen.HTML 12](#_Toc114657296)

[2.4 Scripts 12](#_Toc114657297)

[2.5 Credentials 12](#_Toc114657298)

[2.5 Ticker output 13](#_Toc114657299)

[3 System Reports 14](#_Toc114657300)

[3.1 Overview 14](#_Toc114657301)

[3.2 VMware 14](#_Toc114657302)

[3.2.1 Prerequisites: 14](#_Toc114657303)

[3.2.2 Monitored elements 15](#_Toc114657304)

[3.2.2.1 VCenter checks 15](#_Toc114657305)

[3.2.2.2 DNS: 15](#_Toc114657306)

[3.2.2.3 Uptime 15](#_Toc114657307)

[3.2.2.4 Alarms Check 15](#_Toc114657308)

[3.2.2.5 VM Checks 15](#_Toc114657309)

[3.2.2.6 Host services 15](#_Toc114657310)

[3.2.2.7 Network 15](#_Toc114657311)

[3.2.2.8 Hardware 15](#_Toc114657312)

[3.2.2.9 Disk space 15](#_Toc114657313)

[3.2.2.10 Datastore Checks 15](#_Toc114657314)

[3.2.3 Configuration file 15](#_Toc114657315)

[3.2.4 Output 17](#_Toc114657316)

[3.3 NetApp 18](#_Toc114657317)

[3.3.1 Prerequisites 18](#_Toc114657318)

[3.3.2 Monitored elements 18](#_Toc114657319)

[3.3.2.1 DNS 18](#_Toc114657320)

[3.3.2.2 Health Check 18](#_Toc114657321)

[3.3.2.3 Uptime 18](#_Toc114657322)

[3.3.2.4 Snapmirrors 18](#_Toc114657323)

[3.3.2.5 Snapshots 18](#_Toc114657324)

[3.3.2.6 Snapvaults 18](#_Toc114657325)

[3.3.2.7 Controller Alarms 18](#_Toc114657326)

[3.3.2.8 Networks 18](#_Toc114657327)

[3.3.2.9 Hardware 18](#_Toc114657328)

[3.3.2.10 Volumes 19](#_Toc114657329)

[3.3.2.11 SVMs 19](#_Toc114657330)

[3.3.2.12 Shares 19](#_Toc114657331)

[3.3.2.13 Aggregates 19](#_Toc114657332)

[3.3.3 Configuration file 19](#_Toc114657333)

[3.3.4 Output 20](#_Toc114657334)

[3.4 Simplivity 21](#_Toc114657335)

[3.4.1 Prerequisites 21](#_Toc114657336)

[3.4.2 Monitored elements 21](#_Toc114657337)

[3.4.2.1 Uptime 21](#_Toc114657338)

[3.4.2.2 Backup Checks 21](#_Toc114657339)

[3.4.2.3 VM Match Check 21](#_Toc114657340)

[3.4.2.4 HA VM Check 21](#_Toc114657341)

[3.4.2.5 Alarms Check 21](#_Toc114657342)

[3.4.2.6 Host services 21](#_Toc114657343)

[3.4.2.7 Network 21](#_Toc114657344)

[3.4.2.8 ILO Health Check 21](#_Toc114657345)

[3.4.2.9 Disk space 21](#_Toc114657346)

[3.4.3 Configuration file 21](#_Toc114657347)

[3.5.4 Output 22](#_Toc114657348)

[3.5 PURE Storage 24](#_Toc114657349)

[3.5.1 Prerequisites 24](#_Toc114657350)

[3.5.2 Monitored elements 24](#_Toc114657351)

[3.5.2.1 DNS 24](#_Toc114657352)

[3.5.2.2 System Check 24](#_Toc114657353)

[3.5.2.3 Volumes 24](#_Toc114657354)

[3.5.2.4 Controller Alarms 24](#_Toc114657355)

[3.5.2.5 Networks 24](#_Toc114657356)

[3.5.2.6 Hosts 24](#_Toc114657357)

[3.5.2.7 Ports 24](#_Toc114657358)

[3.5.2.8 Hardware 24](#_Toc114657359)

[3.5.3 Configuration file 24](#_Toc114657360)

[3.5.4 Output 25](#_Toc114657361)

[3.6 CommVault 26](#_Toc114657362)

[3.6.1 Prerequisites 26](#_Toc114657363)

[3.6.2 Monitored elements 26](#_Toc114657364)

[3.6.2.1 Missed Backups 26](#_Toc114657365)

[3.6.3 Configuration file 26](#_Toc114657366)

[3.6.4 Output 26](#_Toc114657367)

[3.7 NetBackup 27](#_Toc114657368)

[3.7.1 Prerequisites 27](#_Toc114657369)

[3.7.2 Monitored elements 27](#_Toc114657370)

[3.7.2.1 Missed Backups 27](#_Toc114657371)

[3.7.3 Configuration file 27](#_Toc114657372)

[3.7.4 Output 27](#_Toc114657373)

[3.8 Windows 28](#_Toc114657374)

[3.8.1 Prerequisites 28](#_Toc114657375)

[3.8.2 Monitored elements 28](#_Toc114657376)

[3.8.2.1 DNS 28](#_Toc114657377)

[3.8.2.2 Uptime 28](#_Toc114657378)

[3.8.2.3 Services 28](#_Toc114657379)

[3.8.2.4 Events 28](#_Toc114657380)

[3.8.2.5 Networks 28](#_Toc114657381)

[3.8.2.6 Hardware 28](#_Toc114657382)

[3.8.2.7 Performance 28](#_Toc114657383)

[3.8.2.8 Disk Space 28](#_Toc114657384)

[3.8.2.9 Certificates 28](#_Toc114657385)

[3.8.3 Configuration file 28](#_Toc114657386)

[3.8.4 Output 29](#_Toc114657387)

[3.9 Cisco UCS 30](#_Toc114657388)

[3.9.1 Prerequisites 30](#_Toc114657389)

[3.9.2 Monitored elements 30](#_Toc114657390)

[3.9.2.1 System 30](#_Toc114657391)

[3.9.2.2 Uptime 30](#_Toc114657392)

[3.9.2.3 Faults 30](#_Toc114657393)

[3.9.2.4 Networks 30](#_Toc114657394)

[3.9.2.5 Hardware 30](#_Toc114657395)

[3.9.3 Configuration file 30](#_Toc114657396)

[3.9.4 Output 31](#_Toc114657397)

[4 Daily Report 32](#_Toc114657398)

[4.1 Prerequisites 32](#_Toc114657399)

[4.2 Monitored elements 32](#_Toc114657400)

[4.2.1 Reports output by system health checks 32](#_Toc114657401)

[4.3 Configuration file 32](#_Toc114657402)

[4.4 Output 33](#_Toc114657403)

[5 Interactive Mode 34](#_Toc114657404)

[Appendix 1 – greenscreen.html 35](#_Toc114657405)

# 1 Introduction and Design

## Introduction

The GreenScreen scripts are designed to fill a hole in the monitoring sphere. Namely, there is no simple way to have a simple over-arching monitoring screen that tells me “Is there something wrong with my environment?”. VROPs and SCOM are too complex to configure and create a snapshot roll-up screen of the kind that we want.

## Design

### 1.3 Environment GreenScreen



This screen shows all configured components in your environment. It simply shows whether there are any problems with any of the components in your environment. Clicking on the title of the component will fire up another web page showing the component details (see below).

To get this to work you will need to alter the GreenSceen.html file attached and run this in your chosen web browser. The full version of the greenscreen.html is attached in the appendix. To get the greenscreen.html to “Point” to your component web pages you will need to alter the following lines:

<head><title>GreenScreen</title></head>

* Alter the page title to something appropriate for you company if you desire

<iframe id="nhc0vmwarehealth" scrolling="no" style="width:300px; height:60px;border: 4px ridge lightblue;" src="http://nhc0-eud-vman01/monitor/vmwarehealth.html"></iframe>

* Change the id field to one that is unique (we will use this later)
* Change the src to the web page that contains the contents
* Add one of these lines for each component web page

var txtURL = `http://nhc0-eud-vman01/monitor/ticker.txt?${nowtime}`;

* This should be changed to the location of the file you will use for the ticker functionality

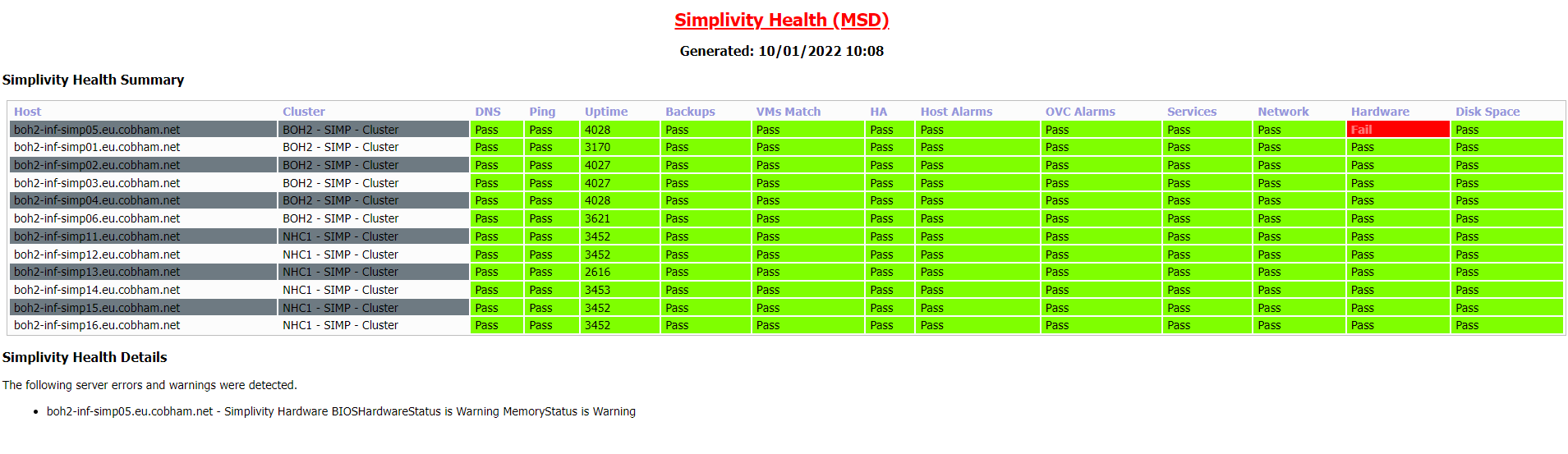
document.getElementById('nhc0vmwarehealth').contentWindow.location.reload(true);

* For each iframe id change this line to reload

### Component GreenScreen

The scripts use a RAG (Red/Amber/Green) screen to alert users to errors and potential problems in their environment. It’s best to liken this to the engine management light in your car. Each component in the monitored system is processed and a roll-up status presented in a screen. For instance, a VM may not be HA-compliant or a volume on an array maybe over 90%. For these errors a Red flashing cell is displayed. Further details of the actual error can be found in the details portion of the screen.

Below is an example screen to demonstrate the idea:



The screen is divided into three main sections:

* Heading – giving details of what report this is (the heading will be colour coded depending on whether the health check has failed or passed)and the time the report was generated
* Health Summary – this shows the RAG screen and will immediately alert users to potential problems
* Health Details – further details about errors/problems ordered to synchronise with the Health Summary

Clearly, for each system different configuration items are tested/monitored. Details of each report are found later in this document.

A few tips:

The screens can be, indeed should be, used on a central screen where all of the team can see. The screens are designed to provide near-time monitoring and are best viewed where they can be seen all of the time.

# Configuration

## Scheduled Task

Each system can be processed on a separate schedule and that is dictated by the details of the task scheduler job (which you will need to set up).

For convenience, the jobs are each put into a batch file to allow you to pass parameters to the script such as whether you want email output of errors or whether you want a log file produced. These parameters are dealt with later in the following section.

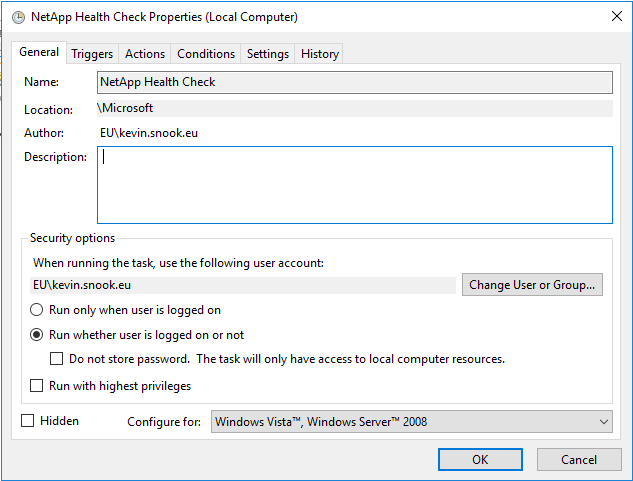
The most popular configuration is to just run the script and produce emails if errors are found. Here we create a batch file thus:

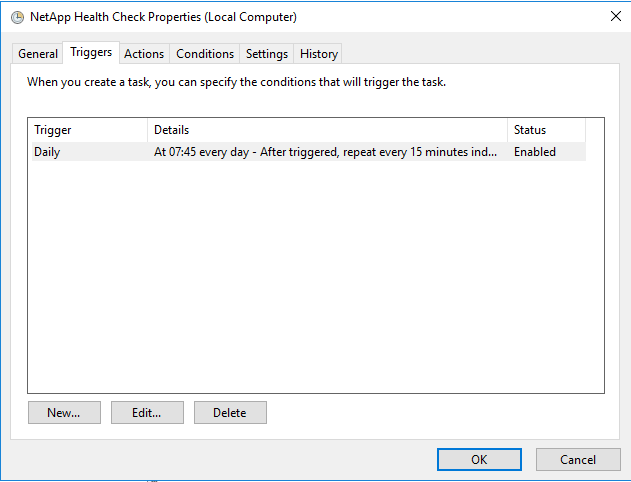
C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe -command "C:\Source\Scripts\netapp\test-netapphealth.ps1 –Configfile *location of config file*"

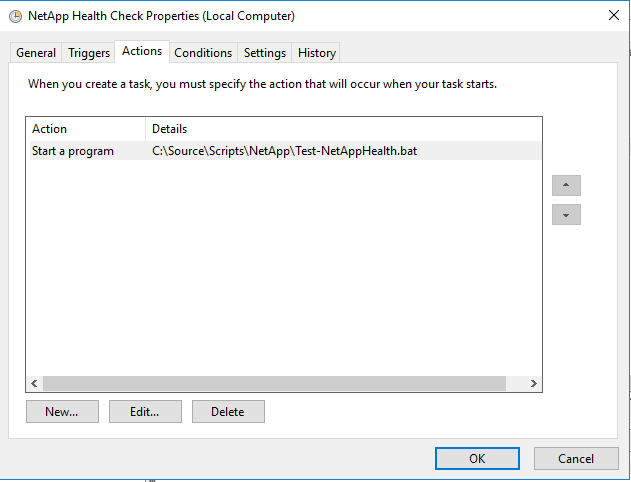
The other popular option is to create a log as well. Here we create a batch file thus:

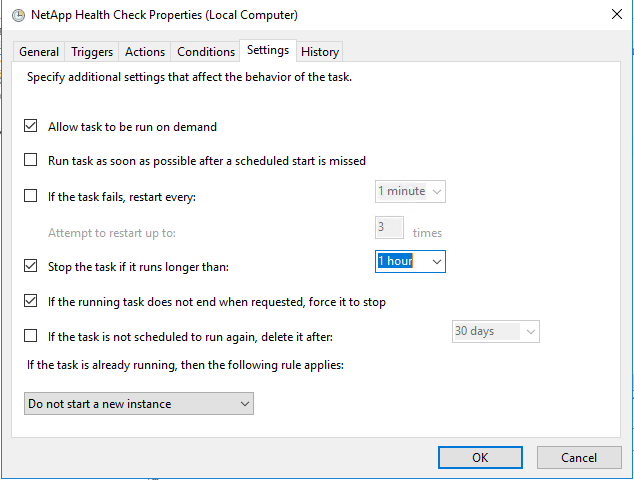
C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe -command "C:\Source\Scripts\netapp\test-netapphealth.ps1 –Configfile *location of config file*"

Here are the steps to creating a scheduled job for production of the screens (we use a NetApp screen for illustration):









## IIS

The HTML produced for the alert screens can be used with any Web server. We use IIS as it is freely available on the platforms we use.

This is not intended to be an instruction manual on IIS but we note a few things that it is recommended you enable to get these reports to present properly:

* Static content must be enabled
* Create a folder under the Default Web Site
* Point that folder to a suitable Virtual Directory and setup permissions so anyone wanting to access the monitoring pages can do so.
* As this is static RO content, http access is sufficient although this may need to be reviewed if you are sharing this server with other more sensitive content
* By default, all of the scripts output to a folder called C:\inetpub\wwwroot\monitor\ so it makes sense to create your Virtual Folder using that information.
* Ensure your content is not cached via the Output Caching of the virtual directory. Your web.config should look like this:

<?xml version="1.0" encoding="UTF-8"?>

<configuration>

<system.webServer>

<staticContent>

<clientCache cacheControlMode="DisableCache" />

</staticContent>

</system.webServer>

</configuration>

## GreenScreen.HTML

To get this to work you will need to alter the GreenSceen.html file attached and run this in your chosen web browser. The full version of the greenscreen.html is attached in the appendix. The greenscreen.html webpage runs locally and will need to be configured with the names of the webpages to display. To get the greenscreen.html to “Point” to your component web pages you will need to alter the following lines:

<head><title>GreenScreen</title></head>

* Alter the page title to something appropriate for you company if you desire

<iframe id="nhc0vmwarehealth" scrolling="no" style="width:300px; height:60px;border: 4px ridge lightblue;" src="http://nhc0-eud-vman01/monitor/vmwarehealth.html"></iframe>

* Change the id field to one that is unique (we will use this later)
* Change the src to the web page that contains the contents
* Add one of these lines for each component web page

var txtURL = `http://nhc0-eud-vman01/monitor/ticker.txt?${nowtime}`;

* This should be changed to the location of the file you will use for the ticker functionality

document.getElementById('nhc0vmwarehealth').contentWindow.location.reload(true);

* For each iframe id change this line to reload

## Scripts

Depending on which scripts you need to run, you will need to configure their individual parameters. Each script has a config file which is pointed to by using the command line parameter –ConfigFile e.g. C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe -command "C:\Source\Scripts\simplivity\test-simplivityhealth.ps1 -ConfigFile C:\Source\Scripts\simplivity\test-simplivityhealth-cfg.ps1"

Mostly the scripts will be run from a scheduled task (see above) – so you will need to create a batch file with the correct script and config file.

## 2.5 Credentials

Clearly, reports of this nature run interactively with calls for usernames and passwords. The usernames and passwords are passed into the script using encrypted xml files. These are created before running the scripts. You must create a separate file for each set of credentials required then point the script to them in the config file. Important: the credentials are encrypted on a per-user basis – what this means is that you should create credentials when logged in as the user that will run the scripts. The XML files will not work with any other user.

The Powershell command used to create the credentials is:

Get-Credential | Export-CliXml -Path c:\source\scripts\monitoring-user\_cred.xml

An alternative path can be used and this then should be reflected in the Test-*ReportName*-cfg.ps1 file.

The sections below show the credential requirements for each report.

## Ticker output

A ticker.txt file can be created in the web directory. This file will produce the ticker output at the bottom of the GreenScreen web page. It is simply a text file – each line will be shown separately in the ticker output. This can be useful for on-going issues that need to be noted. The ticker is configured to stop when the mouse hovers over it. This makes reading the notes a little easier:



# System Reports

## Overview

As you have seen above in the configuration section, each script needs a set of parameters to be set up in their configuration file. At the very least, each config file will need:

* Targets to be monitored
* Credentials to access these targets
* Email settings
* Report file location and web details
* Report title
* Log file location

## 3.2 VMware

### 3.2.1 Prerequisites:

* Local user on each NetApp system monitored called “monitoring-user”
* Password for monitoring user saved in xml file using:

Get-Credential | Export-CliXml -Path c:\source\scripts\monitoring-user\_cred.xml (or alternative path but remember to update the Test-VMwareHealth-cfg.ps1 script)

* Config file set up with correct parameters.

$Credential - This is the SSO credential for your VMware SSO domain

$ILOCredential - This is the credential for the ILO card

$VCCredential - This is the user with VC logons rights (it needs to be able to stop/start services so an admin account is usually best)

$ESXiMonitorCredential - This is a local user setup on the local ESXi host. Use the following to create a monitoring user on each host:

$esxSrv = Connect-VIServer -Server $vmhost.name -User root -Password $pswd

$user = New-VMHostAccount -Server $esxSrv -Id monitoring-user -Password VMware1! -UserAccount -GrantShellAccess

$rootFolder = Get-Folder -Name root -Server $esxSrv

$role = Get-VIRole -Name Admin -Server $esxSrv #This can be a different role

New-VIPermission -Entity $rootFolder -Principal $user -Role $role

$VCRootCredential - This is the root user on the VCenter

* All credentials should be saved in xml files for use in the script:

Get-Credential | Export-CliXml -Path C:\Source\Scripts\Credentials\admin@sso.xml (or alternative path but remember to update the Test-VMwareHealth-cfg.ps1 script)

Get-Credential | Export-CliXml -Path C:\Source\Scripts\Credentials\hpeloginonly.xml (or alternative path but remember to update the Test-VmwareHealth-cfg.ps1 script)

Get-Credential | Export-CliXml -Path C:\Source\Scripts\Credentials\ks\_cred.xml (or alternative path but remember to update the Test-VmwareHealth-cfg.ps1 script)

Get-Credential | Export-CliXml -Path C:\Source\Scripts\Credentials\monitoring-user\_cred.xml (or alternative path but remember to update the Test-VmwareHealth-cfg.ps1 script)

Get-Credential | Export-CliXml -Path C:\Source\Scripts\Credentials\vc\_root\_cred.xml (or alternative path but remember to update the Test-VmwareHealth-cfg.ps1 script)

Get-Credential | Export-CliXml -Path C:\Source\Scripts\Credentials\idrac\_root.xml (or alternative path but remember to update the Test-VmwareHealth-cfg.ps1 script)

* Putty installed - the install directory for Putty should be entered in the Test-VMwareHealth-cfg.ps1 (see below) - default is "C:\PROGRA~1\PUTTY\plink.exe"
* Config file set up with correct parameters (see below)

### 3.2.2 Monitored elements

#### 3.2.2.1 VCenter checks

Checks certificate and reports if certificate will expire in less than $CertificateTimeToAlert days (configured in Test-VMwareHealth-cfg.ps1 file - see below)

Checks we can acces the REST API URL

Reports any services that are set to Automatic but not running

Reports on any health statuses that are not green

Reports disk partitions above $PartitionPercentFull (default=90%)

#### 3.2.2.2 DNS:

Checks each hosts is resolvable in DNS

Checks each host is pingable

#### 3.2.2.3 Uptime

Checks uptime for each host

#### 3.2.2.4 Alarms Check

Finds any Host alarms

Finds any VM alarms on each host

#### 3.2.2.5 VM Checks

Reports on powered off or faulty VMs (unless $CheckPowerOffVMs is set to $false)

#### 3.2.2.6 Host services

Checks all services are up

#### 3.2.2.7 Network

Checks that, if the host is in a cluster, vMotion is enabled

Checks that, if the host is part of a cluster, then HA, DRS, HA Admission control are enabled

Find any inactive/standby/dead paths on HBA paths

Checks if portgroups have been exhausted of ports (less than 10)

#### 3.2.2.8 Hardware

Reports any errors being reported through sensors etc in VMware

#### 3.2.2.9 Disk space

Checks disk space

#### 3.2.2.10 Datastore Checks

Checks for space on datastores and reports any below ($datastorePercentFree

### 3.2.3 Configuration file

Test-VMwareHealth-cfg.ps1 script:

#...................................

# Variables

#...................................

#Max days since last full backup

$MaxDaysSinceBackup = 1

#Max hours to go back and alert in logs

$MaxHoursToScanLog = 2

#Warn if uptime is below this number of hours

$MinimumUptime = 2

#Vcenter server to monitor

$VCServer = "ORY1-EUD-VCM001.eu.cobham.net"

#Number of days back before we alert about a cert being about to expire

$CertificateTimeToAlert = 20

#Set to true to alert on powered off VMs

$CheckPowerOffVMs = $false

#Minimum Percentage free on datastores before raising alert

$datastorePercentFree = 1

#Maximum Percentage full on Host/vCenter partitions before raising alert

$PartitionPercentFull = 90

#Comma separated array of Hosts to ignore (if host has permanenet known condition or is offline) (Please ensure this is exactly the same case and format as the hostname(s))

$IgnoreHosts = @("cph2-inf-simp01.eu.cobham.net","cph2-inf-simp02.eu.cobham.net")

#Comma separated array of VMs to ignore (if VM has permanent known condition or is offline) (Please ensure this is exactly the same case and format as the VM name(s))

$IgnoreVMs = @("ARTEMIS\_restore\_restore","Compta-sage500")

#Comma separated array of VM alarms to ignore

$IgnoreVMAlarms = @("Virtual machine memory usage","Virtual machine CPU usage")

#Comma separated array of Host alarms to ignore

$IgnoreHostAlarms = @("Virtual machine memory usage","Host memory usage","Host CPU usage")

#Comma separated array of hosts with hardware errors to ignore

$IgnoreHardwareErrors = @("cph1-inf-esx451.eu.cobham.net","10.192.64.17")

#Comma separated array of Hosts to ignore - (this will not logon to these hosts to check disk space)

$IgnoreHostsDiskSpace = @("tnf1-inf-simp02.eu.cobham.net")

#Run IPMI Checks

$IPMIHardwareErrors = $true

#Number of items not replicated before alert raised

$MaxReplicationItemsLagging = 100

#Alert on evaluation licences in use

$IgnoreEvaluationLicences = $true

#Path to PuttyLink executable

$PuttyLinkPath = "C:\PROGRA~1\PUTTY\plink.exe"

#Report settings

#Determines whether the script logs it's activities using logfile described below

$Log = $true

#Location of log file for script

$logfile = "C:\Source\Scripts\vmware\script\_health\VMware\_health.log"

#Location of transcript file for script (shows command line output for script

$TranscriptFile = "C:\Source\Scripts\vmware\script\_health\VMware\_health\_transcript.log"

#Reportsubject = the heading for the html report

$reportsubject = "VMware Health (CACm)"

#Enter the URL of the report for use from the Web

$ReportURL = "http://ORY1-EUD-VMAN01/Monitor/vmwareHealth.html"

#ReportMode - Set to $true to generate a HTML report. Uses the name of the Report file (see below)

$ReportMode=$true

#ReportFile - name of file to output html report to

$ReportFile="C:\inetpub\wwwroot\monitor\vmwarehealth.html"

#Determines whether we send the HTML report via email using the SMTP configuration within the config file.

$SendEmail=$true

#Only sends the email report if at least one error or warning was detected.

$AlertsOnly=$true

#...................................

# Email Settings

#...................................

#Send email to this address

$recipients = "ITS.Datacentre.Services@cobham.com","CAC.IT.Notifications@cobham.com"

#$recipients = "kevin.snook@cobham.com"

#Send email from this address

$fromaddress = "CACm-VMware-Alerts@cobham.com"

#Send email using this relay host

$smtpserver = "smtp.eu.cobham.net"

#...................................

#Credentials

#....................................

#VMware SSO Credential

$Credential = Import-CliXml -Path C:\Source\Scripts\Credentials\Cred\_administrator@EU.CACm.vSphere.xml

#HPe ILO Credential

$ILOCredential = Import-CliXml -Path c:\source\scripts\Credentials\hpeilo.xml

#vCenter Credential

$VCCredential = Import-CliXml -Path C:\Source\Scripts\Credentials\Cred\_administrator@EU.CACm.vSphere.xml

#Monitoring credential on ESXi Hosts

$ESXiMonitorCredential = Import-CliXml -Path c:\source\scripts\Credentials\monitoring-user\_cred.xml

#Root credential on vCenter

$VCRootCredential = Import-CliXml -Path c:\source\scripts\Credentials\vc\_root\_cred.xml

$iDRACCredential = Import-CliXml -Path c:\source\scripts\Credentials\idrac\_root.xml

### 3.2.4 Output

As well as the html report file, the script outputs other files to monitor behaviour:

A log file (selected in config file) - this monitors all of the functions/commands performed by the script.

A transcript file - this is effectively the output of the command and may be used to troubleshoot - it is always on.

## 3.3 NetApp

### 3.3.1 Prerequisites

* Local user on each NetApp system monitored called monitoring-user
* Password for monitoring user saved in xml file using:

Get-Credential | Export-CliXml -Path c:\source\scripts\Credentials\monitoring-user\_cred.xml (or alternative path but remember to update the Test-NetAppHealth-cfg.ps1 script)

* Config file set up with correct parameters.

### 3.3.2 Monitored elements

#### 3.3.2.1 DNS

Check controller address is resolvable in DNS

If IP is returned from DNS, perform PING on it

#### 3.3.2.2 Health Check

Looks at IsNodeHealthy attribute and reports accordingly

#### 3.3.2.3 Uptime

Records uptime and warns if less than 24 hours

#### 3.3.2.4 Snapmirrors

Checks a volume has some snapmirror snapshots pulled according to it's schedules

For each volume checks the last scheduled snapshot and that the actual last snapshot held matches

For each volume checks each schedule has retained the correct amount of snapshots

#### 3.3.2.5 Snapshots

Checks snapshots created according to it's schedules

For all volumes, checks the last scheduled snapshot and that the actual last snapshot held matches

For all volumes, checks each schedule has retained the correct amount of snapshots

#### 3.3.2.6 Snapvaults

Checks snapvault snapshots pulled according to it's schedules

For policy types vault and mirror\_vault, checks the last scheduled snapshot and that the actual last snapshot held matches

For policy types vault and mirror\_vault, checks each schedule has retained the correct amount of snapshots

#### 3.3.2.7 Controller Alarms

Reads messages and returns any alarms that meet the $AlarmSeverity or greater and are within the $MaxHoursToScanLog threshold (Default=24 hours)

#### 3.3.2.8 Networks

Finds any interfaces that are not currently homed on the correct node

Pings all data and nodemgmt interfaces

Finds peer SVM or interfaces and reports if they are not pingable or operational

#### 3.3.2.9 Hardware

Reports on Service Processor problems

Pings Service Processor to ensure it operational

Reports on Shelf problems

Reports on disks that have failed

#### 3.3.2.10 Volumes

Reports on any volumes over thresholds set in scripts as $VolumeFullPercentageWarning (default=85%) and $VolumeFullPercentageError (default=95%)

Reports on any SnapshotReserve over $VolumeSnapReserveFullPercentageError threshold set in script (default=95%)

Checks autodelete parameters

#### 3.3.2.11 SVMs

Checks SVMs are operational

Checks domain controller settings and access for each SVM

#### 3.3.2.12 Shares

Checks data shares are accessible from the network

#### 3.3.2.13 Aggregates

Reports on any volumes over thresholds set in scripts as $AggregateFullPercentageWarning (default=85%) and $AggregateFullPercentageError (default=95%)

### 3.3.3 Configuration file

**Test-NetAppHealth-cfg.ps1 script:**

#...................................

# Variables

#...................................

#Maxima and minima

$MaxMinutesSinceSnapshot = 60 #Max minutes since last snapshot

$MaxMinutesSnapMirrorLag = 60 #Max minutes lag for snapmirrors

$MaxHoursToScanLog = 2 #Max hours to go back and alert in logs

$VolumeFullPercentageError = 95 #Percentage full before Error

$VolumeFullPercentageWarning = 85 #Percentage full before Warning

$VolumeSnapReserveFullPercentageError = 95 #Percentage full before Error on snap reserve

$AggregateFullPercentageError = 95 #Percentage full before Error

$AggregateFullPercentageWarning = 85 #Percentage full before Warning

$NetAppControllers = "10.172.2.185","10.136.18.60"

#Report settings

#Determines whether the script logs it's activities using logfile described below

$Log = $true

#Location of log file for script

$logfile = "C:\Source\Scripts\netapp\netapp\_health.log"

#Location of transcript file for script (shows command line output for script

$TranscriptFile = "C:\Source\Scripts\netapp\netapp\_health\_transcript.log"

#Reportsubject = the heading for the html report

$reportsubject = "NetApp Health (MSD)"

#Enter the URL of the report for use from the Web

$ReportURL = "http://BOH2-EUD-VMAN01\Monitor\netapphealth.html"

#ReportMode - Set to $true to generate a HTML report. Uses the name of the Report file (see below)

$ReportMode=$true

#ReportFile - name of file to output html report to

$ReportFile="C:\inetpub\wwwroot\monitor\netapphealth.html"

#Determines whether we send the HTML report via email using the SMTP configuration within the config file.

$SendEmail=$true

#Only sends the email report if at least one error or warning was detected.

$AlertsOnly=$true

$ErrorsURL = "http://BOH2-EUD-VMAN01\Monitor\netappreporterrors.html"

$ErrorsFile="C:\inetpub\wwwroot\monitor\netappreporterrors.html"

#...................................

# Email Settings

#...................................

#Send email to this address

$recipients = "ITS.Datacentre.Services@cobham.com"

#$recipients = "kevin.snook@cobham.com"

#Send email from this address

$fromaddress = "CMS-NetApp-Alerts@cobham.com"

#Send email using this relay host

$smtpserver = "smtp.eu.cobham.net"

#...................................

# Credentials

#...................................

#Login to monitoring user (RO user "monitoring-user" setup on NetApp clusters)

$NetAppCredential = Import-CliXml -Path c:\source\scripts\Credentials\netapp-monitoring-user.xml

### 3.3.4 Output

As well as the html report file, the script outputs other files to monitor behaviour:

A log file - this monitors all of the functions/commands performed by the script.

A transcript file - this is effectively the output of the command and may be used to troubleshoot - it is on by default.

## 3.4 Simplivity

### 3.4.1 Prerequisites

$Credential - This is the SSO credential for your VMware SSO domain

$ILOCredential - This is the credential for the ILO card on the Simplivity Omnicube

$VCCredential - This is the user with VC logons rights (it needs to be able to stop/start services so an admin account is usually best)

All credentials should be saved in xml files for use in the script:

Get-Credential | Export-CliXml -Path c:\source\scripts\credentials\admin@sso.xml (or alternative path but remember to update the Test-SimplivityHealth-cfg.ps1 script)

Get-Credential | Export-CliXml -Path c:\source\scripts\credentials\\hpeloginonly.xml (or alternative path but remember to update the Test-SimplivityHealth-cfg.ps1 script)

Get-Credential | Export-CliXml -Path c:\source\scripts\credentials\ks\_cred.xml (or alternative path but remember to update the Test-SimplivityHealth-cfg.ps1 script)

Config file set up with correct parameters.

### 3.4.2 Monitored elements

#### 3.4.2.1 Uptime

Checks uptime for each host

#### 3.4.2.2 Backup Checks

Reports failed backups that are within threshold of $MaxDaysSinceBackup

#### 3.4.2.3 VM Match Check

Checks that VMware matches with the Simplivity list of VMs on this host

#### 3.4.2.4 HA VM Check

Reports on any VMs that are not storage HA compliant

#### 3.4.2.5 Alarms Check

Finds any OVC alarms

Finds Host alarms

#### 3.4.2.6 Host services

Checks all services are up

#### 3.4.2.7 Network

Check OCV controller address is resolvable in DNS

If IP is returned from DNS, perform PING on it

#### 3.4.2.8 ILO Health Check

Checks for access to ILO

Checks fan status

Checks temperature status

Checks power supply status

Checks the ILO Health SUmmary for errors reported

Checks the IML Log for errors reported

Checks ILO event log for erros reported

#### 3.4.2.9 Disk space

Checks disk space

### 3.4.3 Configuration file

**Test-SimplivityHealth-cfg.ps1 script:**

#...................................

# Variables

#...................................

$MaxDaysSinceBackup = 1 #Max days since last full backup

$MaxHoursToScanLog = 24 #Max hours to go back and alert in logs

#Warn if uptime is below this number of hours

$MinimumUptime = 2

$VCServer = "BOH2-EUD-VCM001.eu.cobham.net"

#Comma separated array of Hosts to ignore (if host has permanenet known condition or is offline) (Please ensure this is eactly the same case and format as the hostname(s))

$IgnoreHosts = @("poc1-inf-simp01.eu.cobham.net","poc1-inf-simp02.eu.cobham.net")

#Comma separated array of hosts with hardware errors to ignore

$IgnoreHardwareErrors = @("boh2-inf-simp05.eu.cobham.net")

#Report settings

#Determines whether the script logs it's activities using logfile described below

$Log = $true

#Location of log file for script

$logfile = "C:\Source\Scripts\simplivity\Simplivity\_health.log"

#Location of transcript file for script (shows command line output for script

$TranscriptFile = "C:\Source\Scripts\simplivity\Simplivity\_health\_transcript.log"

#Reportsubject = the heading for the html report

$reportsubject = "Simplivity Health (MSD)"

#Enter the URL of the report for use from the Web

$ReportURL = "http://BOH2-EUD-VMAN01\Monitor\simplivityhealth.html"

#ReportMode - Set to $true to generate a HTML report. Uses the name of the Report file (see below)

$ReportMode=$true

#ReportFile - name of file to output html report to

$ReportFile="C:\inetpub\wwwroot\monitor\simplivityhealth.html"

#Determines whether we send the HTML report via email using the SMTP configuration within the config file.

$SendEmail=$true

#Only sends the email report if at least one error or warning was detected.

$AlertsOnly=$true

#...................................

# Email Settings

#...................................

#Send email to this address

#$recipients = "kevin.snook@cobham.com"

$recipients = "ITS.Datacentre.Services@cobham.com"

#Send email from this address

$fromaddress = "CMS-SIMP-Alerts@cobham.com"

#Send email using this relay host

$smtpserver = "smtp.eu.cobham.net"

#...................................

#Credentials

#....................................

#...................................

# Credentials

#...................................

#SSO Login for vCenter

$Credential = Import-CliXml -Path c:\source\scripts\Credentials\admin@sso\_boh2.xml

$ILOCredential = Import-CliXml -Path c:\source\scripts\Credentials\hpeilo.xml

$VCCredential = Import-CliXml -Path c:\source\scripts\Credentials\admin@sso\_boh2.xml

### 3.5.4 Output

As well as the html report file, the script outputs other files to monitor behaviour:

A log file - this monitors all of the functions/commands performed by the script.

A transcript file - this is effectively the output of the command and may be used to troubleshoot - it is on by default.

## 3.5 PURE Storage

### 3.5.1 Prerequisites

Local user on each PURE system monitored. This needs only RO permissions.

Password for monitoring user saved in xml file using:

Get-Credential | Export-CliXml -Path c:\source\scripts\credentials\pureuser.xml (or alternative path but remember to update the Test-PUREHealth-cfg.ps1 script)

Config file set up with correct parameters.

### 3.5.2 Monitored elements

#### 3.5.2.1 DNS

Check controller address is resolvable in DNS

If IP is returned from DNS, perform PING on it

#### 3.5.2.2 System Check

Checks if NTP servers are setup.

Checks if Remote Assist is on.

Checks if phone home is on.

Checks if array alerts are enabled

Checks if SMTP relay is set

Checks if email is setup

Checks if SNMP is setup

Checks if Syslog servers are setup

Checks if DNS is setup

Checks if AD integration is setup

#### 3.5.2.3 Volumes

Reports on any volumes over thresholds set in scripts as $VolumeFullPercentageWarning (default=85%) and $VolumeFullPercentageError (default=95%)

Reports on any snapshot issues

Reports on any Protection Group issues

#### 3.5.2.4 Controller Alarms

Reads messages and returns any alarms that are within the $MaxDaysToScanLog threshold (Default=1)

#### 3.5.2.5 Networks

Pings all data and array mgmt interfaces

#### 3.5.2.6 Hosts

Checks for hosts that do not have presentations made

#### 3.5.2.7 Ports

Nothing as yet (TBA)

#### 3.5.2.8 Hardware

Reports on Controller problems

Reports on Bad disks

### 3.5.3 Configuration file

Test-PUREHealth-cfg.ps1 script:

#...................................

# Variables

#...................................

#Maxima and minima

$MaxMinutesSinceSnapshot = 60 #Max minutes since last snapshot

$MaxHoursToScanLog = 4 #Max hours to go back and alert in logs

$VolumeFullPercentageError = 95 #Percentage full before Error

$VolumeFullPercentageWarning = 85 #Percentage full before Warning

#...................................

# PURE controllers

$PUREControllers = "10.172.2.160"

#...................................

#Report settings

#Determines whether the script logs it's activities using logfile described below

$Log = $true

#Location of log file for script

$logfile = "C:\Source\Scripts\PURE\PURE\_health.log"

#Location of transcript file for script (shows command line output for script

$TranscriptFile = "C:\Source\Scripts\PURE\PURE\_health\_transcript.log"

#Reportsubject = the heading for the html report

$reportsubject = "PURE Health (MSD)"

#Enter the URL of the report for use from the Web

$ReportURL = "http://BOH2-EUD-VMAN01\Monitor\PUREhealth.html" #Enter the name of the server where ae are saving the errors (probably the server where this script is running)

#ReportMode - Set to $true to generate a HTML report. Uses the name of the Report file (see below)

$ReportMode=$true

#ReportFile - name of file to output html report to

$ReportFile="C:\inetpub\wwwroot\monitor\PUREhealth.html"

#Determines whether we send the HTML report via email using the SMTP configuration within the config file.

$SendEmail=$true

#Only sends the email report if at least one error or warning was detected.

$AlertsOnly=$true

$ErrorsURL = "http://BOH2-EUD-VMAN01\Monitor\PUREreporterrors.html"

$ErrorsFile="C:\inetpub\wwwroot\monitor\PUREreporterrors.html"

#...................................

# Email Settings

#...................................

#Send email to this address

$recipients = "ITS.Datacentre.Services@cobham.com"

#$recipients = "kevin.snook@cobham.com"

#Send email from this address

$fromaddress = "CMS-PURE-Alerts@cobham.com"

#Send email using this relay host

$smtpserver = "smtp.eu.cobham.net"

#...................................

#Credentials

#....................................

#Login to monitoring user (RO user setup on PURE clusters)

$PURECredential = Import-CliXml -Path c:\source\scripts\Credentials\pureuser.xml

### 3.5.4 Output

As well as the html report file, the script outputs other files to monitor behaviour:

A log file - this monitors all of the functions/commands performed by the script.

A transcript file - this is effectively the output of the command and may be used to troubleshoot - it is on by default.

## 3.6 CommVault

### 3.6.1 Prerequisites

User credentials that have read access to the VM level in vCenter.

$VCCredential = Import-CliXml -Path c:\source\scripts\credentials\ks\_cred.xml

### 3.6.2 Monitored elements

#### 3.6.2.1 Missed Backups

Any VMs with their last backup time greater than 24 hours ago are reported on. Any VMs with the BackupAllowed parameter set to DOMAIN CONTROLLER|CDRIVE|FALSE are ignored.

### 3.6.3 Configuration file

For this script, there is NO configuration file. As the script is so small, configuration is done inside the script in the following lines:

$VCCredential = Import-CliXml -Path c:\source\scripts\Credentials\SVC-VMVRRW-001.xml

$servers = "boh2-eud-vcm001","boh2-eud-vcm002"

$sendEmail = $true

$ReportFile="C:\inetpub\wwwroot\monitor\CommVaulthealth.html"

$ReportURL = "http://BOH2-EUD-VMAN01/Monitor/CommVaulthealth.html"

$reportsubject = "CommVault Health (MSD)"

### 3.6.4 Output

As well as the html report file, the script outputs other files to monitor behaviour:

A log file - this monitors all of the functions/commands performed by the script.

A transcript file - this is effectively the output of the command and may be used to troubleshoot - it is on by default.

## 3.7 NetBackup

### 3.7.1 Prerequisites

User credentials that have read access to the VM level in vCenter.

$VCCredential = Import-CliXml -Path c:\source\scripts\credentials\ks\_cred.xml

### 3.7.2 Monitored elements

#### 3.7.2.1 Missed Backups

Any VMs with their last backup time greater than 24 hours ago are reported on. Any VMs with the BackupAllowed parameter set to DOMAIN CONTROLLER|CDRIVE|FALSE are ignored.

### 3.7.3 Configuration file

For this script, there is NO configuration file. As the script is so small, configuration is done inside the script in the following lines:

$VCCredential = Import-CliXml -Path c:\source\scripts\Credentials\SVC-VMVRRW-001.xml

$servers = "nhc4-inf-vcm001","nhc5-inf-vcm001"

$sendEmail = $true

$ReportFile="C:\inetpub\wwwroot\monitor\NetBackuphealth.html"

$ReportURL = "http://NHC4-NAD-VMAN01/Monitor/NetBackuphealth.html"

$reportsubject = "NetBackup Health (US HoldCo)"

### 3.7.4 Output

As well as the html report file, the script outputs other files to monitor behaviour:

A log file - this monitors all of the functions/commands performed by the script.

A transcript file - this is effectively the output of the command and may be used to troubleshoot - it is on by default.

## 3.8 Windows

### 3.8.1 Prerequisites

Password for monitoring user saved in xml file using:

Get-Credential | Export-CliXml -Path c:\source\scripts\Credentials\SVC-VMVRRW-001.xml

(or alternative path but remember to update the Test-WIndowsHealth-cfg.ps1 script)

Config file set up with correct parameters.

### 3.8.2 Monitored elements

#### 3.8.2.1 DNS

Check controller address is resolvable in DNS

If IP is returned from DNS, perform PING on it

#### 3.8.2.2 Uptime

Checks uptime on the systems and reports if below the $MinimumUptime in the config file.

#### 3.8.2.3 Services

Checks if the services configured in the config file ($DesiredStateServices ) are running

#### 3.8.2.4 Events

Reports on any non-informational events in the Application Log reported going back $MaxHoursToScanLog

#### 3.8.2.5 Networks

Pings all data and array mgmt interfaces.

#### 3.8.2.6 Hardware

Checks for any errors on the IPMI (HPe ILO only)

#### 3.8.2.7 Performance

Reports on any CPUs above threshold set in scripts as $CPUUsagePercent (default=90%)

Reports on Memory below threshold set in scripts as $MemoryFreeUsagePercent (default=10%)

#### 3.8.2.8 Disk Space

Reports on any volumes under thresholds set in scripts as $volumePercentFree (default=10%)

#### 3.8.2.9 Certificates

Reports on any certificates that are about to expire

### 3.8.3 Configuration file

Test-WINDOWShEALTHlth-cfg.ps1 script:

#Max hours to go back and alert in logs

$MaxHoursToScanLog = 4

#Warn if uptime is below this number of hours

$MinimumUptime = 2

#Number of days back before we alert about a cert being about to expire

$CertificateTimeToAlert = 30

#Fill in WIndows servers to monitor

$targets = "boh2-eud-cvma01.eu.cobham.net","boh2-eud-cvma02.eu.cobham.net","boh2-eud-cvma11.eu.cobham.net","boh2-eud-cvma12.eu.cobham.net"

#$targets = "boh2-eud-cvma01.eu.cobham.net"

#Minimum Percentage free on volumes before raising alert

$volumePercentFree = 10

#Maximum Percentage CPU usage before raising alert

$CPUUsagePercent = 90

#Minimum Percentage Free Memory usage before raising alert

$MemoryFreeUsagePercent = 10

#Comma separated array of network interfaces to be ignored

$IgnoreNetworkFailures = @("Embedded LOM 1 Port 4","Embedded LOM 1 Port 3")

#Comma separated array of IP addresses to NOT ping (by default GreenScreen will ping all addresses on each server)

$IgnoreIPaddresses = @("10.216.7.4","10.191.128.104","10.216.7.5","10.191.128.105","10.136.73.4","10.216.7.4","10.216.7.5","10.136.73.5")

#Comma separated array of server event numbers to ignore

$IgnoreServerEvents = @("10016")

#Comma separated array of servers to not log disk space

$IgnoreServerDiskSpace = @("boh2-eud-cvma01.eu.cobham.net","boh2-eud-cvma02.eu.cobham.net")

#Comma separated array of servers to not log hardware errors

$IgnoreHardwareErrors = @("boh2-eud-cvma01.eu.cobham.net","boh2-eud-cvma02.eu.cobham.net")

#Services to monitor

$DesiredStateServices = @("GxBlr(Instance001)","GxClMgrS(Instance001)","GxCVD(Instance001)","GXMMM(Instance001)","GxFWD(Instance001)")

#Report settings

#Determines whether the script logs it's activities using logfile described below

$Log = $true

#Location of log file for script

$logfile = "C:\Source\Scripts\Windows\Windows\_health.log"

#Location of transcript file for script (shows command line output for script

$TranscriptFile = "C:\Source\Scripts\\Windows\Windows\_health\_transcript.log"

#Reportsubject = the heading for the html report

$reportsubject = "Windows Health (MSD)"

#Enter the URL of the report for use from the Web

$ReportURL = "http://BOH2-EUD-VMAN01\Monitor\windowshealth.html"

#ReportMode - Set to $true to generate a HTML report. Uses the name of the Report file (see below)

$ReportMode=$true

#ReportFile - name of file to output html report to

$ReportFile="C:\inetpub\wwwroot\monitor\windowshealth.html"

#Determines whether we send the HTML report via email using the SMTP configuration within the config file.

$SendEmail=$true

#Only sends the email report if at least one error or warning was detected.

$AlertsOnly=$true

#...................................

# Email Settings

#...................................

#Send email to this address

$recipients = "ITS.Datacentre.Services@cobham.com"

#Send email from this address

$fromaddress = "MSD-Windows-Alerts@cobham.com"

#Send email using this relay host

$smtpserver = "smtp.eu.cobham.net"

#Credentials

#Credential for IPMI

$IPMICredential = Import-CliXml -Path c:\source\scripts\Credentials\hpeilo.xml

#Credential for Servers

$Credential = Import-CliXml -Path c:\source\scripts\Credentials\SVC-VMVRRW-001.xml

### 3.8.4 Output

As well as the html report file, the script outputs other files to monitor behaviour:

A log file - this monitors all of the functions/commands performed by the script.

A transcript file - this is effectively the output of the command and may be used to troubleshoot - it is on by default.

## 3.9 Cisco UCS

### 3.9.1 Prerequisites

Password for monitoring user saved in xml file using:

Get-Credential | Export-CliXml -Path c:\source\scripts\Credentials\ciscoadmin.xml

(or alternative path but remember to update the Test-UCSHealth-cfg.ps1 script)

Config file set up with correct parameters.

### 3.9.2 Monitored elements

#### 3.9.2.1 System

Check component management addresse are resolvable in DNS

If IP is returned from DNS, perform PING on it

Checks all FICs/chassis/IOMs for errors

#### 3.9.2.2 Uptime

Checks uptime on the systems and reports if below the $MinimumUptime in the config file.

#### 3.9.2.3 Faults

Checks for any recorded faults on the UCS of severity “Error” and “Critical”

#### 3.9.2.4 Networks

Checks for any recorded Network faults on the UCS of severity “Error” and “Critical”

#### 3.9.2.5 Hardware

Checks for any recorded inoperable hardware

### 3.9.3 Configuration file

Test-UCSHealth-cfg.ps1 script:

#...................................

# Variables

#...................................

#Warn if uptime is below this number of hours

$MinimumUptime = 2

$UCSMServers = @(“10.251.234.36”,“10.250.234.36”)

#$UCSMServers = @(“10.251.234.36”)

#Comma separated array of servids to ignore in the form chassis-1/blade-1@NHC0-NET-FIC001 for blades or rack-unit-1@NHC0-NET-FIC001 for rack servers

$IgnoreServerIDs = @("rack-unit-5@NHC1-NET-FIC001")

#Comma separated array of objects to ignore (this will ignore any FIC errors containing the relevant text) (remember the \* character)

$IgnoreObjects = @("\*NHC0-INF-VDI010\*","\*testing\*","\*another\*")

#Report settings

#Determines whether the script logs it's activities using logfile described below

$Log = $true

#Location of log file for script

$logfile = "C:\Source\Scripts\Cisco\UCS\UCS\_health.log"

#Location of transcript file for script (shows command line output for script

$TranscriptFile = "C:\Source\Scripts\Cisco\UCS\UCS\_health\_transcript.log"

#Reportsubject = the heading for the html report

$reportsubject = "UCS Health (UK HoldCo)"

#Enter the URL of the report for use from the Web

$ReportURL = "http://NHC0-EUD-VMAN01/Monitor/UCSHealth.html" #Enter the name of the server where ae are saving the errors (probably the server where this script is running)

#ReportMode - Set to $true to generate a HTML report. Uses the name of the Report file (see below)

$ReportMode=$true

#ReportFile - name of file to output html report to

$ReportFile="C:\inetpub\wwwroot\monitor\UCSHealth.html"

#Determines whether we send the HTML report via email using the SMTP configuration within the config file.

$SendEmail=$true

#Only sends the email report if at least one error or warning was detected.

$AlertsOnly=$true

#...................................

# Email Settings

#...................................

#Send email to this address

#$recipients = "kevin.snook@cobham.com"

$recipients = "ITS.DCI.Datacentre.Services@cobham.com"

#$recipients = "nobody@cobham.com"

#Send email from this address

$fromaddress = "HoldCo-UCS-Alerts@cobham.com"

#Send email using this relay host

$smtpserver = "smtp.eu.cobham.net"

#...................................

#Credentials

#....................................

#...................................

# Credentials

#...................................

#UCSM credentials

$UCSCredential = Import-CliXml -Path c:\source\scripts\Credentials\ciscoadmin.xml

### 3.9.4 Output

As well as the html report file, the script outputs other files to monitor behaviour:

A log file - this monitors all of the functions/commands performed by the script.

A transcript file - this is effectively the output of the command and may be used to troubleshoot - it is on by default.

# 4 Daily Report

For convenience a daily roll-up report is sent.

### 4.1 Prerequisites

This report will need to be added to the task scheduler. A morning and evening report is thought to be the best practice. A batch file with contents as follows will run the report:

C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe -command "C:\Source\Scripts\Send\_WrapUp\_Status\_Email.ps1"

### 4.2 Monitored elements

#### 4.2.1 Reports output by system health checks

This script checks the IIS folder for reports. It then wraps them up into one email and sends that to the configured recipients.

### 4.3 Configuration file

For this script, there is NO configuration file. As the script is so small, configuration is done inside the script. The entire script is shown here:

# Email Settings

#...................................

$recipients = @("ITS.DCI.Datacentre.Services@cobham.com")

#$recipients = @("kevin.snook@cobham.com")

$smtpsettings = @{

From = "CMS-System-Alerts@cobham.com"

SmtpServer = "smtp.eu.cobham.net"

}

$reportFolder="C:\inetpub\wwwroot\monitor"

#Find any reports that have not been updated in 1 hour

#$Result = Get-ChildItem $reportFolder | Where-Object {$\_.LastWriteTime -lt (Get-Date).AddHours(-1)}

#$Result

#exit

#$serversummaryhtml = "<h3>VMware Health Details</h3>

# <p>$ignoretext</p>

# <p>No VMware health errors or warnings.</p>"

$htmlreport = ""

if (Get-ChildItem $reportFolder -Filter \*Error\_Status\_Fail\*){

$reportemailsubject = "FAIL - CMS Daily Health Report"

}

else{

if (Get-ChildItem $reportFolder -Filter \*.html| Where-Object {$\_.LastWriteTime -lt (Get-Date).AddHours(-1)}){

#$results = Get-ChildItem $reportFolder | Where-Object {$\_.LastWriteTime -lt (Get-Date).AddHours(-1) -and $\_.name -notmatch "ticker.txt"}

#$results

$reportemailsubject = "FAIL - CMS Daily Health Report"

$htmlreport += "<html>

<style>

H4{font-size: 20px;color: #FF0000;font-weight: bold;}

</style>

<body>

<h4 align=`"center`">At least one report has not been generated for over 1 hour</h4>

</body>

</html>"

}

else{

$reportemailsubject = "PASS - CMS Daily Health Report"

}

}

$reportime = (Get-Date).ToString("dd/MM/yyyy HH:mm")

foreach ($reportfile in (Get-ChildItem $reportFolder -Filter \*.html| Sort name)){

$htmlreport += get-content $reportfile.FullName

}

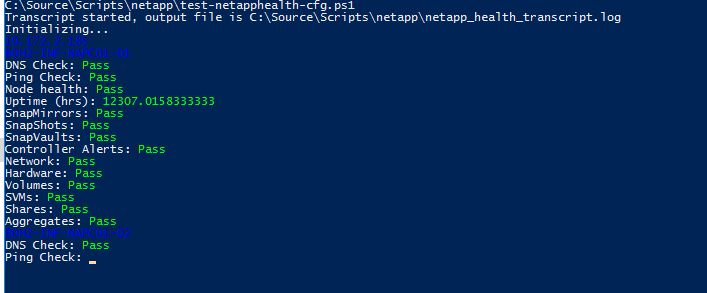
Send-MailMessage @smtpsettings -To $recipients -Subject "$reportemailsubject - $reportime" -Body $htmlreport -BodyAsHtml -Encoding ([System.Text.Encoding]::UTF8) -Priority High

### 4.4 Output

An email is sent to recipients showing all health check output in the configured folder. A subject is added which shows whether there is anything wrong in the environment (FAIL) or everything is good (PASS). This will save busy admins from even having to read the report(s).

# 5 Interactive Mode

The reports can be run from a Powershell command line. This produces output like this:



Elements that Pass the check are shown in Green and any failures are shown in Red. This is useful when you need a quick check to see if there are any issues.

# Appendix 1 – greenscreen.html

<meta http-equiv="refresh" content="20"> <!-- Refresh every 20 secs -->

<meta http-equiv="Cache-Control" content="no-cache, no-store, must-revalidate">

<meta http-equiv="Pragma" content="no-cache">

<meta http-equiv="Expires" content="0">

<head><title>GreenScreen(HoldCo)</title></head>

<body style="background-color:lightgray"></body>

<style>

/\* (A) FIXED WRAPPER \*/

.hwrap {

overflow: hidden; /\* HIDE SCROLL BAR \*/

background: lightgray;

border: 4px ridge lightblue;

position:absolute;

bottom:0;

width: 96%;

height: 20px

}

/\* (B) MOVING TICKER WRAPPER \*/

.hmove { display: flex; }

/\* (C) ITEMS - INTO A LONG HORIZONTAL ROW \*/

.hitem {

flex-shrink: 0;

width: 100%;

padding: 0px;

font-family: Tahoma;

font-size: 16px;

}

/\* (D) ANIMATION - MOVE ITEMS FROM RIGHT TO LEFT \*/

/\* 4 ITEMS -400%, CHANGE THIS IF YOU ADD/REMOVE ITEMS \*/

@keyframes tickerh {

0% { transform: translate3d(100%, 0, 0); }

100% { transform: translate3d(-400%, 0, 0); }

}

.hmove { animation: tickerh linear 29s infinite ; }

.hmove:hover { animation-play-state: paused; }

</style>

<div class="hwrap" id="hwrap"><div class="hmove">

<div class="hitem" id="hitem"><object id="filecontents" width="800px"></object></div>

</div></div>

<iframe id="nhc0vmwarehealth" scrolling="no" style="width:300px; height:60px;border: 4px ridge lightblue;" src="http://nhc0-eud-vman01/monitor/vmwarehealth.html"></iframe>

<iframe id="nhc0simplivityhealth" scrolling="no" style="width:300px; height:60px;border: 4px ridge lightblue;" src="http://nhc0-eud-vman01/monitor/simplivityhealth.html"></iframe>

<iframe id="nhc0purehealth" scrolling="no" style="width:300px; height:60px;border: 4px ridge lightblue;" src="http://nhc0-eud-vman01/monitor/purehealth.html"></iframe>

<iframe id="nhc0ucshealth" scrolling="no" style="width:300px; height:60px;border: 4px ridge lightblue;" src="http://nhc0-eud-vman01/monitor/ucshealth.html"></iframe>

<iframe id="boh2vmwarehealth" scrolling="no" style="width:300px; height:60px;border: 4px ridge lightblue;" src="http://boh2-eud-vman01/monitor/vmwarehealth.html"></iframe>

<iframe id="boh2simplivityhealth" scrolling="no" style="width:300px; height:60px;border: 4px ridge lightblue;" src="http://boh2-eud-vman01/monitor/simplivityhealth.html"></iframe>

<iframe id="boh2purehealth" scrolling="no" style="width:300px; height:60px;border: 4px ridge lightblue;" src="http://boh2-eud-vman01/monitor/purehealth.html"></iframe>

<iframe id="boh2netapphealth" scrolling="no" style="width:300px; height:60px;border: 4px ridge lightblue;" src="http://boh2-eud-vman01/monitor/netapphealth.html"></iframe>

<iframe id="ory1vmwarehealth" scrolling="no" style="width:300px; height:60px;border: 4px ridge lightblue;" src="http://ory1-eud-vman01/monitor/vmwarehealth.html"></iframe>

<iframe id="ory1simplivityhealth" scrolling="no" style="width:300px; height:60px;border: 4px ridge lightblue;" src="http://ory1-eud-vman01/monitor/simplivityhealth.html"></iframe>

<iframe id="ory1windowshealth" scrolling="no" style="width:300px; height:60px;border: 4px ridge lightblue;" src="http://ory1-eud-vman01/monitor/windowshealth.html"></iframe>

<iframe id="boh2windowshealth" scrolling="no" style="width:300px; height:60px;border: 4px ridge lightblue;" src="http://boh2-eud-vman01/monitor/windowshealth.html"></iframe>

<iframe id="gva1vmwarehealth" scrolling="no" style="width:300px; height:60px;border: 4px ridge lightblue;" src="http://gva1-eud-vman01/monitor/vmwarehealth.html"></iframe>

<iframe id="gva1simplivityhealth" scrolling="no" style="width:300px; height:60px;border: 4px ridge lightblue;" src="http://gva1-eud-vman01/monitor/simplivityhealth.html"></iframe>

<iframe id="nhc4vmwarehealth" scrolling="no" style="width:300px; height:60px;border: 4px ridge lightblue;" src="http://nhc4-nad-vman01.na.cobham.net/monitor/vmwarehealth.html"></iframe>

<iframe id="nhc4ucshealth" scrolling="no" style="width:300px; height:60px;border: 4px ridge lightblue;" src="http://nhc4-nad-vman01.na.cobham.net/monitor/ucshealth.html"></iframe>

<iframe id="nhc0vmwarehealth2" scrolling="no" style="width:300px; height:60px;border: 4px ridge lightblue;" src="http://nhc0-eud-vman01/monitor/vmwarehealth2.html"></iframe>

<iframe id="nhc0commvaulthealth" scrolling="no" style="width:300px; height:60px;border: 4px ridge lightblue;" src="http://nhc0-eud-vman01/monitor/CommVaulthealth.html"></iframe>

<iframe id="boh2commvaulthealth" scrolling="no" style="width:300px; height:60px;border: 4px ridge lightblue;" src="http://boh2-eud-vman01/monitor/CommVaulthealth.html"></iframe>

<iframe id="ory1commvaulthealth" scrolling="no" style="width:300px; height:60px;border: 4px ridge lightblue;" src="http://ory1-eud-vman01/monitor/CommVaulthealth.html"></iframe>

<iframe id="nhc4netbackuphealth" scrolling="no" style="width:300px; height:60px;border: 4px ridge lightblue;" src="http://nhc4-nad-vman01/monitor/NetBackuphealth.html"></iframe>

<script>

window.onload = function loadFunction() {

const d = new Date();

let nowtime = d.getTime();

console.log(nowtime);

var txtURL = `http://nhc0-eud-vman01/monitor/ticker.txt?${nowtime}`;

var txtFile = new XMLHttpRequest();

console.log(txtURL);

txtFile.open("GET", txtURL, true);

txtFile.onreadystatechange = function() {

if (txtFile.readyState === 4) { // Makes sure the document is ready to parse.

if (txtFile.status === 200) { // Makes sure it's found the file.

allText = txtFile.responseText;

console.log(allText);

divider = document.getElementById("filecontents")

/\*lines = txtFile.responseText.split("\n"); // Will separate each line into an array\*/

tidyText = allText.replace(/[\n\r]+/g, " ++++++ ");

console.log(tidyText);

/\*var interval = 20000; // how much time should the delay between two iterations be (in milliseconds)?

lines.forEach(function (line, index) {

setTimeout(function () {

divider.textContent = line

marqueescroll.textContent = line

console.log(line);

}, index \* interval);

});

\*/

divider.textContent = tidyText

}

}

}

txtFile.send(null);

document.getElementById('nhc0vmwarehealth').contentWindow.location.reload(true);

document.getElementById('nhc0simplivityhealth').contentWindow.location.reload(true);

document.getElementById('nhc0purehealth').contentWindow.location.reload(true);

document.getElementById('nhc0ucshealth').contentWindow.location.reload(true);

document.getElementById('boh2vmwarehealth').contentWindow.location.reload(true);

document.getElementById('boh2simplivityhealth').contentWindow.location.reload(true);

document.getElementById('boh2purehealth').contentWindow.location.reload(true);

document.getElementById('boh2netapphealth').contentWindow.location.reload(true);

document.getElementById('ory1vmwarehealth').contentWindow.location.reload(true);

document.getElementById('ory1simplivityhealth').contentWindow.location.reload(true);

document.getElementById('ory1windowshealth').contentWindow.location.reload(true);

document.getElementById('gva1vmwarehealth').contentWindow.location.reload(true);

document.getElementById('gva1simplivityhealth').contentWindow.location.reload(true);

document.getElementById('nhc4vmwarehealth').contentWindow.location.reload(true);

document.getElementById('nhc4ucshealth').contentWindow.location.reload(true);

document.getElementById('nhc0vmwarehealth2').contentWindow.location.reload(true);

document.getElementById('boh2windowshealth').contentWindow.location.reload(true);

document.getElementById('nhc0commvaulthealth').contentWindow.location.reload(true);

document.getElementById('boh2commvaulthealth').contentWindow.location.reload(true);

document.getElementById('ory1commvaulthealth').contentWindow.location.reload(true);

document.getElementById('nhc4netbackuphealth').contentWindow.location.reload(true);

}

</script>