#!/usr/bin/python

# nbreport.py

Version = 'b.3'

#

# Author: Terry Carter/Roberto Garcia

#

# v b.3 3/5/2019 Added tape scratch and availability per library

# v b.2.3 3/4/2019 Added verbose log, FOD email preferences and fixed missed tape backups

# v b.2.2 2/13/2019 Fixes for newly provisioned server faults

# v b.2.1 fixes for domains without tape backups.

# v b.2 added tape backup checks.

# v b.1 added Prakash and Pushpinder to email. Revised error code section

#

# This script:

# Gets a list of all assigned clients

# A) Reports if any clients do not have backups for the past two days

# B) Read in a list of clients from active directory (report provided)

# Reports on any hosts in AD that are NOT a client

# Reports on any hosts that are clients that are not in AD

#

# Future Update requests

# 1) Report on the status of the last backup (status 0, status 1 or other)

# 2) For status 1, what files were missed

# 3) Log time of backup - POM reporting

# 4) Report on volume of space used on disk storage

#

############################################################################################

#

# Init

# Here are all of the things needed to initialize the processes used.

#

############################################################################################

# Local imports #

from column import indent # Local script that prints in columns

from nbjobs import parse\_jobs # Lists long running jobs

from nblibdr import library\_status # Lists library status

from nblibdr import drive\_status # Lists drive status

from nblibdr import get\_robot\_connections # Lists robot connections

from nblibdr import get\_library\_name # Lists the common name of the robot

from nblibdr import do\_inventory # Tells the robot to inventory

from nblibdr import list\_scratch # Lists number of scratch tapes in the library

from nblibdr import free\_slots # Lists number of free slots in the library

# Global imports #

import re # Used to replace items

import shlex # used to format lines for Popen

import subprocess # Used for Popen, directions to the command line

import datetime # Used for date and relative date information

#import smtplib # Import smtplib for the actual sending function

#from email.MIMEMultipart import MIMEMultipart

#from email.MIMEText import MIMEText

import socket # Used to get host information

Hostname = socket.gethostname() # Get host name

FQDN = socket.getfqdn() # Get fully qualified host name

#print Hostname

#print FQDN

from whereami import am\_i\_fod # Is this running on a FOD system?

if am\_i\_fod():

MailTo = "fod\_backups\_us\_grp@oracle.com" # List the people to mail to for FOD

else:

MailTo = "cit\_backups\_ww\_grp@oracle.com" # List the people to mail to for NON-FOD

ReportFileName = "/tmp/nbreportfile.tmp" # Location of report file used for mailing

verbose\_logging = 1 # Verbose logging flag

# 0 Off - General status

# 1 Commands being ran

# 2 Results of commands

############################################################################################

#

# ListClients

# This gets a list of all of the clients assigned in NetBackukp

#

# Input: None

# Output: Array of clinet names, Int total number of clients

#

############################################################################################

def ListClients():

print "Getting a list of clients"

OutputArray = [] # Clears the output array

TotalClients = 0

# cmd ="/usr/openv/netbackup/bin/admincmd/bperror -U -backstat -by\_statcode -hoursago ${HOURS} >> ${TEMP}"

cmd="/usr/openv/netbackup/bin/admincmd/bpplclients" # Insert whatever bp command you want to run here

args = shlex.split(cmd) # Formats the string to be used with Popen - splits on spaces

output,error=subprocess.Popen (args,stdout = subprocess.PIPE, stderr= subprocess.PIPE).communicate()

output = re.sub(' +',' ',output) # Takes multiple spaces and converts to one space

output = output.splitlines() # Splits multiple line var into one line per array element

for line in output: # For each line in the array

line = line.split() # Split each item on each line

OutputArray.append(line [2]) # Save this to an array

TotalClients = TotalClients + 1 # Add up Total Clients

return OutputArray, TotalClients

############################################################################################

#

# LastBackup

# For all of the host names in List, we look up the latest and latest full backups.

# If the latest backup is more than 2 days ago, we throw an error

# If the latest full backup is more than a week ago (Roughly 192 hours), we throw an error

# If the request for backups throws an error, we pass it on

#

# Input: Array of NetBackup hosts

# Output: Array of all hosts and the backup status

# Array of failed hosts

# Integer number of clients having errors

#

############################################################################################

def LastBackup(List):

print "Checking clients for backups"

Today = datetime.date.today() # Get today's date

TwoDaysAgo = Today - datetime.timedelta(days=2) # Get the date for 2 days ago

FullOutArray = [['Host','Last Backup','Full Backup','Status']] # Create headers for arrays

ErrorOutArray = [['Host','Last Backup','Full Backup','Status']]

NumErrorClients = 0 # Zero the number of clients with errors counter

### Loop through all hosts ###

for x in range(2,len(List)): # Skip the first two header lines. Cycle through the rest of the array

HostErrorFlag = False

# cmd="/usr/openv/netbackup/bin/admincmd/bperror -U -backstat -by\_statcode -hoursago ${HOURS} >> ${TEMP}"

cmd="/usr/openv/netbackup/bin/admincmd/bpimagelist -U -client " + List[x] + " -hoursago 192" # Insert bp command to run here

if verbose\_logging >= 1: # For troubleshooting

print cmd # Print command

args = shlex.split(cmd) # Formats the string to be used with Popen - splits on spaces

Output,Error=subprocess.Popen (args,stdout = subprocess.PIPE, stderr= subprocess.PIPE).communicate()

if verbose\_logging >= 2: # For troubleshooting

print Output # Print what was returned

if ((Error == "") and (Output != "")): # No error from the Popen command

Output = Output.splitlines() # Splits multiple line var into one line per array element

OutputArray = [] # Clears the output array

for Line in Output: # For each line in the array

Line = Line.split() # Split each item on each line

OutputArray.append(Line) # Save this to an array

LastBackupDate = OutputArray [2][0] # Capture the last valid backup date

FormattedLastBackupDate = datetime.datetime.strptime(LastBackupDate, "%m/%d/%Y") # Convert last backup date into a date variable

### Was the latest backup more than two days ago? ###

if FormattedLastBackupDate.date() < TwoDaysAgo: # Was the last backup more than 2 days ago?

HostStatus = "Last backup too old, "

HostErrorFlag = True # Flag there was a backup error with this host

else: # Backup is fine

HostStatus = "Last backup is fine, "

ValidFullBackup = False # Capture the last full backup date

### Loop through all of the backups for the host to look for Full backup ###

for y in range(len(OutputArray)): # Loop through all the lines in the output

if OutputArray[y][6] == "Full": # Check to see if it's a full backup

ValidFullBackup = True # Set flag that it's a full backup

ValidFullBackupDate = OutputArray[y][0] # Get the date of the full backup

if ValidFullBackup == True: # Print info on full backup condition

HostStatus = HostStatus + "Full backup is fine"

else: # No Full backups in the last week

HostStatus = HostStatus + "Missing Full backup"

HostErrorFlag = True # Flag there was a backup error with this host

ValidFullBackupDate = '00/00/0000'

FullOutArray.append([List[x],LastBackupDate,ValidFullBackupDate,HostStatus]) # Append the status to the array

else: # We got an error from the Popen command

Error = re.sub('\r','',Error) # take out carriage returns

Error = re.sub('\n','',Error) # take out Line feeds

HostErrorFlag = True # Flag there was a backup error with this host

LastBackupDate = '00/00/0000'

ValidFullBackupDate = '00/00/0000'

HostStatus = Error

FullOutArray.append([List[x],LastBackupDate,ValidFullBackupDate,HostStatus]) # Append the status to the array

### If there was an error on the Full or last backup, log it in the error list and count ###

if HostErrorFlag:

# ErrorOutArray.append([List[x],'00/00/0000','00/00/0000',Error])

ErrorOutArray.append([List[x],LastBackupDate,ValidFullBackupDate,HostStatus]) # Append the status to the array

NumErrorClients = NumErrorClients + 1 # Increment number of clients with errors count

return FullOutArray, ErrorOutArray, NumErrorClients # Return full results, error results and number of clients with errors.

############################################################################################

#

# GetStsCode

# List error clients results with status code

#

# Input: Array of hosts having errors

# Output: Array report of errored hosts sorted by error type

#

############################################################################################

def GetStsCode(ErrorOutArray):

ErrCodeReport\_array = [] # initialize array

print "Getting a list of clients with errors"

ErrCodeReport = "\n" # Initializing error code report variable

for line in range(1,len(ErrorOutArray)): # Loop through all of the hosts reporting a failure

hostname = ErrorOutArray[line][0] # Get the host name

# Get the status of the failed host over the last 24 hours

cmd ="/usr/openv/netbackup/bin/admincmd/bperror -client " + hostname + " -backstat"

args = shlex.split(cmd) # Formats the string to be used with Popen - splits on spaces

Output,Error=subprocess.Popen (args,stdout = subprocess.PIPE, stderr= subprocess.PIPE).communicate()

if Output: # Only sort when we get output, not errors

Output\_words = Output.split() # Split the output into words

host\_error = Output\_words[len(Output\_words)-1] # The error nubmer is from the last backup, last item in the string

ErrCodeReport\_array.append([int(host\_error),hostname]) # write error number and host to an array

ErrCodeReport\_array.sort() # Sort the array after it's completely written

ErrCodeReport = "\n" # Initializing error code report variable

last\_error = 400 # init the last\_error to something that will be new on first test

for line in range(len(ErrCodeReport\_array)): # Sift through all the lines of the array and format for test

if last\_error <> ErrCodeReport\_array[line][0]: # Is this the same error we had last time

last\_error = ErrCodeReport\_array[line][0] # Update the last error code we got

ErrCodeReport += "\n" # Add newline for each error code

cmd ="/usr/openv/netbackup/bin/admincmd/bperror -statuscode " + str(last\_error)

args = shlex.split(cmd) # Formats the string to be used with Popen - splits on spaces

Output,Error=subprocess.Popen (args,stdout = subprocess.PIPE, stderr= subprocess.PIPE).communicate()

if Output: # If we get a description of the error code...

ErrCodeReport += str(last\_error) + " " + Output.splitlines()[0] + "\n" # save the number AND the short error description

else: # Otherwise

ErrCodeReport += str(last\_error) + "\n" # Just print the error number

ErrCodeReport += str(ErrCodeReport\_array[line][1]) + "\n" # Add the host beneath the error code/desc heading

return ErrCodeReport # Return the formatted output

########################################################################

#

# Get\_Policies

# Return a list of Policies

#

# Input: None

# Output: List of policies

#

########################################################################

def get\_policies():

cmd = "/usr/openv/netbackup/bin/admincmd/bppllist" # Insert whatever bp command you want to run here

args = shlex.split(cmd) # Formats the string to be used with Popen - splits on spaces

output, error = subprocess.Popen(args, stdout=subprocess.PIPE, stderr=subprocess.PIPE).communicate()

if error:

raise ValueError(error) # Pass the Popen error

else:

return output

########################################################################

#

# Check\_Policy\_Library

# Checks to see if there is a vault policy which means there is a library attached

# Returns if found (True) or not (False)

#

# Input: None

# Output: Boolean status of attached library

#

########################################################################

def check\_policy\_library():

output = get\_policies()

output = output.splitlines() # Splits multiple line var into one line per array element

for line in output: # For each line in the array

if "Vault" in line: # the word "Vault" means we have a library attached

if verbose\_logging:

print line + " has the word Vault in it"

return True # Policy name found

return False # Policy name not found

############################################################################################

#

# LastTapeBackup

# For all of the host names in List, we look up the latest and latest full backups.

# If the latest backup is more than 2 days ago, we throw an error

# If the latest full backup is more than a week ago (Roughly 192 hours), we throw an error

# If the request for backups throws an error, we pass it on

#

# Input: Array of host names

# Output: Array listing all hosts and their backup to tape status

# Array of Failing hosts with their tape status

# Integer number of clients with failures

#

############################################################################################

def LastTapeBackup(List):

print "Checking clients for tape backups"

Today = datetime.date.today() # Get today's date

TwoDaysAgo = Today - datetime.timedelta(days=2) # Get the date for 2 days ago

FullOutArray = [['Host','Last Backup','Full Backup','Status']] # Create headers for arrays

ErrorOutArray = [['Host','Last Backup','Full Backup','Status']]

NumErrorClients = 0 # Zero the number of clients with errors counter

### Loop through all hosts ###

for x in range(2,len(List)): # Skip the first two header lines. Cycle through the rest of the array

HostErrorFlag = False

# cmd="/usr/openv/netbackup/bin/admincmd/bperror -U -backstat -by\_statcode -hoursago ${HOURS} >> ${TEMP}"

cmd="/usr/openv/netbackup/bin/admincmd/bpimagelist -U -client " + List[x] + " -hoursago 192 -tape" # Insert bp command to run here

if verbose\_logging >= 1: # For troubleshooting

print cmd # Print command

args = shlex.split(cmd) # Formats the string to be used with Popen - splits on spaces

Output,Error=subprocess.Popen (args,stdout = subprocess.PIPE, stderr= subprocess.PIPE).communicate()

if verbose\_logging >=2: # For troubleshooting

print "Output", Output # Print results

if ((Error == "") and (Output != "")): # No error from the Popen command

Output = Output.splitlines() # Splits multiple line var into one line per array element

OutputArray = [] # Clears the output array

for Line in Output: # For each line in the array

Line = Line.split() # Split each item on each line

OutputArray.append(Line) # Save this to an array

LastBackupDate = OutputArray [2][0] # Capture the last valid backup date

FormattedLastBackupDate = datetime.datetime.strptime(LastBackupDate, "%m/%d/%Y") # Convert last backup date into a date variable

### Was the latest backup more than two days ago? ###

if FormattedLastBackupDate.date() < TwoDaysAgo: # Was the last backup more than 2 days ago?

HostStatus = "Last backup too old, "

HostErrorFlag = True # Flag there was a backup error with this host

else: # Backup is fine

HostStatus = "Last backup is fine, "

ValidFullBackup = False # Capture the last full backup date

### Loop through all of the backups for the host to look for Full backup ###

for y in range(len(OutputArray)): # Loop through all the lines in the output

if OutputArray[y][6] == "Full": # Check to see if it's a full backup

ValidFullBackup = True # Set flag that it's a full backup

ValidFullBackupDate = OutputArray[y][0] # Get the date of the full backup

if ValidFullBackup == True: # Print info on full backup condition

HostStatus = HostStatus + "Full backup is fine"

else: # No Full backups in the last week

HostStatus = HostStatus + "Missing Full backup"

HostErrorFlag = True # Flag there was a backup error with this host

ValidFullBackupDate = '00/00/0000'

FullOutArray.append([List[x],LastBackupDate,ValidFullBackupDate,HostStatus]) # Append the status to the array

else: # We got an error from the Popen command

Error = re.sub('\r','',Error) # take out carriage returns

Error = re.sub('\n','',Error) # take out Line feeds

HostErrorFlag = True # Flag there was a backup error with this host

LastBackupDate = '00/00/0000'

ValidFullBackupDate = '00/00/0000'

HostStatus = Error

FullOutArray.append([List[x],LastBackupDate,ValidFullBackupDate,HostStatus]) # Append the status to the array

### If there was an error on the Full or last backup, log it in the error list and count ###

if HostErrorFlag:

# ErrorOutArray.append([List[x],'00/00/0000','00/00/0000',Error])

ErrorOutArray.append([List[x],LastBackupDate,ValidFullBackupDate,HostStatus]) # Append the status to the array

NumErrorClients = NumErrorClients + 1 # Increment number of clients with errors count

return FullOutArray, ErrorOutArray, NumErrorClients # Return full results, error results and number of clients with errors.

############################################################################################

#

# get\_jobs

# Retrieve long running jobs

# Format for nice printing

#

# Input: Integer, constant for lenth of hours judged to be "long running"

# Calls: parse\_jobs from nbjobs.py library

# Retrieves: Array of jobs lasting longer than given time

# Output: Text Formatted, ready for report

#

############################################################################################

def get\_jobs(job\_time):

long\_jobs = parse\_jobs(job\_time) # Find long running jobs

job\_return = "Active Backup Jobs\n" # Include header in report

if long\_jobs == []: # Test if there are any long running jobs

job\_return += "All jobs running on time" # Print something instead of empty contents

else: # There are long running jobs

for line in long\_jobs: # Read through all the lines of long running jobs

job\_return += " ".join(("Host", line[0] , "/ Backup job", line[1], " has been running for over", str(line[2]), "Hours\n"))

return job\_return

############################################################################################

#

# MakeEmail

# Open a file and create the body of the email to send

#

# Input: Total - Integer total number of josts

# Full - Array with the disk backup status of each host

# Error - Array with the disk backup status of failed hosts

# NumError - Integer number of jobs with errors

# StsCode - Text preformatted report of failed hosts sorted by status code

# has\_library - Boolean True if this domain has a library attached

# job\_status - Text preformatted report of the status of long running jobs

# Calls: LastTapeBackup - Returns tape backup status'

# Ouptut: File Name and path described in 'ReportFileName' in init section

#

############################################################################################

def MakeEmail(Total, Full, Error, NumError, StsCode, has\_library, job\_status):

print "Creating file to mail" # Message to user for section start

SuccessHost = Total - NumError # Compute successfull completed hosts backup to disk

SuccessRate = int(float(SuccessHost) / Total \* 100) # Compute successful percentage

FailureRate = 100 - SuccessRate # Compute failure rate

# Get tape data

if has\_library: # Does this environment have a tape library

print "This domain has a tape library" # Message to user for this section

FullTape, ErrorTape, NumErrorTape = LastTapeBackup(ClientsInPolicy) # Retrieve tape backup stats

SuccessHostTape = Total - NumErrorTape # Compute successful completed hosts backup to tape

SuccessRateTape = int(float(SuccessHostTape) / Total \* 100) # Compute successful percentage

SuccessRateTapeJob = 100 - (SuccessRate - SuccessRateTape) # Compute filure rate

else:

print "This domain has no tape library attached" # Message to user for this section

with open(ReportFileName,'w') as ReportFile: # Open the file in write mode

# Print Header

ReportFile.write("\*\*\* Netbackup report for " + Hostname + " \*\*\*\n")

if has\_library: # Some extra information is printed if we have a library

ReportFile.write("Tape library attached\n") # Note if library is attached or not

else:

ReportFile.write("No tape library attached\n")

ReportFile.write("\n")

# Host Count

ReportFile.write("Total Configured host - " + str(Total) + "\n") # Print overall statistic

# Library status

if has\_library: # Some extra information is printed if we have a library

ReportFile.write("\n") # Write the status of the library

ReportFile.write("\*\*\* Library Status \*\*\*\n")

status = library\_status()

for line in status: # Print the status of each library

ReportFile.write(line[0])

ReportFile.write("\n")

# Tape drive status

ReportFile.write("\n") # Buffer line

ReportFile.write("\*\*\* Tape Drive Status \*\*\*\n") # Tape drive status header

status = drive\_status() # Get the drive status from nblib.py

labels = ('Drive Name', 'Server', 'Dr#', 'Device', 'Status') # Set headers

ReportFile.write(indent([labels]+status, hasHeader=True)) # Format and write the contents of the report

# Scratch and space count

ReportFile.write("\n") # Buffer line

ReportFile.write("\*\*\* Tape Scratch and Space Count \*\*\*\n") # Tape numbers header

connections = get\_robot\_connections() # Get robot connection from nblibdr.py

for line in connections: # Read through each robot

if line[1] != "No": # No invalid robot found

# \*\*\* Inventory each library

library\_name = get\_library\_name (line[0], line[2]) # Send robot number, host name, get library name

ReportFile.write("Library " + library\_name + "\n") # Print library name header

do\_inventory (line[0], line[2]) # Run an inventory

# \*\*\* List scratch in each library

scratch = list\_scratch (line[0]) # Figure number of scrach in each TLD

ReportFile.write("Scratch " + str(scratch) + "\n") # Print number of scratch in that TLD

# \*\*\* List free slots of each library

f\_slots = free\_slots (line[0], line[2]) # Get number of free slots

ReportFile.write("Empty slots " + str(f\_slots) + "\n") # Print free slots

ReportFile.write("\n") # Buffer line

# Job status - long running jobs

ReportFile.write(job\_status) # Write the status of long running jobs

# Disk backup overview

ReportFile.write("\n") # Buffer line

ReportFile.write("\*\*\* Disk Backups \*\*\*\n") # Header for disk backups

ReportFile.write("Successful backed up hosts to disk- " + str(SuccessHost) + "\n") # Metrics for disk backups

ReportFile.write("Failed hosts - " + str(NumError) + "\n")

ReportFile.write("Host protection rate - " + str(SuccessRate) + "%\n")

ReportFile.write("Host at risk - " + str(FailureRate) + "%\n")

# Tape backup overview

if has\_library: # Some extra information is printed if we have a library

ReportFile.write("\n") # Buffer line

ReportFile.write("\*\*\* Tape Backups \*\*\n") # Header for tape backups

ReportFile.write("Successful backed up hosts to tape- " + str(SuccessHostTape) + "\n") # Metrics for tape backups

ReportFile.write("Failed hosts - " + str(NumErrorTape) + "\n")

ReportFile.write("Tape Protection Rate - " + str(SuccessRateTape) + "%\n")

ReportFile.write("Tape Job Success Rate - " + str(SuccessRateTapeJob) + "%\n")

# Disk backup failure

ReportFile.write("\n") # Print blank line

ReportFile.write("\*\*\* Disk Backup Failures \*\*\*\n") # Header for backup failures

ReportFile.write(indent(Error, hasHeader=True)) # Write formatted disk backup failure report

# Disk backup failure sorted by code

ReportFile.write("\n") # Print blank line

ReportFile.write("\*\*\* Disk Failure Status Code \*\*\*") # Header for disk backup failure sorted by status

ReportFile.write(StsCode + "\n") # Write formatted report

# Tape backup failures

if has\_library: # Some extra information is printed if we have a library

ReportFile.write("\n") # Print blank line

ReportFile.write("\*\*\* Tape Backup Failures \*\*\*\n") # Header for tape backup failures

ReportFile.write(indent(ErrorTape, hasHeader=True)) # Write formatted report for tape backup failures

# Disk backup logs

ReportFile.write("\n") # Print blank line

ReportFile.write("\*\*\* Disk Backup Logs \*\*\*\n") # Header for all disk backup jobs

ReportFile.write(indent(Full, hasHeader=True)) # Write formatted report for all disk backup jobs

# Tape backup logs

if has\_library: # Some extra information is printed if we have a library

ReportFile.write("\n") # Print blank line

ReportFile.write("\*\*\* Tape Backup Logs \*\*\*\n") # Header for tape backup jobs

ReportFile.write(indent(FullTape, hasHeader=True)) # Write formatted report for all tape backup jobs

# Version information

ReportFile.write("\n") # Print blank line

ReportFile.write(Version + "\n") # Print version number

############################################################################################

#

# SendEmail

# Send a mail the the recipients with the previously formatted body

#

# Input: None

# Output: Mail - File 'ReportFileName' sent to 'MailTo' Distribution List, both described in init

#

############################################################################################

def SendEmail():

print "Sending mail"

Header = "\"Windows NetBackup results for " + Hostname + "\"" # Title of the mail

cmd="mail -s " + Header + " " + MailTo + " < " + ReportFileName # Insert bp command to run here

print cmd

Output,Error=subprocess.Popen (cmd,shell=True, stdout = subprocess.PIPE, stderr= subprocess.PIPE).communicate() # Sends cmd to the subprocess

# Shell=True waits for the response and allows direct output.

# Example of sendmail vs mail. Allows for fixed width fonts

""" [root@epdc01nbadm01 ~]# (

> echo "From: tscarter@epdc01nbadm01";

> echo "To: terry.carter@oracle.com";

> echo "Subject: Testing Sendmail"

> echo "Content-Type: text/html";

> echo "MIME-Version: 1.0";

> cat /tmp/nbreportfile.tmp

> ) | sendmail -t

"""

############################################################################################

#

# Main

#

############################################################################################

### Overall Data

ClientsInPolicy, TotalNumClients = ListClients() # Get a list and number of clients.

has\_library = check\_policy\_library() # Does this domain have a library?

### Disk Data

FullOutArray, ErrorOutArray, NumErrorClients = LastBackup(ClientsInPolicy) # Get two arrays of all backups and those that failed

StsCodeErr = GetStsCode(ErrorOutArray) # Get a list of status code errors

### Get Job Information

job\_status = get\_jobs(24) # List jobs older than 24 hours

### Make it prety

MakeEmail(TotalNumClients, FullOutArray, ErrorOutArray, NumErrorClients, StsCodeErr, has\_library, job\_status) # Create file to email results

SendEmail() # Format and send the mail

print "cat " + ReportFileName

#end nbreport.py