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

#

# NAME: CONFORMITY

#

# AUTHOR: DAAS

#

# COMMENT: VDI - 3DEXP - Pools X - Pools W

#

# VERSION HISTORY:

# 1.0 16/01/2020 - Initial release

#

#

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

. "D:\\_DaaS\_Scheduled\_Tasks\Modules\Connect\_HvServer\_Param.ps1"

$VCtab = New-Object System.Collections.ArrayList

foreach ($broker in $brokers){

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

######### Connections Brokers #####################

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

$conn = Connect-HVServer -Server $broker -Credential $cred -Domain "eu" -Force

################################### Get vCenter ID #########

. "D:\\_DaaS\_Scheduled\_Tasks\Modules\Get-Vcenter\_from ID.ps1" ######### get-VcenterName par ID #####################

}

$Parents = New-Object System.Collections.ArrayList

$Masters = New-Object System.Collections.ArrayList

$Snapshots = New-Object System.Collections.ArrayList

$Naming = New-Object System.Collections.ArrayList

$EmptyFolders = New-Object System.Collections.ArrayList

$LinkedFolders = New-Object System.Collections.ArrayList

$MasterSpace = New-Object System.Collections.ArrayList

$WrongDNS = New-Object System.Collections.ArrayList

$DuplicateFolders = New-Object System.Collections.ArrayList

#$poolWArray = New-Object System.Collections.ArrayList

$ListParents = New-Object System.Collections.ArrayList

$CountParents = New-Object System.Collections.ArrayList

$IVM = New-Object System.Collections.ArrayList

$vmRecap = New-Object System.Collections.ArrayList

$ISO = New-Object System.Collections.ArrayList

$Parent = New-Object System.Collections.ArrayList

foreach ($vcenter in $VCtab) {

$vc = Connect-VIServer $vcenter.name -Credential $cred

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

######################### SNAPSHOTS #####################

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

Write-Host "Getting Old Snaphots from" $Vcenter.name

#$ParentsSnapshots = get-vm -server $vc -Location "parents" | Get-Snapshot

#foreach ($ParentsSnapshot in $ParentsSnapshots)

#{

#if ($ParentsSnapshot.created -lt (get-date).AddDays(-150))

#{

# $Age = (get-date) - ($ParentsSnapshot.created)

# $Obj = New-Object PSCustomObject

# $Obj | add-member -MemberType NoteProperty -name "vCenter" -Value $vCenter.name

# $Obj | add-member -MemberType NoteProperty -name "Type" -Value "Parent"

# $Obj | add-member -MemberType NoteProperty -name "VM" -Value $ParentsSnapshot.vm

# $Obj | add-member -MemberType NoteProperty -name "snapshot" -Value $ParentsSnapshot.name

# $Obj | add-member -MemberType NoteProperty -name "Creation Date" -Value $ParentsSnapshot.created

# $Obj | add-member -MemberType NoteProperty -name "Age" -Value $Age.days

# [void]$Snapshots.add($Obj)

#}

#}

if (!($vcenter.name -match "fr0-vsdaas-p130.eu.airbus.corp" -or $vcenter.name -match "fr0-vsdaas-p131.eu.airbus.corp"))

{

write-host "pris en compte"

$MastersSnapshots = get-vm -server $vc -Location "masters" | Get-Snapshot

foreach ($MastersSnapshot in $MastersSnapshots)

{

if ($MastersSnapshot.created -lt (get-date).AddDays(-30))

{

$Age = (get-date) - ($MastersSnapshot.created)

$Obj = New-Object PSCustomObject

$Obj | add-member -MemberType NoteProperty -name "vCenter" -Value $vCenter.name

$Obj | add-member -MemberType NoteProperty -name "Type" -Value "Master"

$Obj | add-member -MemberType NoteProperty -name "VM" -Value $MastersSnapshot.vm

$Obj | add-member -MemberType NoteProperty -name "snapshot" -Value $MastersSnapshot.name

$Obj | add-member -MemberType NoteProperty -name "Creation Date" -Value $MastersSnapshot.created

$Obj | add-member -MemberType NoteProperty -name "Age" -Value $Age.days

[void]$Snapshots.add($Obj)

}

}

$DesktopsSnapshots = get-vm -server $vc -Location "desktops" | Get-Snapshot

foreach ($DesktopsSnapshot in $DesktopsSnapshots)

{

$Age = (get-date) - ($DesktopsSnapshot.created)

$Obj = New-Object PSCustomObject

$Obj | add-member -MemberType NoteProperty -name "vCenter" -Value $vCenter.name

$Obj | add-member -MemberType NoteProperty -name "Type" -Value "Desktop"

$Obj | add-member -MemberType NoteProperty -name "VM" -Value $DesktopsSnapshot.vm

$Obj | add-member -MemberType NoteProperty -name "snapshot" -Value $DesktopsSnapshot.name

$Obj | add-member -MemberType NoteProperty -name "Creation Date" -Value $DesktopsSnapshot.created

$Obj | add-member -MemberType NoteProperty -name "Age" -Value $Age.days

[void]$Snapshots.add($Obj)

}

}

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

###################### POWER Status #####################

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

Write-Host "Getting PoweredOn Parents"

$ParentsPoweredOn = get-vm -server $vc -Location "parents" | Where-Object PowerState -match "PoweredOn"

foreach ($ParentPoweredOn in $ParentsPoweredOn)

{

$lastboot = ($ParentPoweredOn | Get-View).runtime.boottime

$Age = (get-date) - ($lastboot)

$Obj = New-Object PSCustomObject

$Obj | add-member -MemberType NoteProperty -name "vCenter" -Value $vCenter.name

$Obj | add-member -MemberType NoteProperty -name "Type" -Value "Parent"

$Obj | add-member -MemberType NoteProperty -name "VM" -Value $ParentPoweredOn.name

$Obj | add-member -MemberType NoteProperty -name "Last Boot Date" -Value $lastboot

$Obj | add-member -MemberType NoteProperty -name "Age" -Value $Age.days

if ($Age.days -gt 0)

{

[void]$Parents.add($Obj)

}

}

Write-Host "Getting PoweredOff Masters"

$MastersPoweredOn = get-vm -server $vc -Location "masters" | Where-Object PowerState -match "PoweredOff"

foreach ($MasterPoweredOn in $MastersPoweredOn)

{

$Obj = New-Object PSCustomObject

$Obj | add-member -MemberType NoteProperty -name "vCenter" -Value $vCenter.name

$Obj | add-member -MemberType NoteProperty -name "Type" -Value "Master"

$Obj | add-member -MemberType NoteProperty -name "VM" -Value $MasterPoweredOn.name

$Obj | add-member -MemberType NoteProperty -name "Error" -Value "Master Powered Off"

[void]$Masters.add($Obj)

}

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

###################### TYPOLOGIE #####################

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

write-host "Getting Wrongly named VMs"

$namingDesktops = get-vm -server $vc -Location "Desktops" | Where-Object name -NotMatch "[EI][PT]V\w{7}[A-Y]\d{3}"

foreach ($namingDesktop in $namingDesktops)

{

$Obj = New-Object PSCustomObject

$Obj | add-member -MemberType NoteProperty -name "vCenter" -Value $vCenter.name

$Obj | add-member -MemberType NoteProperty -name "Type" -Value "Desktop"

$Obj | add-member -MemberType NoteProperty -name "VM" -Value $namingDesktop.name

$Obj | add-member -MemberType NoteProperty -name "Error" -Value "Wrong Pattern"

[void]$Naming.add($Obj)

}

$namingMasters = get-vm -server $vc -Location "Masters" | Where-Object{ ($\_.Name -notmatch "[EI]PV\w{6}Z\d{3}") -and ($\_.Name -notmatch "GM.\*") -and ($\_.Name -notmatch "[EI]PV\w{6}[A-W]000")}

foreach ($namingMaster in $namingMasters)

{

$Obj = New-Object PSCustomObject

$Obj | add-member -MemberType NoteProperty -name "vCenter" -Value $vCenter.name

$Obj | add-member -MemberType NoteProperty -name "Type" -Value "Master"

$Obj | add-member -MemberType NoteProperty -name "VM" -Value $namingMaster.name

$Obj | add-member -MemberType NoteProperty -name "Error" -Value "Wrong Pattern"

[void]$Naming.add($Obj)

}

$namingParents = get-vm -server $vc -Location "Parents" | Where-Object name -NotMatch "\w{7}[A-Y] - [EI]PV\w{6}Z\d{3} - \d{6}"

foreach ($namingParent in $namingParents)

{

$Obj = New-Object PSCustomObject

$Obj | add-member -MemberType NoteProperty -name "vCenter" -Value $vCenter.name

$Obj | add-member -MemberType NoteProperty -name "Type" -Value "Parent"

$Obj | add-member -MemberType NoteProperty -name "VM" -Value $namingParent.name

$Obj | add-member -MemberType NoteProperty -name "Error" -Value "Wrong Pattern"

[void]$Naming.add($Obj)

}

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

###################### FOLDERS #########################

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

write-host "Getting Duplicate Folders"

$DuplicateVMFolders = get-folder | Where-Object {($\_.parent -match $\_.name) -and ($\_.type -eq 'vm')}

foreach ($DuplicateVMFolder in $DuplicateVMFolders)

{

$Obj = New-Object PSCustomObject

$Obj | add-member -MemberType NoteProperty -name "vCenter" -Value $vCenter.name

$Obj | add-member -MemberType NoteProperty -name "Folder" -Value $DuplicateVMFolder.name

$Obj | add-member -MemberType NoteProperty -name "Child VMs" -Value $DuplicateVMFolder.ExtensionData.ChildEntity.Count

$Obj | add-member -MemberType NoteProperty -name "Error" -Value "Duplicate Folder"

[void]$DuplicateFolders.add($Obj)

}

Write-host "Getting Orphan Folders"

$MasterFolders = Get-Folder -Location "Masters"

$ParentFolders = Get-Folder -Location "Parents"

$DesktopFolder = Get-Folder -Location "Desktops"

$DesktopFolders = @()

foreach ($item in $DesktopFolder.name)

{

if($item.length -ge 6)

{

$DesktopFolders += $item.substring(0,6)

}

}

if ($MasterFolders.Count -gt 0) {

$MastersToParents = Compare-Object -ReferenceObject $MasterFolders.name -DifferenceObject $ParentFolders.name

foreach ($item in $MastersToParents)

{

switch ($item.SideIndicator)

{

"<="

{

$Obj = New-Object PSCustomObject

$Obj | add-member -MemberType NoteProperty -name "vCenter" -Value $vCenter.name

$Obj | add-member -MemberType NoteProperty -name "Type" -Value "Master"

$Obj | add-member -MemberType NoteProperty -name "VM" -Value $item.InputObject

$Obj | add-member -MemberType NoteProperty -name "Error" -Value "No Parent folder linked"

[void]$LinkedFolders.add($Obj)

}

"=>"

{

$Obj = New-Object PSCustomObject

$Obj | add-member -MemberType NoteProperty -name "vCenter" -Value $vCenter.name

$Obj | add-member -MemberType NoteProperty -name "Type" -Value "Parent"

$Obj | add-member -MemberType NoteProperty -name "VM" -Value $item.InputObject

$Obj | add-member -MemberType NoteProperty -name "Error" -Value "No Master folder linked"

[void]$LinkedFolders.add($Obj)

}

}

}

$MastersToDesktops = Compare-Object -ReferenceObject $MasterFolders.name -DifferenceObject $DesktopFolders

foreach ($item in $MastersToDesktops)

{

if ($item.SideIndicator -match "<=" -and $item.InputObject -notmatch "GPMINI")

{

$Obj = New-Object PSCustomObject

$Obj | add-member -MemberType NoteProperty -name "vCenter" -Value $vCenter.name

$Obj | add-member -MemberType NoteProperty -name "Type" -Value "Master"

$Obj | add-member -MemberType NoteProperty -name "VM" -Value $item.InputObject

$Obj | add-member -MemberType NoteProperty -name "Error" -Value "No Desktop folder linked"

#[void]$LinkedFolders.add($Obj)

}

}

$ParentsToDesktops = Compare-Object -ReferenceObject $ParentFolders.name -DifferenceObject $DesktopFolders

foreach ($item in $ParentsToDesktops)

{

if ($item.SideIndicator -match "<=" -and $item.InputObject -notmatch "GPMINI")

{

$Obj = New-Object PSCustomObject

$Obj | add-member -MemberType NoteProperty -name "vCenter" -Value $vCenter.name

$Obj | add-member -MemberType NoteProperty -name "Type" -Value "Parent"

$Obj | add-member -MemberType NoteProperty -name "VM" -Value $item.InputObject

$Obj | add-member -MemberType NoteProperty -name "Error" -Value "No Desktop folder linked"

[void]$LinkedFolders.add($Obj)

}

}

}

write-host "Getting Empty Folders"

$EmptyMasters = $MasterFolders | Where-Object {$\_.extensiondata.childentity.count -eq 0 -and $\_.parent -match "Masters" }

foreach ($EmptyMaster in $EmptyMasters)

{

$Obj = New-Object PSCustomObject

$Obj | add-member -MemberType NoteProperty -name "vCenter" -Value $vCenter.name

$Obj | add-member -MemberType NoteProperty -name "Type" -Value "Master"

$Obj | add-member -MemberType NoteProperty -name "VM" -Value $EmptyMaster.name

$Obj | add-member -MemberType NoteProperty -name "Error" -Value "Empty Folder"

[void]$EmptyFolders.add($Obj)

}

$EmptyParents = $ParentFolders | Where-Object {$\_.extensiondata.childentity.count -eq 0 -and $\_.parent -match "Parents" }

foreach ($EmptyParent in $EmptyParents)

{

$Obj = New-Object PSCustomObject

$Obj | add-member -MemberType NoteProperty -name "vCenter" -Value $vCenter.name

$Obj | add-member -MemberType NoteProperty -name "Type" -Value "Parent"

$Obj | add-member -MemberType NoteProperty -name "VM" -Value $EmptyParent.name

$Obj | add-member -MemberType NoteProperty -name "Error" -Value "Empty Folder"

[void]$EmptyFolders.add($Obj)

}

$EmptyDesktops = $DesktopFolders | Where-Object {$\_.extensiondata.childentity.count -eq 0 -and $\_.parent -match "Desktops" }

foreach ($EmptyDesktop in $EmptyDesktops)

{

$Obj = New-Object PSCustomObject

$Obj | add-member -MemberType NoteProperty -name "vCenter" -Value $vCenter.name

$Obj | add-member -MemberType NoteProperty -name "Type" -Value "Desktop"

$Obj | add-member -MemberType NoteProperty -name "VM" -Value $EmptyDesktop.name

$Obj | add-member -MemberType NoteProperty -name "Error" -Value "Empty Folder"

[void]$EmptyFolders.add($Obj)

}

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

###################### Disk Issue #########################

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

write-host "Getting Lack Of Free Space"

$VmMasters = get-vm -server $vc -Location 'Masters' | Where-Object {($\_.Guest.Disks.FreeSpaceGB -lt 10) -and ($\_.Guest.Disks.path -match "C:\\")}

foreach ($VmMaster in $VmMasters)

{

$Obj = New-Object PSCustomObject

$Obj | add-member -MemberType NoteProperty -name "vCenter" -Value $vCenter.name

$Obj | add-member -MemberType NoteProperty -name "Type" -Value "Master"

$Obj | add-member -MemberType NoteProperty -name "VM" -Value $VmMaster.name

$Obj | add-member -MemberType NoteProperty -name "CapacityGB" -Value $VmMaster.Guest.Disks.CapacityGB

$Obj | add-member -MemberType NoteProperty -name "FreeSpaceGB" -Value $VmMaster.Guest.Disks.FreeSpaceGB

$Obj | add-member -MemberType NoteProperty -name "OS" -Value $VmMaster.guestid

$Obj | add-member -MemberType NoteProperty -name "Error" -Value "Space Left"

[void]$MasterSpace.add($Obj)

}

$VmDesktops = get-vm -server $vc -Location 'Desktops' | Where-Object {($\_.Guest.Disks.FreeSpaceGB -lt 10) -and ($\_.Guest.Disks.path -match "C:\\") -and ($\_.name -match "[E,I]PV.\*") -and ($\_.name -notmatch "[EI]PV\w{7}Z\d{3}")}

foreach ($VmDesktop in $VmDesktops)

{

$Obj = New-Object PSCustomObject

$Obj | add-member -MemberType NoteProperty -name "vCenter" -Value $vCenter.name

$Obj | add-member -MemberType NoteProperty -name "Type" -Value "Desktop"

$Obj | add-member -MemberType NoteProperty -name "VM" -Value $VmDesktop.name

$Obj | add-member -MemberType NoteProperty -name "CapacityGB" -Value $VmDesktop.Guest.Disks.CapacityGB

$Obj | add-member -MemberType NoteProperty -name "FreeSpaceGB" -Value $VmDesktop.Guest.Disks.FreeSpaceGB

$Obj | add-member -MemberType NoteProperty -name "OS" -Value $VmDesktop.guestid

$Obj | add-member -MemberType NoteProperty -name "Error" -Value "Space Left"

[void]$DesktopSpace.add($Obj)

}

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

###################### Wrong DNS #########################

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

write-host "Getting wrong DNS name"

$VMSDNS = get-vm -server $vc | Where-Object {(![string]::IsNullOrWhitespace($\_.guest.hostname)) -and ($\_.name -notmatch "GM.\*") -and ($\_.PowerState -match "PoweredOn") -and ($\_.name -notmatch "vCLS-\*")}

foreach ($VMDNS in $VMSDNS)

{

if ($VMDNS.name -notmatch $VMDNS.guest.Hostname.split('.')[0])

{

$Obj = New-Object PSCustomObject

$Obj | add-member -MemberType NoteProperty -name "vCenter" -Value $vCenter.name

$Obj | add-member -MemberType NoteProperty -name "VM" -Value $VMDNS.name

$Obj | add-member -MemberType NoteProperty -name "DNS name" -Value $VMDNS.guest.Hostname

$Obj | add-member -MemberType NoteProperty -name "Error" -Value "DNS and VM name match"

[void]$WrongDNS.add($Obj)

}

}

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

###################### Pools W #########################

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

<#

write-host "Getting pool W"

$poolW = get-vm -server $vc -Location "desktops" | Where-Object name -Match "[EI][TP]V\w{7}W\d{3}"

foreach($pool in $poolW)

{

$matches = [regex]::Matches($pool.Name, "[EI][PT]V(\w{7}W)\d{3}")

$p = $matches.Groups[1].Value

if($poolWArray.Pool -notcontains $p)

{

$Obj = New-Object PSCustomObject

$Obj | add-member -MemberType NoteProperty -name "vCenter" -Value $vCenter.name

$Obj | add-member -MemberType NoteProperty -name "Pool" -Value $p

[void]$poolWArray.add($Obj)

}

}

#>

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

########### If More than 2 Parents ######################

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

write-host "Checking if more than 3 Parents"

$AllParents = Get-vm -server $vc -Location "Parents" | Where-Object Name -notlike "\*GPMINI\*"

foreach ($par in $AllParents) {

$Obj = New-Object PSCustomObject

$Obj | add-member -MemberType NoteProperty -name "Pool" -Value $Par.name.substring(0,8)

$Obj | add-member -MemberType NoteProperty -name "VM" -Value $Par.name

$Obj | add-member -MemberType NoteProperty -name "Date" -Value $Par.name.substring(26,6)

$Obj | add-member -MemberType NoteProperty -name "vCenter" -Value $vCenter.name

[void]$ListParents.add($Obj)

}

$CountParents = $ListParents | Group-Object -Property pool,vcenter | Where-Object {$\_.count -gt 2} | Select count, name | Sort count

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

###Get Invalid VM Vcenter

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

$VMs = (Get-View -ViewType VirtualMachine)| ?{$\_.Runtime.ConnectionState -eq "invalid" -or $\_.Runtime.ConnectionState -eq "inaccessible" -or $\_.Runtime.ConnectionState -eq "orphaned"}

Foreach ($VM in $VMs) {

$InvalidMachines = New-Object PSCustomObject

$InvalidMachines | add-member -MemberType NoteProperty -name "name" -Value $VM.name

$InvalidMachines | add-member -MemberType NoteProperty -name "Status" -Value $VM.Runtime.connectionState

$InvalidMachines | add-member -MemberType NoteProperty -name "Vcenter" -Value $Vcenter

[void]$IVM.add($InvalidMachines)

}

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

###Get e1000\* Adapter

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

$vms = Get-VM -Location "Masters","Desktops" | where-object {($\_.name -match "^[EI]PV.\*")-and ($\_.GuestId -match "^windows.\*") -or ($\_.name -match "GM\*")}

foreach ($vm in $vms) {

$vmAdapter = Get-NetworkAdapter -VM $vm | select -ExpandProperty type

$obj = New-Object PSCustomObject

$obj | Add-Member -type NoteProperty -Name VM -Value $vm

$obj | Add-Member -type NoteProperty -Name Vcenter -Value $vcenter

$obj | Add-Member -type NoteProperty -Name Adapter -Value $vmAdapter[0]

[void]$vmRecap.add($obj)

}

DisConnect-VIServer $vcenter.name -Confirm:$false

}

DisConnect-VIServer -server $vctab.name -Confirm:$false

$ADWrongOU = Get-ADComputer -SearchBase "CN=Computers,DC=eu,DC=airbus,DC=corp" -filter \* | Where-Object {($\_.name -match "[EI][PT]V\w{7}[A-Z]\d{3}" -or $\_.name -match "[EI][PT]V\w{6}[A-Z]\d{3}" -or $\_.name -match "[EI]PVCAE(FR|DE)[BTFW][A-Z]\d{3}" -or $\_.name -match "CPV[A-Z]\d{3}\w{4}([VF])\d{3}" -or $\_.name -match "(DE|FR)0-(VS|CS|CS\w{1})DAAS-p\d+") }

$ADWrongOU += Get-ADComputer -SearchBase "OU=VDI,OU=Others,OU=Workstations,DC=eu,DC=airbus,DC=corp" -filter \* -SearchScope OneLevel | Where-Object {($\_.name -match "[EI][PT]V\w{7}[A-Z]\d{3}" -or $\_.name -match "[EI][PT]V\w{6}[A-Z]\d{3}" -or $\_.name -match "[EI]PVCAE(FR|DE)[BTFW][A-Z]\d{3}" -or $\_.name -match "CPV[A-Z]\d{3}\w{4}([VF])\d{3}" -or $\_.name -match "(DE|FR)0-(VS|CS|CS\w{1})DAAS-p\d+") }

$vme1000 = $vmrecap | Where-Object -property Adapter -NE "Vmxnet3"

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

######### Mail et LOG ##############################################

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

$date = Get-Date -format dd-MM-yyyy

#$date =(Get-Date).AddDays(-1).ToString('dd-MM-yyyy')

$RootFolder = "D:\\_DaaS\_Scheduled\_Tasks\Conformity\Logs\"

$RootFolder1 = "D:\Scripts\LGU\ISO\_Parent\_replica\"

$SnapshotsFile = $RootFolder + $date + "\_Snapshots.csv"

$ParentsFile = $RootFolder + $date + "\_Parent\_PoweredOn.csv"

$MastersFile = $RootFolder + $date + "\_Master\_PoweredOff.csv"

$NamingFile = $RootFolder + $date + "\_Naming.csv"

$EmptyFoldersFile = $RootFolder + $date + "\_Empty\_Folders.csv"

$LinkedFoldersFile = $RootFOlder + $date + "\_Linked\_Folders.csv"

$MasterSpaceFile = $RootFolder + $date + "\_Master\_Space.csv"

$DesktopSpaceFile = $RootFolder + $date + "\_Desktop\_Space.csv"

$WrongDNSFile = $RootFolder + $date + "\_Wrong\_DNS.csv"

$DuplicateFoldersFile = $RootFolder + $date + "\_Duplicate\_Folders.csv"

$ADWrongOUFile = $RootFolder + $date + "\_AD\_Wrong\_OU.csv"

#$poolWFile = $RootFolder + $date + "\_poolW.csv"

$CountParentsFile = $RootFolder + $date + "\_More\_Than\_2\_Parents.csv"

$IVMFile = $RootFolder + $date + "\_Invalid\_Vms.csv"

$vme1000File = $RootFolder + $date + "\_e1000\_Vms.csv"

$ISO = $RootFolder1 + $date + "-Machines\_Attached\_ISO.csv"

$Parent = $RootFolder1 + $date + "-Parent\_Replica\_in\_POD.csv"

$Snapshots| sort $\_.Type | Export-csv $SnapshotsFile -Delimiter ';' -NoTypeInformation

$Parents | Export-csv $ParentsFile -Delimiter ';' -NoTypeInformation

$Masters | Export-csv $MastersFile -Delimiter ';' -NoTypeInformation

$Naming | Export-csv $NamingFile -Delimiter ';' -NoTypeInformation

$EmptyFolders | Export-csv $EmptyFoldersFile -Delimiter ';' -NoTypeInformation

$LinkedFolders | Export-csv $LinkedFoldersFile -Delimiter ';' -NoTypeInformation

$MasterSpace | Export-csv $MasterSpaceFile -Delimiter ';' -NoTypeInformation

$DesktopSpace | Export-csv $DesktopSpaceFile -Delimiter ';' -NoTypeInformation

$WrongDNS | Export-csv $WrongDNSFile -Delimiter ';' -NoTypeInformation

$DuplicateFolders | Export-csv $DuplicateFoldersFile -Delimiter ';' -NoTypeInformation

$ADWrongOU | Select-Object name| Export-csv $ADWrongOUFile -Delimiter ';' -NoTypeInformation

#$poolWArray | Export-csv $poolWFile -Delimiter ';' -NoTypeInformation

$CountParents | Export-csv $CountParentsFile -Delimiter ';' -NoTypeInformation

$IVM | Export-Csv -Path $IVMFile -Delimiter ';' -NoTypeInformation

$vme1000 | Export-Csv -Path $vme1000File -Delimiter ';' -NoTypeInformation

$smtp = "smtp.airbus.corp"

$smtpDev = "novmail.ham.de.eu.airbus.corp"

$from = "Conformity <no-reply@airbus.com>"

$to = "daas.support@airbus.com", "daas.infra@airbus.com", "daas.specialops@airbus.com","daas.3dx@airbus.com"

#$to = "srinath.vasanthan.external@airbus.com"

#$to = "gokulkannan.krishnamoorthi.external@airbus.com"

$subject = "DaaS 2.0 CONFORMITY - "+$date

$Body = @"

Conformity Daas 2.0

REPORT :

Snapshots : $($Snapshots.count) errors / warnings

Parent\_PoweredOn : $($Parents.count) errors

Master\_PoweredOff : $($Masters.count) errors

Naming : $($Naming.count) errors

Empty\_Folders : $($EmptyFolders.count) errors

Linked\_Folders : $($LinkedFolders.count) errors

Master\_Space : $($MasterSpace.count) errors / warnings

Desktop\_Space : $($DesktopSpace.count) errors / warnings

Wrong\_DNS : $($WrongDNS.count) errors

Duplicate\_Folders : $($DuplicateFolders.count) errors

AD\_Wrong\_OU : $($ADWrongOU.count) errors

CountParent : $($CountParents.count) errors

Invalid VMs : $($IVM.count) errors

e1000 Vms : $($vme1000.count) errors

ISO attached Vms : $($ISO.count) errors

Parent & Replica Vms : $($Parent.count) errors

DEFINITION :

Snapshots : Snapshots that shouldn't exist anymore, with the following rules, desktops > 0 days, masters > 30 days, parents > 90 days

Parent\_PoweredOn : Parents powered On, state that shouldn't exist in RUN mode

Master\_PoweredOff : Masters powered Off, state that shouldn't exist in RUN mode

Naming : VD which aren't matching DaaS naming pattern

Empty\_Folders : Folders which are empty and therefore not used anymore

Linked\_Folders : All Standard Dynamic pools must have a Pool/Parent/Master set of folders, the report show the folders where the chain is broken

Master\_Space : Masters where the space is below 10 GB

Desktop\_Space : Desktop where the space is below 10 GB

Wrong\_DNS : VD where the VM name doesn't match its DNS name

Duplicate\_Folders : reports the pools where a pool folder is misplaced in its own folder, EX : Desktops/ESPAG1A/ESPAG1A/ETVESPAG1Axxx

AD\_Wrong\_OU : Computers / Servers placed in the wrong OU

CountParent : Pools with more than 2 Parents

Invalid VMs : VMs on vcenter with Status Invalid, Inaccessible or Orphaned

e1000 Vms : Machines with adapter type other than Vmxnet3 Type

ISO attached Vms : Machines attached with ISO

Parent & Replica : Parent and Replica VMs in Horizon console

"@

send-MailMessage -SmtpServer $smtp -To $to -Attachments $SnapshotsFile, $ParentsFile, $MastersFile, $NamingFile, $EmptyFoldersFile, $LinkedFoldersFile, $MasterSpaceFile, $DesktopSpaceFile, $WrongDNSFile, $DuplicateFoldersFile, $ADWrongOUFile, $CountParentsFile, $IVMFile, $vme1000File, $ISO, $Parent -From $from -Subject $subject -Body $Body