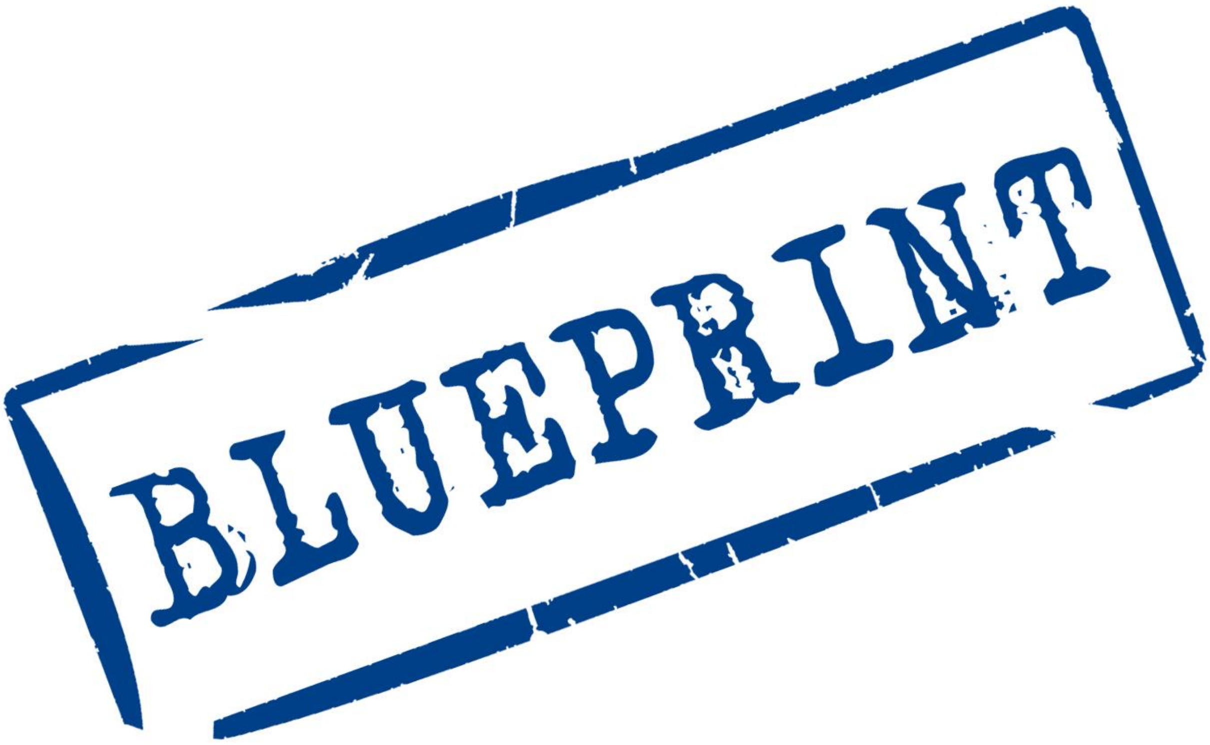
ukcloud

Pure commitment.

Programmatic Control of the UKCloud platform



UKC-GEN-125

INTRODUCTION

The autonomous control of highly elastic infrastructure enables customers to experience unparalleled agility through the use of cloud services. Compared to traditional hosting solutions, customers using UKCloud's cloud services benefit from true utility pricing (per VM per hour) which means customers only pay for the resources that they are consuming. Before cloud, customers would have to size their solution for peak demand, and once the capex investment is made there is no benefit to dynamically rightsizing workloads. Cloud provides a real commercial incentive for customers to turn off virtual machines when they're not required and scale back solutions when peak demand subsides.

One of the challenges which customers face in this agile and elastic world is how they will control and manage such a dynamically changing environment. It would not be feasible to rely on operational staff to manually reconfigure the cloud platform in response to changing requirements – for example, manually starting and stopping virtual machines via the UKCloud Portal. Instead, customers need to consider automated solutions which can dynamically respond to changing requirements by programmatically driving the UKCloud cloud platform.

UKCloud facilitate this by providing an Application Programming Interface (API) which customers can use to implement programmatic control of the UKCloud cloud platform. Our API is based on VMware vCloud Director and so benefits from being well documented, used by various other Cloud Service Providers and natively supported by a wide number of applications and toolsets. Customers can leverage this wide support and comprehensive documentation to quickly and simply create their own scripts to control various aspects of the UKCloud cloud platform.

The main purpose of this Blueprint is to provide an example that illustrates to customers how easy it is to implement and benefit from an automated environment. The examples used in this Blueprint can apply to a number of use-cases such as:

- Automatically turning off test and development environments at night and at weekends. Doing so can reduce the cost of the solution by over two-thirds compared with running the virtual machines 24/7.
- Automatically scaling down parts of a citizen facing application that is predominantly used during the day, perhaps 0800-1800 including weekends. While the core application is required 24/7, aspects of it could be scaled down, such as running fewer web servers when demand is low.

IN THIS BLUEPRINT ▶

INTRODUCING POWERCLI AND POWERSHELL

Since 2006, Microsoft has provided Windows PowerShell, a task automation and configuration management framework, consisting of a command line shell and scripting language built on the .NET framework. While Windows has always provided a powerful graphical user interface, many routine tasks need to be automated, and PowerShell provides that capability.

PowerShell provides a number of .NET classes, called cmdlets, which implement particular operations. VMware has introduced additional cmdlets to PowerShell called PowerCLI, which map to the vCloud Director API functions – building blocks of UKCloud cloud. These extensions make it easy for administrators to control their UKCloud cloud environment in a programmatic manner. It should be noted however that not all vCloud Director API functions have corresponding PowerCLI cmdlets. In response to this, both VMware and contributors from the community are regularly adding additional cmdlets to address common requirements.

For the purpose of this Blueprint, PowerCLI has been used as a way to automate activities within the UKCloud cloud platform. PowerCLI mandates the use of a Microsoft Windows operating system. For Linux users, UKCloud exposes the API (based on vCloud Director) via the UKCloud Portal. This API can be addressed via a number of modern language SDKs, but their use is not covered here.

At the vCloud Director level, organisation users are able to manipulate entities within their own organisation, without necessarily being an organisation Administrator or Cloud Administrator. Functionality will be limited by their username and role/privilege.

INSTALLING POWERCLI

VMware PowerCLI is available from the VMware website at http://vmware.com/go/powercli

Having installed PowerCLI, users will soon discover that by default, Microsoft Windows does not allow the execution of non-signed scripts. This is a standard security policy which can be amended for appropriate requirements. It is necessary to run a command to set a registry setting to allow scripts to be executed, depending on your security requirements.

- Set-ExecutionPolicy Unrestricted
  - Will allow unsigned PowerShell scripts to run.
- Set-ExecutionPolicy Restricted
  - Will not allow unsigned PowerShell scripts to run.
- Set-ExecutionPolicy RemoteSigned
  - Will allow only remotely signed PowerShell scripts to run.

These commands can be run directly from the PowerShell command line.

To start a PowerShell window, use the Windows Start button and type 'powershell' into the command box.

Customers might also wish to consider Quest PowerGUI. Quest PowerGUI is a free application which facilitates developing or editing PowerShell and PowerCLI scripts by providing visibility of all the arrays and variables you are using, and simplifying debugging options. Quest PowerGUI is available at http://www.quest.com/powergui-freeware/

INTRODUCING THE SCRIPTS

Now that you have installed PowerCLI, we can move onto the actual scripts. In all the scripts in this document, it is necessary to enter customer specific information for the UKCloud cloud platform such as organisation name, username and password. These have been removed for security, but it is clear where these are required.

Logic behind the scripts

There are numerous ways that users can automate and manipulate virtual machines (or vApps) based on certain criteria. As a simple demonstration of this, for this Blueprint automated control will be based on:

- Day of the week to action on
- Hour to start a vApp
- Hour to stop a vApp
- The actual minute within the hour will depend on when the script is run

In order to decide which vApps to action on, this Blueprint uses the capability of a vApp to store metadata about itself, through the use of 'Key/Value' pairs. These can be set either through the GUI (shown below) or by using one of the scripts provided in this Blueprint.

The Key/Value pairs used for the scripts are shown in the table below.

Key	Value	Usage examples
AutoOnOff	Yes or No	Setting to No would 'disable' start/stop
StartTime	24 hour clock hour value	07 for 0700, 19 for 1900, 12 for midday
StopTime	24 hour clock hour value	07 for 0700, 19 for 1900, 12 for midday
Days	Day of week as three letters	Mon, Tue, Fri, Sat, Sun, Every

Setting the metadata on the vApps

In order for the scripts to work, they will rely on metadata being created as above. There are two ways to do this.

1. Use the UKCloud Portal to add the metadata manually to each vApp OR

2. Run the script to write out the current values (even if there aren't any) to a CSV

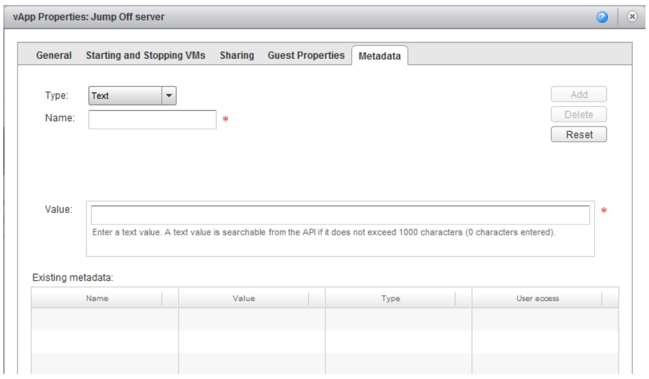
   a. Update the spreadsheet with your requirements for start/stop

   b. Run the script that will read a CSV file (spreadsheet) and apply the metadata that way

Using the UKCloud Portal to set the metadata

To set the metadata via the UKCloud Portal:

1. Log in to the UKCloud Portal
2. Select your vCloud organisation
3. For each vApp you want to enable this functionality for:
   a. Right-click on the vApp
   b. Select Properties. A panel will open
   c. Select the 'Metadata' tab at the top right.


vApp Properties: Jump Off server

General | Starting and Stopping VMs | Sharing | Guest Properties | Metadata

Type: Text

Name: *

Value: *

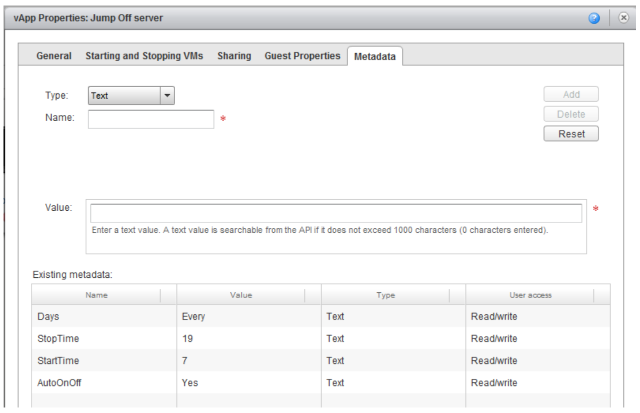
Enter a text value. A text value is searchable from the API if it does not exceed 1000 characters (0 characters entered).

Existing metadata:

Name	Value	Type	User access

In most cases, there will be no metadata present, but be cognisant that other users may be using metadata for other purposes.

d. In the ‘Name’ box enter the Key name ‘AutoOnOff’
e. In the ‘Value’ box enter ‘Yes’.
f. Leave the ‘Type’ as ‘Text’.
g. Then click ‘Add’.

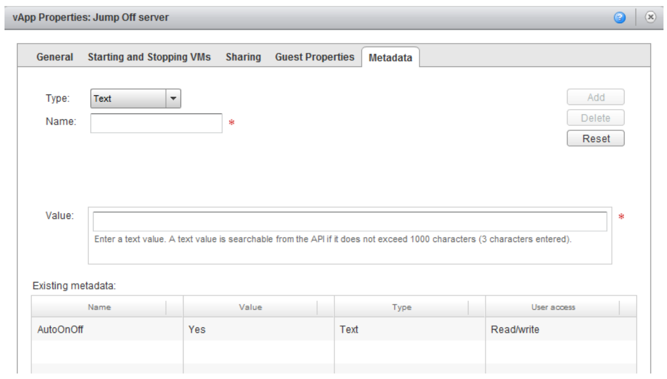

vApp Properties: Jump Off server

Type: Text
Name:
Value:
Enter a text value. A text value is searchable from the API if it does not exceed 1000 characters (0 characters entered).

Existing metadata:

Name	Value	Type	User access
Days	Every	Text	Read/write
StopTime	19	Text	Read/write
StartTime	7	Text	Read/write
AutoOnOff	Yes	Text	Read/write

Now, do the same for ‘StartTime’, ‘StopTime’ and ‘Days’ as in the table above. When finished, the panel should look as shown below


vApp Properties: Jump Off server

Type: Text
Name:
Value:
Enter a text value. A text value is searchable from the API if it does not exceed 1000 characters (3 characters entered).

Existing metadata:

Name	Value	Type	User access
AutoOnOff	Yes	Text	Read/write

Using scripts to programmatically set the metadata

Two scripts have been developed to:

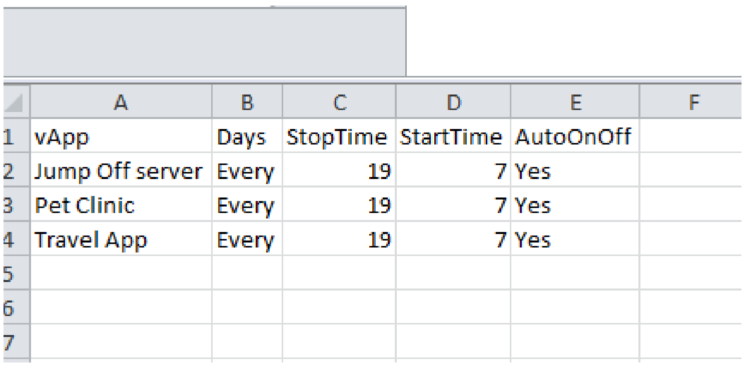
- Read the metadata from the vApp and write out to a CSV file — starts on page 13
- Read a CSV file and update the metadata of the vApp — starts on page 16

These are used within your PowerShell application.

The script for reading metadata from the vApp and writing it out to a CSV file will produce a CSV file as below.

This spreadsheet can then be edited to reflect when to start or stop the vApps.

Remember to save the file as a CSV for subsequent reading by the next script.



	A	B	C	D	E	F
1	vApp	Days	StopTime	StartTime	AutoOnOff	
2	Jump Off server	Every	19	7	Yes	
3	Pet Clinic	Every	19	7	Yes	
4	Travel App	Every	19	7	Yes	
5						
6						
7						

The main start/stop script

The main script will read the metadata from the vApp, and take action appropriately. Generally, it will take action retrospectively. For example, if it is run at 0900 and finds vApps that should have been started at 0700, and are still powered off, it will start them. Similarly, if it is run at 2100 and finds vApps that should have been powered off at 1900 and are still on, it will power them off.

It will also always look for ‘AutoOnOff’ to be set to ‘Yes’ before taking action. Using this Key/Value pair, a vApp could be made to be ignored by the script without the need to delete all the Key/Value pairs.

The main stop/start script is available on page 16.

ADDITIONAL SCRIPTS

Powering all vApps off

This script will power off all vApps with Key/Value pair set to ‘AutoOnOff’ and ‘Yes’ irrespective of time or day, and ignores any other metadata key/value pairs. It could be run manually, or from a job scheduler at a specific time of day. It could easily be customised to power on vApps instead.

The script is available from page 19.

Saving a password to a file securely

Scripts can often require sensitive information such as usernames, passwords and organisation names to be specified either in the script itself or asked for through interaction using the Read-Host cmdlet. This is acceptable for interactive situations, but not secure or viable when the script is being automatically run from a job scheduler.

This script allows for a password to be entered and saved to a specified file for subsequent use in a script. While the file can be opened by a text editor, it is meaningless. If the script is executed using a username reserved for scheduled jobs, then the password file should be stored within that user’s home folder structure, such that it will protected from other users.

The script is available from page 21.

RUNNING SCRIPTS FROM WINDOWS TASK SCHEDULER

Microsoft provides a task/job scheduler on Windows desktop and server operating systems. This allows the capability to create a task or job that will be run daily, and potentially several times a day.

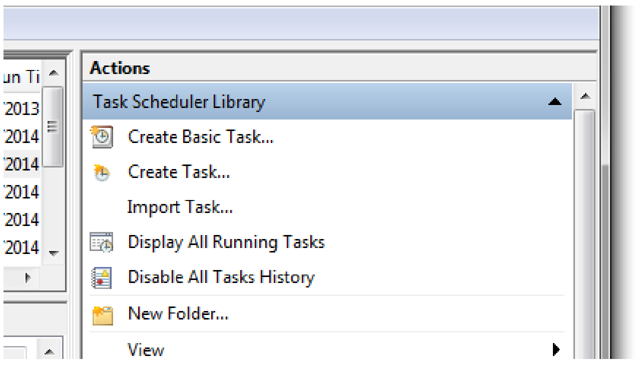
In the case of the 'Just power off everything now' script, this could be executed at the same time every evening, say at 21:00, and then vApps with 'AutoOnOff' set to 'Yes' would be gracefully shut down.

In the more complicated script discussed above, users may want to execute the script several times a day (morning and evening), and the script logic and metadata would take care of which vApps to start or stop.

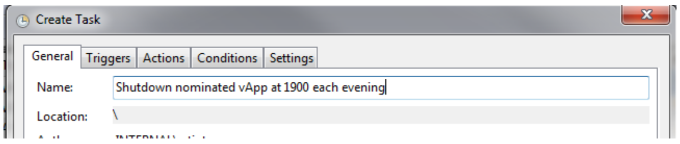
When running the scripts from a Task Scheduler, as opposed to within the PowerGUI environment, it is necessary to 'include' the vCloud cmdlets into the PowerShell environment. This is covered in a short line of code at the top of the scripts. Simply remove the # comment to enable the line of code.

```
# Uncomment line below when running as a scheduled
task
# Add-PSSnapin VMware.VimAutomation.Cloud
```

Within Task Scheduler, create a new task:

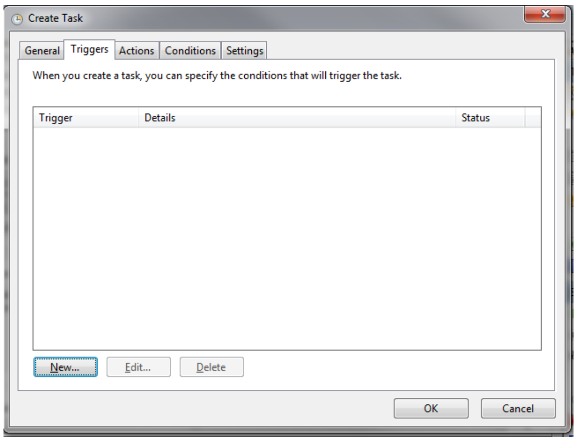


Give the new task a name:



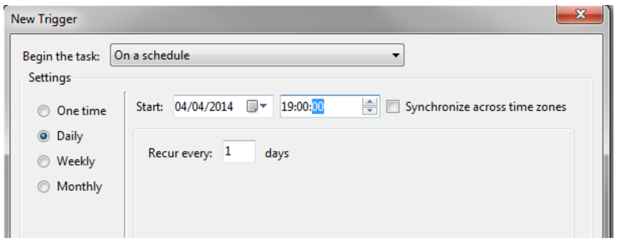
Name: Shutdown nominated vApp at 1900 each evening
Location: \

Next, click on the Triggers tab, and click New



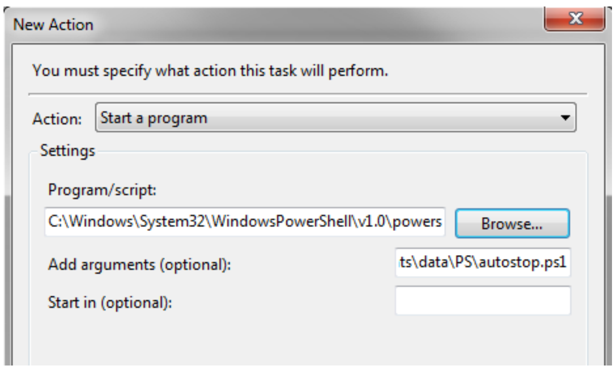
When you create a task, you can specify the conditions that will trigger the task.

Define the task to run Daily at 19:00:00, with 'Begin the task' set to 'On a schedule'



Begin the task: On a schedule
Daily [✓]
Start: 04/04/2014 19:00:00
Synchronize across time zones [ ]
Recur every: 1 days

Then, on the 'Actions' tab, create a new action.



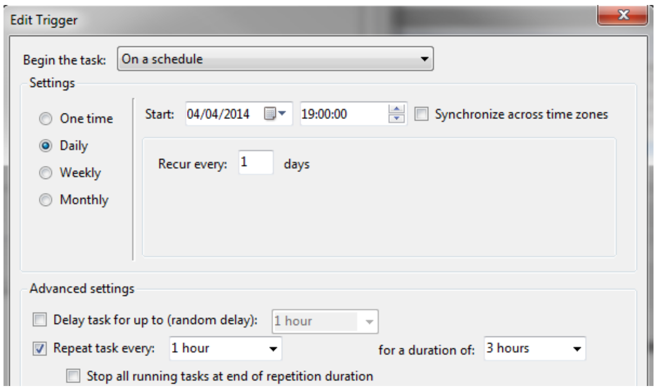
You must specify what action this task will perform.

Action: Start a program
Program/script: C:\Windows\System32\WindowsPowerShell\v1.0\powers
Add arguments (optional): ts\data\PS\autostop.ps1
Start in (optional):

The action will be to ‘Start a program’.

The program/script to run is:

C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
and the argument is the path to the script to be run C:\<full path to file>\autostop.ps1



Edit Trigger

Begin the task: On a schedule

Settings

One time [✓ Daily] Weekly Monthly

Start: 04/04/2014 19:00:00 ☐ Synchronize across time zones

Recur every: 1 days

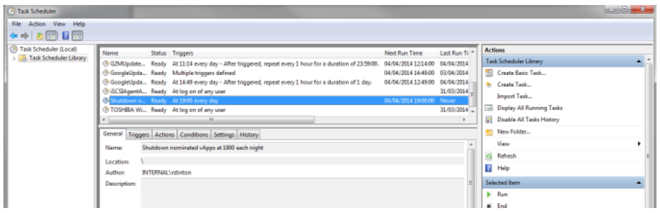
Advanced settings

☐ Delay task for up to (random delay): 1 hour

☑ Repeat task every: 1 hour for a duration of: 3 hours

☐ Stop all running tasks at end of repetition duration

Having clicked ‘OK’ several times, the new entry can be tested, using the ‘Run’ Actions, over on the top right of this screen capture.



Task Scheduler

Name	Status	Triggers	Next Run Time	Last Run Ti
G2MUpdate...	Ready	At 11:14 every day - After triggered, repeat every 1 hour for a duration of 23:59:00.	04/04/2014 12:14:00	04/04/2014
GoogleUpda...	Ready	Multiple triggers defined	04/04/2014 14:49:00	03/04/2014
GoogleUpda...	Ready	At 14:49 every day - After triggered, repeat every 1 hour for a duration of 1 day.	04/04/2014 12:49:00	04/04/2014
iSCSIAgentA...	Ready	At log on of any user		31/03/2014
Shutdown n...	Ready	At 19:00 every day	04/04/2014 19:00:00	Never
TOSHIBA Wi...	Ready	At log on of any user		31/03/2014

Name: Shutdown nominated vApps at 1900 each night

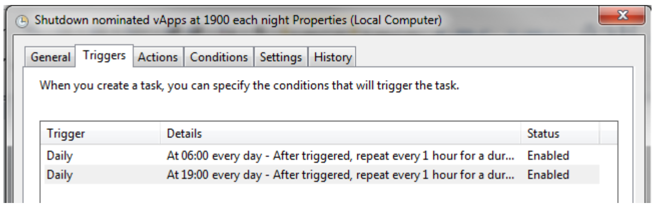
Location: \

Author: INTERNAL\rstinton

Description:

A window will launch and show the task running successfully. If you see some red error text, ensure that the ‘Add-PSSnapin’ line has been uncommented as discussed above.

For the more complex script, that may want to be run every hour for several hours in the morning and evening, simply define a more complex trigger.



Shutdown nominated vApps at 1900 each night Properties (Local Computer)

When you create a task, you can specify the conditions that will trigger the task.

Trigger	Details	Status
Daily	At 06:00 every day - After triggered, repeat every 1 hour for a dur...	Enabled
Daily	At 19:00 every day - After triggered, repeat every 1 hour for a dur...	Enabled

APPENDIX: THE SCRIPTS

You can copy these scripts and past them into your PowerShell application. You will need to enter your specific information for the UKCloud cloud platform such as organisation name, username and password. These have been removed for security, but it is clear where these are required.

Script to read the metadata from the vApp and write out to a CSV file

```
## UKCloud PowerShell/PowerCLI example script
## Date written: April 2014 by UKCloud
##
## Purpose: Read metadata from vApps in an Org, collect metadata relevant to AutoOnOff and
write out to a csv file
## Input required: Filename to write out to


### Define Functions that will be used in the script


Function Get-CIMetaData {
    <#
    .SYNOPSIS
        Retrieves all Metadata Key/Value pairs.
    .DESCRIPTION
        Retrieves all custom Metadata Key/Value pairs on a specified vCloud object
    .PARAMETER  CIObject
        The object on which to retrieve the Metadata.
    .PARAMETER  Key
        The key to retrieve.
    .EXAMPLE
        PS C:\> Get-CIMetadata -CIObject (Get-Org Org1)
    #>
    param(

[parameter(Mandatory=$true,ValueFromPipeline=$true,ValueFromPipelineByPropertyName=$true)]
            [PSObject[]]$CIObject,
            $Key
        )
    Process {
        Foreach ($Object in $CIObject) {
            If ($Key) {
                ($Object.ExtensionData.GetMetadata()).MetadataEntry | Where {$_.Key -eq $key }
| Select @{N="CIObject";E={$Object.Name}}, Key, Value
            } Else {
                ($Object.ExtensionData.GetMetadata()).MetadataEntry | Select
@{N="CIObject";E={$Object.Name}}, Key, Value
            }
        }
    }
}

### End of function definitions

### Start of script !!

### Connect to customer's Org (need Org admin user/password and Org name)

# Connect to UKCloud Cloud
# Uncomment line below for rapid connection for testing purposes
# Connect-CIServer -server api.vcd.portal.UKCloud.com -User "username" -Password "password" -
Org “your_org_name”
#
# Uncomment below for connection with username/password prompted for
# $creds = Get-Credential
# Connect-CIServer -server api.vcd.portal.UKCloud.com -Org “your_org_name” -Credential $creds
#
# Uncomment below for connection using a stored secure password file
$credsfile = 'C:\temp\password.txt'
$username = ‘your_username’
$password = get-content $CredsFile | convertto-securestring
$creds = new-object -typename System.Management.Automation.PSCredential -argumentlist
$username,$password
Connect-CIServer -server api.vcd.portal.UKCloud.com -Org “your_org_name” -Credential $creds


# Set up an array for the final report
$report = @()

# Get all vApp from the Org
$vApps = Get-CIVApp

foreach ($vApp in $vApps) {

       # Get the metadata key/value pairs for all vApps
       $Metadatas = Get-CIMetaData -CIObject $vApp

       # Get the individual key/value pairs for each vApp
       $StopTime = "Not specified"
       $StartTime = "Not specified"
       $Day = "Not specified"
       $AutoOnOff = "Not specified"
       foreach ($Metadata in $Metadatas) {
              $Key = ''
              $Value = ''

              $vAppName = $Metadata.CIObject
              $Key = $Metadata.Key
              $Value = $Metadata.Value


              if ($Key -eq 'StopTime') {$StopTime = $Value}
              if ($Key -eq 'StartTime') {$StartTime = $Value}
              if ($Key -eq 'Days') {$Day = $Value}
              if ($Key -eq 'AutoOnOff') {$AutoOnOff = $Value}
       }
       Write-Host "vApp ",$vApp
       Write-Host "Days ",$Day
       Write-Host "StopTime",$StopTime
       Write-Host "StartTime",$StartTime
       Write-Host "AutoOnOff",$AutoOnOff
       write-host ""
```

```
        $row= " " | select vApp,Days,StopTime,StartTime,AutoOnOff
        $row.vApp = $vApp
        $row.Days = $Day
        $row.StopTime = $StopTime
        $row.StartTime = $StartTime
        $row.AutoOnOff = $AutoOnOff
        # Add this row to the report array
        $report += $row

}

# Write out the report array to the file in CSV format
# $outfile = "C:\temp\metadata.csv"
$outfile = Read-Host -prompt 'Please enter a file path to write out to'


$report | Export-Csv $outfile -NoTypeInformation
```

Script to read a CSV file and update the metadata of the vApp

```
## UKCloud PowerShell/PowerCLI example script
## Date written: April 2014 by UKCloud
##
## Purpose: Import metadata key/value pairs from a CSV file, and apply the metadata to vApps
to enable the AutoOnOff functionality
## Input required: A CSV file with 5 columns: vApp,Days,StopTime,StartTime,AutoOnOff
## vApp = vApp Name
## Days = String containing one or more days in form Mon Tue Wed (eg MonWedFri or
MonTueWedThuFriSatSun)
## StopTime = Hour in 24 hour clock (eg 19 or 21)
## StartTime = Hour in 24 hour clock (eg 19 or 21)
## AutoOnOff = Yes or No




### Define Functions that will be used in the script

Function New-CIMetaData {
    <#
    .SYNOPSIS
        Creates a Metadata Key/Value pair.
    .DESCRIPTION
        Creates a custom Metadata Key/Value pair on a specified vCloud object
    .PARAMETER  Key
        The name of the Metadata to be applied.
    .PARAMETER  Value
        The value of the Metadata to be applied.
    .PARAMETER  CIObject
        The object on which to apply the Metadata.
    .EXAMPLE
        PS C:\> New-CIMetadata -Key "Owner" -Value "Alan Renouf" -CIObject (Get-Org Org1)
    #>
     [CmdletBinding(
         SupportsShouldProcess=$true,
        ConfirmImpact="High"
    )]
    param(

[parameter(Mandatory=$true,ValueFromPipeline=$true,ValueFromPipelineByPropertyName=$true)]
            [PSObject[]]$CIObject,
            $Key,
            $Value
        )
    Process {
        Foreach ($Object in $CIObject) {
            $Metadata = New-Object VMware.VimAutomation.Cloud.Views.Metadata
            $Metadata.MetadataEntry = New-Object
VMware.VimAutomation.Cloud.Views.MetadataEntry
```

```
            $Metadata.MetadataEntry[0].Key = $Key
            $Metadata.MetadataEntry[0].Value = $Value
            $Object.ExtensionData.CreateMetadata($Metadata)
            ($Object.ExtensionData.GetMetadata()).MetadataEntry | Where {$_.Key -eq $key } |
Select @{N="CIObject";E={$Object.Name}}, Key, Value
        }
    }
}

function Select-FileDialog
### $file = Select-FileDialog -Title "Select a file" -Directory "D:\scripts" -Filter
"Powershell Scripts|(*.ps1)"
###
{
      param([string]$Title,[string]$Directory,[string]$Filter="All Files (*.*)|*.*")
      [System.Reflection.Assembly]::LoadWithPartialName("System.Windows.Forms") | Out-Null
      $objForm = New-Object System.Windows.Forms.OpenFileDialog
      $objForm.InitialDirectory = $Directory
      $objForm.Filter = $Filter
      $objForm.Title = $Title
      $Show = $objForm.ShowDialog()
      If ($Show -eq "OK")
      {
             Return $objForm.FileName
      }
      Else
      {
             Write-Error "Operation cancelled by user."
      }
}
### End of external functions

### Start of script !!

# Connect to UKCloud Cloud
# Uncomment line below for rapid connection for testing purposes
# Connect-CIServer -server api.vcd.portal.UKCloud.com -User "username" -Password "password" -
Org "your_org_name"
#
# Uncomment below for connection with username/password prompted for
# $creds = Get-Credential
# Connect-CIServer -server api.vcd.portal.UKCloud.com -Org "your_org_name" -Credential $creds
#
# Uncomment below for connection using a stored secure password file
$credsfile = 'C:\temp\password.txt'
$username = 'your_username'
$password = get-content $CredsFile | convertto-securestring
$creds = new-object -typename System.Management.Automation.PSCredential -argumentlist
$username,$password
Connect-CIServer -server api.vcd.portal.UKCloud.com -Org "your_org_name" -Credential $creds

# Read existing CSV file with metadata requirements
```

```
$file = Select-FileDialog -Title "Select a file to import from"

# Read the CSV file into an array
$requirements = Import-Csv -Path $file

# Trundle through the array, getting the requirements per vApp
foreach ($requirement in $requirements){
       $vApp = $requirement.vApp
       $Days = $requirement.Days
       $StopTime = $requirement.StopTime
       $StartTime = $requirement.StartTime
       $AutoOnOff = $requirement.AutoOnOff

       # Apply the metadata to the vApp (will overwrite previous key/value pairs)
       $target = Get-CIVApp -Name $vApp
       New-CIMetaData -Key "Days" -Value $Days -CIObject $target
       New-CIMetaData -Key "StopTime" -Value $StopTime -CIObject $target
       New-CIMetaData -Key "StartTime" -Value $StartTime -CIObject $target
       New-CIMetaData -Key "AutoOnOff" -Value $AutoOnOff -CIObject $target
}
```

The main start/stop script

```
## UKCloud PowerShell/PowerCLI example script
## Date written: April 2014 by UKCloud
##
## Purpose:Power on or off vApps based on metadata and time of day
## Input required: Script requires 3 metadata key/value pairs to be set
##     StopTime set to 24 hour clock hour value (eg 18 or 20)
##  StartTime set to 24 hour clock hour value (eg 06 or 08)
##  Days set to list of 3 letter abbreviated days (eg MonWed or SatSun or Every for all week)
##  AutoOnOff (Yes or No)
##     Username, password, Org name

# Uncomment line below when running as a scheduled task
# Add-PSSnapin VMware.VimAutomation.Cloud


# Functions required not included in base PowerCLI

Function Get-CIMetaData {
    <#
    .SYNOPSIS
        Retrieves all Metadata Key/Value pairs.
    .DESCRIPTION
        Retrieves all custom Metadata Key/Value pairs on a specified vCloud object
    .PARAMETER  CIObject
        The object on which to retrieve the Metadata.
    .PARAMETER  Key
        The key to retrieve.
    .EXAMPLE
        PS C:\> Get-CIMetadata -CIObject (Get-Org Org1)
    #>
    param(

[parameter(Mandatory=$true,ValueFromPipeline=$true,ValueFromPipelineByPropertyName=$true)]
            [PSObject[]]$CIObject,
            $Key
        )
    Process {
        Foreach ($Object in $CIObject) {
            If ($Key) {
                ($Object.ExtensionData.GetMetadata()).MetadataEntry | Where {$_.Key -eq $key }
| Select @{N="CIObject";E={$Object.Name}}, Key, Value
            } Else {
                ($Object.ExtensionData.GetMetadata()).MetadataEntry | Select
@{N="CIObject";E={$Object.Name}}, Key, Value
            }
        }
    }
}
```

```
# End of additional functions

## Start of UKCloud code

# Connect to UKCloud Cloud
# Uncomment line below for rapid connection for testing purposes (substitute your Org name)
# Connect-CIServer -server api.vcd.portal.UKCloud.com -User "username" -Password "password" -
Org "your_org_name"
#
# Uncomment below for connection with username/password prompted for
# $creds = Get-Credential
# $org = Read-Host "Enter Org name"
# Connect-CIServer -server api.vcd.portal.UKCloud.com -Org $org -Credential $creds
#
# Uncomment below for connection using a stored secure password file, substitute your
credsfile, username, password, org
$credsfile = 'C:\temp\password.txt'
$username = 'your_username'
$org = "your_org_name"
$password = get-content $CredsFile | convertto-securestring
$creds = new-object -typename System.Management.Automation.PSCredential -argumentlist
$username,$password
Connect-CIServer -server api.vcd.portal.UKCloud.com -Org $org -Credential $creds


# Get the current time, and specifically hour in 24 hour format
$time = Get-Date -DisplayHint Time
$today = Get-Date -UFormat %a
$hour = $time.Hour

# Debug
# Write-Host 'Time now',$hour

# Get a list of all the vApps in the Org
$vApps = Get-CIVApp

foreach ($vApp in $vApps) {

# Get the metadata key/value pairs for all vApps
$Metadatas = Get-CIMetaData -CIObject $vApp

# Get the individual key/value pairs for each vApp
$StopTime = -2
$StartTime = -999
$Day = "Not specified"
foreach ($Metadata in $Metadatas) {
$Key = ''
$Value = ''

$vAppName = $Metadata.CIObject
$Key = $Metadata.Key
$Value = $Metadata.Value
# Debug
# Write-Host $vAppName,$Key,$Value

if ($Key -eq 'AutoOnOff') {$AutoOnOff = $Value}
if ($Key -eq 'StopTime') {$StopTime = $Value}
if ($Key -eq 'StartTime') {$StartTime = $Value}
if ($Key -eq 'Days') {$Day = $Value}
if ($Day -like 'Every') {$Day = $today}
}

# Debug
# Write-Host $StartTime,$StopTime,$Day,$AutoOnOff

# For testing purposes, just write out the action
if (($hour -ge $StopTime) -and ($vApp.Status -eq 'PoweredOn') -and ($Day -like '*'+$today+'*')
-and ($AutoOnOff -eq 'Yes')) {write-host 'Stopping...',$vApp}
if (($hour -ge $StartTime) -and ($vApp.Status -eq 'PoweredOff') -and ($Day -like
'*'+$today+'*') -and ($AutoOnOff -eq 'Yes')) {write-host 'Starting...',$vApp}
# Carry out the PowerOn or PowerOff task
# if (($hour -ge $StopTime) -and ($vApp.Status -eq 'PoweredOn') -and ($Day -like
'*'+$today+'*') -and ($AutoOnOff -eq 'Yes')) {Stop-CIVApp -VApp $vApp -Confirm:$false}
# if (($hour -ge $StartTime) -and ($vApp.Status -eq 'PoweredOff') -and ($Day -like
'*'+$today+'*') -and ($AutoOnOff -eq 'Yes')) {Stop-CIVApp -VApp $vApp -Confirm:$false}
}
```

Script for powering all vApps off

```
## UKCloud PowerShell/PowerCLI example script
## Date written: April 2014 by UKCloud
##
## Purpose: Script to read metadata from vApps, and if 'AutoOnOff' is set to 'Yes', power off
the vApp
##            Could be run manually or via a job scheduler
## Input required: username, password, Org name

# Uncomment line below when running as a scheduled task
# Add-PSSnapin VMware.VimAutomation.Cloud

# External Functions required

Function Get-CIMetaData {
    <#
    .SYNOPSIS
        Retrieves all Metadata Key/Value pairs.
    .DESCRIPTION
        Retrieves all custom Metadata Key/Value pairs on a specified vCloud object
    .PARAMETER  CIObject
        The object on which to retrieve the Metadata.
    .PARAMETER  Key
        The key to retrieve.
    .EXAMPLE
        PS C:\> Get-CIMetadata -CIObject (Get-Org Org1)
    #>
    param(

[parameter(Mandatory=$true,ValueFromPipeline=$true,ValueFromPipelineByPropertyName=$true)]
            [PSObject[]]$CIObject,
            $Key
        )
    Process {
        Foreach ($Object in $CIObject) {
            If ($Key) {
                ($Object.ExtensionData.GetMetadata()).MetadataEntry | Where {$_.Key -eq $key }
| Select @{N="CIObject";E={$Object.Name}}, Key, Value
            } Else {
                ($Object.ExtensionData.GetMetadata()).MetadataEntry | Select
@{N="CIObject";E={$Object.Name}}, Key, Value
            }
        }
    }
}
# End of function definition

## Start of script

# Connect to UKCloud Cloud
# Uncomment line below for rapid connection for testing purposes
# Connect-CIServer -server api.vcd.portal.UKCloud.com -User "username" -Password "password" -
Org "your_org_name"
#
# Uncomment below for connection with username/password prompted for
# $creds = Get-Credential
# Connect-CIServer -server api.vcd.portal.UKCloud.com -Org "your_org_name" -Credential $creds
#
# Uncomment below for connection using a stored secure password file
$credsfile = 'C:\temp\password.txt'
$username = 'your_username'
$password = get-content $CredsFile | convertto-securestring
$creds = new-object -typename System.Management.Automation.PSCredential -argumentlist
$username,$password
Connect-CIServer -server api.vcd.portal.UKCloud.com -Org "your_org_name" -Credential $creds


# Get all vApps in this Org
$vApps = Get-CIVApp
foreach ($vApp in $vApps) {

       $Metadatas = Get-CIMetaData -CIObject $vApp
       $status=Get-CIVApp -Name $vApp
       foreach ($Metadata in $Metadatas) {

              $vApp = $Metadata.CIObject
              $Key = $Metadata.Key
              $Value = $Metadata.Value

#Uncomment to move from test to real power off
              if (($Value -like 'Yes') -and ($Key -like 'AutoOnOff') -and ($status.Status -eq
'PoweredOn')) {write-host 'Stopping vApp....',$vApp}
              #if (($Value -like 'Yes') -and ($Key -like 'AutoOnOff') -and ($vApp.Status -eq
'PoweredOn')) {Stop-CIVApp -VApp $vApp -Confirm:$false}
              }
}
```

Script for saving a password to a file securely

```
## UKCloud PowerShell/PowerCLI example script
## Date written: April 2014 by UKCloud
##
## Purpose: Save a secure password for a file for use later in scripts.  Test connection.
## Input required: Enter valid username, password and Org name for a valid Org

## Use this script to save a password to a secure file
## and to test reading the file and logging into vCloud

$AdminName = Read-Host "Enter your API username"
$Org = Read-Host "Enter your Org name/number"

# Choose a location for the secure password file
$CredsFile = "C:\temp\password.txt"
# $CredsFile = Read-Host -Prompt 'Enter location for secure password file'

$FileExists = Test-Path $CredsFile

if  ($FileExists -eq $false) {

    Write-Host 'Credential file not found. Enter your password:' -ForegroundColor Red

    Read-Host -AsSecureString -Prompt 'Enter your password'| ConvertFrom-SecureString | Out-
File $CredsFile

    $password = get-content $CredsFile | convertto-securestring

    $Cred = new-object -typename System.Management.Automation.PSCredential -argumentlist
domain\$AdminName,$password}

else

    {Write-Host 'Using your stored credential file' -ForegroundColor Green

    $password = get-content $CredsFile | convertto-securestring

    $Cred = new-object -typename System.Management.Automation.PSCredential -argumentlist
$AdminName,$password

       Connect-CIServer -server api.vcd.portal.UKCloud.com -Org $Org -Credential $Cred
       }
```

ABOUT UKCLOUD

UKCloud has developed a range of cloud services designed specifically for the UK public sector, to help increase efficiencies, reduce costs, significantly improve procurement times and increase transparency. Our services are *easy to adopt, easy to use and easy to leave* to ensure that our customers remain in complete control with minimum risk. We were one of the first G-Cloud providers to achieve Pan Government Accreditation (PGA) up to Elevated OFFICIAL, and our services continue to achieve formal UK Government accreditations which make them suitable for all data at OFFICIAL (including OFFICIAL-SENSITIVE).

UKCloud's full offering consists of:

1. Infrastructure as a Service (IaaS) – seven offerings around Compute and Storage on demand
2. Software as a Service (SaaS) – offerings for email and collaboration as well as secure sync-and-share of files to help teams work effectively in groups, using a variety of devices
3. Platform as a Service (PaaS) – based upon Open Source Digital Application Platform and Hadoop which provides organisations the benefits of using a commodity cloud platform without the added management overheads

All of UKCloud's UK sovereign cloud computing services are hosted in one (or both) of our highly resilient Tier 3 UK data centres in Farnborough and Corsham. UKCloud services are delivered with leading technologies from UKCloud Alliance Partners: QinetiQ, VMware, Cisco, EMC and Ark Data Centres. The Cloud Alliance also provides a collaborative resource which drives innovation and technical product development, helping to continually improve UKCloud's offering to meet the needs of the UK public sector.

UKCloud is focused on providing cloud services in a more agile, secure and cost effective manner. We strive to deliver solutions that harness technology as a way to facilitate the changes that are needed to streamline processes and reduce costs to support the UK public sector and, ultimately, UK citizens and taxpayers.

MORE INFORMATION

For further information about UKCloud and how we can help you, please send an email to info@ukcloud.com

UKCloud Ltd

A8 Cody Technology Park
Ively Road
Farnborough
Hampshire
GU14 0LX

+44 (0)1252 303300

info@ukcloud.com

www.ukcloud.com

UKC-GEN-125 • 07/2016