Powershell Tasks in Operations Manager 2007

Using Powershell scripts in custom console tasks in Operations Manager 2007

Pete Zerger, MCSE(Messaging), MCTS(SQL 2005), MVP-MOM
http://www.systemcenterforum.org

June 2007
Version 1.0

Table of Contents

Introduction

Tasks are actions executed against monitored objects in Operations Manager 2007. New custom tasks can be created in the Management Pack Objects node in the Authoring area of the Operations Console. Tasks can be executed from the Actions pane in the Monitoring space of the Operations Console.

A task can be created as an **agent task** or a **console task**, which differ as follows:

- Agent tasks can run remotely on an agent or a Management Server
- Console tasks can run only on the local computer.

It is important to remember that while you can execute a batch file or script as a task remotely (agent task) or locally (console task), if a task is generated by an alert or an event, it can only be run locally.

The following is a list of task types that may be authored in the Authoring space of the Operations Console:

- **Run a script** - Runs a vbscript or jscript script on an agent or Management Server.
- **Command line** - Runs a batch file or starts an application on an agent or Management Server. This task can be run locally or remotely. Because of the script engine limitations of the 'run a script' task type (mentioned above), *this is the task type that will be used to execute Powershell scripts.*
- **Alert command line** - Runs a task automatically when a specified alert or alerts are generated. Specify the alert by using the Parameters drop-down list in the Command Line wizard page of the Create Task Wizard. This task can only be run locally.
- **Event command line** - Runs a task automatically when a specified event or events are generated. Specify the event by using the Parameters drop-down list in the Command Line wizard page of the Create Task Wizard. This task can only be run locally.

In this article, we will illustrate the use of Powershell scripts in two examples. The first, will be execution of a simple Powershell script (with script parameters) output to a text file. In the second example, we'll create a Task using a Powershell script containing a script parameter requiring a value be passed for the script to run successfully.

Example 1: Simple Powershell script execution task with file export

In this example, we'll create a console task that executes a simple Powershell script that exports a list of installed management packs to a text file.

NOTE: In order for these console tasks to execute successfully, you will need to install Powershell 1.0 and the Operations Manager UI components on the machine where the task will be executed.

Create the script

First, we need to create the Powershell script that will be used in the custom task. Copy the following Powershell script below into notepad and save to the c:\scripts directory on your computer.

Be sure to replace the value assigned to the **$rootMS** variable in line 1 of the script to that of your root management server.

NOTE: Beware of line wrap in the sample scripts provided in this tutorial

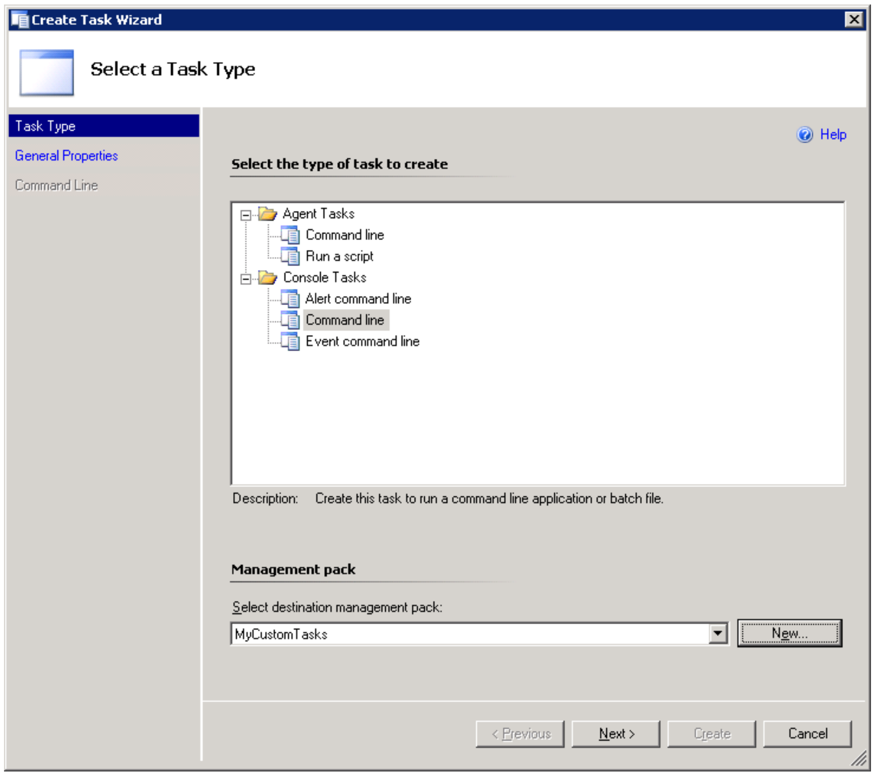
***************Start copy below this line***************

```
$rootMS = 'nocrms01'

#Initializing the Ops Mgr 2007 Powershell provider
    add-pssnapin "Microsoft.EnterpriseManagement.OperationsManager.Client" -ErrorVariable errSnapin;
    set-location "OperationsManagerMonitoring::" -ErrorVariable errSnapin;
    new-managementGroupConnection -ConnectionString:$rootMS -ErrorVariable errSnapin;
    set-location $rootMS -ErrorVariable errSnapin;

#Checks to see if it failed or succedded in loading the provider
    if ($errSnapin.count -eq 0){
            Write-host "`nOpsMgr 2007 PSSnapin initialized!`n";
    }
    else{
          Write-host "`nOpsMgr 2007 PSSnapin failed initialize!`nPlease verify you are running this script on a Ops Mgr
2007 Management Server";
            Write-Host;
    }

get-managementpack | select-object displayname, sealed | export-csv 'c:\mps.csv'
```

***************Start copy above this line***************

Create the Task

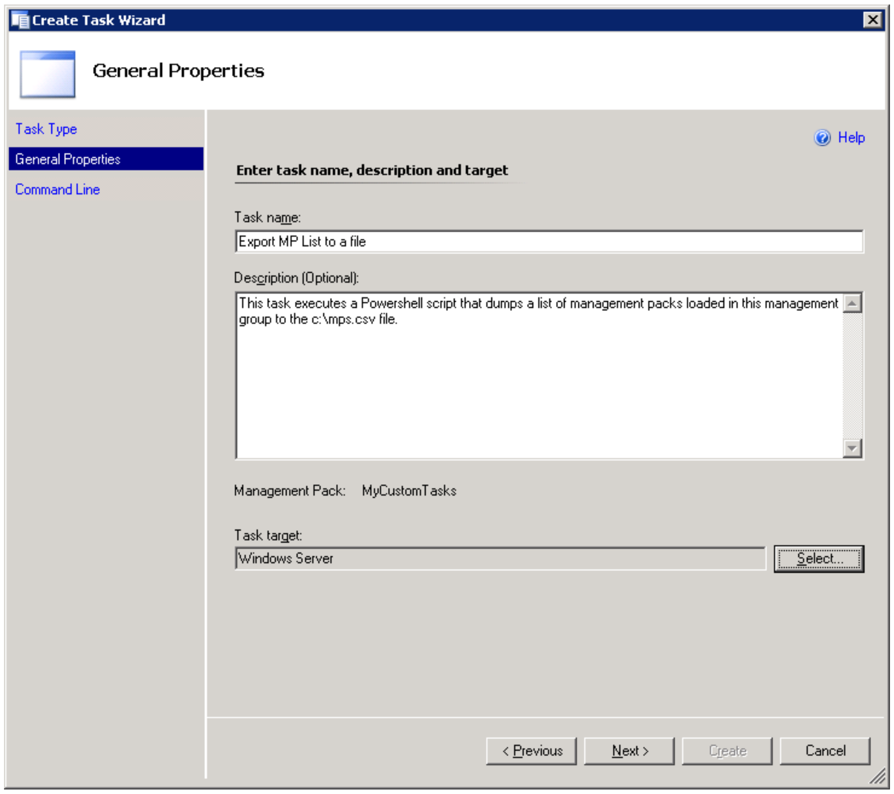
1. Launch the Operations Console and select the Authoring workspace.
2. Expand the Authoring node. Right click the Tasks node and select **Create a New Task**.
3. On the **Select a Task Type** screen, under **Console Tasks**, select **Command line** (see figure 1).
4. In the dropdown provided, select the desired management pack in which to store the task, then click Next.



Description: Create this task to run a command line application or batch file.

Select destination management pack: MyCustomTasks

5. On the **General Properties** page, enter a name and description for the task.
6. In the **Task Target** box, select Windows Server as the target. This will cause our task to appear in the Actions pane every time we select a Windows 2000 or 2003 Server in the Operations Console. Click Next.


Create Task Wizard

General Properties

Enter task name, description and target

Task name:
Export MP List to a file

Description (Optional):
This task executes a Powershell script that dumps a list of management packs loaded in this management group to the c:\mps.csv file.

Management Pack: MyCustomTasks

Task target:
Windows Server

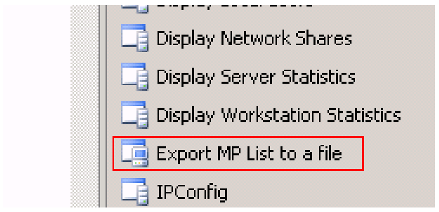
7. On the **Command Line** screen, enter the following values in the spaces provided:

   Application: %windir%\system32\windowspowershell\v1.0\powershell.exe
   Parameters: c:\scripts\exportmps.ps1
   Working directory: %windir%\system32\windowspowershell\v1.0

8. Click the **Create** button to create your custom task.

Test the Task

1. In the Operations Console, browse to the Monitoring space.
2. Click on the Monitoring state view and in the Results pane, select any Windows Server.
3. In the Actions pane, click the task 'Export MP List to a file'



Example 2: Powershell script execution with script parameters

In this example, we'll create a console task that executes a Powershell script that exports a list open alerts and key alert details to an HTML web page (c:\alerts.html in this example). As an example this time, we are going to use a Powershell script that accepts the name of your Root Management Server name as a parameter

This examplewill demonstrate the targeting functionality within Operations Manager at a basic level and how to employ it in running tasks.

Create the Script

Here is the sample script. Save as **alertstohtml.ps1** to the **c:\scripts** directory. It requires a value be passed to parameter –rootMS: to run successfully, where the value passed is the DNS or NetBIOS name of the Root Management Server.

***************Start copy below this line***************

```
param ($rootMS)

#Initializing the Ops Mgr 2007 Powershell provider

    add-pssnapin "Microsoft.EnterpriseManagement.OperationsManager.Client" -ErrorVariable errSnapin;
    set-location "OperationsManagerMonitoring::" -ErrorVariable errSnapin;
    new-managementGroupConnection -ConnectionString:$rootMS -ErrorVariable errSnapin;
    set-location $rootMS -ErrorVariable errSnapin;

#Checks to see if it failed or succedded in loading the provider

    if ($errSnapin.count -eq 0){
            Write-host "`nOpsMgr 2007 PSSnapin initialized!`n";
    }
    else{
            Write-host "`nOpsMgr 2007 PSSnapin failed initialize!`nPlease verify you are running this script on a Ops
Mgr 2007 Management Server";
            Write-Host;
    }

 #Next, retrieve all active alerts from the management group  and output to c:\alerts.html

Get-alert | where-Object {$_.ResolutionState -ne 255 }| select-Object MonitoringObjectName, Name, Description,
ResolutionState | ConvertTo-html | set-content c:\alerts.html
```

***************Start copy above this line***************

Create the Task

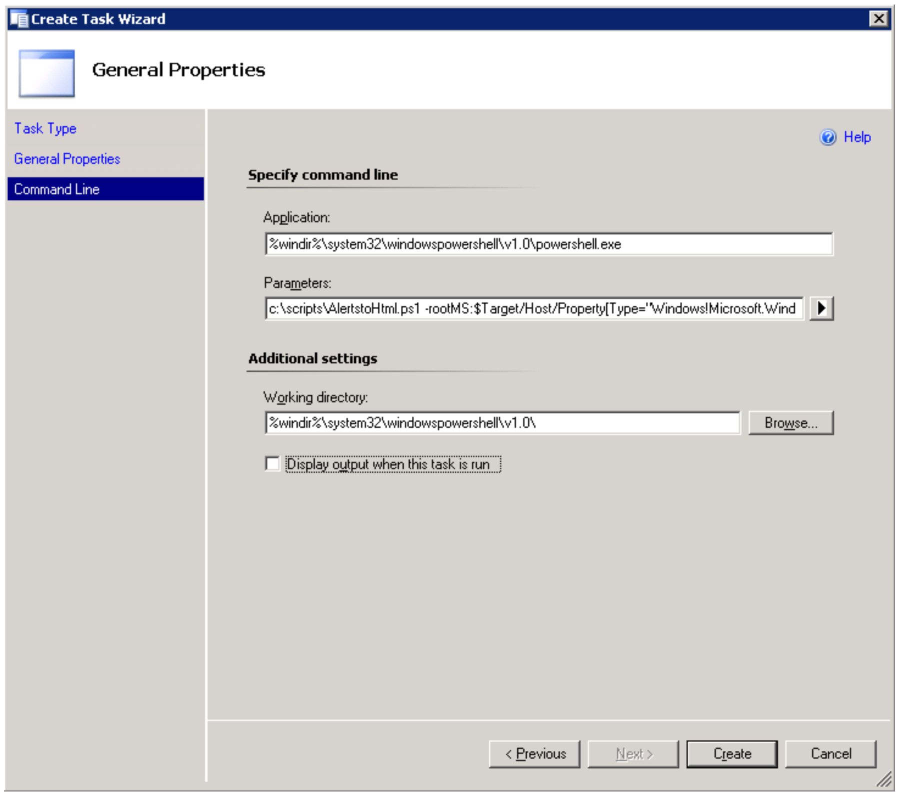
You'll create this task you did the first, but with a minor difference. Paste the values shown into the Create Task Wizard fields provided.

Task Target: Windows Server

Application: %windir%\system32\windowspowershell\v1.0\powershell.exe

Parameters: c:\scripts\AlertstoHtml.ps1 –rootMS $Target/Property[Type="System!System.Entity"]/DisplayName$

working directory: %windir%\system32\windowspowershell\1.0



Application:
%windir%\system32\windowspowershell\v1.0\powershell.exe

Parameters:
c:\scripts\AlertstoHtml.ps1 -rootMS:$Target/Host/Property[Type="Windows!Microsoft.Wind

Working directory:
%windir%\system32\windowspowershell\v1.0\

☐ Display output when this task is run

Notice the string to the right of the **–rootMS** parameter in the Parameters textbox:

$Target/Property[Type="System!System.Entity"]/DisplayName$

This value was added by selecting **Display Name (Entity)** from the arrow-out to the right of the Parameters textbox. This will cause the Display name of the entity given focus in the Operations Console when the task is launched to be passed to this parameter of the script.

IMPORTANT: Since the task is targeted to Windows Server, it will appear in the list of available tasks for any selected windows server. However, it will run successfully ONLY if launched when the Root Management Server is highlighted in the Operations Console GUI.

Hopefully this example makes the targeting feature a bit clearer.

Test the task as you did with the first example, and check the c:\ directory for the alerts.html output file created when the task runs successfully.

Troubleshooting Task Output

Since in these examples the Powershell task provides output to a text file, it is not required to select the **Display output when this task is run** option in the wizard. However, when you do select this, an output window will capture any error messages generated by the script, making troubleshooting much easier.

If you run the task in example 2 with focus on a server other than the Root Management Server, it will fail and output will be captured in an output window.

Feedback

I hope you find this article helpful. Your feedback is always welcome and appreciated at administrator@systemcenterforum.org