



Creating a System Center Essentials 2007 Command Shell Environment

Step-by-step guidance on how to configure a custom Powershell instance for managing Essentials (SCE) 2007

Version 1.0

Authors:

Pete Zerger, MCSE(Messaging) | MCTS(SQL 2005) | MVP-MOM
Neale Brown, MCSA(Messaging)
Consultant Partners, AKOS Technology Services

Some Rights Reserved: You are free to reference this document, so long as you properly credit the author and provide a link back to the published source.

Introduction

System Center Essentials is a powerful operations and systems management platform optimized for the mid-market enterprise, where specialized IT skills and large IT staffs often just don't exist. To that end, SCE 2007 doesn't ship with the Command Shell, the object-oriented command line administration interface that comes with Operations Manager 2007. However, just because it doesn't come with the custom Command Shell environment doesn't mean we can't make our own.

To that effect, in this tutorial we are going to step you through the process of creating a SCE Command Shell interface you can use to automate administration on your SCE Server and perform some actions that may more laborious, or not possible through the GUI.

Find your SDK .NET class libraries

At the end of the day, you have to remember Powershell is based on .NET. Where you have .NET class libraries, you have a source for Powershell cmdlet development....or just knocking out some ad-hoc one-liners. You'll be surprised how much we can do by simply loading the SCE.NET class libraries into a Powershell instance. In SCE 2007, the SDK binaries we need are easy to find – they're located in a subfolder of the installation directory

Go to this directory on the SCE 2007 Server:

```
%programfiles%\System Center Essentials 2007\SDK Binaries
```

And you should see these files:

```
Microsoft.EnterpriseManagement.OperationsManager.Common.dll
```

```
Microsoft.EnterpriseManagement.OperationsManager.dll
```

Configure Powershell Script Security

By default, script security in PowerShell is set at Restricted, which means any script(any .ps1 file) cannot be executed from within PowerShell. This includes profile scripts (which I'll get to in a moment), so for this example we'll configure the shell to run scripts "Unrestricted".

From a Powershell prompt, type:

```
Set-ExecutionPolicy Unrestricted
```

Configure the SCE Powershell Profile

Powershell allows special scripts, called **Profiles**, to be run to run during Shell startup to allow for user customization of the shell environment. The files that can be used (and are read in this order):

1. Documents and Settings\All Users\Documents\WindowsPowershell\Profile.ps1

2. Documents and Settings\All Users\Documents\ WindowsPowershell \Microsoft.PowerShell_Profile.ps1
3. \$HOME\My Documents\ WindowsPowershell\profile.ps1
4. \$HOME\My Documents\ WindowsPowershell\Microsoft.PowerShell_profile.ps1

We are going to use #2, since this will allow us to customize the shell environment to make it MOM friendly for all SCE administrators that launch Powershell on the Mgmt Server.

There are actually two ways to incorporate the ‘SCE profile’ into Powershell. If you follow the directions in the next 3 steps under option 1, you will actually integrate the SCE profile globally into Powershell, connecting to SCE every time you launch Powershell. If you would like to control whether you connect to SCE, use the option to create a separate shortcut to load the SCE profile contained in next set of steps labeled “**Option 2: Custom SCE Command Shell Shortcut**”.

Option 1: Powershell Profile Script

Perform the following actions on the SCE 2007 Management Server:

1. If it does not already exist, create a file called **Microsoft.PowerShell_Profile.ps1**
2. in the **Documents and Settings\All Users\Documents\Powershell** directory
3. Cut-and-paste the following code into this file, which will automatically load the SCE .NET Class Libraries and instantiate a connection to the SCE Management Group.

*****start copy below this line*****

```
[System.Reflection.Assembly]::LoadFile("$($env:ProgramFiles)\System Center Essentials
2007\SDK Binaries\Microsoft.EnterpriseManagement.OperationsManager.Common.dll") >
$null
```

```
[System.Reflection.Assembly]::LoadFile("$($env:ProgramFiles)\System Center Essentials
2007\SDK Binaries\Microsoft.EnterpriseManagement.OperationsManager.dll") > $null
```

```
$sce = [Microsoft.EnterpriseManagement.ManagementGroup]::connect("localhost")
```

```
Function global:prompt{
    "SCE Management Group: " + $($sce.Name) + "`n" + $(Get-Location) + ">";
}
```

*****end copy above this line*****

Note:

We have added a custom prompt so that the SCE command shell appears different from the default Powershell prompt. We basically took the format of the Ops Mgr 2007 command shell prompt and customized it for SCE.

Option 2: Custom SCE Command Shell Shortcut

If you do not want to connect to your SCE Server every time you launch Powershell, you could save the above commands to a custom script name and incorporate it into a shortcut with a custom run statement, as explained below.

Perform the following action to create a separate shortcut on the SCE 2007 Management Server

1. Save the code snippet from above in the *Documents and Settings\All Users\Documents\WindowsPowershell* and call it *SCE.CustomStartup.ps1*
2. Copy the Powershell shortcut from the Start Menu to your desktop. At this point you can change the name of the shortcut to whatever you want. (i.e. SCE Command Shell)
3. Edit the Powershell shortcut that was copied to the desktop.
4. Set the **Target** to:
C:\WINDOWS\system32\windowspowershell\v1.0\powershell.exe -NoExit
.\SCE.CustomStartup.ps1
5. Set the **Start In** to:
"C:\Documents and Settings\All Users\Documents\WindowsPowershell"
6. Hit **Okay** and it should be ready for use.

Either way, when you launch Powershell on your SCE Server, a SCE administration object is instantiated automatically. To test this, perform the following command to retrieve the properties and methods available to us through this new object

```
$sce | get-member
```

This should return a long list of properties and methods (for full output of this command, click [HERE](#))

Now what can I do? Well, since we don't have any of the custom cmdlets you have with Ops Mgr, some simple Powershell script-writing will be required.

Sample Scripts

Some sample one-liners to get you started are listed below. Note when doing a copy-and-paste that line wrap has pushed some of these one-lines to a 2nd line.

```
# Retrieve list of installed management packs
$sce.GetManagementPacks() |Format-Table FriendlyName, Version, ID, Sealed

# Retrieve list of overrides created on or after 04/25/2007
$sce.GetMonitoringOverrides() |where-object {$_.TimeAdded -ge '04/25/2007'} | Format-Table name, Description

# Retrieve list of overrides with description field matching a given string - "Pete"
in this example (using
```

```
# the description field may be a good way to easily track your own overrides)

$sce.GetMonitoringOverrides() | Where-Object {$_.description -like '*Pete*'} | Format-Table name, Description

#Inventory of agent-managed computers sorted by installation date

$sce.GetAdministration().GetAllAgentManagedComputers() | Sort -desc InstallTime | Format-Table ComputerName, Version, ProxyingEnabled, InstalledBy, InstallTime

# Retrieve a list of Agents with a HealthState of Error

$sce.GetAdministration().GetAllAgentManagedComputers() | Where-Object {$_.HealthState -eq "Error"}

# Retrieve a list of all Network Devices being monitored by SCE

$sce.GetAdministration().GetAllRemotelyManagedDevices()

# Retrieve a list of all open Alerts that match the Severity of "Error"

$sce.GetMonitoringAlerts() | Where-Object {$_.Severity -eq "Error"} | Format-List Name, Description, Severity, NetbiosComputerName
```

NOTE: The good news about the SCE .NET libraries is that they are remotable. This means we should be able to configure the SCE Command Shell on any machine with the SCE Administration Console installed and a local copy of the SDK binaries. We'll attack this scenario in part 2.

Feedback

We hope you've found this document helpful and would like to hear your ideas on how you'd like to leverage Powershell in your SCE environment. Please send your feedback to administrator AT systemcenterforum.org and we'll try to incorporate any feedback into part 2.