MPAuthor   
Orchestrator Sample Management Pack

Completed by Brian Wren, Microsoft Corporation

# Version History

|  |  |  |
| --- | --- | --- |
| Version | Date | Description |
| 1.0.0.74 | May 25, 2012 | Initial version. |
| 1.0.0.79 | August 9, 2013 | Update to Runbook Discovery script to correct problem with Windows Server 2012. |

# Introduction

This document describes the sample management pack MPAuthor.Orchestrator. It provides instructions on setting up the application components monitored by the management pack, a high level discussion of its design strategy, and detailed information on its individual contents.

## Included Files

The following files are included in this sample:

|  |  |
| --- | --- |
| File | Description |
| MPAuthor.Orchestrator.mp | Sealed Management Pack file with all classes, monitors, and rules. |
| MPAuthor. Orchestrator.xml | Unsealed version of MPAuthor. Orchestrator.mp. Included for reference only. The sealed mp file is the one that should be installed. |
| MPAuthor.Orchestrator.zip | Compressed file including the Visual Studio project with the source for the complete management pack. |
| MPAuthor Orchestrator Sample Management Pack.docx | This document. |

## Prerequisites

The following components are required for the use of this sample:

|  |  |
| --- | --- |
| Component | Description |
| Operations Manager 2007 R2 or System Center 2012 Operations Manager | Required for installation of the management pack files. |
| Microsoft Visual Studio 2010 Professional | Required to open the Visual Studio project. Not required to use the sample management pack or view/edit the XML files. |
| [Visual Studio Authoring Extensions](http://social.technet.microsoft.com/wiki/contents/articles/5236.visual-studio-authoring-extensions-for-system-center-2012-operations-manager-en-us.aspx) |
| [Orchestrator Monitoring Pack](http://www.microsoft.com/en-us/download/details.aspx?id=29269) | Required management pack reference. |

# Installing and Configuring the Management Pack

1. Install MPAuthor.Orchestrator.mp by using the [standard process to install a management pack](http://technet.microsoft.com/en-us/library/hh212691.aspx).
2. Select one or more computers that will run all the runbook discoveries, monitors, and rules. Only one computer should be a selected for each Orchestrator management group. This will presumably be the Orchestrator management server, but can be any computer with an Operations Manager 2007 or 2012 agent.
3. On the computer to act as the Runbook Host, create the following registry keys:
   * HKLM\SOFTWARE\MPAuthor
   * HKLM\SOFTWARE\MPAuthor\Orchestrator
4. On the computer to act as the Runbook Host, create the following registry string values:
   * HKLM\SOFTWARE\MPAuthor\Orchestrator\WebServer
   * HKLM\SOFTWARE\MPAuthor\Orchestrator\WebServicePort
5. Populate the registry values with the name of the computer and the port of the Orchestrator Web Service to use for discoveries and monitors. The web service could be located on the same computer as the Runbook Host or a difference computer.
6. Create a Run As Account with a domain account that has the authority to access the Orchestrator web service. The account type should be Windows. This is the account that will be used for all discovery and monitoring of runbooks. Assign the Run As Account to the Run As Profile called Orchestrator Web Service Account.
7. Once you have configured the Run As Profile, then create an override to enable the Object Discovery called Discover Runbooks.
8. If you want to monitor the running state of runbooks that start with a monitor activity, then you must use an override to enable the monitor Runbook Running. Either create a separate override for each runbook that you want to monitor, or create a group with runbooks you want to monitor and then create an override for the group.

# Setting up the Visual Studio Project

The .mp file in this sample has been sealed with key pair that is not included with the download. You can install the .mp files in a management group, but you cannot modify the management pack and then reseal it because you don’t have access to the original key pair.

If you want to modify the management packs and then reseal it, then you must obtain a new key pair and use it to seal the XML files. Any references to the management pack must then be changed to reflect the new public key token.

You can obtain a new key pair by following the procedure documented in [How to Seal a Management Pack File](http://technet.microsoft.com/en-us/library/hh457550.aspx). If you are using Visual Studio Authoring Extensions, then you can use the following procedure to change the key file in the projects.

1. Copy your key file to a known location.
2. In Visual Studio, right click a project and select **Properties**.
3. Select the **Build** tab.
4. Change the path of the **Key File** to the correct one for your file.
5. Repeat steps 2-4 for each project.

# Service Model

Figure 6 illustrates the service model for the Orchestrator sample management pack.



Figure 6 - MPAuthor Orchestrator Service Model

Details on each class are provided in Table 1.

Table 1 - MPAuthor Orchestrator Classes

|  |  |  |
| --- | --- | --- |
| Class Name | Base Class | Description |
| Orchestrator Runbook Host | Microsoft.Windows.LocalApplication | Acts as a target for the runbook discoveries and monitors. Class itself is not monitored. |
| Orchestrator Runbook | Microsoft.Windows.ApplicationComponent | Represents an Orchestrator runbook. |
| MPAuthor Monitor Runbooks | Microsoft.SystemCenter.InstanceGroup | Group containing all runbooks that start with a monitor activity. |

# Discovery

The following table provides the method that is used to discover each class.

Table 2 –Discovery Logic

|  |  |  |
| --- | --- | --- |
| Class | Method | Details |
| Orchestrator Runbook Host | Registry | Discovered from existence of registry key at HKLM\SOFTWARE\MPAuthor\Orchestrator.   * WebServer property collected from HKLM\SOFTWARE\MPAuthor\Orchestrator\WebServer. * WebServicePort property collected from HKLM\SOFTWARE\MPAuthor\Orchestrator\WebServicePort. |
| Orchestrator Runbook | Script | Runs on the Runbook Host and discovers all runbooks available from the Orchestrator web service.  This runbook is disabled by default. The user is expected to enable it through an override once the Run As Profile has been configured properly. |
| Populate Monitor Runbooks | Group Populator | Populate the MPAuthor Monitor Runbooks group with all runbooks that have an IsMonitor value of True. |

# Monitoring

## Scripts

All of the scripts in this management pack uses functions from [Orchestrator Web Service PowerShell module available on CodePlex](http://orchestrator.codeplex.com/releases/view/82959). This module provides critical PowerShell functions using the Orchestrator web service. The functions are left intact with additional logic created in each script to perform the specific required function.

## Unit Monitor

The monitor MPAuthor.Orchestrator.Monitor.RunbookRunning runs a script that returns all runbooks that start with a monitor activity and whether the runbook currently is running. It determines this by looking for any active jobs for the runbook.

This monitor is disabled by default because it is expected that not all runbooks should be monitored. It is intended that it be enabled through an override that is created either for specific runbooks or on a group containing critical runbooks to monitor.

## Rules

The rule MPAuther.Orchestrator.Rule.RunbookDiscovery and MPAuther. Orchestrator.Rule.RunbookMonitor look for an event created by the runbook discovery and monitor indicating that no runbooks were found. This is most likely due to the web server and port being configured incorrectly or the specific credentials not having required access (or no credentials being supplied at all). This is simply an alert to the user that the configuration of the management pack may be incorrect.

## Task

The task MPAuthor.Orchestrator.Task.StartRunbook provides the ability to start a runbook from the Operations Console. If the runbook has no parameters, then it can be started just be running the task. If the runbook requires parameters, then they need to be specified with an override on a single line in the ParameterString parameter on the task. The parameters need to be specified in the following format:

Param1=Value1;Param2=Value2;Param3=Value3

The reason that this format is used is because of the inability to create a more sophisticated user interface in the Operations Console for collecting parameters. The Orchestration Console is designed for exactly this purpose and prompts the user for a value for each parameter.

The StartRunbook task uses a write action module called MPAuthor.Orchestrator.WriteAction.StartRunbook. This module is public and can be used by other management packs that require the ability to start a runbook.