**SCOM Management Pack Creator Guide**

**By Dujon Walsham**

Contents

[Introduction 6](#_Toc12565717)

[How does it Work 6](#_Toc12565718)

[Sample Management Pack Guide 7](#_Toc12565719)

[Import the SCOM Management Pack Creator PS Module 7](#_Toc12565720)

[Create Visual Studio Management Pack Project 8](#_Toc12565721)

[Creation of MPX Files 10](#_Toc12565722)

[Explanation of the Get-SCOMClassID Function 11](#_Toc12565723)

[Create Classes 12](#_Toc12565724)

[New Class 12](#_Toc12565725)

[Add Classes 13](#_Toc12565726)

[Add Properties 13](#_Toc12565727)

[Add a Run as Account 14](#_Toc12565728)

[Add a Computer Group Discovery 14](#_Toc12565729)

[Add an Instance Group Discovery 14](#_Toc12565730)

[Completion of Class Creation 14](#_Toc12565731)

[Create Relationship 15](#_Toc12565732)

[Add Relationship 15](#_Toc12565733)

[Completion of Relationship Creation 16](#_Toc12565734)

[Create Discovery 16](#_Toc12565735)

[Add PowerShell Discovery 17](#_Toc12565736)

[Add Registry Discovery 18](#_Toc12565737)

[Add WMI Discovery 19](#_Toc12565738)

[Add VBScript Discovery 19](#_Toc12565739)

[Add UNIX Discovery 20](#_Toc12565740)

[Add Computer Group Discovery 21](#_Toc12565741)

[Add Instance Group Discovery 21](#_Toc12565742)

[Completion of Discovery File 22](#_Toc12565743)

[Create Folders 22](#_Toc12565744)

[Add Folders 23](#_Toc12565745)

[Completion of Folder File 23](#_Toc12565746)

[Create Views 24](#_Toc12565747)

[Add Views 24](#_Toc12565748)

[Completion of View File 25](#_Toc12565749)

[Create Monitors 25](#_Toc12565750)

[Add Custom Monitor using PowerShell 26](#_Toc12565751)

[Add Windows Event Monitor 29](#_Toc12565752)

[Add Windows Event Manual Reset 30](#_Toc12565753)

[Add Windows Event Timer Reset 31](#_Toc12565754)

[Add Windows Service Monitor 31](#_Toc12565755)

[Add Windows Service Monitor CPU Performance 32](#_Toc12565756)

[Add Windows Service Monitor Memory Performance 32](#_Toc12565757)

[Add Windows Generic Log Monitor 33](#_Toc12565758)

[Add Windows Performance Monitor 33](#_Toc12565759)

[Add Windows Event Rule 34](#_Toc12565760)

[Add Windows PowerShell Script Rule 35](#_Toc12565761)

[Add Windows Performance Rule 36](#_Toc12565762)

[Add UNIX Log File Monitoring Rule 37](#_Toc12565763)

[Create Tasks 37](#_Toc12565764)

[Create Diagnostic/Recovery Tasks 38](#_Toc12565765)

[Functions 39](#_Toc12565766)

[**New-SCOMMPClass** 39](#_Toc12565767)

[Switches 39](#_Toc12565768)

[**Add-SCOMMPClass** 39](#_Toc12565769)

[Switches 39](#_Toc12565770)

[**Get-SCOMClassID** 40](#_Toc12565771)

[Switches 40](#_Toc12565772)

[**Add-SCOMMPClassProperty** 40](#_Toc12565773)

[Switches 41](#_Toc12565774)

[**Add-SCOMMPRunAsAccount** 41](#_Toc12565775)

[Switches 41](#_Toc12565776)

[**New-SCOMMPRelationship** 41](#_Toc12565777)

[Switches 41](#_Toc12565778)

[**Add-SCOMMPRelationship** 42](#_Toc12565779)

[Switches 42](#_Toc12565780)

[**New-SCOMMPDiscovery** 42](#_Toc12565781)

[Switches 42](#_Toc12565782)

[**Add-SCOMMPPowerShellDiscovery** 42](#_Toc12565783)

[Switches 42](#_Toc12565784)

[**Add-SCOMMPRegistryDiscovery** 43](#_Toc12565785)

[Switches 43](#_Toc12565786)

[**Add-SCOMMPRegistryKey** 44](#_Toc12565787)

[Switches 44](#_Toc12565788)

[**Add-SCOMMPWMIDiscovery** 44](#_Toc12565789)

[Switches 44](#_Toc12565790)

[**Add-SCOMMPVBScriptDiscovery** 45](#_Toc12565791)

[Switches 45](#_Toc12565792)

[**Add-SCOMMPUnixShellCommandDiscovery** 46](#_Toc12565793)

[Switches 46](#_Toc12565794)

[**Add-SCOMMPComputerGroupDiscovery** 47](#_Toc12565795)

[Switches 47](#_Toc12565796)

[**Add-SCOMMPInstanceGroupDiscovery** 47](#_Toc12565797)

[Switches 47](#_Toc12565798)

[**Create-PowerShellScript** 48](#_Toc12565799)

[Switches 48](#_Toc12565800)

[**Create-VBScript** 48](#_Toc12565801)

[Switches 48](#_Toc12565802)

[**New-SCOMMPView** 48](#_Toc12565803)

[Switches 48](#_Toc12565804)

[**Add-SCOMMPView** 48](#_Toc12565805)

[Switches 48](#_Toc12565806)

[**New-SCOMMPFolder** 49](#_Toc12565807)

[Switches 49](#_Toc12565808)

[**Add-SCOMMPFolder** 49](#_Toc12565809)

[Switches 49](#_Toc12565810)

[**New-SCOMMPMonitorRule** 50](#_Toc12565811)

[Switches 50](#_Toc12565812)

[**New-SCOMMPCustomProbeAction** 50](#_Toc12565813)

[Switches 50](#_Toc12565814)

[**Add-SCOMMPCustomProbeAction** 50](#_Toc12565815)

[Switches 50](#_Toc12565816)

[**Add-SCOMMPWindowsEventMonitor** 51](#_Toc12565817)

[Switches 51](#_Toc12565818)

[**Add-SCOMMPWindowsEventManualResetMonitor** 52](#_Toc12565819)

[Switches 52](#_Toc12565820)

[**Add-SCOMMPWindowsEventTimerResetMonitor** 53](#_Toc12565821)

[Switches 53](#_Toc12565822)

[**Add-SCOMMPWindowsServiceMonitor** 54](#_Toc12565823)

[Switches 54](#_Toc12565824)

[**Add-SCOMMPWindowsServicePerformanceMonitor** 55](#_Toc12565825)

[Switches 55](#_Toc12565826)

[**Add-SCOMMPWindowsGenericLogMonitor** 56](#_Toc12565827)

[Switches 56](#_Toc12565828)

[**Add-SCOMMPPerformanceMonitor** 57](#_Toc12565829)

[Switches 57](#_Toc12565830)

[**Add-SCOMMPWindowsEventRule** 58](#_Toc12565831)

[Switches 58](#_Toc12565832)

[**Add-SCOMMPWindowsPowerShellScriptRule** 58](#_Toc12565833)

[Switches 58](#_Toc12565834)

[**Add-SCOMMPPerfomanceRule** 59](#_Toc12565835)

[Switches 59](#_Toc12565836)

[**Add-SCOMMPUnixLogFileRule** 61](#_Toc12565837)

[Switches 61](#_Toc12565838)

[**Add-SCOMMPAgentTaskCommandLine** 62](#_Toc12565839)

[Switches 62](#_Toc12565840)

[**Add-SCOMMPAgentTaskScript** 62](#_Toc12565841)

[Switches 62](#_Toc12565842)

[**Add-SCOMMPAgentTaskUnixShell** 62](#_Toc12565843)

[Switches 63](#_Toc12565844)

[**Add-SCOMMPAgentTaskUnixScript** 63](#_Toc12565845)

[Switches 63](#_Toc12565846)

[**Add-SCOMMPDiagnosticTaskCommandLine** 63](#_Toc12565847)

[Switches 63](#_Toc12565848)

[**Add-SCOMMPRecoveryTaskCommandLine** 64](#_Toc12565849)

[Switches 64](#_Toc12565850)

# Introduction

I had created many customized management packs using Visual Studios within combination of the VSAE Authoring extensions application.

Writing an entire management can take a lot of time depending on the complexity requirements such as

* Classes
* Discoveries
* Views
* Monitors
* Rules

There have been methods to speed up and aid in the productivity such as using the VSAE authoring guide available on TechNet, opening a sealed management pack to look at the XML code and also using the existing codes from a management pack which you yourself had created.

One of the issues I sometimes faced was creating Monitors, Rules or discoveries using the templates as sometimes if you make edits directly to the XML file which can control anyone of these it will then be overwritten if you attempt to change anything inside of the template files used to create any of those specific objects within your management pack.

Now we step into the latest version, which is now the “**SCOM Management Pack Creator PowerShell Module”**

# How does it Work

The PowerShell Module is a way of taking the complexity of writing a management pack away, and consolidating into a powershell module which is more user friendly and more efficient when it comes to the speed and quality of the management pack development.

The entire structure of the management pack can be created such as;

* Classes
* Discoveries
* Run As Accounts
* Monitors
* Rules
* Tasks
* Diagnostic Tasks
* Recovery Tasks
* Knowledgebase Articles
* Views
* Folders

The great thing about this module is that it fully supports cross-platform management pack creations from Windows to UNIX/Linux.

# Sample Management Pack Guide

This section shows an example of how to build the entire management pack end to end using some of the CMDLets in this guide.

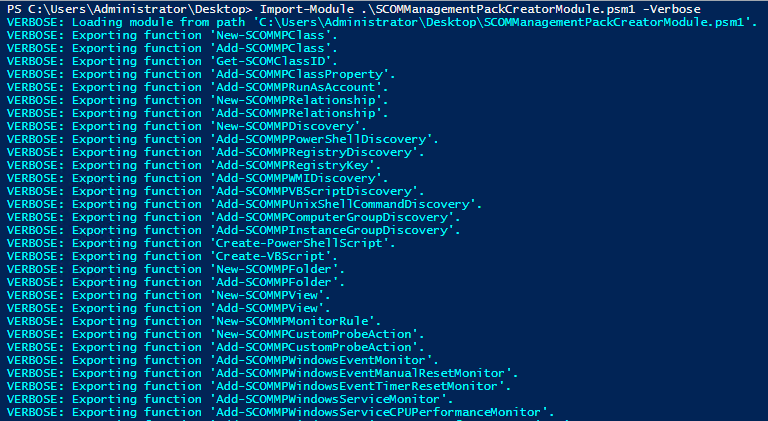
You will understand how to create the whole management pack.

(**Note:** Always best practice to run a “Build – Build Solution” each time you update the management pack so that you are able to pick up on any issues that arise.)

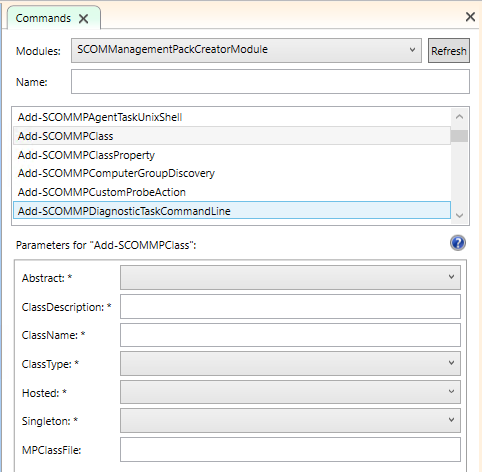
## Import the SCOM Management Pack Creator PS Module

First thing in which we need to do is to import the module into PowerShell

1. Copy the SCOMManagementPackCreatorModule.psm1 to a location which you desire
2. Open PowerShell with Administrative rights.
3. Enter the command **Import-Module .\SCOMManagementPackCreatorModule.psm1**



**Figure 1.1 –** Verbose information for all of the commands in the Module



**Figure 1.2 –** Commands list of the module for the SCOMManagementPackCreatorModule in the PowerShell ISE application

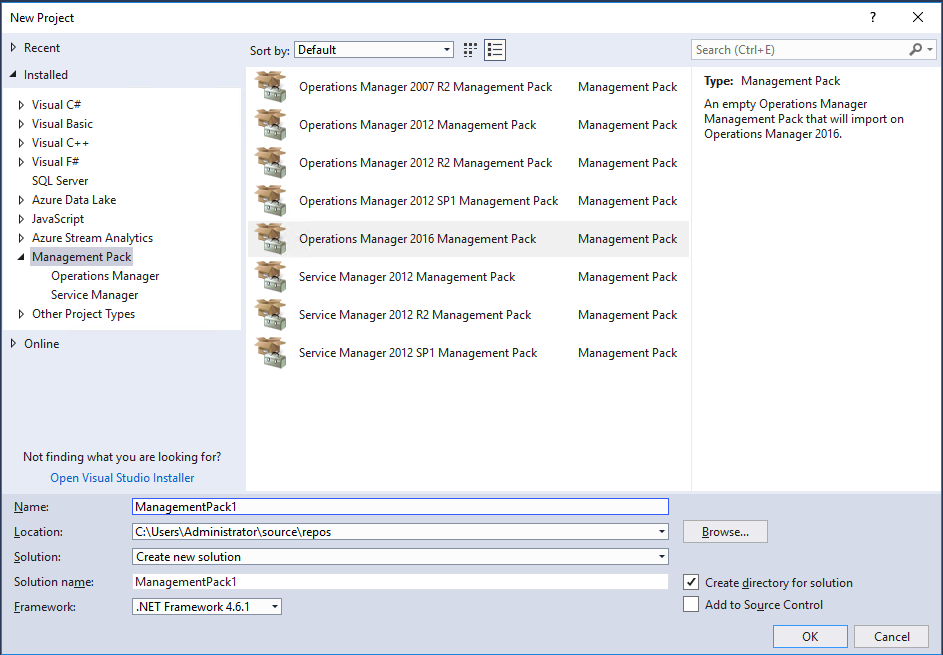
## Create Visual Studio Management Pack Project

Now we need to prepare a Management Pack project so that we can build the actual management pack for SCOM.

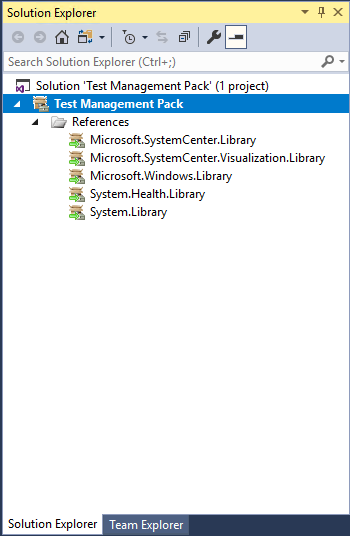
Prerequisites for this are the following;

* Visual Studios
* VSAE (Visual Studio Authoring Extensions) – Latest version can be found here <https://www.microsoft.com/en-gb/download/details.aspx?id=30169>

1. Open up Visual Studios
2. Go to New – Project
3. Select Management Pack – Operations Manager Management Pack which is applicable to the version of SCOM which you are currently using.

  
**Figure 1.3 –** Selection of Operations Manager management pack version for the Management Pack Development Project.

Once created you should see the following screenshot below detailing all of the default references for your new Management Pack Project.

  
**Figure 1.4 –** Management Pack project containing the References.

## Creation of MPX Files

In your management pack project, all files created which contain the XML code for your Management Pack are saved as an .MPX File.

With the module you will be able to create these MPX files beforehand which contain already the foundational code in order for you to not only build your management pack but also to add to each specific part of your management pack.

Each MPX file contains its own representation so that the entire management isn’t just changed, it’s more of a segregated section for each.

We have the following commands which can do this which are;

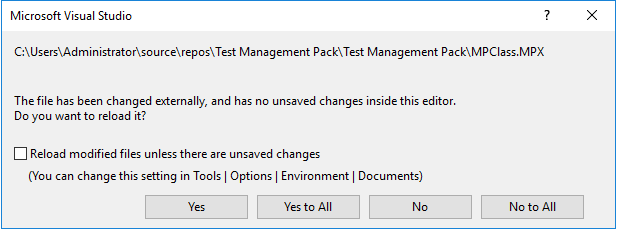
|  |  |
| --- | --- |
| Command | Description |
| New-SCOMMPClass | Creates an MPX file to create Classes for your MP |
| New-SCOMMPRelationship | Creates an MPX file to create Relationships for your MP |
| New-SCOMMPDiscovery | Creates an MPX file to create Discoveries for your MP |
| New-SCOMMPView | Creates an MPX file to create Views for your MP |
| New-SCOMMPFolder | Creates an MPX file to create Folders for your MP |
| New-SCOMMPMonitorRule | Creates an MPX file to create Monitors and Rules for your MP |
| New-SCOMMPCustomProbeMonitor | Creates several MPX files to develop a custom Probe Monitor – specific for PowerShell based monitors |

Not only just MPX files but there are other commands which utilise PowerShell Scripts and VB Scripts and therefore there are commands which also support this function.

|  |  |
| --- | --- |
| Command | Description |
| Create-PowerShellScript | Creates a PowerShell Script for your Discovery if using a PowerShell based Discovery |
| Create-VBScript | Creates a VB Script for your Discovery if using a VB Script based discovery |

Running these commands with pointing them to the directory of your management pack project will allow you to have great control and editing power over the whole process. So in this example my location of the Management Pack is project is based in **C:\Users\Administrator\source\repos\Test Management Pack\Test Management Pack** so this would be the location I would create the MPX files in.

Once importing an MPX file to your project, you should get a notification below which will prompt to refresh or automatically refresh.

  
**Figure 1.5 –** Notification of changes to your MPX file once additional information has been added to your MPX files.

## Explanation of the Get-SCOMClassID Function

The **Get-SCOMClassID** is a very powerful function within this module, and in fact makes the entire process so much easier as well as automated.

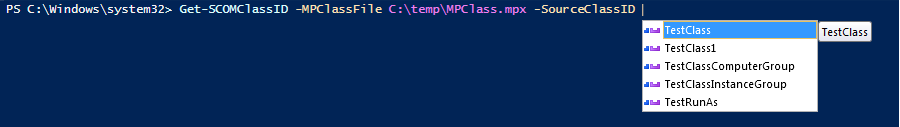
This command is able to analyse the following MPX files

* Class MPX File
* Folder MPX file
* Monitors and Rules MPX file

The main purpose is to be able to read all of the referenced classes within each one, as you will find there are a lot of components to the management pack in which you will want to create, and to look through each part of your Management Pack to find out exactly what ID name you gave a particular class, discovery, monitor or rule etc. can be very time consuming.

So this function can actually dynamically read every single thing in the MPX files and has the ability to pass them over to different parameters in other functions via pipeline so that you don’t need to manually look through any MPX files at any time during the whole process.

For example, if I want to create a property for a class, and I’ve created multiple classes I may not remember the exact name so if use the **Get-SCOMClassID** function we have the following below;

  
**Figure 1.6 –** Get-SCOMClassID function in action

As you can see it shows a dynamic list of every single class I have created within the class MPX file, this has been extended for the Folders and Monitors/Rules MPX files so that we can use them in later functions.

## Create Classes

So we have the following commands available for this process which are

|  |  |
| --- | --- |
| Command | Description |
| Add-SCOMMPClass | Adds a class to your MP |
| Add-SCOMMPClassProperty | Adds a property to a class in your MP |
| Add-SCOMMPRunAsAccount | Adds a Run As Account Profile to your classes in your MP |

### New Class

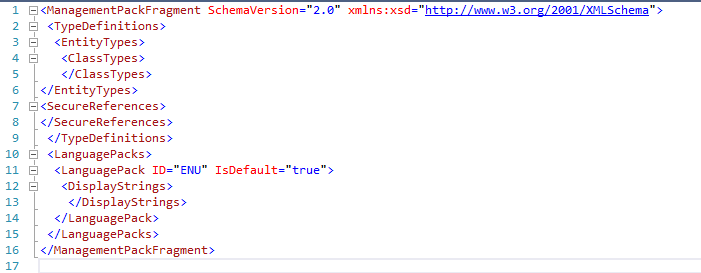
So if we wanted to create a new class we would run the following command;

New-SCOMMPClass -MPClassFile "C:\Users\Administrator\source\repos\Test Management Pack\Test Management Pack\MPClass.mpx"

This will create an MPX file which will reside in your project. Once this is created we would then do the following;

1. Right click the name of the Management Pack and choose Add – Existing Items
2. Select the Class MPX file you have just created

This will then display the class you have just created



**Figure 1.7 –** View of the Class MPX before anything has been added to it

### Add Classes

Now this is created and added into your Management Pack project, we can now add classes to this and we should be able to see either an automatic update or a prompt to refresh the MPX file so that we can see its new changes.

So to add a new class we would use the Add-SCOMMPClass command, here’s an example of this

Add-SCOMMPClass -ClassName TestClass -ClassType WindowsComputer -ClassDescription Test -Abstract false -Hosted true -Singleton false -MPClassFile "C:\Users\Administrator\source\repos\Test Management Pack\Test Management Pack\MPClass.mpx"

You can add additional classes as much as you need, so to create another class we run the following;

Add-SCOMMPClass -ClassName TestClass1 -ClassType WindowsComputer -ClassDescription Test -Abstract false -Hosted true -Singleton false -MPClassFile "C:\Users\Administrator\source\repos\Test Management Pack\Test Management Pack\MPClass.mpx"

### Add Properties

Next we will add a property to the class which you will want the class to be associated with so that you can run discoveries for which creates the attributes for the class.

So for example if we want to discover a property such as “Name” we would run the following which adds a property to each created class.

Get-SCOMClassID -MPClassFile "C:\Users\Administrator\source\repos\Test Management Pack\Test Management Pack\MPClass.mpx" -SourceClassID TestClass | Add-SCOMMPClassProperty -PropertyName Name -PropertyType string -KeyValue true -PropertyDescription Test

The second one below will create a property called “Version” for the secondary test class which will be created.

Get-SCOMClassID -MPClassFile "C:\Users\Administrator\source\repos\Test Management Pack\Test Management Pack\MPClass.mpx" -SourceClassID TestClass1 | Add-SCOMMPClassProperty -PropertyName Version -PropertyType string -KeyValue true -PropertyDescription Test

### Add a Run as Account

Next we will add a Run as Account profile which can be used to associate an account which requires enough privileges to be able to formulate the classes and properties which we have just created.

Get-SCOMClassID -MPClassFile "C:\Users\Administrator\source\repos\Test Management Pack\Test Management Pack\MPClass.mpx" -SourceClassID TestClass | Add-SCOMMPRunAsAccount -SecureReferenceName TestRunAs -SecureReferenceDescription Test

### Add a Computer Group Discovery

To create a Computer Group discovery for the class you have created run the following,

Add-SCOMMPClass -ClassName TestClassComputerGroup -ClassType ComputerGroup -ClassDescription Test -Abstract false -Hosted false -Singleton true -MPClassFile "C:\Users\Administrator\source\repos\Test Management Pack\Test Management Pack\MPClass.mpx"

### Add an Instance Group Discovery

Next you need to add references unless you do not intend to use specific functions in the module i.e. if creating an Instance Group class you will require the **Microsoft.SystemCenter.InstanceGroup.Library** which can be found in the install directory of your System Center Authoring extensions C:\Program Files (x86)\System Center Authoring Extensions\References.

(**Note:** You may need to change the Alias of the reference management pack just in case if the “Build Solution” option in Visual Studio fails as the alias referenced in the “ID” or “TypeID” has to match the alias given on the reference MP i.e. “MSIL!” should match the alias on the Microsoft.SystemCenter.InstanceGroup.Library” alias)

Add-SCOMMPClass -ClassName TestClassInstanceGroup -ClassType InstanceGroup -ClassDescription Test -Abstract false -Hosted false -Singleton true -MPClassFile "C:\Users\Administrator\source\repos\Test Management Pack\Test Management Pack\MPClass.mpx"

### Completion of Class Creation

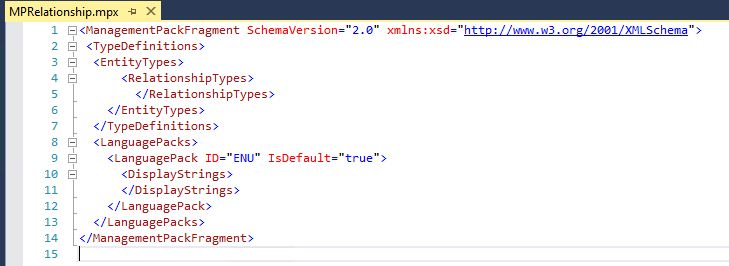
Once finished with the creation of the Class, it should look something like this;

  
**Figure 1.8 –** Completed Class MPX in your Management Pack Project

## Create Relationship

If wanting to create a relationship between both classes so that you can use them for references and nesting/hosting, then we can run the command to create a Relationship MPX file.

New-SCOMMPRelationShip -MPRelationshipFile "C:\Users\Administrator\source\repos\Test Management Pack\Test Management Pack\MPRelationship.mpx"



**Figure 1.9 –** View of the Relationship MPX before anything has been added to it

### Add Relationship

To create a relationship between both classes which we created earlier, you can run the example below

Get-SCOMClassID -MPClassFile "C:\Users\Administrator\source\repos\Test Management Pack\Test Management Pack\MPClass.mpx" -SourceClassID TestClass -TargetClassID TestClass1 | Add-SCOMMPRelationship -RelationshipName testRelationship -RelationshipDescription Test -Abstract false -Accessibility Internal -MPRelationshipFile "C:\Users\Administrator\source\repos\Test Management Pack\Test Management Pack\MPRelationship.mpx"

### Completion of Relationship Creation

Your relationship file should look something similar to this

  
**Figure 1.10 –** Completed Relationship MPX in your Management Pack Project

## Create Discovery

As a Prerequisite you will need to add the following Reference MPs;

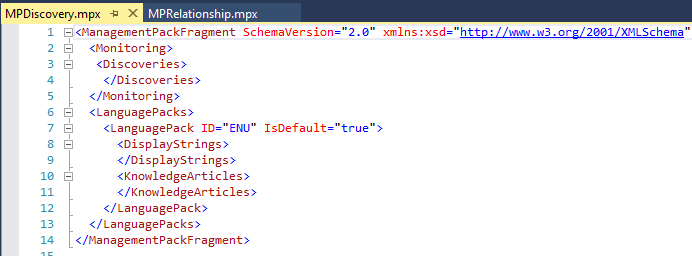
* System.AdminItem.Library
* System.SoftwareItem.Library

Which can be found in **C:\Program Files (x86)\System Center Authoring Extensions\References**

(**Note:** You may need to change the Alias of the reference management pack just in case if the “Build Solution” option in Visual Studio fails as the alias referenced in the “ID” or “TypeID” has to match the alias given on the reference MP i.e. “MSIL!” should match the alias on the Microsoft.SystemCenter.InstanceGroup.Library” alias)

To create the discovery file run the following command;

New-SCOMMPDiscovery -MPDiscoveryFile "C:\Users\Administrator\source\repos\Test Management Pack\Test Management Pack\MPDiscovery.mpx"



**Figure 1.11 –** View of the Discovery MPX before anything has been added to it

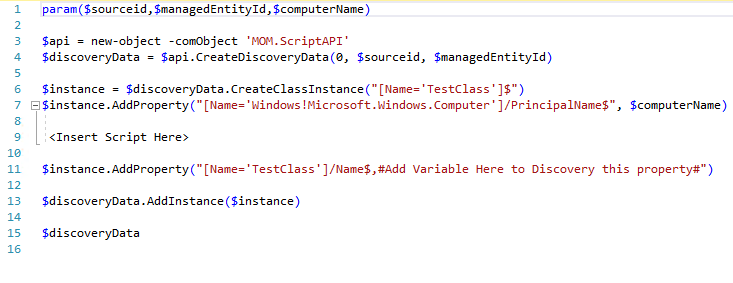
### Add PowerShell Discovery

There are a couple of bits to this as this depends on a PowerShell Script to be created, so in this case we will need to run a command to create the Powershell script which is the following below;

Get-SCOMClassID -MPClassFile "C:\Users\Administrator\source\repos\Test Management Pack\Test Management Pack\MPClass.mpx" -SourceClassID TestClass | Create-PowerShellScript -ScriptName "C:\Users\Administrator\source\repos\Test Management Pack\Test Management Pack\Test.ps1"

Once created you can add into the Management Pack project using the same process as before.

Your script should look something like this;

  
**Figure 1.12 –** PowerShell script which contains the classes and properties which you had created earlier.

Then once this is created we can run the following command line to create the PowerShell Discovery module;

Get-SCOMClassID -MPClassFile "C:\Users\Administrator\source\repos\Test Management Pack\Test Management Pack\MPClass.mpx" -SourceClassID TestClass -RunAsAccount TestRunAs | Add-SCOMMPPowerShellDiscovery -DiscoveryName TestDiscovery -DiscoveryTarget WindowsComputer -DiscoveryDescription TestDiscovery -IntervalSeconds 300 -ScriptName Test.ps1 -TimeoutSeconds 300 -KnowledgeArticle "This is a test article" -MPDiscoveryFile "C:\Users\Administrator\source\repos\Test Management Pack\Test Management Pack\MPDiscovery.mpx"

  
**Figure 1.13 –** PowerShell Discovery Module

### Add Registry Discovery

For creating a registry discovery you would run the following command;

Get-SCOMClassID -MPClassFile "C:\Users\Administrator\source\repos\Test Management Pack\Test Management Pack\MPClass.mpx" -SourceClassID TestClass -RunAsAccount TestRunAs | Add-SCOMMPRegistryDiscovery -DiscoveryName TestRegDiscovery -DiscoveryTarget WindowsComputer -DiscoveryDescription TestDiscovery -Frequency 300 -KnowledgeArticle "This is a test article" -MPDiscoveryFile "C:\Users\Administrator\source\repos\Test Management Pack\Test Management Pack\MPDiscovery.mpx"

In addition to this we will need to add a registry key attribute function which will allow us to discover the actual keys. It could be to find if a certain key exists, or if a key has a particular attribute. But it’s needed to be created so that the properties can be properly discovered.

This example is to check if a key exists;

Get-SCOMClassID -MPClassFile "C:\Users\Administrator\source\repos\Test Management Pack\Test Management Pack\MPClass.mpx" -SourceClassID TestClass | Add-SCOMMPRegistryKey -AttributeName TestName -RegistryPath HKLM\SOFTWARE\Microsoft\Test -PathType KeyExists -AttributeType Boolean -MPDiscoveryFile "C:\Users\Administrator\source\repos\Test Management Pack\Test Management Pack\MPDiscovery.mpx"

   
**Figure 1.14 –** Registry Discovery Module with registry key

### Add WMI Discovery

To add a WMI Discovery you can run the following sample command

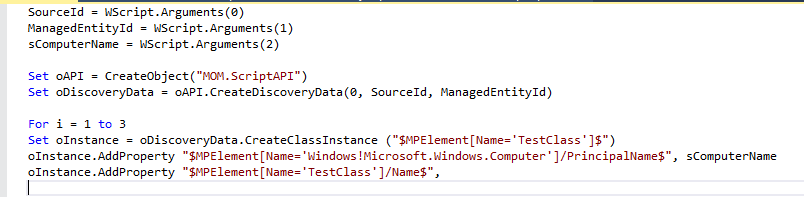
Get-SCOMClassID -MPClassFile "C:\Users\Administrator\source\repos\Test Management Pack\Test Management Pack\MPClass.mpx" -SourceClassID TestClass -RunAsAccount TestRunAs | Add-SCOMMPWMIDiscovery -DiscoveryName TestWMI -DiscoveryTarget WindowsComputer -DiscoveryDescription Test -Namespace SMS\_Collection -Query "select \* from vsms\_R\_system" -Frequency 300 -KnowledgeArticle "This is a test article" -MPDiscoveryFile "C:\Users\Administrator\source\repos\Test Management Pack\Test Management Pack\MPDiscovery.mpx"

  
**Figure 1.15 –** WMI Discovery Module

### Add VBScript Discovery

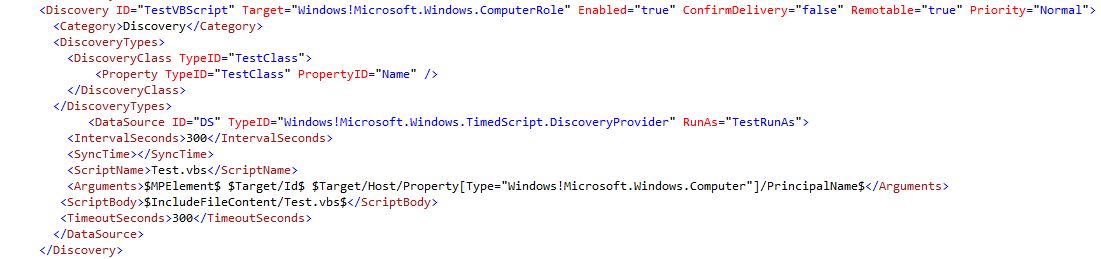
Similar to the Powershell module, you will need a VBScript reference beforehand. To create this run the following sample command;

Get-SCOMClassID -MPClassFile "C:\Users\Administrator\source\repos\Test Management Pack\Test Management Pack\MPClass.mpx" -SourceClassID TestClass | Create-VBScript -ScriptName "C:\Users\Administrator\source\repos\Test Management Pack\Test Management Pack\Test.vbs"

  
**Figure 1.16 –** VB Discovery script containing the custom classes and properties

Once created you can run the following command to create the VB script discovery module

Get-SCOMClassID -MPClassFile "C:\Users\Administrator\source\repos\Test Management Pack\Test Management Pack\MPClass.mpx" -SourceClassID TestClass -RunAsAccount TestRunAs | Add-SCOMMPVBScriptDiscovery -DiscoveryName TestVBScript -DiscoveryTarget WindowsComputer -DiscoveryDescription Test -IntervalSeconds 300 -ScriptName Test.vbs -TimeoutSeconds 300 -KnowledgeArticle "This is a test article" -MPDiscoveryFile "C:\Users\Administrator\source\repos\Test Management Pack\Test Management Pack\MPDiscovery.mpx"

  
**Figure 1.17 –** VB Discovery Module

### Add UNIX Discovery

If you are looking to create a management pack based off UNIX/Linux then you will need to obtain the **Unix.Authoring.Library** which is available on the TechNet Gallery <https://gallery.technet.microsoft.com/UNIXLinux-Authoring-b16fd2e4>

To create this discovery you will need to add the following Reference Management Pack

* Microsoft.Unix.Library

These can be downloaded from the Microsoft website for the UNIX/Linux Management Pack bundle here

(**Note:** You may need to change the Alias of the reference management pack just in case if the “Build Solution” option in Visual Studio fails as the alias referenced in the “ID” or “TypeID” has to match the alias given on the reference MP i.e. “MSIL!” should match the alias on the Microsoft.SystemCenter.InstanceGroup.Library” alias)

Get-SCOMClassID -MPClassFile "C:\Users\Administrator\source\repos\Test Management Pack\Test Management Pack\MPClass.mpx" -SourceClassID TestClass -RunAsAccount TestRunAs | Add-SCOMMPUnixShellCommandDiscovery -DiscoveryName TestUnix -DiscoveryTarget UnixComputer -DiscoveryDescription Test -ShellCommand grep -Interval 300 -Timeout 300 -Pattern \*test -KnowledgeArticle "This is a test article" -MPDiscoveryFile "C:\Users\Administrator\source\repos\Test Management Pack\Test Management Pack\MPDiscovery.mpx"

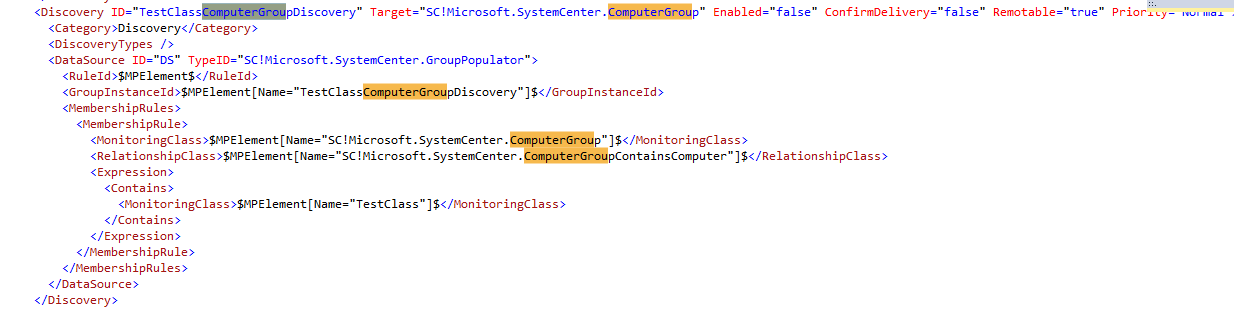
  
**Figure 1.18 –** UNIX Shell Command Discovery module

### Add Computer Group Discovery

To create the discovery related to the computer group you created earlier run the following

# New Computer Group Discovery

Get-SCOMClassID -MPClassFile "C:\Users\Administrator\source\repos\Test Management Pack\Test Management Pack\MPClass.mpx" -SourceClassID TestClass -TargetClassID TestClassComputerGroup | Add-SCOMMPComputerGroupDiscovery -DiscoveryName TestClassComputerGroupDiscovery -DiscoveryTarget ComputerGroup -DiscoveryDescription Test -KnowledgeArticle "This is a test article" -MPDiscoveryFile "C:\Users\Administrator\source\repos\Test Management Pack\Test Management Pack\MPDiscovery.mpx"



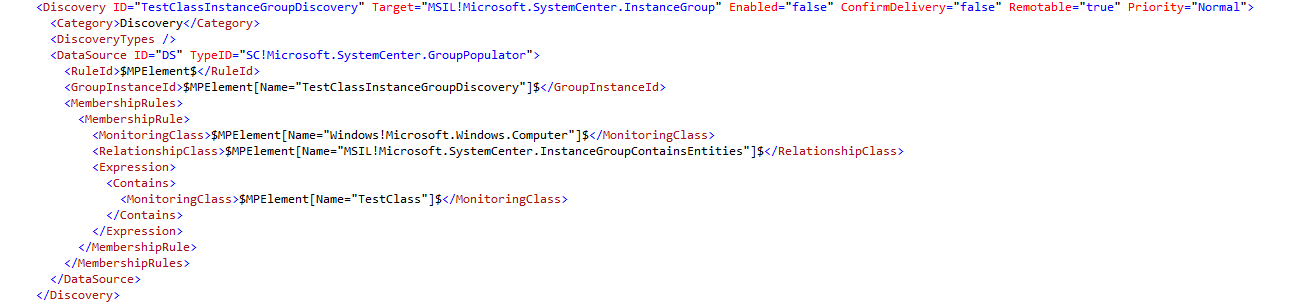
**Figure 1.19 –** ComputerGroup Discovery module

### Add Instance Group Discovery

To create the discovery related to the instance group you created earlier run the following

# New Instance Group Discovery

Get-SCOMClassID -MPClassFile "C:\Users\Administrator\source\repos\Test Management Pack\Test Management Pack\MPClass.mpx" -SourceClassID TestClass -TargetClassID TestClassInstanceGroup | Add-SCOMMPInstanceGroupDiscovery -DiscoveryName TestClassInstanceGroupDiscovery -DiscoveryTarget InstanceGroup -DiscoveryDescription Test -KnowledgeArticle "This is a test article" -MPDiscoveryFile "C:\Users\Administrator\source\repos\Test Management Pack\Test Management Pack\MPDiscovery.mpx"

  
**Figure 1.20 –** InstanceGroup Discovery module

### Completion of Discovery File

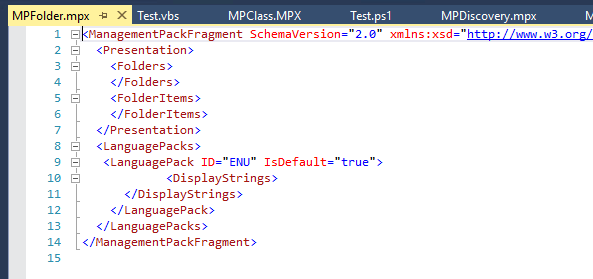
Your discovery MPX should now look something like this

  
**Figure 1.21 –** Completed Discovery MPX for your management pack project

## Create Folders

This is where you create your folders which show in the SCOM console for your custom management pack. To create the foundation for this run the following command below;

New-SCOMMPFolder -MPFolderFile "C:\Users\Administrator\source\repos\Test Management Pack\Test Management Pack\MPFolder.mpx"

  
**Figure 1.22 –** Folder MPX file when nothing has been added into it.

### Add Folders

To add folders in to the MPX file, for this example we can add two folders which will be nested into each other.

Add-SCOMMPFolder -FolderName TestFolder -MPFolderFile "C:\Users\Administrator\source\repos\Test Management Pack\Test Management Pack\MPFolder.mpx"

Get-SCOMClassID -MPFolderFile "C:\Users\Administrator\source\repos\Test Management Pack\Test Management Pack\MPFolder.mpx" -FolderID TestFolder | Add-SCOMMPFolder -FolderName Test1 -MPFolderFile "C:\Users\Administrator\source\repos\Test Management Pack\Test Management Pack\MPFolder.mpx"

### Completion of Folder File

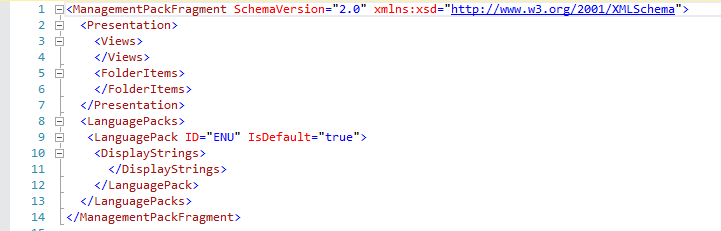
This is what your completed folder MPX file should look like

  
**Figure 1.23 –** CompletedFolder MPX file.

## Create Views

For creating custom views in which your classes will be shown in the SCOM console from run the following command;

New-SCOMMPView -MPViewFile "C:\Users\Administrator\source\repos\Test Management Pack\Test Management Pack\MPView.mpx"

  
**Figure 1.24 –** View MPX file whilst nothing has been added into it yet

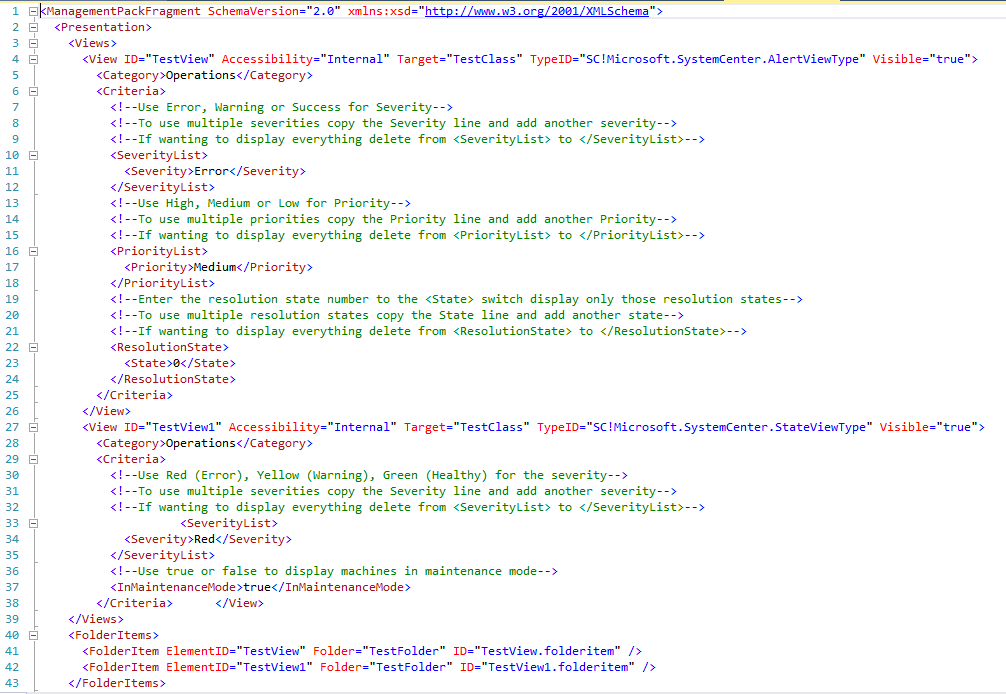
### Add Views

The following sample commands will create two views, one being an alert view and the other being a State View.

Get-SCOMClassID -MPFolderFile "C:\Users\Administrator\source\repos\Test Management Pack\Test Management Pack\MPFolder.mpx" -FolderID TestFolder -MPClassFile "C:\Users\Administrator\source\repos\Test Management Pack\Test Management Pack\MPClass.mpx" -TargetClassID TestClass | Add-SCOMMPView -ViewName TestView -ViewType AlertView -MPViewFile "C:\Users\Administrator\source\repos\Test Management Pack\Test Management Pack\MPView.mpx"

Get-SCOMClassID -MPFolderFile "C:\Users\Administrator\source\repos\Test Management Pack\Test Management Pack\MPFolder.mpx" -FolderID TestFolder -MPClassFile "C:\Users\Administrator\source\repos\Test Management Pack\Test Management Pack\MPClass.mpx" -TargetClassID TestClass | Add-SCOMMPView -ViewName TestView1 -ViewType StateView -MPViewFile "C:\Users\Administrator\source\repos\Test Management Pack\Test Management Pack\MPView.mpx"

### Completion of View File

  
**Figure 1.25 –** CompletedView MPX file

## Create Monitors

The foundation for creating Monitors, Rules, Tasks, and Diagnostic and recovery tasks.

To build the foundation run the following command

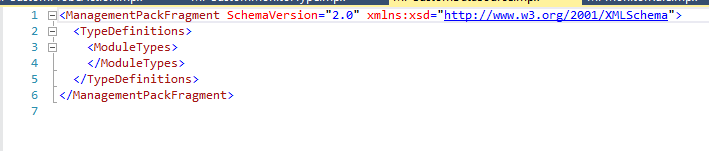
New-SCOMMPMonitorRule -MPMonitorRuleFile "C:\Users\Administrator\source\repos\Test Management Pack\Test Management Pack\MPMonitorRule.mpx"

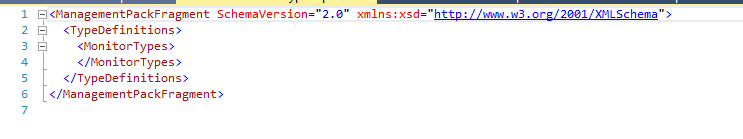
  
**Figure 1.26 –** Monitor/Rule MPX file before anything has been added to it

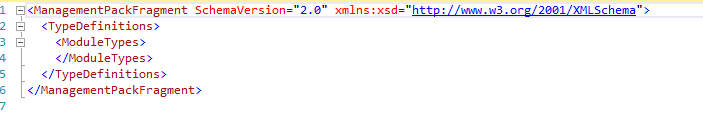
### Add Custom Monitor using PowerShell

To create a PowerShell based Monitor you will need to create a Custom Probe action, below is a command which will build out all of the relevant MPX files to construct the entire monitor.

New-SCOMMPCustomProbeAction -MPCustomDataSourceFile "C:\Users\Administrator\source\repos\Test Management Pack\Test Management Pack\MPCustomDataSource.mpx" -MPCustomMonitorTypeFile "C:\Users\Administrator\source\repos\Test Management Pack\Test Management Pack\MPCustomMonitorType.mpx" -MPCustomProbeActionFile "C:\Users\Administrator\source\repos\Test Management Pack\Test Management Pack\MPCustomProbeAction.mpx"





  
**Figure 1.27 –** CustomProbeAction, CustomDataSource and CustomMonitorType MPX files with nothing added to them yet.

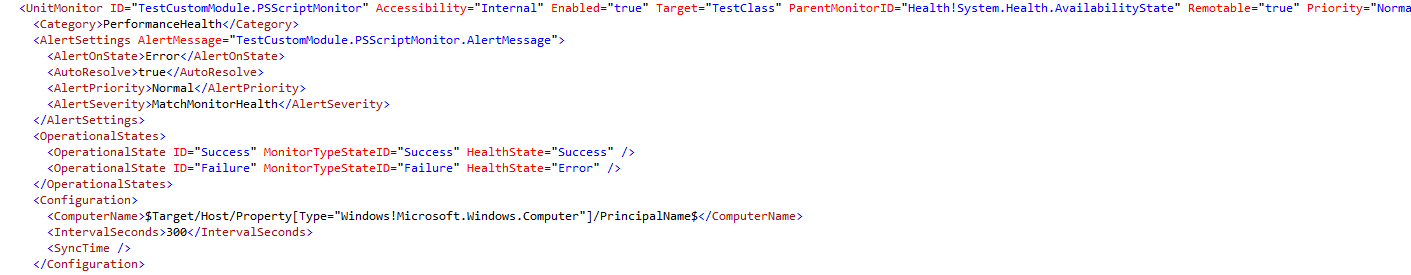
To build the monitor run the following command;

Get-SCOMClassID -MPClassFile "C:\Users\Administrator\source\repos\Test Management Pack\Test Management Pack\MPClass.mpx" -TargetClassID TestClass -RunAsAccount TestRunAs | Add-SCOMMPCustomProbeAction -CustomModuleName TestCustomModule -AlertName CustomModuleAlert -AlertMessage "This is a Test" -KnowledgeArticle "This is a Test Article" -TimeoutSeconds 300 -MPCustomDataSourceFile "C:\Users\Administrator\source\repos\Test Management Pack\Test Management Pack\MPCustomDataSource.mpx" -MPCustomMonitorTypeFile "C:\Users\Administrator\source\repos\Test Management Pack\Test Management Pack\MPCustomMonitorType.mpx" -MPCustomProbeActionFile "C:\Users\Administrator\source\repos\Test Management Pack\Test Management Pack\MPCustomProbeAction.mpx" -MPMonitorRuleFile "C:\Users\Administrator\source\repos\Test Management Pack\Test Management Pack\MPMonitorRule.mpx" -ScriptName Test.ps1 -ScriptOutput "C:\Users\Administrator\source\repos\Test Management Pack\Test Management Pack\test.ps1"









  
**Figure 1.28 –** Completed MPX files. Also you will see a PowerShell script which is used as the base for your monitor which will be controlled through a PowerShell script.

### Add Windows Event Monitor

To create a Windows Event Monitor run the following command;

Get-SCOMClassID -MPClassFile "C:\Users\Administrator\source\repos\Test Management Pack\Test Management Pack\MPClass.mpx" -TargetClassID TestClass -RunAsAccount TestRunAs | Add-SCOMMPWindowsEventMonitor -MonitorName TestEventMonitor -MonitorEnabled true -AlertOnState Error -AlertPriority High -AlertSeverity Error -AlertName TestEventAlert -UnhealthyLogName Application -UnhealthyEventDisplayNumber 1000 -UnhealthyPublisherName Test -HealthyLogName Application -HealthyEventDisplayNumber 1001 -HealthyPublisherName Test -MPMonitorRuleFile "C:\Users\Administrator\source\repos\Test Management Pack\Test Management

Pack\MPMonitorRule.mpx"

  
**Figure 1.29 –** Windows Event Monitor Module

### Add Windows Event Manual Reset

To create a Windows Event Manual Reset Monitor run the following command;

Get-SCOMClassID -MPClassFile "C:\Users\Administrator\source\repos\Test Management Pack\Test Management Pack\MPClass.mpx" -TargetClassID TestClass -RunAsAccount TestRunAs | Add-SCOMMPWindowsEventManualResetMonitor -MonitorName TestManualReset -MonitorEnabled true -AlertOnState Error -AlertPriority High -AlertSeverity Error -AlertName TestEventAlert1 -UnhealthyLogName Application -UnhealthyEventDisplayNumber 1003 -UnhealthyPublisherName Test -KnowledgeArticle "This is a test article" -MPMonitorRuleFile "C:\Users\Administrator\source\repos\Test Management Pack\Test Management Pack\MPMonitorRule.mpx"

  
**Figure 1.30 –** Windows Event Manual Reset Monitor Module

### Add Windows Event Timer Reset

To create a Windows Event Timer Reset Monitor run the following command;

Get-SCOMClassID -MPClassFile "C:\Users\Administrator\source\repos\Test Management Pack\Test Management Pack\MPClass.mpx" -TargetClassID TestClass -RunAsAccount TestRunAs | Add-SCOMMPWindowsEventTimerResetMonitor -MonitorName TestTimerReset -MonitorEnabled true -AlertOnState Error -AlertPriority High -AlertSeverity Error -AlertName TestTimer -UnhealthyLogName Application -UnhealthyEventDisplayNumber 1005 -UnhealthyPublisherName Test -TimerWaitInSeconds 300 -KnowledgeArticle "This is a test article" -MPMonitorRuleFile "C:\Users\Administrator\source\repos\Test Management Pack\Test Management Pack\MPMonitorRule.mpx"

  
**Figure 1.31 –** Windows Event Timer Reset Monitor Module

### Add Windows Service Monitor

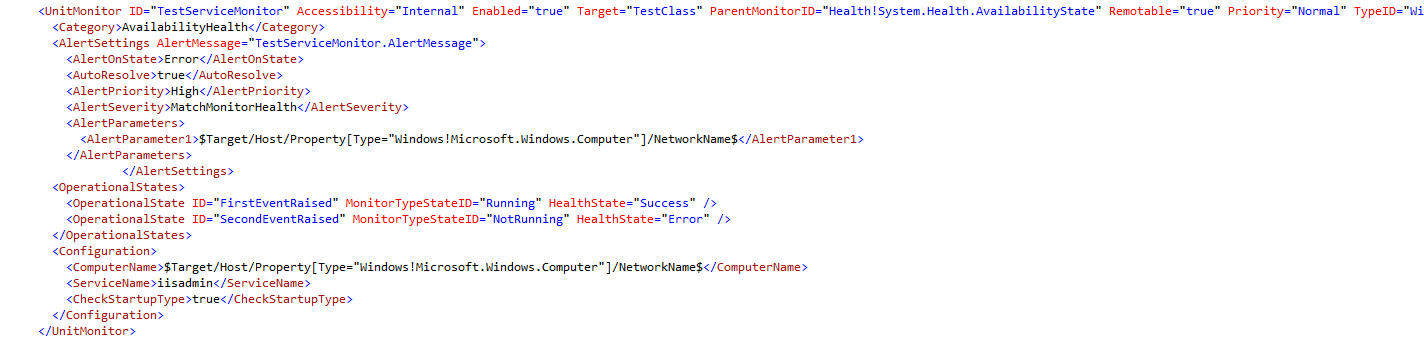
You will need to add the following reference management pack

* Microsoft.SystemCenter.NTService.Library

This can be found in the installation of the Visual Studio Authoring extensions **C:\Program Files (x86)\System Center Visual Studio Authoring Extensions\References**

(**Note:** You may need to change the Alias of the reference management pack just in case if the “Build Solution” option in Visual Studio fails as the alias referenced in the “ID” or “TypeID” has to match the alias given on the reference MP i.e. “MSIL!” should match the alias on the Microsoft.SystemCenter.InstanceGroup.Library” alias)

Get-SCOMClassID -MPClassFile "C:\Users\Administrator\source\repos\Test Management Pack\Test Management Pack\MPClass.mpx" -TargetClassID TestClass -RunAsAccount TestRunAs | Add-SCOMMPWindowsServiceMonitor -MonitorName TestServiceMonitor -MonitorEnabled true -AlertOnState Error -AlertPriority High -AlertSeverity Error -AlertName TestServiceAlert -AlertMessage "This Service is down" -ServiceName "iisadmin" -AlertOnAuto true -MPMonitorRuleFile "C:\Users\Administrator\source\repos\Test Management Pack\Test Management Pack\MPMonitorRule.mpx"

  
**Figure 1.32 –** Windows Service Monitor Module

### Add Windows Service Monitor CPU Performance

To create a Windows Service CPU Performance Monitor run the following command;

Get-SCOMClassID -MPClassFile "C:\Users\Administrator\source\repos\Test Management Pack\Test Management Pack\MPClass.mpx" -TargetClassID TestClass -RunAsAccount TestRunAs | Add-SCOMMPWindowsServiceCPUPerformanceMonitor -MonitorName TestServiceMonitorCPU -MonitorEnabled true -AlertOnState Error -AlertName TestCPU -AlertMessage Test -AlertPriority High -AlertSeverity Error -AlertOnAuto true -ServiceName iisadmin -Frequency 300 -Threshold 10 -NumSamples 3 -KnowledgeArticle "this is a test article" -MPMonitorRuleFile "C:\Users\Administrator\source\repos\Test Management Pack\Test Management Pack\MPMonitorRule.mpx"

  
**Figure 1.33 –** Windows Service Monitor CPU Performance Module

### Add Windows Service Monitor Memory Performance

To create a Windows Service memory Performance Monitor run the following command;

Get-SCOMClassID -MPClassFile "C:\Users\Administrator\source\repos\Test Management Pack\Test Management Pack\MPClass.mpx" -TargetClassID TestClass -RunAsAccount TestRunAs | Add-SCOMMPWindowsServiceMemoryPerformanceMonitor -MonitorName TestServiceMonitorMemory -MonitorEnabled true -AlertOnState Error -AlertName TestMemory -AlertMessage Test -AlertPriority High -AlertSeverity Error -AlertOnAuto true -ServiceName iisadmin -Frequency 300 -Threshold 10 -NumSamples 3 -KnowledgeArticle "this is a test article" -MPMonitorRuleFile "C:\Users\Administrator\source\repos\Test Management Pack\Test Management Pack\MPMonitorRule.mpx"

  
**Figure 1.34 –** Windows Service Monitor Memory Performance Module

### Add Windows Generic Log Monitor

You will need to add the following reference management pack

* System.ApplicationLog.Library

This can be found in the installation of the Visual Studio Authoring extensions **C:\Program Files (x86)\System Center Visual Studio Authoring Extensions\References**

(**Note:** You may need to change the Alias of the reference management pack just in case if the “Build Solution” option in Visual Studio fails as the alias referenced in the “ID” or “TypeID” has to match the alias given on the reference MP i.e. “MSIL!” should match the alias on the Microsoft.SystemCenter.InstanceGroup.Library” alias)

Get-SCOMClassID -MPClassFile "C:\Users\Administrator\source\repos\Test Management Pack\Test Management Pack\MPClass.mpx" -TargetClassID TestClass -RunAsAccount TestRunAs | Add-SCOMMPWindowsGenericLogMonitor -MonitorName TestGenericLogMonitor -MonitorEnabled true -AlertOnState Error -AlertName Logmonitortest -AlertMessage Test -AlertPriority High -AlertSeverity Error -LogFileDirectory C:\Temp -LogPattern \*.test -LogIsUTF8 false -ErrorMessagePattern Error -KnowledgeArticle "this is a test article" -MPMonitorRuleFile "C:\Users\Administrator\source\repos\Test Management Pack\Test Management Pack\MPMonitorRule.mpx"

  
**Figure 1.35 –** Windows Generic Log Monitor Module

### Add Windows Performance Monitor

You will need to add the following reference management pack

* System.Performance.Library

This can be found in the installation of the Visual Studio Authoring extensions **C:\Program Files (x86)\System Center Visual Studio Authoring Extensions\References**

(**Note:** You may need to change the Alias of the reference management pack just in case if the “Build Solution” option in Visual Studio fails as the alias referenced in the “ID” or “TypeID” has to match the alias given on the reference MP i.e. “MSIL!” should match the alias on the Microsoft.SystemCenter.InstanceGroup.Library” alias)

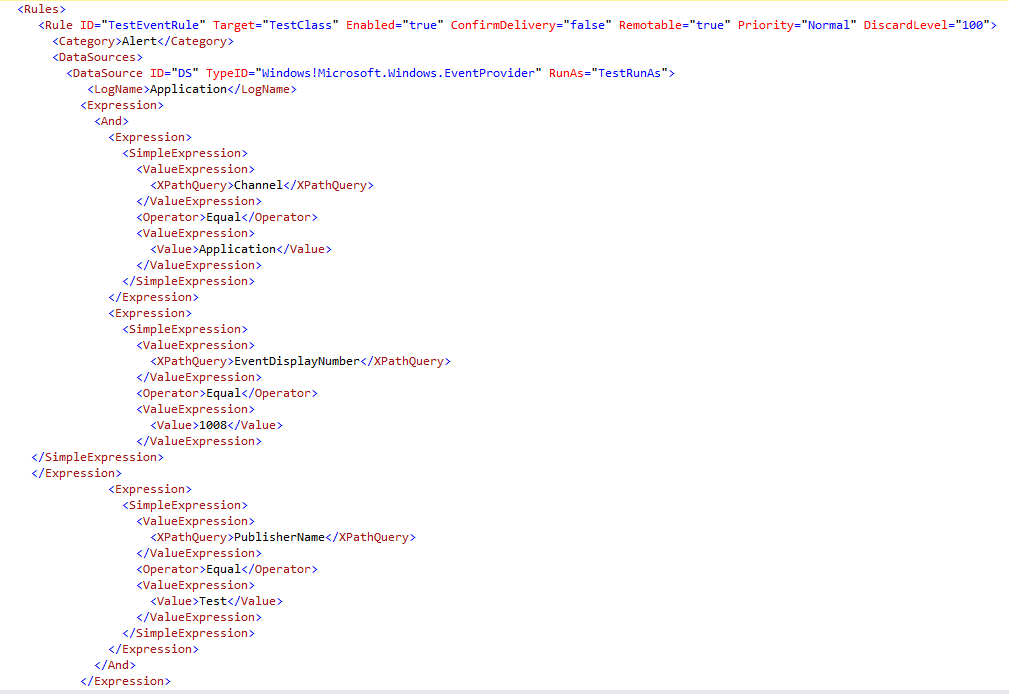
Get-SCOMClassID -MPClassFile "C:\Users\Administrator\source\repos\Test Management Pack\Test Management Pack\MPClass.mpx" -TargetClassID TestClass -RunAsAccount TestRunAs | Add-SCOMMPPerformanceMonitor -MonitorName TestPerformanceMonitor -MonitorEnabled true -AlertOnState Error -AlertPriority High -AlertSeverity Error -ComputerName Test -CounterName cpu -ObjectName Process -InstanceName Test -AllInstances true -Frequency 10 -Threshold 10 -KnowledgeArticle "this is a test article" -MPMonitorRuleFile "C:\Users\Administrator\source\repos\Test Management Pack\Test Management Pack\MPMonitorRule.mpx"

  
**Figure 1.36 –** Windows Performance Monitor Module

### Add Windows Event Rule

To create a Windows Event Rule run the following command;

Get-SCOMClassID -MPClassFile "C:\Users\Administrator\source\repos\Test Management Pack\Test Management Pack\MPClass.mpx" -TargetClassID TestClass -RunAsAccount TestRunAs | Add-SCOMMPWindowsEventRule -RuleName TestEventRule -RuleDescription Test -RuleEnabled true -LogName Application -EventDisplayNumber 1008 -PublisherName Test -AlertName TestAlert -RulePriority High -RuleSeverity Critical -KnowledgeArticle "This is a test article" -MPMonitorRuleFile "C:\Users\Administrator\source\repos\Test Management Pack\Test Management Pack\MPMonitorRule.mpx"

  
**Figure 1.37 –** Windows Event Rule Module

### Add Windows PowerShell Script Rule

To create a Windows PowerShell Script Rule run the following command;   
(Note: You will need to create a PowerShell script for this so you can attach it to this module)

Get-SCOMClassID -MPClassFile "C:\Users\Administrator\source\repos\Test Management Pack\Test Management Pack\MPClass.mpx" -TargetClassID TestClass -RunAsAccount TestRunAs | Add-SCOMMPWindowsPowerShellScriptRule -RuleName TestPowershellRule -RuleDescription Test -RuleEnabled true -RulePriority High -RuleSeverity Critical -AlertName Test -AlertMessage TestMessage -ScriptName Test.ps1 -IntervalSeconds 300 -TimeoutSeconds 300 -KnowledgeArticle "This is a test article" -MPMonitorRuleFile "C:\Users\Administrator\source\repos\Test Management Pack\Test Management Pack\MPMonitorRule.mpx"

  
**Figure 1.38 –** Windows PowerShell Script rule Module

### Add Windows Performance Rule

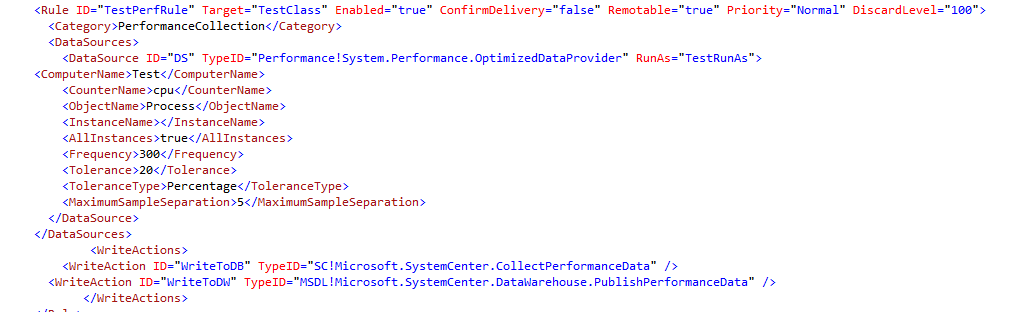
You will need to add the following reference management pack

* Microsoft.SystemCenter.DataWarehouse.Library

This can be found in the installation of the Visual Studio Authoring extensions **C:\Program Files (x86)\System Center Visual Studio Authoring Extensions\References**

(**Note:** You may need to change the Alias of the reference management pack just in case if the “Build Solution” option in Visual Studio fails as the alias referenced in the “ID” or “TypeID” has to match the alias given on the reference MP i.e. “MSIL!” should match the alias on the Microsoft.SystemCenter.InstanceGroup.Library” alias)

Get-SCOMClassID -MPClassFile "C:\Users\Administrator\source\repos\Test Management Pack\Test Management Pack\MPClass.mpx" -TargetClassID TestClass -RunAsAccount TestRunAs | Add-SCOMMPPerformanceRule -RuleName TestPerfRule -RuleDescription Test -RuleEnabled true -AlertName Test -AlertMessage TestMessage -ComputerName Test -CounterName cpu -ObjectName Process -Frequency 300 -Threshold 10 -InstanceProperty 20 -Tolerance -MaxSampleSeparation 5 -AllInstances true -KnowledgeArticle "This is a test article" -MPMonitorRuleFile "C:\Users\Administrator\source\repos\Test Management Pack\Test Management Pack\MPMonitorRule.mpx"

  
**Figure 1.39 –** Windows Performance rule Module

### Add UNIX Log File Monitoring Rule

To create a UNIX Log File Monitoring Rule run the following command;

Get-SCOMClassID -MPClassFile "C:\Users\Administrator\source\repos\Test Management Pack\Test Management Pack\MPClass.mpx" -TargetClassID TestClass -RunAsAccount TestRunAs | Add-SCOMMPUnixLogFileRule -RuleName TestRuleUnixLog -RuleDescription Test -RuleEnabled true -LogFile grep -RegExpFilter \*Test -PublisherName Test -RulePriority High -RuleSeverity Critical -AlertName TestAlert5 -KnowledgeArticle "THis is a test article" -MPMonitorRuleFile "C:\Users\Administrator\source\repos\Test Management Pack\Test Management Pack\MPMonitorRule.mpx"

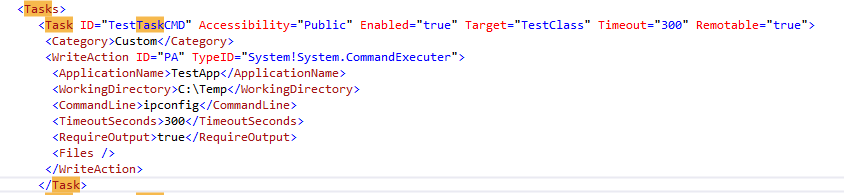
  
**Figure 1.40 –** UNIX Log File Monitoring rule Module

### Create Tasks

#### Add SCOM Agent Tasks

To create a SCOM Agent task based on command line run the following;

Get-SCOMClassID -MPClassFile "C:\Users\Administrator\source\repos\Test Management Pack\Test Management Pack\MPClass.mpx" -TargetClassID TestClass | Add-SCOMMPAgentTaskScript -TaskName TestTaskScript -TaskEnabled true -TaskApplicationName TestApp -TaskWorkingDirectory C:\Temp -Scriptname Test.bat -TimeoutSeconds 300 -MPMonitorRuleFile "C:\Users\Administrator\source\repos\Test Management Pack\Test Management Pack\MPMonitorRule.mpx"

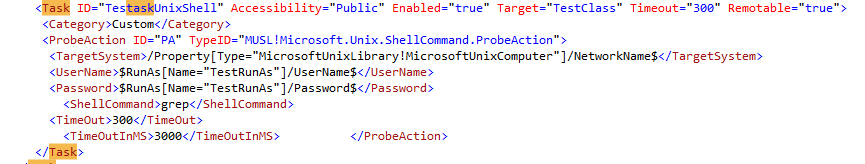


**Figure 1.41 –** SCOM Agent Task based on command line

You will require the Microsoft.UnixShell.Command.Library which can be downloaded here <https://systemcenter.wiki/?Get-ManagementPack=Microsoft.Unix.ShellCommand.Library&Version=7.5.1005.0>

To create a SCOM Agent task based on a UNIX Shell run the following;

Get-SCOMClassID -MPClassFile "C:\Users\Administrator\source\repos\Test Management Pack\Test Management Pack\MPClass.mpx" -TargetClassID TestClass -RunAsAccount TestRunAs | Add-SCOMMPAgentTaskUnixShell -TaskName TestaskUnixShell -TaskEnabled true -ShellCommand grep -TimeoutSeconds 300 -TimeoutMilliSeconds 3000 -MPMonitorRuleFile "C:\Users\Administrator\source\repos\Test Management Pack\Test Management Pack\MPMonitorRule.mpx"

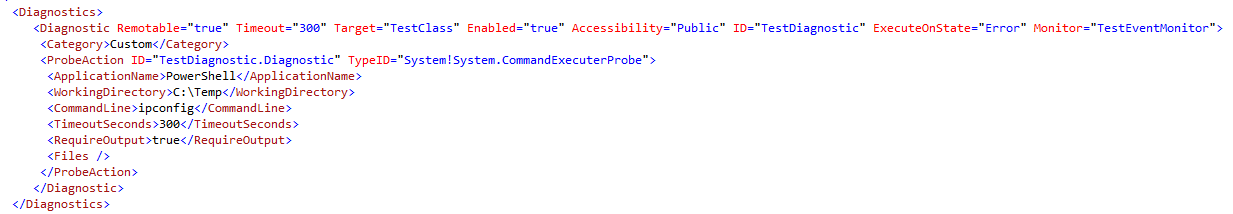
  
**Figure 1.42 –** SCOM Agent task based on UNIX Shell

### Create Diagnostic/Recovery Tasks

#### Add SCOM Diagnostic Tasks

To create a diagnostic task for your monitor run the following command;

Get-SCOMClassID -MPClassFile "C:\Users\Administrator\source\repos\Test Management Pack\Test Management Pack\MPClass.mpx" -TargetClassID TestClass -MPMonitorRuleFile "C:\Users\Administrator\source\repos\Test Management Pack\Test Management Pack\MPClass.mpx" -MonitorRuleID TestEventMonitor | Add-SCOMMPDiagnosticTaskCommandLine -DiagnosticName TestDiagnostic -DiagnosticEnabled true -DiagnosticMonitorTarget TestEventMonitor -DiagnosticExecuteOnState Error -DiagnosticApplicationName PowerShell -DiagnosticWorkingDirectory C:\Temp -DiagnosticCommandLine ipconfig -TimeoutSeconds 300 -MPMonitorRuleFile "C:\Users\Administrator\source\repos\Test Management Pack\Test Management Pack\MPMonitorRule.mpx"

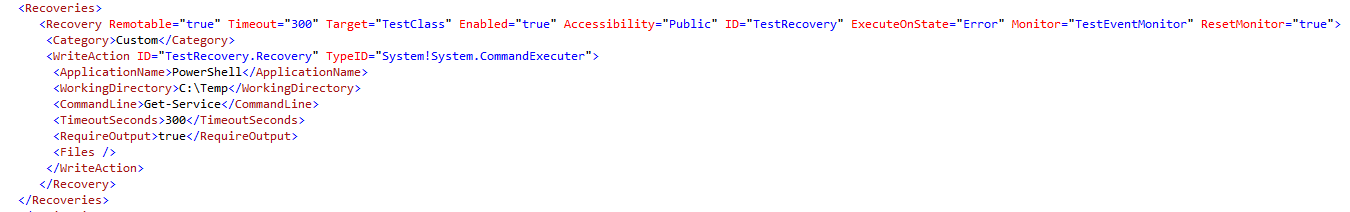


**Figure 1.43 –** SCOM Diagnostic task module

#### Add SCOM Recovery Tasks

To create a recovery task for your monitor run the following command;

Get-SCOMClassID -MPClassFile "C:\Users\Administrator\source\repos\Test Management Pack\Test Management Pack\MPClass.mpx" -TargetClassID TestClass -MPMonitorRuleFile "C:\Users\Administrator\source\repos\Test Management Pack\Test Management Pack\MPClass.mpx" -MonitorRuleID TestEventMonitor | Add-SCOMMPRecoveryTaskCommandLine -RecoveryName TestRecovery -RecoveryMonitorTarget TestEventMonitor -RecoveryEnabled true -RecoveryApplicationName PowerShell -RecoveryWorkingDirectory C:\Temp -RecoveryCommandLine Get-Service -RecoveryResetMonitor true -RecoveryExecuteOnState Error -TimeoutSeconds 300 -MPMonitorRuleFile "C:\Users\Administrator\source\repos\Test Management Pack\Test Management Pack\MPMonitorRule.mpx"

  
**Figure 1.44 –** SCOM recovery task module

# Functions

Here are a list of all of the functions alongside its switches which can be used to build out the management pack.

A lot of these functions contain pre-set answers within the Intellisense in order to make the process a lot easier.

## **New-SCOMMPClass**

This function is used to create the XML file which will contain all of the information for your SCOM management pack Classes, properties and Run as Accounts

### Switches

|  |  |  |
| --- | --- | --- |
| Name | Description | Expected Value |
| MPClassFile | Location of where the MPX file will be created | String value i.e. C:\Temp\Visualstudioclass.mpx |

## **Add-SCOMMPClass**

This functions allow you to create a new class within the class file which would have been created from the New-SCOMMPClass function

### Switches

|  |  |  |
| --- | --- | --- |
| Name | Description | Expected Value |
| ClassName | Name of the class | String value i.e. Visual Studio Computer |
| ClassType | Type of class | WindowsComputer WindowsApplicationComponent WindowsLocalApplication UnixComputer ComputerGroup InstanceGroup ComputerHealthRollup CustomClass |
| ClassDescription | Description of the class | String value i.e. computer which has visual studios installed |
| Abstract | Will this class be abstract | true false. |
| Hosted | Will this class be a hosted class | true false. |
| Singleton | Will this class be a singleton class | true false. |
| MPClassFile | Location of where the MPX file will be created | String value i.e. C:\Temp\Visualstudioclass.mpx |

## **Get-SCOMClassID**

A dynamic parameter function which is able to investigate all of the classes created so that they can be pipelined to all other commands in the module

### Switches

|  |  |  |
| --- | --- | --- |
| Name | Description | Expected Value |
| MPClassFile | Location of where the MPX file will be created | String value i.e. C:\Temp\Visualstudioclass.mpx |
| MPFolderFile | Location of where the MPX file will be created | String value i.e. C:\Temp\Visualstudioclass.mpx |
| MPMonitorRuleFile | Location of where the MPX file will be created | String value i.e. C:\Temp\Visualstudioclass.mpx |
| SourceClassID | The Source class which a pipelined parameter will reference | Dynamic parameter of class investigation |
| TargetClassID | The Target class which a pipelined parameter will reference | Dynamic parameter of class investigation |
| RunAsAccount | The Run As Account class which a pipelined parameter will reference | Dynamic parameter of class investigation |
| FolderID | The FolderID class which a pipelined parameter will reference | Dynamic parameter of class investigation |
| MonitorRuleID | The MonitorRuleID class which a pipelined parameter will reference | Dynamic parameter of class investigation |

## **Add-SCOMMPClassProperty**

This function allows you to create properties for the class which you have created which will be the CI attribute information that would be captured alongside its discovery

### Switches

|  |  |  |
| --- | --- | --- |
| Name | Description | Expected Value |
| PropertyName | Name of the Property | Name of the property i.e. Version |
| PropertyType | Type of property | int decimal double String datetime guid bool enum richtext binary |
| KeyValue | Is the value a key value | true false. |
| PropertyDescription | Description of the property | String value i.e. Version of Visual Studios installed |
| AffectedClassID | The class which this property is linked to | Pipelined from **Get-SCOMClassID SourceClassID** |
| MPClassFile | Location of where the MPX files will be created | String value i.e. C:\Temp\Visualstudioclass.mpx |

## **Add-SCOMMPRunAsAccount**

This function will add a Run as Account to your class file

### Switches

|  |  |  |
| --- | --- | --- |
| Name | Description | Expected Value |
| SecureReferenceName | Name of the Run As Account | String value i.e. Visual Studio RunAs Account |
| SecureReferenceDescription | Description of the Run As Account | String value i.e. Run As Account used to discover Visual Studios |
| MPClassFile | Location of where the MPX files will be created | String value i.e. C:\Temp\Visualstudioclass.mpx |

## **New-SCOMMPRelationship**

This function creates a relationship between two classes which is needed if you are looking to have nested classes which will be used to reference others when creating further discoveries or monitors

### Switches

|  |  |  |
| --- | --- | --- |
| Name | Description | Expected Value |
| MPRelationshipFile | Location of where the MPX file will be created | String value i.e. C:\Temp\Visualstudioclass.mpx |

## **Add-SCOMMPRelationship**

This function adds a relationship class to your relationship container that you would have created on the New-SCOMMPRelationship function

### Switches

|  |  |  |
| --- | --- | --- |
| Name | Description | Expected Value |
| RelationshipName | Name of the Relationship | String value i.e. Visual Studio Computer User Relationship |
| RelationshipDescription | Description of the relationship class | String value i.e. Relationship between Visual Studio computer and user |
| Abstract | Will this class be abstract | true false. |
| Accessibility | What is the accessibility of the class | Internal Public |
| SourceType |  | Pipelined from **Get-SCOMClassID SourceClassID** |
| TargetType |  | Pipelined from **Get-SCOMClassID TargetClassID** |
| MPRelationshipFile | Location of where the MPX files will be created | String value i.e. C:\Temp\Visualstudioclass.mpx |

## **New-SCOMMPDiscovery**

This function creates the discovery class for which you will use to discover all of your objects within your SCOM Management Pack

### Switches

|  |  |  |
| --- | --- | --- |
| Name | Description | Expected Value |
| DiscoveryTarget | ClassID which the discovery will run for | String value i.e. Visual.Studio.Computer |
| MPDiscoveryFile | Location of where the XML files will be created | String value i.e. C:\Temp\Visualstudioclass.xml |

## **Add-SCOMMPPowerShellDiscovery**

Creates a PowerShell based discovery for your discovery container which would use a PowerShell script to perform the object discovery within your management pack.

### Switches

|  |  |  |
| --- | --- | --- |
| Name | Description | Expected Value |
| DiscoveryName | Name of the discovery | String value |
| DiscoveryTarget | Which class will the discovery run for | Pipelined from **Get-SCOMClassID TargetClassID** |
| DiscoveryDescription | Description of the discovery | String value |
| DiscoveryClass | Discovery Class Name | Pipelined from **Get-SCOMClassID SourceClassID** |
| DiscoveryRunAsAccount | Run As account used to run the discovery | Pipelined from **Get-SCOMClassID RunAsAccount** |
| IntervalSeconds | Time to run the discovery | Integer i.e. 300 |
| SyncTime | Specify a time to run at (optional) | Time i.e. 03:00 |
| ScriptName | Name of the script | String value i.e. VisualStudioPSDiscovery.ps1 |
| ScriptBody | Script body content |  |
| TimeoutSeconds | Time it takes to no longer run | Integer i.e. 300 |
| KnowledgeArticle | Contains words which pertain to the discovery for knowledgebase sharing | String i.e. This discovery is used to discover this object utilizing stuff |
| MPClassFile | Location of the class file | String value i.e. C:\Temp\Visualstudioclass.mpx |
| MPDiscoveryFile | Location of where the discovery file will be created | String value i.e. C:\Temp\Visualstudioclass.mpx |

## **Add-SCOMMPRegistryDiscovery**

This function creates a Registry discovery which uses a Registry key to identify if an object belongs to the class you have created for your SCOM Management Pack

### Switches

|  |  |  |
| --- | --- | --- |
| Name | Description | Expected Value |
| DiscoveryName | Name of the discovery | String value |
| DiscoveryTarget | Which class will the discovery run for | Pipelined from **Get-SCOMClassID TargetClassID** |
| DiscoveryDescription | Description of the discovery | String value |
| DiscoveryClass | Discovery Class Name | Pipelined from **Get-SCOMClassID SourceClassID** |
| DiscoveryRunAsAccount | Run As account used to run the discovery | Pipelined from **Get-SCOMClassID RunAsAccount** |
| Frequency | Time to run the discovery | Integer i.e. 300 |
| KnowledgeArticle | Contains words which pertain to the discovery for knowledgebase sharing | String i.e. This discovery is used to discover this object utilizing stuff |
| MPClassFile | Location of the class file | String value i.e. C:\Temp\Visualstudioclass.mpx |
| MPDiscoveryFile | Location of where the discovery file will be created | String value i.e. C:\Temp\Visualstudioclass.mpx |

## **Add-SCOMMPRegistryKey**

This function creates the Registry key which will be added to your Registry Discovery which will be used to interrogate the classes’ registry to see if there is a match

### Switches

|  |  |  |
| --- | --- | --- |
| Name | Description | Expected Value |
| AttributeName | Name of the discovery | String Value. Visual Studio KeyExists |
| RegistryPath | Path of where the Key is in the registry | String Value i.e. HKLM\Software |
| PathType | Type of path discovery | KeyExists KeyValue |
| AttributeType | Attribute Type | Boolean String Integer Float |
| ClassID | Class which the Registry discovery will be ran against | Pipelined from **Get-SCOMClassID SourceClassID** |

## **Add-SCOMMPWMIDiscovery**

This functions creates a WMI discovery which will utilise a WMI query that will be used against the class object WMI to discover the object for the designated class

### Switches

|  |  |  |  |
| --- | --- | --- | --- |
| Name | | Description | Expected Value |
| DiscoveryName | Name of the discovery | | String value |
| DiscoveryTarget | Which class will the discovery run for | | Pipelined from **Get-SCOMClassID TargetClassID** |
| DiscoveryDescription | Description of the discovery | | String value |
| DiscoveryClass | Discovery Class Name | | Pipelined from **Get-SCOMClassID SourceClassID** |
| DiscoveryRunAsAccount | Run As account used to run the discovery | | Pipelined from **Get-SCOMClassID RunAsAccount** |
| Namespace | | WMI namespace to connect to | String value i.e. \\root\cimv2 |
| Query | | Query to use for WMI | String value i.e. select \* from win32\_operatingsystem |
| KnowledgeArticle | Contains words which pertain to the discovery for knowledgebase sharing | | String i.e. This discovery is used to discover this object utilizing stuff |
| Frequency | | Time to run the discovery | Integer value i.e. 300 |
| MPClassFile | Location of the class file | | String value i.e. C:\Temp\Visualstudioclass.mpx |
| MPDiscoveryFile | Location of where the discovery file will be created | | String value i.e. C:\Temp\Visualstudioclass.mpx |

## **Add-SCOMMPVBScriptDiscovery**

This function creates a VB Script discovery which will use a VB Script to run the discovery for your designated class

### Switches

|  |  |  |  |
| --- | --- | --- | --- |
| Name | | Description | Expected Value |
| DiscoveryName | | Name of the discovery | String value i.e. Visual Studio Discovery |
| DiscoveryTarget | | Which class will the discovery run for | Pipelined from **Get-SCOMClassID TargetClassID** |
| DiscoveryDescription | | Description of the discovery | String value i.e. Discovery using VBScript |
| DiscoveryClass | | Discovery Class Name | Pipelined from **Get-SCOMClassID SourceClassID** |
| DiscoveryRunAsAccount | | Run As account used to run the discovery | Pipelined from **Get-SCOMClassID RunAsAccount** |
| IntervalSeconds | | Time to run the discovery | Integer i.e. 300 |
| SyncTime | | Specifcy a time to run at (optional) | Date Time i.e. 03:00 |
| KnowledgeArticle | Contains words which pertain to the discovery for knowledgebase sharing | | String i.e. This discovery is used to discover this object utilizing stuff |
| MPClassFile | | Location of the class file | String value i.e. C:\Temp\Visualstudioclass.mpx |
| MPDiscoveryFile | | Location of where the discovery file will be created | String value i.e. C:\Temp\Visualstudioclass.mpx |
| ScriptName | | Name of the script | String value i.e. Vbscript.vbs |
| TImeoutSeconds | | Time it takes to no longer run | Integer i.e. 300 |

## **Add-SCOMMPUnixShellCommandDiscovery**

This function creates a Unix Shell Command discovery used to discover UNIX/Linux objects using a shell command

### Switches

|  |  |  |  |
| --- | --- | --- | --- |
| Name | | Description | Expected Value |
| DiscoveryName | | Name of the discovery | String value i.e. Visual Studio Discovery |
| DiscoveryTarget | | Which class will the discovery run for | Pipelined from **Get-SCOMClassID TargetClassID** |
| DiscoveryDescription | | Description of the discovery | String value i.e. Discovery using VBScript |
| DiscoveryClass | | Discovery Class Name | Pipelined from **Get-SCOMClassID SourceClassID** |
| DiscoveryRunAsAccount | | Run As account used to run the discovery | Pipelined from **Get-SCOMClassID RunAsAccount** |
| Interval | | Time to run the discovery | Integer i.e. 300 |
| ShellCommand | | Shell Command used for discovery | String i.e. grep |
| Pattern | | Pattern for which the shell needs to discover | String i.e. \*test |
| Timeout | | Time elapsed which it will timeout | Integer i.e. 300 |
| KnowledgeArticle | Contains words which pertain to the discovery for knowledgebase sharing | | String i.e. This discovery is used to discover this object utilizing stuff |
| MPClassFile | | Location of the class file | String value i.e. C:\Temp\Visualstudioclass.mpx |
| MPDiscoveryFile | | Location of where the discovery file will be created | String value i.e. C:\Temp\Visualstudioclass.mpx |

## **Add-SCOMMPComputerGroupDiscovery**

This function creates a computer group for all of the objects which have been discovered by your SCOM Management Pack.

### Switches

|  |  |  |
| --- | --- | --- |
| Name | Description | Expected Value |
| DiscoveryName | Name of the discovery | String value i.e. Visual Studio Discovery |
| DiscoveryTarget | Which class will the discovery run for | Pipelined from **Get-SCOMClassID TargetClassID** |
| DiscoveryDescription | Description of the discovery | String value i.e. Discovery using VBScript |
| DiscoveryClass | Pass through details of the class used | Pipelined from **Get-SCOMClassID SourceClassID** |
| KnowledgeArticle | Contains words which pertain to the discovery for knowledgebase sharing | String i.e. This discovery is used to discover this object utilizing stuff |
| MPDiscoveryFile | Discovery file location where the computer group discovery will be added | String value i.e. C:\Temp\Visualstudioclass.mpx |

## **Add-SCOMMPInstanceGroupDiscovery**

This function creates an instance group for all of the instances which have been discovered by your SCOM Management Pack.

### Switches

|  |  |  |
| --- | --- | --- |
| Name | Description | Expected Value |
| DiscoveryName | Name of the discovery | String value i.e. Visual Studio Discovery |
| DiscoveryTarget | Which class will the discovery run for | Pipelined from **Get-SCOMClassID TargetClassID** |
| DiscoveryDescription | Description of the discovery | String value i.e. Discovery using VBScript |
| DiscoveryClass | Pass through details of the class used | Pipelined from **Get-SCOMClassID SourceClassID** |
| KnowledgeArticle | Contains words which pertain to the discovery for knowledgebase sharing | String i.e. This discovery is used to discover this object utilizing stuff |
| MPDiscoveryFile | Discovery file location where the computer group discovery will be added | String value i.e. C:\Temp\Visualstudioclass.mpx |

## **Create-PowerShellScript**

This function is used to create a PowerShell script for your PowerShell discovery. It adds all of the lines required for the discovery template script to work.

### Switches

|  |  |  |
| --- | --- | --- |
| Name | Description | Expected Value |
| ScriptName | Name of the script | String value i.e. VisualStudioPSDiscovery.ps1 |
| DiscoveryClass | Pass through details of the class used | Pipelined from **Get-SCOMClassID SourceClassID** |
| MPClassFile | Location of where the MPX files will be created | String value i.e. C:\Temp\Visualstudioclass.mpx |

## **Create-VBScript**

This function is used to create a VB script for your VBScript discovery. It adds all of the lines required for the discovery template script to work.

### Switches

|  |  |  |
| --- | --- | --- |
| Name | Description | Expected Value |
| ScriptName | Name of the script | String value i.e. VisualStudioPSDiscovery.vbs |
| DiscoveryClass | Pass through details of the class used | Pipelined from **Get-SCOMClassID SourceClassID** |
| MPClassFile | Location of where the MPX files will be created | String value i.e. C:\Temp\Visualstudioclass.mpx |

## **New-SCOMMPView**

This function creates the View XML file so that you are able to create views for your classes in your SCOM Management pack.

### Switches

|  |  |  |
| --- | --- | --- |
| Name | Description | Expected Value |
| MPViewFile | Location of where the MPX files will be created | String value i.e. C:\Temp\Visualstudioclass.mpx |

## **Add-SCOMMPView**

This function creates new views and adds them to your view XML file which was created using the New-SCOMMPView function

### Switches

|  |  |  |
| --- | --- | --- |
| Name | Description | Expected Value |
| ViewName | Name of the View | String value i.e. Visual Studio Alerts View |
| ViewTarget | Class which the view will be created for | Pipelined from **Get-SCOMClassID TargetClassID** |
| ViewType | Type of view to create | AlertView DashboardView DiagramView EventView InventoryView ManagedObjectView PerformanceView OverridesView StateView StateDetailView TaskStatusView URLView |
| FolderID | Folder ID for where the view will be placed | Pipelined from **Get-SCOMClassID FolderID** |
| MPViewFile | Output location of the View MPX file | String value i.e. C:\Temp\Visualstudioclass.mpx |

## **New-SCOMMPFolder**

This function creates folders xml file for where you can store your different views within you SCOM Management Pack

### Switches

|  |  |  |
| --- | --- | --- |
| Name | Description | Expected Value |
| MPFolderFile | Location of where the MPX files will be created | String value i.e. C:\Temp\Visualstudioclass.mpx |

## **Add-SCOMMPFolder**

This function allows you to create different views and store them in your views XML which you had just created with the New-SCOMMPFolder function

### Switches

|  |  |  |
| --- | --- | --- |
| Name | Description | Expected Value |
| FolderName | Name of the folder to create | String value i.e. Visual Studio Views |
| FolderParent | The folder ID of where the folder will be placed. By default it will go to the Root of the SCOM Console | Pipelined from **Get-SCOMClassID FolderID** Also default value is to place folder on the root |
| MPFolderFile | Location of where the MPX files will be created | String value i.e. C:\Temp\Visualstudioclass.mpx |

## **New-SCOMMPMonitorRule**

This function creates the Monitor/Rules XML file which allows you to add additional Monitors and Rules to

### Switches

|  |  |  |
| --- | --- | --- |
| Name | Description | Expected Value |
| MPMonitorRuleFile | Location of where the MPX files will be created | String value i.e. C:\Temp\Visualstudioclass.mpx |

## **New-SCOMMPCustomProbeAction**

This function is used to create the XML file for a Custom probe action monitor. Primarily used in this script as a PowerShell Script Monitor

### Switches

|  |  |  |
| --- | --- | --- |
| Name | Description | Expected Value |
| MPCustomDataSourceFile | Location of where the MPX files will be created | String value i.e. C:\Temp\Visualstudioclass.mpx |
| MPCustomProbeActionFile | Location of where the MPX files will be created | String value i.e. C:\Temp\Visualstudioclass.mpx |
| MPCustomMonitorTypeFile | Location of where the MPX files will be created | String value i.e. C:\Temp\Visualstudioclass.mpx |

## **Add-SCOMMPCustomProbeAction**

This function adds the following

* Probe Action Type
* Data Source Module
* Unit Monitor Type

These allow you to use a PowerShell script which can be used within a monitor.

### Switches

|  |  |  |
| --- | --- | --- |
| Name | Description | Expected Value |
| CustomModuleName | Name of the custom monitor you will use | String value i.e. Visual Studio PowerShellMonitor |
| MonitorTarget | ID of your Class that the monitor will be used with | Pipelined from **Get-SCOMClassID TargetClassID** |
| MonitorRunAsAccount | ID of the Run As Account that will be used | Pipelined from **Get-SCOMClassID RunAsAccount** |
| AlertName | Name of the alert | String value i.e. ComputerIssue |
| AlertMessage | Message of the alert | String value i.e. Computer is not working |
| KnowledgeArticle | Contains words which pertain to the discovery for knowledgebase sharing | String i.e. This discovery is used to discover this object utilizing stuff |
| TimeoutSeconds | Timeout in Seconds | Integer i.e. 300 |
| ScriptName | Name of the PowerShell script | String i.e. Test.ps1 |
| ScriptOutput | Location of the PowerShell script ONLY | String i.e. C:\temp |
| MPCustomDataSourceFile | Location of where the MPX files will be created | String value i.e. C:\Temp\Visualstudioclass.mpx |
| MPCustomProbeActionFile | Location of where the MPX files will be created | String value i.e. C:\Temp\Visualstudioclass.mpx |
| MPCustomMonitorTypeFile | Location of where the MPX files will be created | String value i.e. C:\Temp\Visualstudioclass.mpx |

## **Add-SCOMMPWindowsEventMonitor**

This function allows you to create a Windows Event 2 state monitor

### Switches

|  |  |  |
| --- | --- | --- |
| Name | Description | Expected Value |
| MontiorName | Name of monitor | String value i.e. Visual Studio EventMonitor |
| MonitorEnabled | If monitored is enabled by default or not “true” or “false” | true false |
| MonitorTarget | Class ID which the monitor will be targeted to | Pipelined from **Get-SCOMClassID TargetClassID** |
| MonitorRunAsAccount | Run as account which will run the monitor | Pipelined from **Get-SCOMClassID RunAsAccount** |
| AlertOnState | If it should alert on Error or Warning | Error Warning |
| AlertSeverity | Alert Severity | Error Warning |
| AlertPriority | Alert Priority, High, Normal or Low | High Normal Low |
| Alert Name | Name of the Alert | String value i.e. Visual Studio Event Monitor Bad Health |
| AlertMessage | Message shown when alert is created | String value i.e. Visual Studio Event Monitor in bad health call administrator. |
| UnHealthyLogName | Log Name for unhealthy status | String Value i.e Application |
| UnHealthyPublisherName | Publisher name for unhealthy status | String Value i.e. VisualStudio |
| UnHealthyEventDisplayNumber | Event Number for unhealthy status | Integer Value i.e. VisualStudio |
| HealthyLogName | Log Name for healthy status | String Value i.e Application |
| HealthyPublisherName | Publisher name for healthy status | String Value i.e. VisualStudio |
| HealthyEventDisplayNumber |  | Integer Value i.e. VisualStudio |
| KnowledgeArticle | Contains words which pertain to the discovery for knowledgebase sharing | String i.e. This discovery is used to discover this object utilizing stuff |
| MPMonitorRuleFile | Output for MPX for Monitor file | String value i.e. C:\Temp\Visualstudioclass.mpx |

## **Add-SCOMMPWindowsEventManualResetMonitor**

This function allows you to create a Windows Event Manual Reset Monitor

### Switches

|  |  |  |
| --- | --- | --- |
| Name | Description | Expected Value |
| MontiorName | Name of monitor | String value i.e. Visual Studio EventMonitor |
| MonitorEnabled | If monitored is enabled by default or not “true” or “false” | true false |
| MonitorTarget | Class ID which the monitor will be targeted to | Pipelined from **Get-SCOMClassID TargetClassID** |
| MonitorRunAsAccount | Run as account which will run the monitor | Pipelined from **Get-SCOMClassID RunAsAccount** |
| AlertOnState | If it should alert on Error or Warning | Error Warning |
| AlertSeverity | Alert Severity | Error Warning |
| AlertPriority | Alert Priority, High, Normal or Low | High Normal Low |
| Alert Name | Name of the Alert | String value i.e. Visual Studio Event Monitor Bad Health |
| AlertMessage | Message shown when alert is created | String value i.e. Visual Studio Event Monitor in bad health call administrator. |
| UnHealthyLogName | Log Name for unhealthy status | String Value i.e Application |
| UnHealthyPublisherName | Publisher name for unhealthy status | String Value i.e. VisualStudio |
| UnHealthyEventDisplayNumber | Event Number for unhealthy status | Integer Value i.e. VisualStudio |
| KnowledgeArticle | Contains words which pertain to the discovery for knowledgebase sharing | String i.e. This discovery is used to discover this object utilizing stuff |
| MPMonitorRuleFile | Output for MPX for Monitor file | String value i.e. C:\Temp\Visualstudioclass.mpx |

## **Add-SCOMMPWindowsEventTimerResetMonitor**

This function allows you to create a Windows Event Manual Reset Monitor

### Switches

|  |  |  |
| --- | --- | --- |
| Name | Description | Expected Value |
| MontiorName | Name of monitor | String value i.e. Visual Studio EventMonitor |
| MonitorEnabled | If monitored is enabled by default or not “true” or “false” | true false |
| MonitorTarget | Class ID which the monitor will be targeted to | Pipelined from **Get-SCOMClassID TargetClassID** |
| MonitorRunAsAccount | Run as account which will run the monitor | Pipelined from **Get-SCOMClassID RunAsAccount** |
| AlertOnState | If it should alert on Error or Warning | Error Warning |
| AlertSeverity | Alert Severity | Error Warning |
| AlertPriority | Alert Priority, High, Normal or Low | High Normal Low |
| Alert Name | Name of the Alert | String value i.e. Visual Studio Event Monitor Bad Health |
| AlertMessage | Message shown when alert is created | String value i.e. Visual Studio Event Monitor in bad health call administrator. |
| UnHealthyLogName | Log Name for unhealthy status | String Value i.e Application |
| UnHealthyPublisherName | Publisher name for unhealthy status | String Value i.e. VisualStudio |
| UnHealthyEventDisplayNumber | Event Number for unhealthy status | Integer Value i.e. VisualStudio |
| TimerWaitInSeconds | Duration to reset the monitor | Integer i.e. 300 |
| KnowledgeArticle | Contains words which pertain to the discovery for knowledgebase sharing | String i.e. This discovery is used to discover this object utilizing stuff |
| MPMonitorRuleFile | Output for MPX for Monitor file | String value i.e. C:\Temp\Visualstudioclass.mpx |

## **Add-SCOMMPWindowsServiceMonitor**

This function will create a Windows Service Monitor that will monitor if a windows service is up or down

### Switches

|  |  |  |
| --- | --- | --- |
| Name | Description | Expected Value |
| MontiorName | Name of monitor | String value i.e. Visual Studio EventMonitor |
| MonitorEnabled | If monitored is enabled by default or not “true” or “false” | true false |
| MonitorTarget | Class ID which the monitor will be targeted to | Pipelined from **Get-SCOMClassID TargetClassID** |
| MonitorRunAsAccount | Run as account which will run the monitor | Pipelined from **Get-SCOMClassID RunAsAccount** |
| AlertOnState | If it should alert on Error or Warning | Error Warning |
| AlertSeverity | Alert Severity, Error or Warning | Error Warning |
| AlertPriority | Alert Priority, High, Normal or Low | High Normal Low |
| Alert Name | Name of the Alert | String value i.e. Visual Studio Event Monitor Bad Health |
| AlertMessage | Message shown when alert is created | String value i.e. Visual Studio Event Monitor in bad health call administrator. |
| ServiceName | Name of the service to monitor | String value – i.e. AppInfo |
| AlertOnAuto | If it will monitor an automatic service | true false |
| KnowledgeArticle | Contains words which pertain to the discovery for knowledgebase sharing | String i.e. This discovery is used to discover this object utilizing stuff |
| MPMonitorRuleFile | Output for MPX for Monitor file | String value i.e. C:\Temp\Visualstudioclass.mpx |

## **Add-SCOMMPWindowsServicePerformanceMonitor**

This function will create a Windows Service performance monitor which will monitor the resources that a Windows Service uses i.e. CPU, Memory, Logical disk etc

### Switches

|  |  |  |
| --- | --- | --- |
| Name | Description | Expected Value |
| MontiorName | Name of monitor | String value i.e. Visual Studio EventMonitor |
| MonitorEnabled | If monitored is enabled by default or not “true” or “false” | true false |
| MonitorTarget | Class ID which the monitor will be targeted to | Pipelined from **Get-SCOMClassID TargetClassID** |
| MonitorRunAsAccount | Run as account which will run the monitor | Pipelined from **Get-SCOMClassID RunAsAccount** |
| AlertOnState | If it should alert on Error or Warning | Error Warning |
| AlertSeverity | Alert Severity, Error or Warning | Error Warning |
| AlertPriority | Alert Priority, High, Normal or Low | High Normal Low |
| Alert Name | Name of the Alert | String value i.e. Visual Studio Event Monitor Bad Health |
| AlertMessage | Message shown when alert is created | String value i.e. Visual Studio Event Monitor in bad health call administrator. |
| ServiceName | Name of the service to monitor | String value – i.e. AppInfo |
| AlertOnAuto | If it will monitor an automatic service | true false |
| CounterName | Counter name of the resource to monitor |  |
| ObjectName | Object name of the resource to monitor | String value i.e. LogicalDisk |
| Frequency | Frequency to which the monitor will run | Integer value i.e. 300 |
| Threshold | Threshold max it should reach before alerting | Interger value i.e. 50 |
| NumSamples | Number of samples to compare with | Integer value i.e. 5 |
| KnowledgeArticle | Contains words which pertain to the discovery for knowledgebase sharing | String i.e. This discovery is used to discover this object utilizing stuff |
| MPMonitorRuleFile | Output for MPX for Monitor file | String value i.e. C:\Temp\Visualstudioclass.mpx |

## **Add-SCOMMPWindowsGenericLogMonitor**

This function will create a Windows generic log file monitor which can monitor for patterns on log files which can alert if found

### Switches

|  |  |  |
| --- | --- | --- |
| Name | Description | Expected Value |
| MontiorName | Name of monitor | String value i.e. Visual Studio EventMonitor |
| MonitorEnabled | If monitored is enabled by default or not “true” or “false” | true false |
| MonitorTarget | Class ID which the monitor will be targeted to | Pipelined from **Get-SCOMClassID TargetClassID** |
| MonitorRunAsAccount | Run as account which will run the monitor | Pipelined from **Get-SCOMClassID RunAsAccount** |
| AlertOnState | If it should alert on Error or Warning | Error Warning |
| AlertSeverity | Alert Severity, Error or Warning | Error Warning |
| AlertPriority | Alert Priority, High, Normal or Low | High Normal Low |
| Alert Name | Name of the Alert | String value i.e. Visual Studio Event Monitor Bad Health |
| AlertMessage | Message shown when alert is created | String value i.e. Visual Studio Event Monitor in bad health call administrator. |
| LogFileDirectory | Location of the log files | String value i.e. C:\Temp\Logs |
| LogPattern | Log file extension files | Strng value i.e. \*.log |
| LogIsUTF8 | Is the log format of UTF8 | Boolean value – true or false |
| KnowledgeArticle | Contains words which pertain to the discovery for knowledgebase sharing | String i.e. This discovery is used to discover this object utilizing stuff |
| MPMonitorRuleFile | Output for MPX for Monitor file | String value i.e. C:\Temp\Visualstudioclass.mpx |

## **Add-SCOMMPPerformanceMonitor**

This function will create a Windows performance monitor which will monitor the resources such as i.e. CPU, Memory, Logical disk etc

### Switches

|  |  |  |
| --- | --- | --- |
| Name | Description | Expected Value |
| MontiorName | Name of monitor | String value i.e. Visual Studio EventMonitor |
| MonitorEnabled | If monitored is enabled by default or not “true” or “false” | true false |
| MonitorTarget | Class ID which the monitor will be targeted to | Pipelined from **Get-SCOMClassID TargetClassID** |
| MonitorRunAsAccount | Run as account which will run the monitor | Pipelined from **Get-SCOMClassID RunAsAccount** |
| AlertOnState | If it should alert on Error or Warning | Error Warning |
| AlertSeverity | Alert Severity, Error or Warning | Error Warning |
| AlertPriority | Alert Priority, High, Normal or Low | High Normal Low |
| Alert Name | Name of the Alert | String value i.e. Visual Studio Event Monitor Bad Health |
| AlertMessage | Message shown when alert is created | String value i.e. Visual Studio Event Monitor in bad health call administrator. |
| ComputerName | Where to get computer name value | String value i.e. $Target/Property[Type=”MicrosoftWindowsLibrary!Microsoft.Windows.Computer”]/NetworkName$ |
| ObjectName | Name of object to resource monitor | String value i.e. LogicalDisk |
| AllInstances | Monitor all instances of the object/counter | true false |
| Frequency | Frequency the monitor runs | Integer value i.e. 30 |
| Threshold | Threshold in percentage | Integer value i.e. 40 |
| InstanceName | Name of the object instance |  |
| KnowledgeArticle | Contains words which pertain to the discovery for knowledgebase sharing | String i.e. This discovery is used to discover this object utilizing stuff |
| MPMonitorRuleFile | Output for MPX for Monitor file | String value i.e. C:\Temp\Visualstudioclass.mpx |

## **Add-SCOMMPWindowsEventRule**

This function will create a Windows Event Rule. This has the ability to suppress alerts depending on the variables

### Switches

|  |  |  |
| --- | --- | --- |
| Name | Description | Expected Value |
| RuleName | Name of Rule | String value i.e. Visual Studio PowerShellRule |
| RuleEnabled | If the rule is enabled by default | true false |
| RuleTarget | Class ID to which the rule will apply to | Pipelined from **Get-SCOMClassID TargetClassID** |
| RuleRunAsAccount | Run as account which will run the Rule | Pipelined from **Get-SCOMClassID RunAsAccount** |
| LogName | Name of Windows event Log to monitor | String i.e. Application |
| EventDisplayNumber | Event Number of the event to monitor | String i.e. 1000 |
| Publisher | Name of publisher of event to monitor | String i.e. Test |
| AlertName | Name of the Alert | String i.e. TestAlert |
| AlertOnState | If it should alert on Error or Warning | Error Warning |
| RuleSeverity | Alert Severity, Error or Warning | Error Warning |
| RulePriority | Alert Priority, High, Normal or Low | High Normal Low |
| KnowledgeArticle | Contains words which pertain to the discovery for knowledgebase sharing | String i.e. This discovery is used to discover this object utilizing stuff |
| MPMonitorRuleFile | Output for MPX for Monitor file | String value i.e. C:\Temp\Visualstudioclass.mpx |

## **Add-SCOMMPWindowsPowerShellScriptRule**

This function will create a Windows PowerShell Script rule

### Switches

|  |  |  |
| --- | --- | --- |
| Name | Description | Expected Value |
| RuleName | Name of Rule | String value i.e. Visual Studio PowerShellRule |
| RuleEnabled | If the rule is enabled by default | true false |
| RuleTarget | Class ID to which the rule will apply to | Pipelined from **Get-SCOMClassID TargetClassID** |
| RuleRunAsAccount | Run as account which will run the Rule | Pipelined from **Get-SCOMClassID RunAsAccount** |
| Alert Name | Name of the Alert | String value i.e. Visual Studio Event Monitor Bad Health |
| Alert Message | Message which the alert will display | String value i.e. Visual Studio Event Monitor in bad health call administrator. |
| RuleSeverity | Alert Severity | Error Warning |
| RulePriority | Alert Priority | High Normal Low |
| Interval Seconds | Interval which the rule will run | Integer value i.e. 300 |
| SyncTime | Time in which the rule will run | Time value i.e. 03:00 |
| TimeoutSeconds | Interval which the rule will run | Integer value i.e. 300 |
| KnowledgeArticle | Contains words which pertain to the discovery for knowledgebase sharing | String i.e. This discovery is used to discover this object utilizing stuff |
| MPMonitorRuleFile | Output for MPX for Monitor file | String value i.e. C:\Temp\Visualstudioclass.mpx |

## **Add-SCOMMPPerfomanceRule**

This function will create a Windows Performance Rule

### Switches

|  |  |  |
| --- | --- | --- |
| Name | Description | Expected Value |
| RuleName | Name of Rule | String value i.e. Visual Studio PowerShellRule |
| RuleEnabled | If the rule is enabled by default | true false |
| RuleTarget | Class ID to which the rule will apply to | Pipelined from **Get-SCOMClassID TargetClassID** |
| RuleRunAsAccount | Run as account which will run the Rule | Pipelined from **Get-SCOMClassID RunAsAccount** |
| Alert Name | Name of the Alert | String Value i.e Visual Studio AlertTooHigh |
| AlertMessage | Message shown when alert is created | String Value ie. Alert has breached threshold |
| ComputerName | Where to obtain computer name | String value i.e. $Target/Property[Type=”MicrosoftWindowsLibrary!Microsoft.Windows.Computer”]/NetworkName$ |
| CounterName | Name of counter | String value i.e. % Disk Time |
| ObjectName | Name of object to resource monitor | String value i.e. LogicalDisk |
| AllInstances | Monitor all instances of the object/counter | true false |
| Frequency | Frequency the monitor runs | Integer value i.e. 30 |
| Threshold | Threshold of performance monitor | Integer value i.e. 300 |
| Tolerance | Tolerance based on percentage | Integer value i.e 10 |
| MaxSampleSeparation | Maximum samples to compare with | Integer value i.e. 4 |
| MPMonitorRuleFile | Output for mpx for Monitor file | String value i.e. C:\Temp\Visualstudioclass.mpx |

## **Add-SCOMMPUnixLogFileRule**

This function will create a UNIX Log File Rule to monitor log files in UNIX systems

### Switches

|  |  |  |
| --- | --- | --- |
| Name | Description | Expected Value |
| RuleName | Name of Rule | String value i.e. Visual Studio PowerShellRule |
| RuleEnabled | If the rule is enabled by default | true false |
| RuleTarget | Class ID to which the rule will apply to | Pipelined from **Get-SCOMClassID TargetClassID** |
| RuleRunAsAccount | Run as account which will run the Rule | Pipelined from **Get-SCOMClassID RunAsAccount** |
| LogFile | Name of Log file to monitor | String i.e etc\log.log |
| RegExpFilter | Expression to filter for | String i.e. \*test |
| PublisherName | Name of the publisher | String i.e. Test |
| RuleSeverity | Alert Severity | Error Warning |
| RulePriority | Alert Priority | High Normal Low |
| Interval Seconds | Interval which the rule will run | Integer value i.e. 300 |
| SyncTime | Time in which the rule will run | Time value i.e. 03:00 |
| TimeoutSeconds | Interval which the rule will run | Integer value i.e. 300 |
| KnowledgeArticle | Contains words which pertain to the discovery for knowledgebase sharing | String i.e. This discovery is used to discover this object utilizing stuff |
| MPMonitorRuleFile | Output for MPX for Monitor file | String value i.e. C:\Temp\Visualstudioclass.mpx |

## **Add-SCOMMPAgentTaskCommandLine**

This function will create a SCOM agent task based on command line

### Switches

|  |  |  |
| --- | --- | --- |
| Name | Description | Expected Value |
| TaskName | Name of Task | String value i.e. Visual Studio PowerShellRule |
| TaskEnabled | If the task is enabled by default | true false |
| TaskTarget | Class ID to which the task will apply to | Pipelined from **Get-SCOMClassID TargetClassID** |
| TaskApplicationName | Name of the application which the Task will use | String i.e. C:\Wndows\system32\ping.exe |
| TaskWorkingDirectory | Directory which the task is based from | String i.e. C:\temp |
| TaskCommandLine | Command which the Task will run | String i.e. ipconfig |
| MPMonitorRuleFile | Output for MPX for Monitor file | String value i.e. C:\Temp\Visualstudioclass.mpx |

## **Add-SCOMMPAgentTaskScript**

This function will create a SCOM agent task based on script

### Switches

|  |  |  |
| --- | --- | --- |
| Name | Description | Expected Value |
| TaskName | Name of Task | String value i.e. Visual Studio PowerShellRule |
| TaskEnabled | If the task is enabled by default | true false |
| TaskTarget | Class ID to which the task will apply to | Pipelined from **Get-SCOMClassID TargetClassID** |
| TaskApplicationName | Name of the application which the Task will use | String i.e. C:\Wndows\system32\ping.exe |
| TaskWorkingDirectory | Directory which the task is based from | String i.e. C:\temp |
| ScriptName | Name of the script the task will use | String i.e. C:\temp\task.bat |
| MPMonitorRuleFile | Output for MPX for Monitor file | String value i.e. C:\Temp\Visualstudioclass.mpx |

## **Add-SCOMMPAgentTaskUnixShell**

This function will create a SCOM agent task for UNIX systems based on Shell commands

### Switches

|  |  |  |
| --- | --- | --- |
| Name | Description | Expected Value |
| TaskName | Name of Task | String value i.e. Visual Studio PowerShellRule |
| TaskEnabled | If the task is enabled by default | true false |
| TaskTarget | Class ID to which the task will apply to | Pipelined from **Get-SCOMClassID TargetClassID** |
| TaskRunAsAccount | Run as account which the Task will use | Pipelined from **Get-SCOMClassID RunAsAccount** |
| ShellCommand | The Shell command the task will use | String i.e.grep |
| Timeout | Timeout in seconds | String i.e. 300 |
| TimeoutMilliSeconds | Timeout in mill seconds | String i.e. 30000 |
| MPMonitorRuleFile | Output for MPX for Monitor file | String value i.e. C:\Temp\Visualstudioclass.mpx |

## **Add-SCOMMPAgentTaskUnixScript**

This function will create a SCOM agent task for UNIX systems based on a UNIX Script

### Switches

|  |  |  |
| --- | --- | --- |
| Name | Description | Expected Value |
| TaskName | Name of Task | String value i.e. Visual Studio PowerShellRule |
| TaskEnabled | If the task is enabled by default | true false |
| TaskTarget | Class ID to which the task will apply to | Pipelined from **Get-SCOMClassID TargetClassID** |
| TaskRunAsAccount | Run as account which the Task will use | Pipelined from **Get-SCOMClassID RunAsAccount** |
| ScriptName | The script the task will use | String i.e etc\script |
| Timeout | Timeout in seconds | String i.e. 300 |
| TimeoutMilliSeconds | Timeout in mill seconds | String i.e. 30000 |
| MPMonitorRuleFile | Output for MPX for Monitor file | String value i.e. C:\Temp\Visualstudioclass.mpx |

## **Add-SCOMMPDiagnosticTaskCommandLine**

This function will create a SCOM Diagnostic task for a monitor using a command line function

### Switches

|  |  |  |
| --- | --- | --- |
| Name | Description | Expected Value |
| DiagnosticName | Name of Task | String value i.e. Visual Studio PowerShellRule |
| DiagnosticEnabled | If the task is enabled by default | true false |
| DiagnosticTarget | Class ID to which the task will apply to | Pipelined from **Get-SCOMClassID TargetClassID** |
| DiagmosticMonitorTarget | Run as account which the Task will use | Pipelined from **Get-SCOMClassID MonitorRuleID** |
| DiagnosticExecuteOnState | What state will the task run on | Error Warning |
| DiagnosticApplicationName | Name of the application which the Task will use | String i.e. C:\Wndows\system32\ping.exe |
| DiagnosticWorkingDirectory | Directory which the task is based from | String i.e. C:\temp |
| DiagnosticCommandLine | Command which the Task will run | String i.e. ipconfig |
| TimeoutInSeconds | Timeout in seconds | String i.e. 300 |
| MPMonitorRuleFile | Output for MPX for Monitor file | String value i.e. C:\Temp\Visualstudioclass.mpx |

## **Add-SCOMMPRecoveryTaskCommandLine**

This function will create a SCOM Diagnostic task for a monitor using a command line function

### Switches

|  |  |  |
| --- | --- | --- |
| Name | Description | Expected Value |
| RecoveryName | Name of Task | String value i.e. Visual Studio PowerShellRule |
| RecoveryEnabled | If the task is enabled by default | true false |
| RecoveryTarget | Class ID to which the task will apply to | Pipelined from **Get-SCOMClassID TargetClassID** |
| RecoveryMonitorTarget | Run as account which the Task will use | Pipelined from **Get-SCOMClassID MonitorRuleID** |
| RecoveryExecuteOnState | What state will the task run on | Error Warning |
| RecoveryResetMonitor | Whether to reset the health of the monitor once the task completes | true false |
| RecoveryApplicationName | Name of the application which the Task will use | String i.e. C:\Wndows\system32\ping.exe |
| RecoveryWorkingDirectory | Directory which the task is based from | String i.e. C:\temp |
| RecoveryCommandLine | Command which the Task will run | String i.e. ipconfig |
| TimeoutInSeconds | Timeout in seconds | String i.e. 300 |
| MPMonitorRuleFile | Output for MPX for Monitor file | String value i.e. C:\Temp\Visualstudioclass.mpx |