



Workshop: Authoring SCOM Agent Tasks Using Visual Studio Authoring Extension

Author: Tao Yang

Managing Consultant, TY Consulting Pty Ltd.

Microsoft MVP: Cloud and Datacenter Management



Email: tao.yang@tyconsulting.om.au

Version: 1.0

April, 2016

COPYRIGHT © Squared Up Ltd & TY Consulting Pty Ltd 2016

This document is copyright protected. Other than for the purpose of and subject to the conditions prescribed under the Copyright Act, no part of this document may in any form or by any means be reproduced, stored in a retrieval system or transmitted without prior written permission from Squared Up or TY Consulting.

Revision History

Revision	Date	Status	Author	Change
0.1	30/03/2016	Draft	Tao Yang	Initial Draft
0.2	31/03/2016	Draft	Tao Yang	Minor updates
0.3	05/04/2016	Draft	Tao Yang	Added Section 4 for simple tasks creation using templates.
0.4	11/04/2016	Draft	Tao Yang	Moved agent installation to optional software section.
1.0	12/04/2016	Release	Tao Yang	Updated the process creating managementpack.mpx

Table of Contents

1	Introduction.....	3
2	Setting up Authoring Computer	3
2.1	Installing Required Software	3
2.1.1	Installing Visual Studio 2015 Enterprise	3
2.1.2	Installing Visual Studio Authoring Extension (VSAE)	4
2.2	Installing Optional Software	5
2.2.1	Installing SCOM 2012 R2 Agent	5
2.2.2	Installing Other Optional Software.....	7
2.3	Creating Reference Management Packs Folder	9
2.4	Creating a Key File for Sealing Management Packs.....	9
2.5	Download and Install Dependency MPs.....	10
3	Creating Visual Studio Management Pack Project	13
3.1	Creating VSAE Project and Defining Basic MP Properties	13
3.2	Build the Management Pack	18
3.3	Import Management Pack into SCOM.....	19
4	Creating a Simple Agent Task Using VSAE Template.....	20
5	Creating Complex Agent Tasks	24
5.1	Add Reference Management Packs.....	25
5.2	Create PowerShell Script Used by the Agent Task	26
5.3	Create Custom Write Action Module for the Agent Task.....	28
5.4	Create Agent Task for Pre SQL 2014 Database Engines	30
5.5	Create Agent Task for SQL 2014 Database Engines.....	31
5.6	Build and Import the Management Pack	32
6	Test Agent Tasks	32
7	Summary.....	34

1 Introduction

In this workshop, you will learn how to author a Microsoft System Center Operations Manager management pack that contains agent tasks, which can be executed on an SCOM Windows agent locally and present the task output within SCOM consoles.

This guide firstly walks through the steps required to prepare your authoring computer before demonstrating how to author agent tasks in Visual Studio Authoring Extension.

We will go through the steps of creating an agent task to get long running queries on SQL 2008/2012 or SQL 2014 database engines.

2 Setting up Authoring Computer

In order to author SCOM management packs using Visual Studio Authoring Extension, the following software must be installed on your authoring computer:

- Visual Studio 2015 (Professional / Enterprise / Community)
- Microsoft System Center Operations Manager 2012 R2 agent with the latest Update Rollup
- Visual Studio Authoring Extension v1.2 (<https://www.microsoft.com/en-us/download/details.aspx?id=30169>)

Additionally, although the following applications are not required, but you may also find them useful during your authoring experience.

- Microsoft System Center Operations Manager 2012 R2 Console with the latest Update Rollup
- Notepad++ (<https://notepad-plus-plus.org/>)
- SCSM Entity Explorer (<https://gallery.technet.microsoft.com/SCSM-Entity-Explorer-68b86bd2>)
- MPViewer (blogs.msdn.com/cfs-filessystemfile.ashx/?key/communityserver-blogs-components-weblogfiles/00-00-00-41-89-SCOM+Tools/6560.MPViewer.2.3.3.zip)
- PowerShell Tools for Visual Studio 2015

2.1 Installing Required Software

2.1.1 Installing Visual Studio 2015 Enterprise

We will start with installing Visual Studio 2015. Although it is also OK to use Visual Studio 2015 Professional edition, in this workshop, we will be **using Visual Studio 2015 Enterprise Edition with Update 1**.

Note: If you do not have a licensed version of Visual Studio 2015 Professional or Enterprise, you may use the free Visual Studio 2015 Community Edition instead (<https://www.visualstudio.com/en-us/downloads/download-visual-studio-vs#d-community>). Please also be aware of the licensing terms for Visual Studio Community Edition. You can find details from this blog post: http://blogs.msdn.com/b/quick_thoughts/archive/2014/11/12/visual-

[studio-community-2013-free.aspx](#). The authors of this workshop are not responsible if you have misused Visual Studio Community edition for commercial purposes.

You can simply install Visual Studio 2015 with the default options:

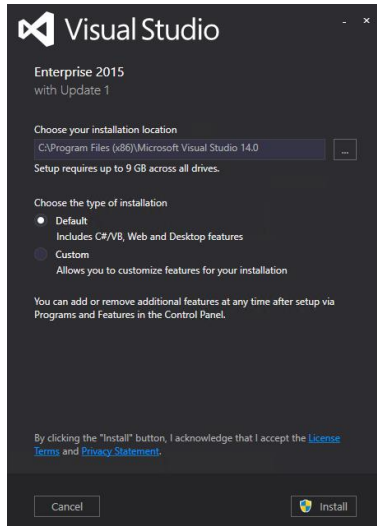


FIGURE 2.1 INSTALLING VISUAL STUDIO 2015 ENTERPRISE WITH DEFAULT OPTIONS

2.1.2 Installing Visual Studio Authoring Extension (VSAE)

Visual Studio Authoring Extension v1.2 can be downloaded from

<http://www.microsoft.com/en-us/download/details.aspx?id=30169>

Note: You MUST download and install version 1.2 of the VSAE because previous versions are not compatible with Visual Studio 2015.

After VSAE is installed, you will see the various SCOM and System Center 2012 R2 Service Manager (SCSM) MP templates when creating new Visual Studio projects, as shown in figure 2.1.5.

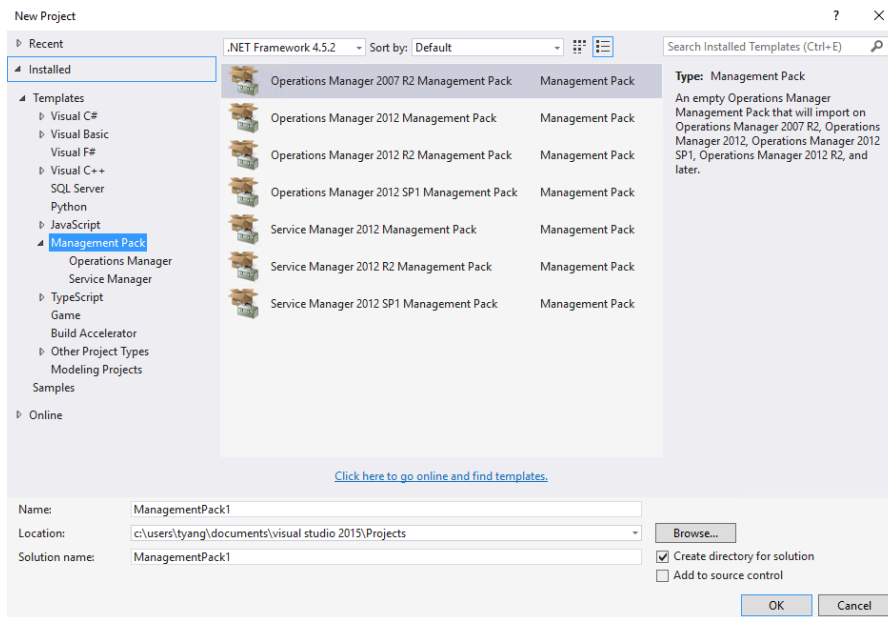


FIGURE 2.1.5 NEW VISUAL STUDIO MANAGEMENT PACK PROJECT TEMPLATES

2.2 Installing Optional Software

2.2.1 Installing SCOM 2012 R2 Agent

Note: The Microsoft Monitoring Agent (MMA), previously known as the agent for SCOM. Currently, there are two versions of MMA available:

SCOM 2012 Agent

OMS Directly Connected Agent

From the management pack authoring requirement point of view, it does not matter which version do we need to install on the authoring computer. In this workshop, we will use the native SCOM agent.

The SCOM agent (Microsoft Monitoring Agent or MMA) is required because the MP Simulator within Visual Studio Authoring Extension (VASE) requires the assemblies that come with MMA. However, once you have installed the MMA on the authoring computer, it is not required to connect this computer to your SCOM management group. If you use the SCOM 2012 console to push the MMA to your authoring computer, the computer will be automatically connected to your SCOM management group. Alternatively, if you do not want to connect the computer to your SCOM management group, you can manually install the agent from SCOM installation media (<Installation media>\agent\AMD64 or i386\OMMAgent.msi”).

For the same reason, .Net Framework 3.5 is also required when you are using the MP Simulator within Visual Studio. Visual Studio will crash when you launch the MP Simulator if .Net Framework 3.5 is not installed.

Next, we will install the SCOM 2012 R2 agent and the latest Update Rollup. The SCOM 2012 R2 agent install msi can be located in the SCOM 2012 installation media, under “<Install Media>\agent\AMD64\MOMAgent.msi”.

During the install, please untick “Connect the agent to System Center Operations Manager” and leave other options as default:

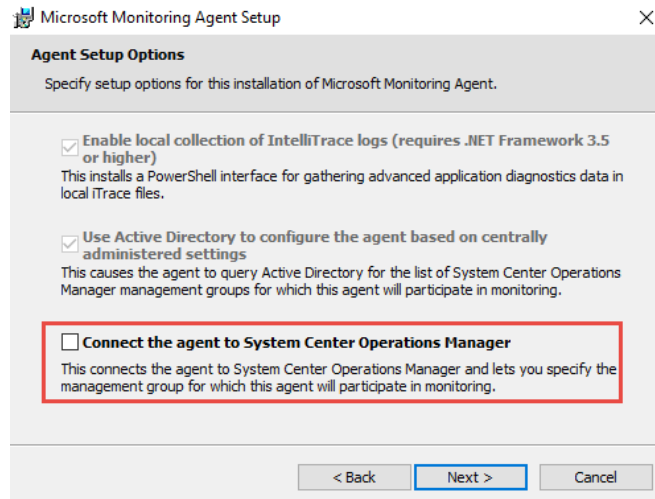


FIGURE 2.2.1 SCOM 2012 R2 AGENT INSTALLATION OPTIONS

After the SCOM 2012 R2 agent is installed, we will then install the latest Update Rollup. At the time of writing, the latest Update Rollup is UR9, which can be downloaded from Microsoft Update Catalog:

<http://catalog.update.microsoft.com/v7/site/Search.aspx?q=3129774>. Please open this URL in Internet Explorer and download the agent update:

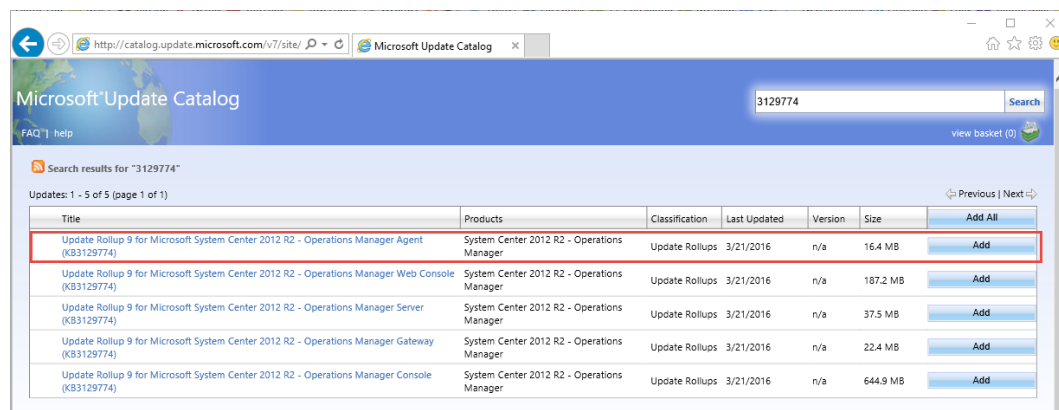
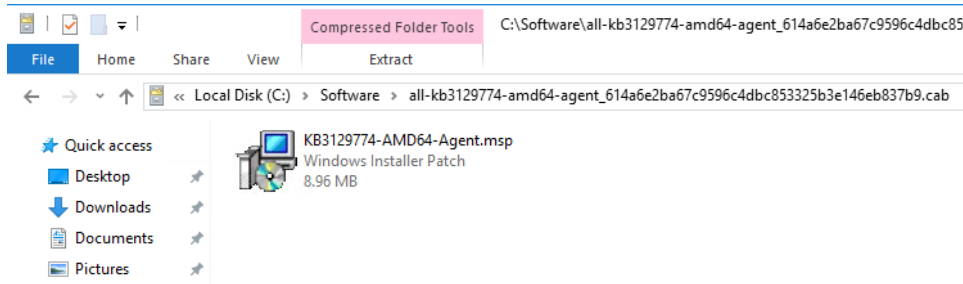
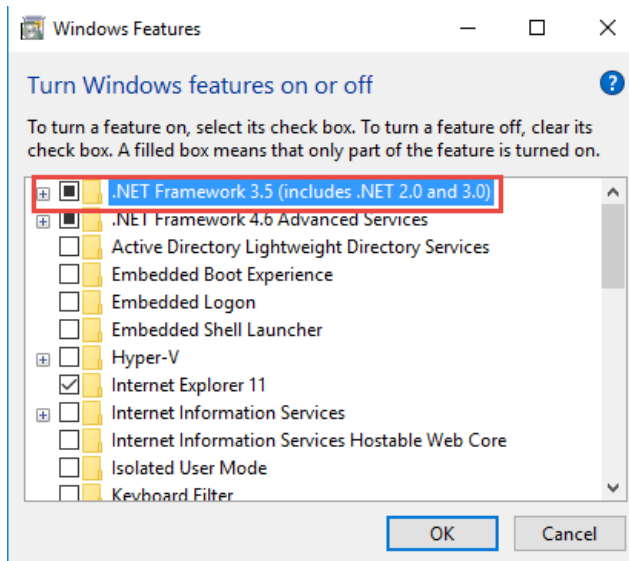


FIGURE 2.2.2 DOWNLOADING SCOM 2012 R2 UPDATE ROLLUP 9

Once the agent update is downloaded, we need to extract the update .msp file from the downloaded cab file. We can simply double click on the .cab file and then copy the agent update .msp to another location and then launch the update as administrator:

**FIGURE 2.2.3 EXTRACTING UR9 FROM CAB FILE**

After the Update Rollup is installed, please make sure .NET Framework 3.5 is also installed:

**FIGURE 2.2.4 INSTALLING .NET FRAMEWORK 3.5**

2.2.2 Installing Other Optional Software

Although it is not required when authoring MPs, we also recommend the following software for your authoring PCs:

Notepad++

Note: You can download Notepad ++ from: <https://notepad-plus-plus.org/>

When authoring management packs, Notepad++ is a great tool for viewing the unsealed MP XML files. We recommend you to install it on your authoring PC as well.

SCSM Entity Explorer

Note: You can download SCSM Entity Explorer from: <https://gallery.technet.microsoft.com/SCSM-Entity-Explorer-68b86bd2>

We can also use SCSM Entity Explorer to navigate SCOM monitoring classes (targets for our OMS collection rules). Although it was written primarily for SCSM, it also works with SCOM. There is no need to install it, please download it from TechNet Gallery and place it on your authoring PC. We will use it later.

MPViewer

Note: You can download MPViewer v 2.3.3 from:
http://blogs.msdn.com/cfs-file.ashx/_key/communityserver-blogs-components-weblogfiles/00-00-00-41-89-SCOM+Tools/6560.MPViewer.2.3.3.zip

MPViewer is a very handy utility when you need to view the content of an MP. You can also use it to unseal a sealed MP or MP bundle to XML files.

SCOM 2012 Operations Console

Although it is not required, we also recommend you to install the SCOM Operations Console installed on your authoring machine. It is convenient when importing and testing your MPs. It is important that you also patch the console to the same Update Rollup level as your management group.

PowerShell Tools for Visual Studio 2015

As we will be writing PowerShell scripts to be used in the demo MP, we also recommend you to install the PowerShell Tools for Visual Studio 2015. This is a Visual Studio Extension. You can find it within Visual Studio by going to “Tools → Extensions and Updates”, as shown in figure 2.2.5.

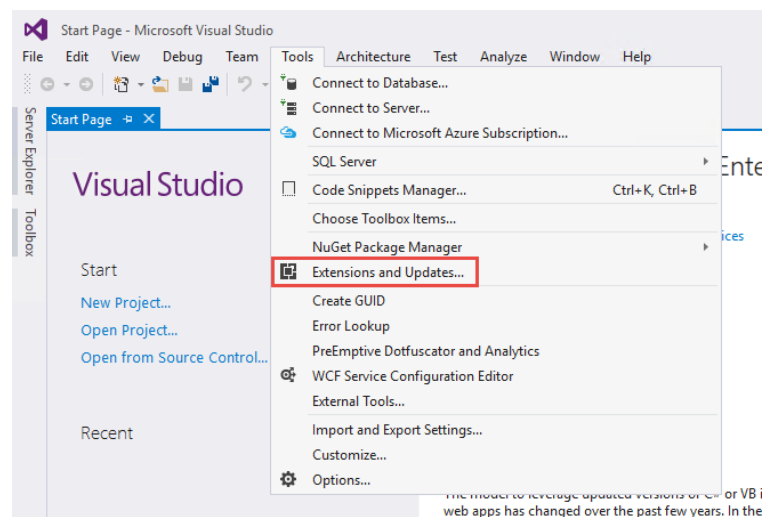


FIGURE 2.2.5 ACCESSING VISUAL STUDIO EXTENSIONS AND UPDATES

In the Extensions and Updates window, select “Online” category from the left, and you can find it by searching “PowerShell” in the search box, as shown in figure 2.2.6.

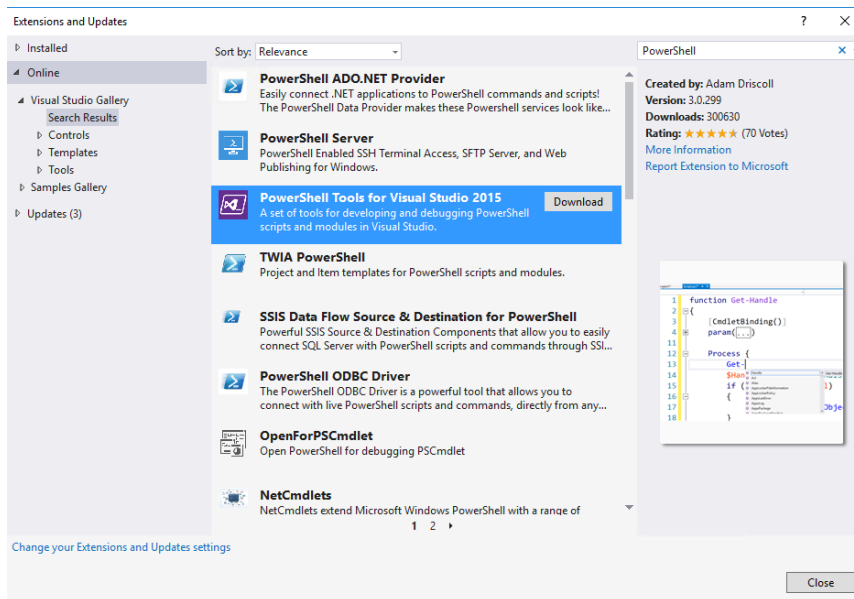


FIGURE 2.2.6 INSTALLING POWERSHELL TOOLS FOR VISUAL STUDIO 2015

2.3 Creating Reference Management Packs Folder

Create a folder **C:\MPs** on your authoring machine. This folder will be used to store any additional reference management packs for your VSAE projects.

Note: When authoring management packs, you will often need to reference MP elements from other MPs. When this is required, you will need to locate either sealed management packs (.mp files) or management pack bundles (.mpb files) that contain the elements you need to reference, and add them in your MP Visual Studio project as references. VSAE ships with many built-in MPs. After the VSAE is installed, you will find these MPs located in “**C:\Program Files (x86)\System Center 2012 Visual Studio Authoring Extensions\References**” folder. Since VSAE only comes with the built-in essential MPs for different versions of SCOM and SCSM, you will not be able to find application specific MPs in VSAE, such as SQL or SharePoint MPs. Therefore, we recommend you to create another folder on your authoring computer and place all other required reference MPs in this folder. If you are using multiple computers when authoring MPs, using a folder that can be accessed by multiple computers might be a better option (i.e. a network share or OneDrive).

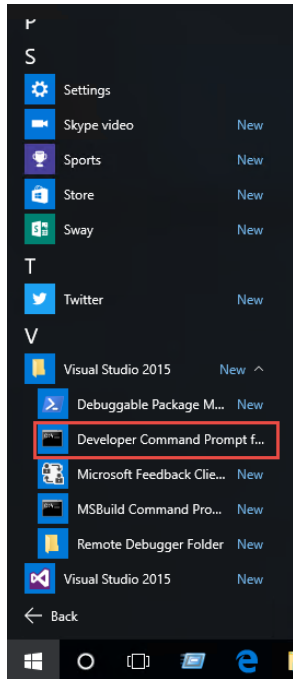
2.4 Creating a Key File for Sealing Management Packs

If you do not have an existing key file for sealing management packs, we will create one so we can generate sealed management packs within VSAE.

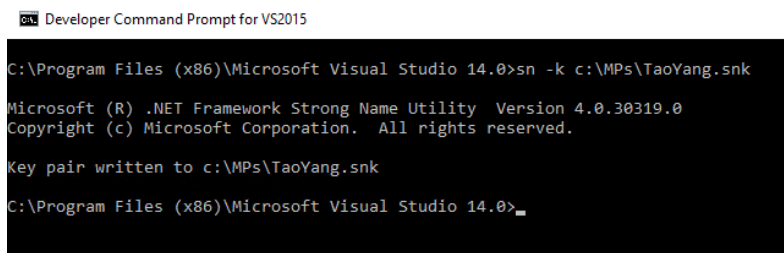
Note: If you already have a key file (.snk file) that you have used to seal MPs before, you can skip this section.

Please follow the following steps to create a key file:

1. Launch the “Developer Command Prompt for VS2015”

**FIGURE 2.4.1 DEVELOPER COMMAND PROMPT FOR VS2015**

2. Type command “`sn -k C:\MPs\<YourName>.snk`”

**FIGURE 2.4.2 CREATING KEY FILES USING SN.EXE**

Note: After the key file (.snk) is created, please save it securely. Please do not share this key publicly as it ensures the integrity of the MPs sealed by this key.

2.5 Download and Install Dependency MPs

Since we will create an agent task for SQL database servers, the management pack we are going to author will be depended on the Microsoft SQL management packs. Therefore, we must download and install the SQL management packs from Microsoft’s download site. Please follow the following steps to download and install the SQL MPs:

1. Download the SQL 2005/2008/2012 MP from <https://www.microsoft.com/en-us/download/details.aspx?id=10631>
2. Download the SQL 2014 MP from <https://www.microsoft.com/en-us/download/details.aspx?id=42573>
3. Unblock the downloaded files
 - a. Launch PowerShell and change to the folder to where the downloaded files are placed
 - b. Run command “`dir | unblock-file`” to unblock all the files in the folder


 Windows PowerShell
Windows PowerShell
Copyright (C) 2015 Microsoft Corporation. All rights reserved.
PS C:\Users\tyang> cd .\Downloads\
PS C:\Users\tyang\Downloads> dir | Unblock-File
PS C:\Users\tyang\Downloads> _

FIGURE 2.5.1 UNBLOCK DOWNLOADED FILES IN POWERSHELL

4. Make sure folder “C:\Temp” exists, create it if it does not exist.
5. Extract the SQL 2005/2008/2012 MP files from the Windows Installer (.msi) package.
 - a. Launch command prompt as administrator
 - b. Change the working directory to the folder where MP msi files are downloaded
 - c. Run “msiexec /a SQLServerMP.msi /qb TARGETDIR=C:\Temp”


 Administrator: Command Prompt
Microsoft Windows [Version 10.0.10586]
(c) 2015 Microsoft Corporation. All rights reserved.
C:\Windows\system32>cd c:\users\tyang\Downloads
c:\Users\tyang\Downloads>msiexec /a SQLServerMP.msi /qb TARGETDIR=C:\Temp\
c:\Users\tyang\Downloads>_

FIGURE 2.5.2 EXTRACTING SQL 2005/2008/2012 MPS FROM MSI

6. Browse to “C:\Temp\System Center Management Packs\Microsoft System Center Management Pack for SQL Server\6.6.4.0” and move all the files to “C:\MPs”
7. Extract the SQL 2014 MP files from the Windows Installer (msi) package.
 - a. In the previous command window, run


 Administrator: Command Prompt
Microsoft Windows [Version 10.0.10586]
(c) 2015 Microsoft Corporation. All rights reserved.
C:\Windows\system32>cd c:\users\tyang\Downloads
c:\Users\tyang\Downloads>msiexec /a SQLServerMP.msi /qb TARGETDIR=C:\Temp\
c:\Users\tyang\Downloads>msiexec /a SQLServer2014MP.msi /qb TARGETDIR=C:\Temp
c:\Users\tyang\Downloads>_

FIGURE 2.5.3 EXTRACTING SQL 2014 MPS FROM MSI

8. Browse to “C:\Temp\System Center Management Packs\Microsoft System Center Management Pack for SQL Server 2014\6.6.4.0” and move all the files to “C:\MPs”. When prompted that destination folder contains files with same names, choose the option to replace these files.
9. Import all the management packs extracted in previous steps into your SCOM management group
 - a. Open SCOM console, then go to **Administration** workspace, right click on ‘**Management Packs**’, then choose ‘**Import Management Packs...**’

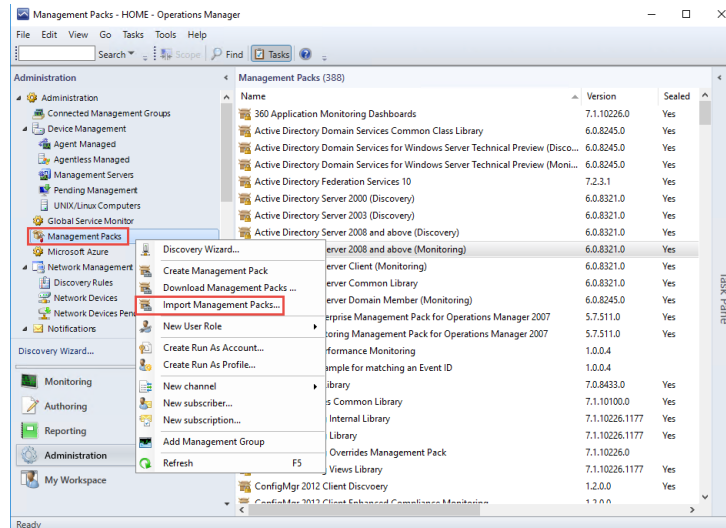


FIGURE 2.5.4 IMPORT MANAGEMENT PACKS

- b. In the 'Import Management Packs' dialog, click on 'Add' then 'Add from disk...'

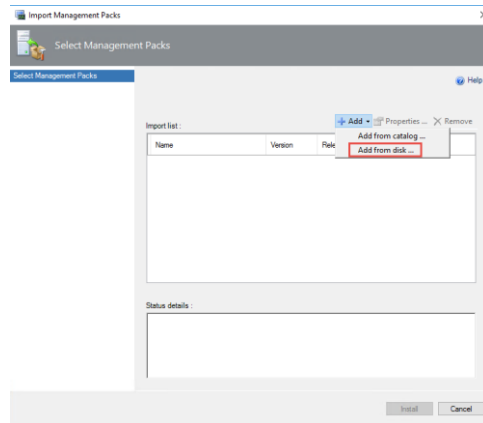


FIGURE 2.5.5 ADD MANAGEMENT PACKS FROM DISK

- c. Choose 'No' in the 'Online Catalog Connection' message box

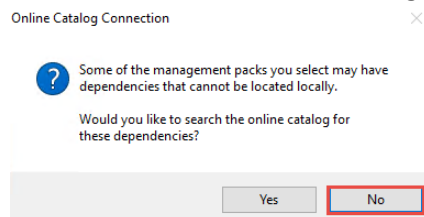


FIGURE 2.5.6 ONLINE CATALOG CONNECTION MESSAGE BOX

- d. Browse to C:\MPs and select all the management packs in this folder. Then click on 'Install'. This will take few minutes to complete.

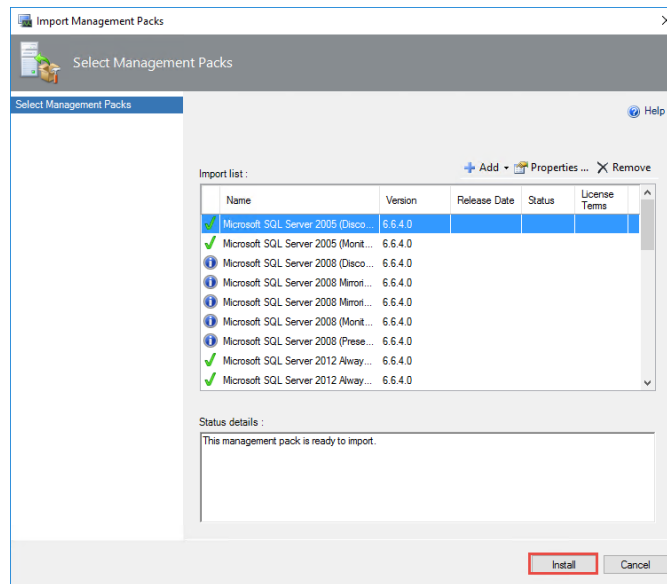


FIGURE 2.5.7 INSTALL MANAGEMENT PACKS

Note: If any management packs with the same or higher versions are already installed in your management group, they will be skipped in the installation process.

3 Creating Visual Studio Management Pack Project

Now that all the pre-requisites have been configured, we can start creating the management pack in Visual Studio 2015.

Note: We are going to create a project that is stored locally on the authoring computer. Although you can also store Visual Studio projects in Source Control systems such as TFS, Visual Studio Online or Github, source control is not in the scope of this workshop.

3.1 Creating VSAE Project and Defining Basic MP Properties

To create a new VSAE management pack project, firstly launch Visual Studio 2015, and then choose **File**→ **New**→ **Project**.

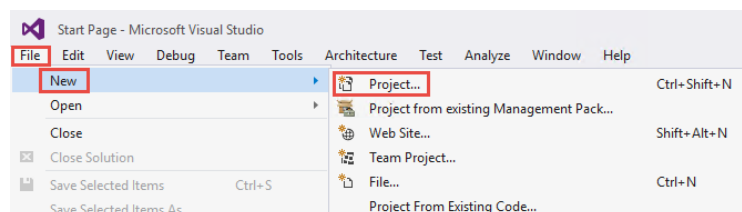


FIGURE 3.1.1 CREATING NEW VISUAL STUDIO PROJECT

In the 'New Project' dialog box, choose '**Operations Manager 2012 Management Pack**' and specify the following properties:

Name: <YourName>.SquaredUp.VSAE.Workshop.Demo

Location: C:\documents\visual studio 2015\Projects

Solution Name: Leave as default

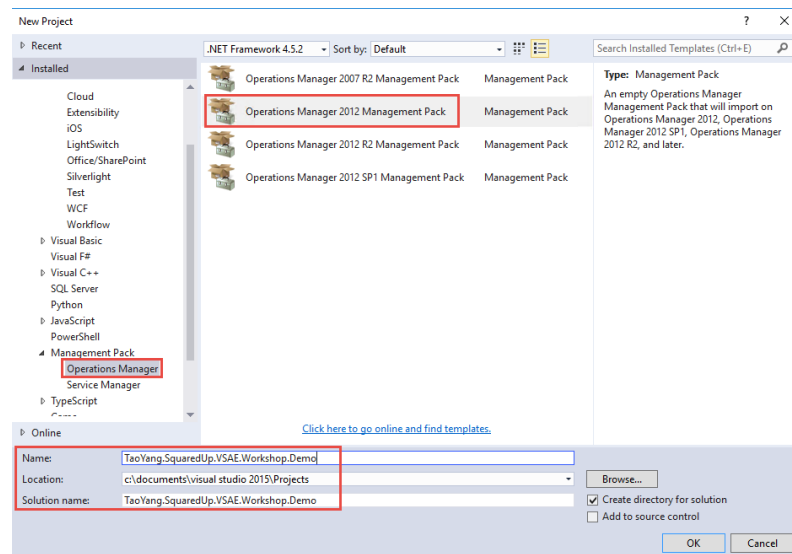


FIGURE 3.1.2 CREATING NEW MANAGEMENT PACK SOLUTION

Note: When selecting the management pack solutions, please always select the minimum supported SCOM version that you wish your management pack to run. This will ensure your MP runs on earlier versions of SCOM.

We will then set the management pack version to 0.0.0.1 and give it a friendly name called '<YourName> SquaredUp VSAE Workshop Demo'. We can do so by right clicking on the management pack project that you have just created from the Solution Explorer and choose 'Properties', as shown in figure 3.1.3 and 3.1.4

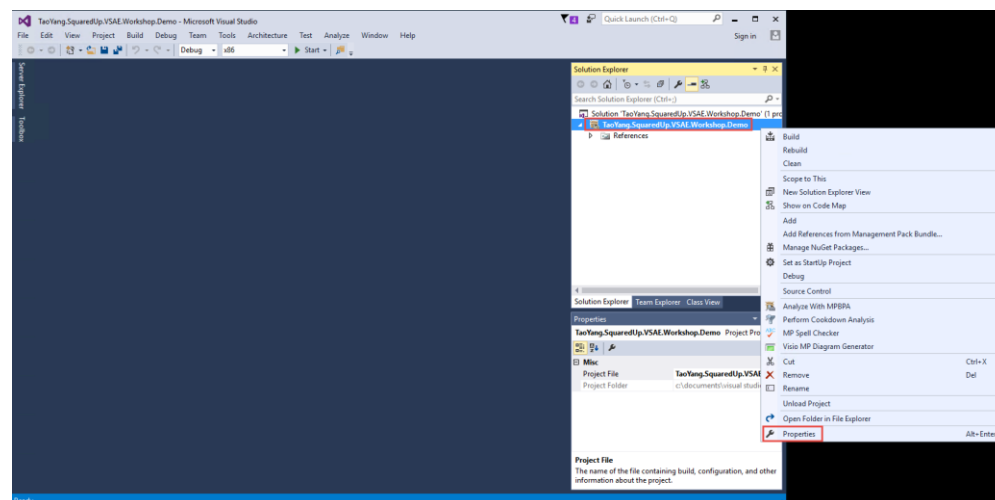


FIGURE 3.1.3 CONFIGURING MANAGEMENT PACK PROJECT PROPERTIES

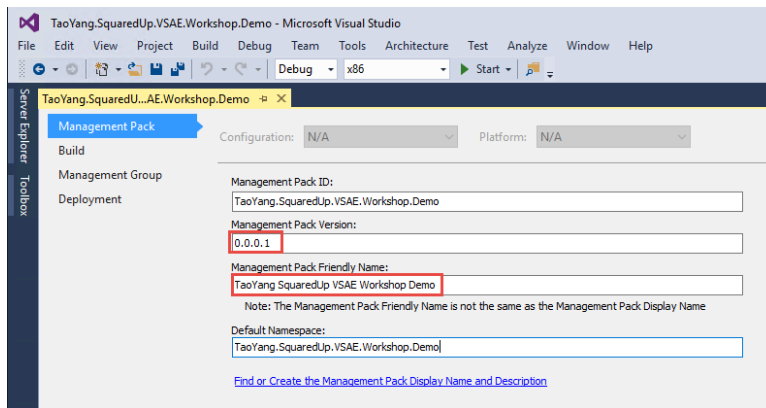


FIGURE 3.1.4 SPECIFYING MANAGEMENT PACK VERSION AND FRIENDLY NAME

Next, since we are going to build a sealed management pack, we will have to fill out the fields under 'Build' tab, as shown in figure 3.1.5

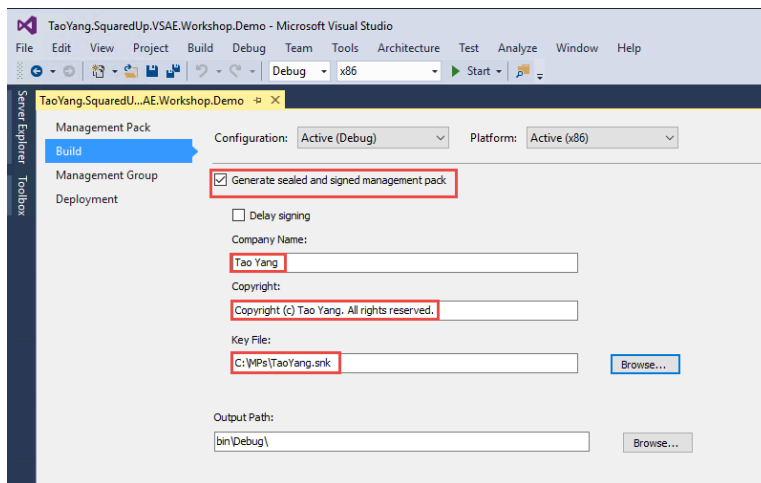


FIGURE 3.1.5 CONFIGURING PROPERTIES FOR SEALED MPS

- Check 'Generate sealed and signed management pack'
- **Company Name:** <Your name>
- **Copyright:** Copyright (c) <Your name>. All rights reserved.
- **Key File:** <Path to the key file created in section 2.4>

Lastly, go to the 'Deployment' tab and check the '**Auto-increment version**' check box.

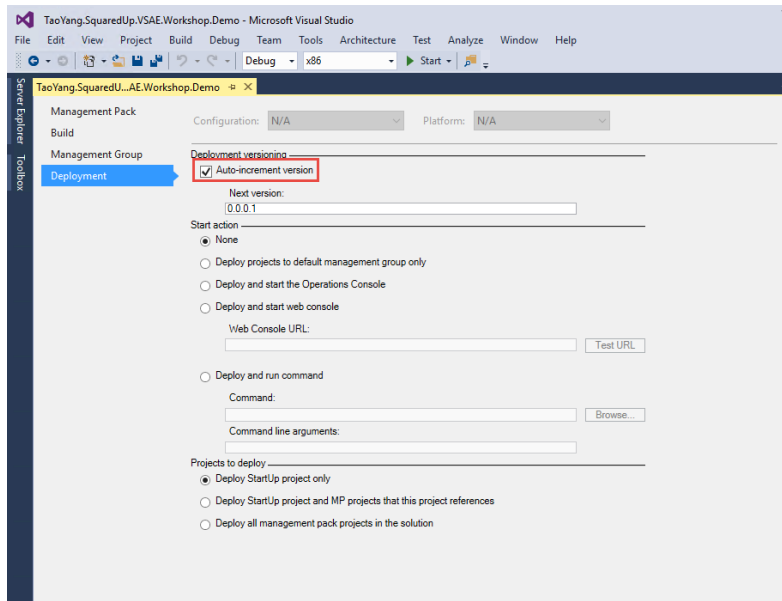


FIGURE 3.1.6 CONFIGURING MP VERSION AUTO-INCREMENT

Note: To simplify the process of incrementing MP versions when authoring sealed MPs, the “Auto-increment version” feature may come very handy. When you have enabled this feature, the MP version will automatically increase by 0.0.0.1 when every time you click “Build Solution” to build the MP.

After we have configured all the properties for the management pack, we will then configure the MP display name and description. The display name and the description that you are going to specify here is what people are going to see in SCOM console once you have imported the MP into your management group. To do so, right click the management pack project, and choose **Properties**, then click on 'Find or Create the ManagementPack Display Name and Description' under the 'Management Pack' tab as shown in figure 3.1.7

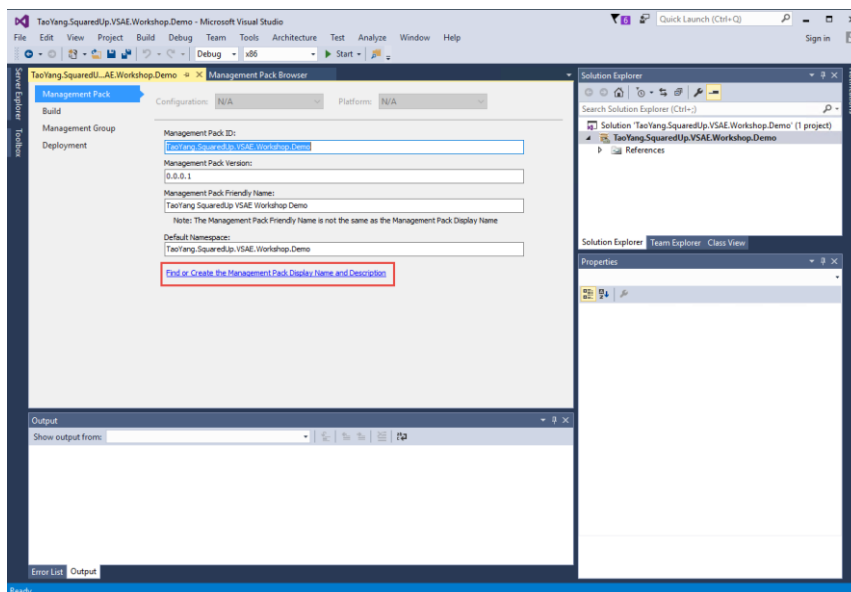


FIGURE 3.1.7 LINK TO CREATE MP FRAGMENT FOR DISPLAY NAME AND DESCRIPTION

This will create a management pack fragment called “**ManagementPack.mpx**”, as shown in figure 3.1.8.

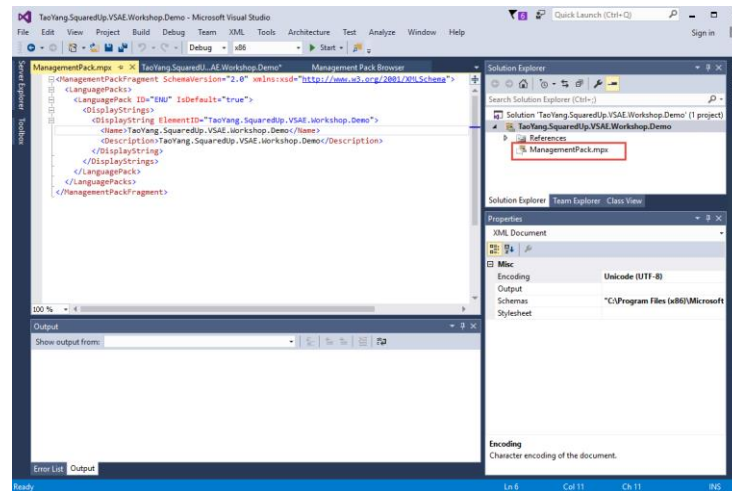


FIGURE 3.1.8 MANAGEMENTPACK.MPX

After the ManagementPack.mpx file is created, change the display name and description from the <Name> and <Description> tag respectively:

Display Name: YourName SquaredUp VSAE Workshop Demo

Description: This is a demo management pack created in the SquaredUp VSAE authoring workshop.

Note: Please replace ‘YourName’ with your name in the display name field.

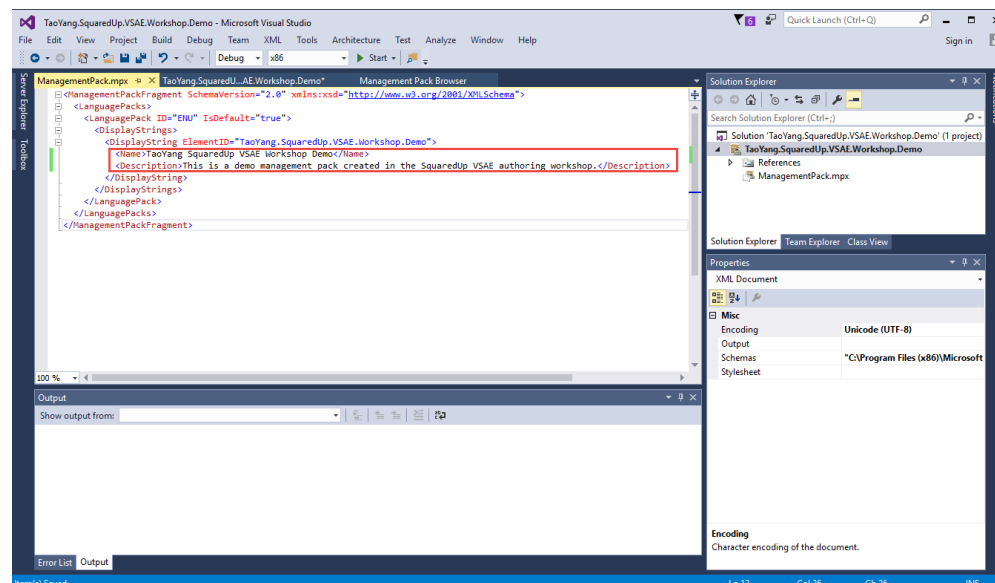


FIGURE 3.1.9 UPDATED MANAGEMENTPACK.MPX

3.2 Build the Management Pack

Now that we have defined all necessary properties of the management pack and defined the display name in the <LanguagePacks> tag, we can build and test the management pack.

Please follow the following steps to build the management pack.

1. In Visual Studio, click on **'Build'** menu and select **'Build Solution'**.

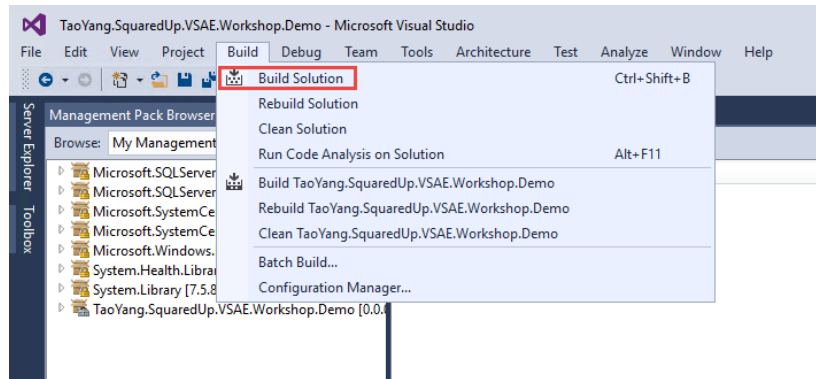


FIGURE 3.2.1 BUILD VSAE SOLUTION

2. Wait until the build process finishes. The result is displayed in the output pane. Make sure the last line of the build result is displayed as **'Build: 1 succeeded, 0 failed, 0 up-to-date, 0 skipped'**.

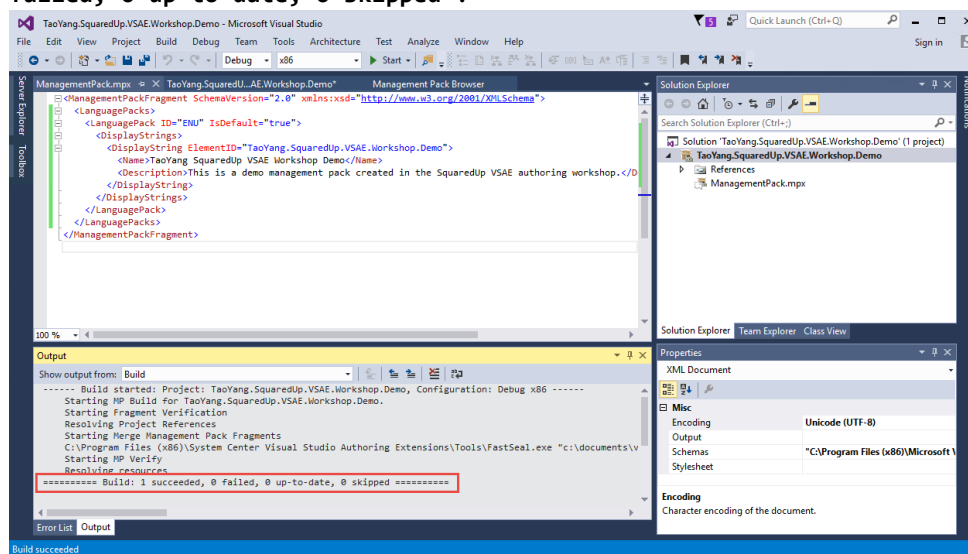


FIGURE 3.2.2 MANAGEMENT PACK BUILD RESULT

3. Browse to **'C:\Documents\Visual Studio 2015\Projects\<YourName>.SquaredUp.VSAE.Workshop.Demo\<YourName>.SquaredUp.VSAE.Workshop.Demo\bin\Debug'** folder, and locate the sealed MP **<YourName>.SquaredUp.VSAE.Workshop.Demo.mp**

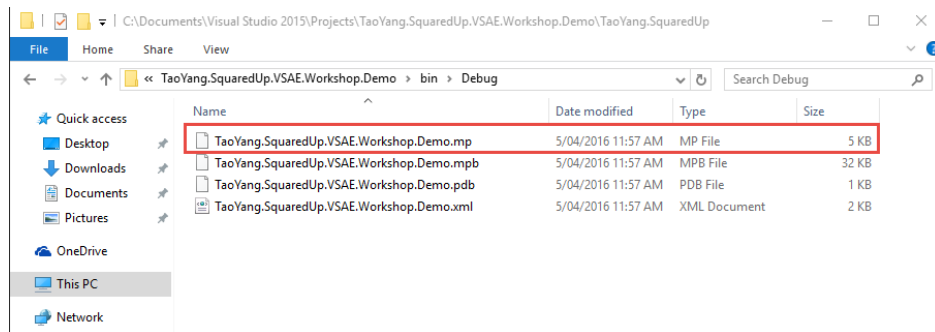


FIGURE 3.2.3 SEALED MANAGEMENT PACK GENERATED BY VSAE

3.3 Import Management Pack into SCOM

Now that we have the management pack ready, we can import it into SCOM. Please follow the following steps to import it into SCOM:

1. Launch SCOM console and go to **'Administration'** pane.
2. Right click on **'Management Packs'** and select **'Import Management Packs...'**
3. Click on **'Add'** then **'Add from disk...'**
4. Click on **'No'** on the 'Online Catalog Connection' message box
5. Select the management pack **'<YourName>.SquaredUp.VSAE.Workshop.Demo.mp'** from the folder stated in the previous section.
6. Click on **'Install'** to begin importing the management pack.
7. Make sure the MP is imported successfully and then close the **'Import Management Packs'** window.

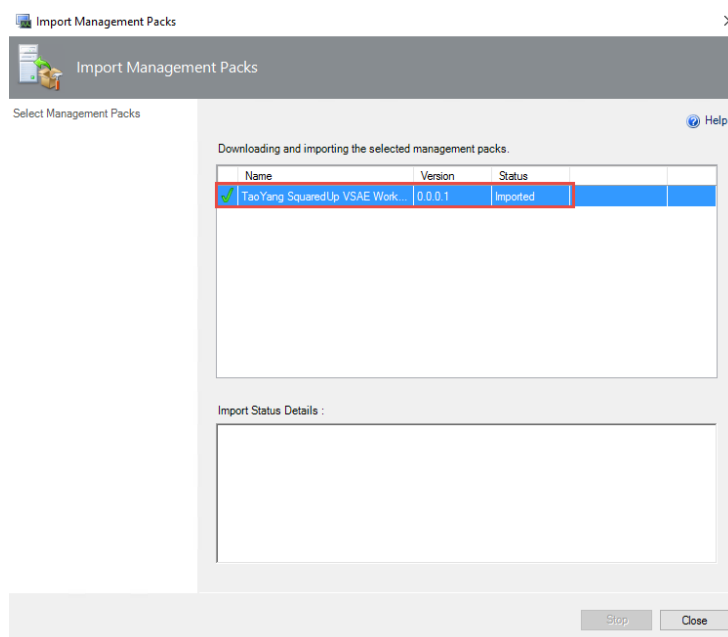


FIGURE 3.3.1 MANAGEMENT PACK IMPORT RESULT

Now you should see the newly created sealed (blank) management pack in your OpsMgr console.

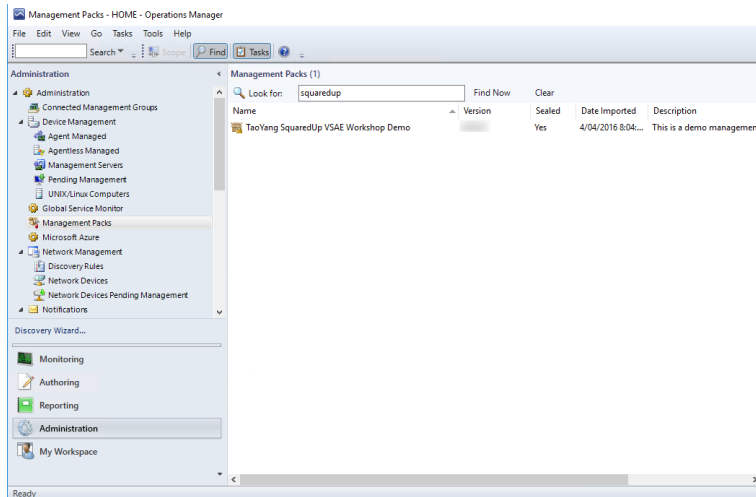


FIGURE 3.3.2 NEWLY CREATED MP IN OPSMGR CONSOLE

4 Creating a Simple Agent Task Using VSAE Template

Let's start with a simple task first. We will create an agent task against Windows Computers that execute a simple PowerShell command:

```
Invoke-Expression Systeminfo.exe
```

Before we start creating this agent task in VSAE, let's test run this command in PowerShell command window:

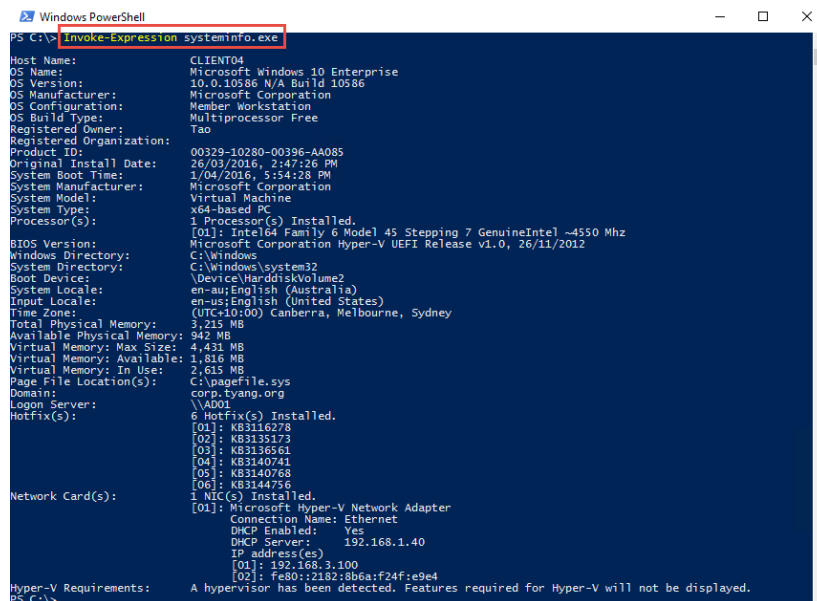


FIGURE 4.1 INVOKE-EXPRESSION SYSTEMINFO.EXE

Now, to start creating the agent task, we will firstly create a PowerShell script within the Visual Studio project.

Please follow the following steps to create the PowerShell script:

1. Create a folder called 'Scripts' in the management pack solution. To do so, right click on the MP solution in Visual Studio Solution Explorer, and select 'Add' → 'New Folder', then name the folder 'Scripts'.

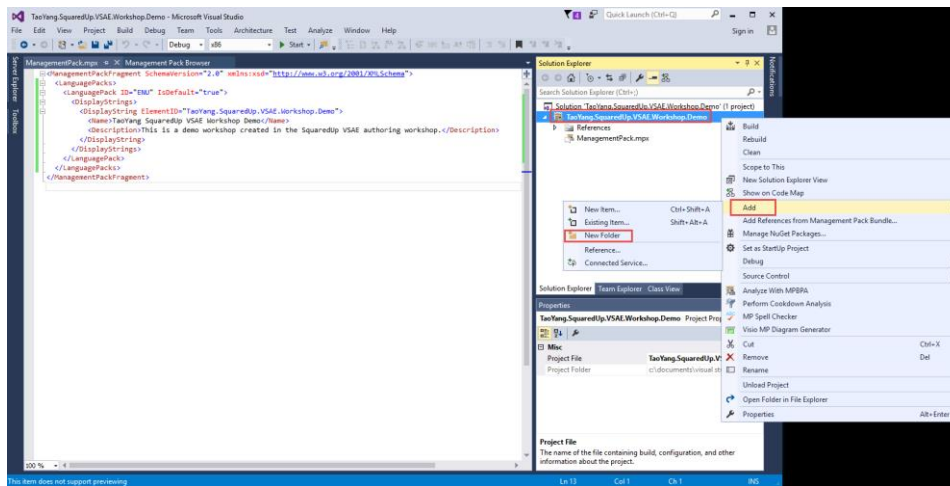


FIGURE 4.2 CREATE NEW FOLDERS IN VSAE SOLUTIONS

2. Create a PowerShell script by righting click on the newly created 'Scripts' folder, select **'Add' → 'New' select 'PowerShell script file'** and name the file **'Invoke-SystemInfo.ps1'**

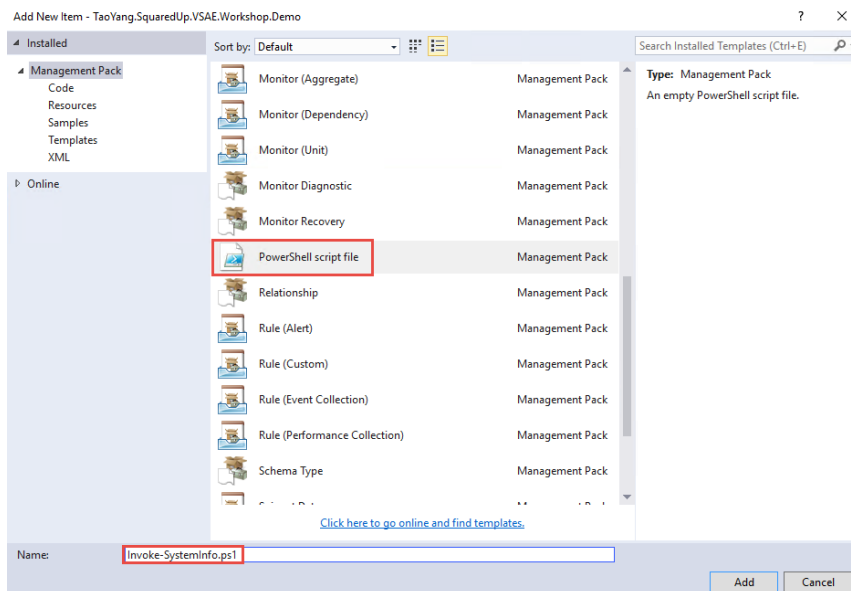


FIGURE 4.3 CREATE NEW POWERSHELL SCRIPT FILE

3. Add 'Invoke-Expression SystemInfo.exe' in the newly created PowerShell script and save it.
4. Create Another folder call "Tasks"
5. Add an **'Agent Task (PowerShell Script)'** template and name it **'SystemInfoTask.mptg'**.

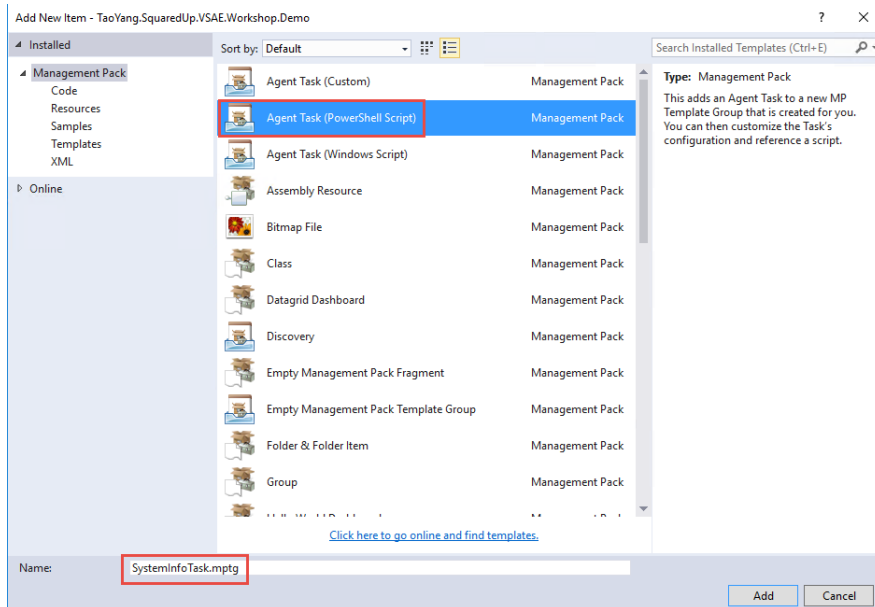


FIGURE 4.4 ADD AGENT TASK (POWERSHELL SCRIPT) TEMPLATE

6. Click on the 'NewPowerShellTask' and update the following properties:
 - Remotable: **False**
 - Category: **Operations**
 - Description: **Invoking SystemInfo.exe on Windows computers.**
 - Display Name: **Invoke Systeminfo - <YourName>**
 - ID: **Invoke.Systeminfo.Agent.Task**
 - Target: Click on '...' button and choose Microsoft.Windows.Computer from the list

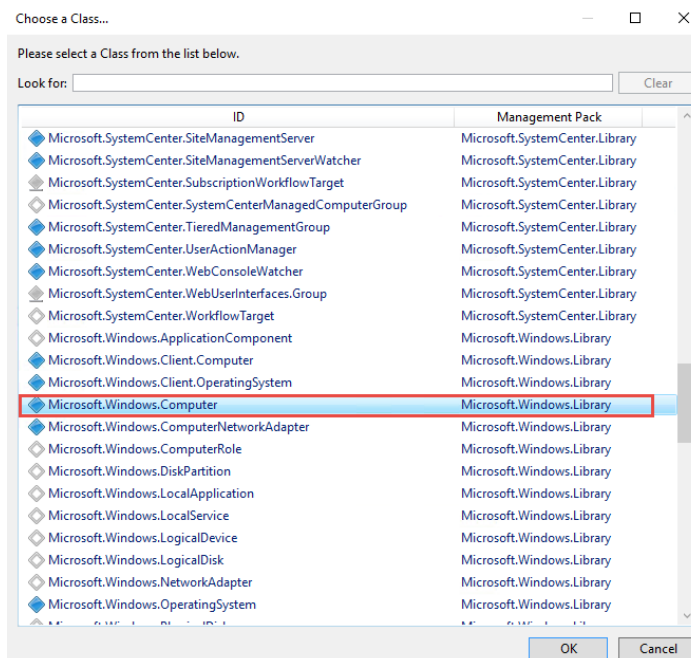


FIGURE 4.5 CHOOSING TASK TARGET

- Script: Click on '...' button and choose the 'Invoke-SystemInfo.ps1' that we have just created.

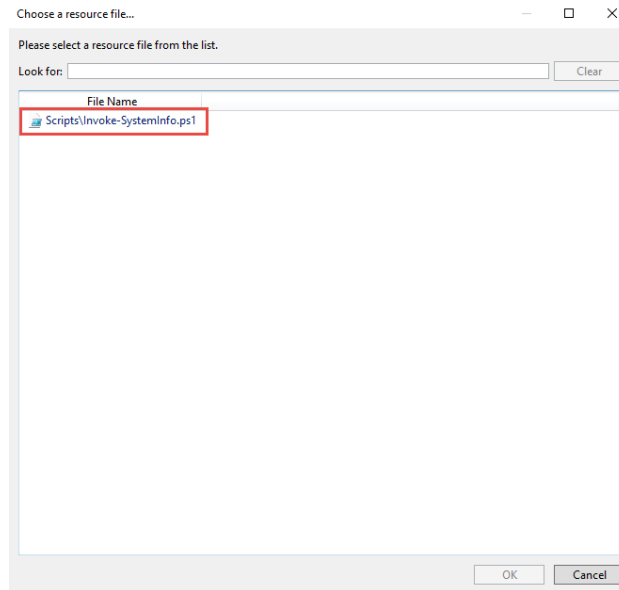


FIGURE 4.6 CHOOSING TASK SCRIPT

- Click Save button to save the agent task inside the template.

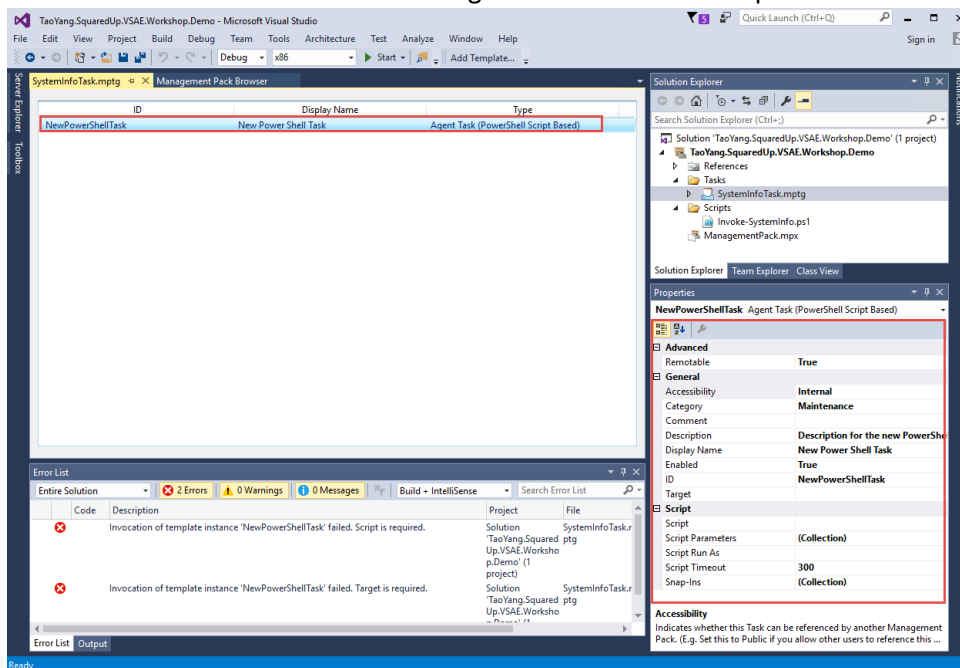


FIGURE 4.6 CREATING TASKS WITHIN THE TEMPLATE

7. Build the management pack again and import it into the OpsMgr management group.
8. Test the 'Invoke Systeminfo' agent task in OpsMgr console.
 - a. In the 'Monitoring' pane, browse to the 'Windows Computer' state view and locate the 'Invoke Systeminfo' agent task in the task pane.

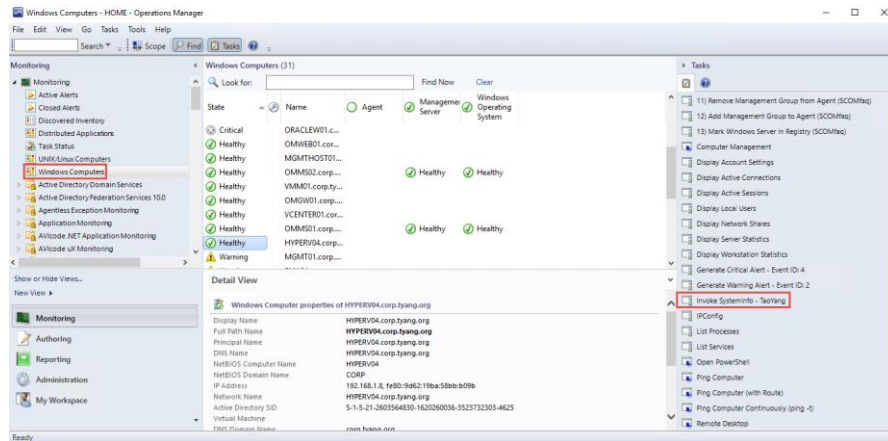


FIGURE 4.7 LOCATING THE INVOKE SYSTEMINFO AGENT TASK IN OPSMGR CONSOLE

- b. Run the 'Invoke Systeminfo' agent task against a Windows computer of your choice.

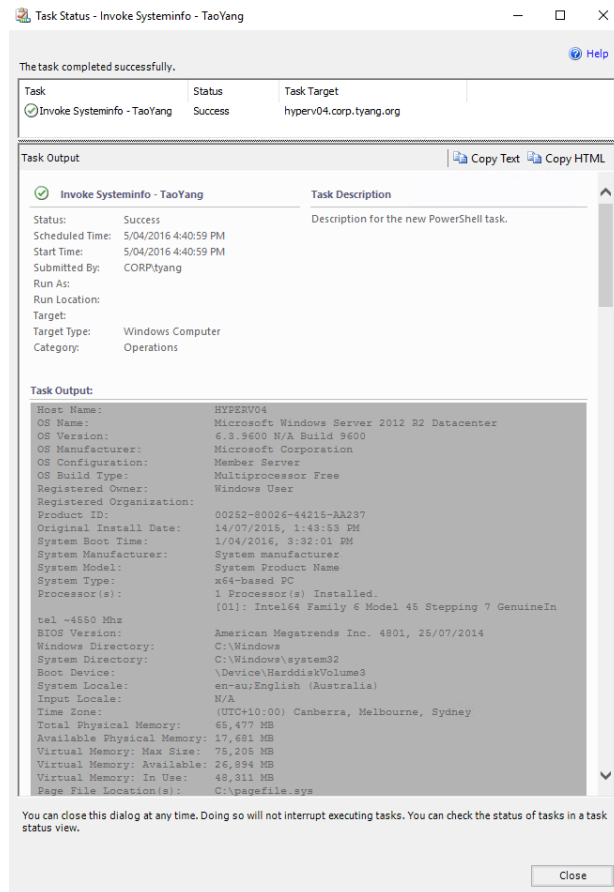


FIGURE 4.8 AGENT TASK RESULT

5 Creating Complex Agent Tasks

Now that we have created a simple PowerShell based agent task using the VSAE templates, we will now create some more complex agent tasks. We will create two new almost identical

tasks to get the long running queries on SQL 2014 database engines and Pre SQL 2014 (2005/2008/2012) database engines.

5.1 Add Reference Management Packs

Because we are creating agent tasks for SQL database engines, we will be referencing the objects defined in the native Microsoft SQL management packs. Therefore, these management packs must be added as references. In this case, we will need to add the following sealed management pack (.mp) files:

- Microsoft.SQLServer.Library.mp
- Microsoft.SQLServer2014.Discovery.mp

Please follow the following steps to add these sealed management packs as references:

1. In the Solution Explorer, right click on **'References'** under the management pack solution and select **'Add References...'**

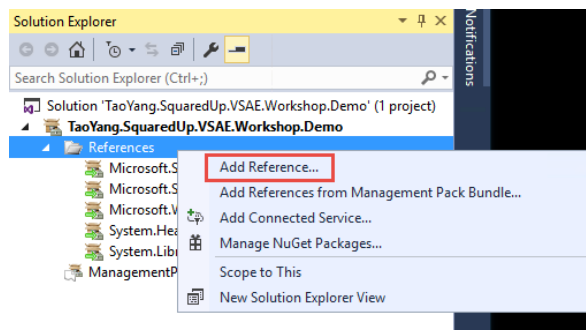


FIGURE 5.1.1 ADDING REFERENCING MANAGEMENT PACKS

2. In the **'Add Reference'** dialog box, go to the **'Browse'** tab and select the **'Microsoft.SQLServer.Library.mp'** located in **'C:\MPs'** folder and click **'OK'**.

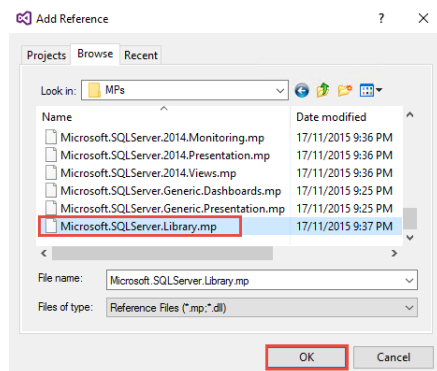


FIGURE 5.1.2 SELECT REFERENCING SEALED MP FILE

3. Click on the newly added reference **'Microsoft.SQLServer.Library'** and change the Alias to **'SQL'** under Properties.

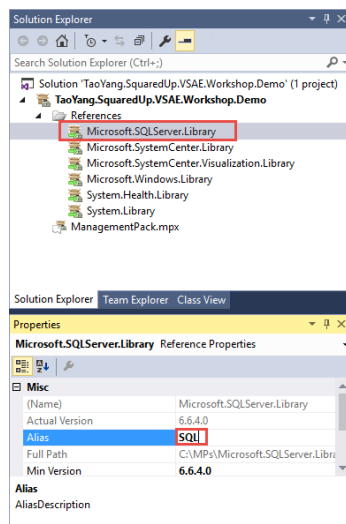


FIGURE 5.1.3 SET REFERENCING MP ALIAS

- Repeat step 1-3 for the '**Microsoft.SQLServer.2014.Discovery.mp**' and set the alias as '**SQL2014Disc**'

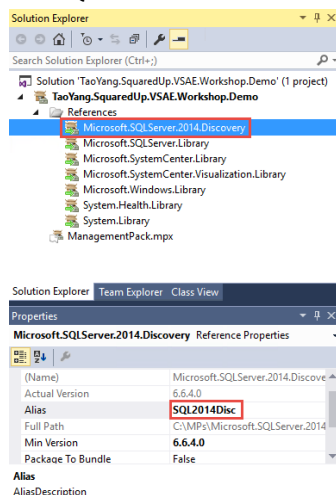


FIGURE 5.1.4 SET SQ 2014 DISCOVERY MP REFERENCE ALIAS

5.2 Create PowerShell Script Used by the Agent Task

Please follow the following steps to create a PowerShell script to get SQL database server long running queries:

- Create a folder called 'Scripts' in the management pack solution if you have not created it in the previous section. To do so, right click on the MP solution in Visual Studio Solution Explorer, and select **'Add' → 'New Folder'**, then name the folder **'Scripts'**.
- Create a PowerShell script by righting click on the newly created 'Scripts' folder, select **'Add' → 'New'** select **'PowerShell script file'** and name the file **'GetDBEngineLongRunningQueries.ps1'**
- Populate the newly created PowerShell script as shown below:

```
#=====
# AUTHOR:          Tao Yang
# Script Name:      GetDBEngineLongRunningQueries.ps1
# DATE:            28/09/2015
```

```
# Version: 1.0
# COMMENT: - Script to get the long running queries on a SQL
DB Engine
#=====
Param ([string]$ConnectionString, [String]$TCPPort, [int]$TopN,
[int]$SQLQueryTimeoutSeconds)

$SQLQuery = @"
SELECT TOP $TopN creation_time
      ,last_execution_time
      ,total_physical_reads
      ,total_logical_reads
      ,total_logical_writes
      , execution_count
      , total_worker_time
      , total_elapsed_time
      , total_elapsed_time / execution_count avg_elapsed_time
      ,SUBSTRING(st.text, (qs.statement_start_offset/2) + 1,
      ((CASE statement_end_offset
      WHEN -1 THEN DATALength(st.text)
      ELSE qs.statement_end_offset END
      - qs.statement_start_offset)/2) + 1) AS statement_text
FROM sys.dm_exec_query_stats AS qs
CROSS APPLY sys.dm_exec_sql_text(qs.sql_handle) st
ORDER BY total_elapsed_time / execution_count DESC;
"@

#Connect to the DB engine
If ($TCPPort -ne $NULL -AND $TCPPort -ne 0)
{
    $conString = "Server`=$ConnectionString,$TCPPort;Integrated
Security=true;Initial Catalog=master"
} else {
    $conString = "Server`=$ConnectionString;Integrated
Security=true;Initial Catalog=master"
}

Write-Output "Connection String: '$conString'"
#Write-Output $SQLQuery
$SQLCon = New-Object System.Data.SqlClient.SqlConnection
$SQLCon.ConnectionString = $conString
$SQLCon.Open()

#execute SQL queries
$sqlCmd = $SQLCon.CreateCommand()
$sqlCmd.CommandTimeout=$SQLQueryTimeoutSeconds
$SqlAdapter = New-Object System.Data.SqlClient.SqlDataAdapter

#Alert Staging table
$sqlCmd.CommandText = $SQLQuery
$SqlAdapter.SelectCommand = $sqlCmd
$DataSet = New-Object System.Data.DataSet
$SqlAdapter.Fill($DataSet) | Out-Null

#Close SQL Connection
$SQLCon.Close()

#Process result
$arrSQLResult = New-Object System.Collections.ArrayList

Foreach ($set in $DataSet.Tables[0])
{
```

```
$objDS = New-object psobject
Foreach ($objProperty in (Get-member -InputObject $set -
MemberType Property))
{
    $PropertyName = $objProperty.Name
    Add-Member -InputObject $objDS -MemberType NoteProperty -Name
$PropertyName -Value $set.$PropertyName
}
[void]$arrSQLResult.Add($objDS)
}
Write-Output "Top $TopN Long Running Queries in the DB Engine:"
Write-Output "-----"
Foreach ($item in $arrSQLResult)
{
    Write-Output $item
    Write-Output "-----"
    Write-Output ""
}
}
```

4. Save and close the PowerShell script file.

5.3 Create Custom Write Action Module for the Agent Task

Next, we will create a custom Write Action Module and wrap the PowerShell script within this module.

Note: Although we can just use a native SCOM PowerShell Write Action Module 'Microsoft.Windows.PowerShellWriteAction', we are going to create a custom module in this workshop. We are doing so to help you understand the process of creating custom modules. Additionally, having a custom module have the following benefits in this case:

More flexible when defining input and overrideable parameters

Ability to hide the actual PowerShell script from the SCOM users in the console.

Please follow the following steps to create the custom Write Action module:

1. In the management pack solution, create a new folder called '**Modules**'.
2. Add a new Empty Management Pack Fragment in side of the 'Modules' folder and name it '**GetDBEngineLongRunningQueries.mpx**'
3. Place the following XML code inside the <ManagementPackFragment> tag:

```
<TypeDefinitions>
  <ModuleTypes>
    <WriteActionModuleType
ID="YourName.SquaredUp.VSAE.Workshop.Demo.Get.DB.Engine.Long.Running.
Queries.WA" Accessibility="Internal" Batching="false"
RunAs="SQL!Microsoft.SQLServer.SQLDefaultAccount">
      <Configuration>
        <xsd:element name="ConnectionString" type="xsd:string"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" />
        <xsd:element name="TCPPort" type="xsd:string"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" />
      </Configuration>
    </WriteActionModuleType>
  </ModuleTypes>
</TypeDefinitions>
```

```

        <xsd:element name="TopN" type="xsd:int"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" />
        <xsd:element name="SQLQueryTimeoutSeconds" type="xsd:int"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" />
        <xsd:element name="TimeoutSeconds" type="xsd:int"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" />
    </Configuration>
    <OverrideableParameters>
        <OverrideableParameter ID="TopN" Selector="$Config/TopN$"
ParameterType="int" />
        <OverrideableParameter ID="SQLQueryTimeoutSeconds"
Selector="$Config/SQLQueryTimeoutSeconds$" ParameterType="int" />
        <OverrideableParameter ID="TimeoutSeconds"
Selector="$Config/TimeoutSeconds$" ParameterType="int" />
    </OverrideableParameters>
    <ModuleImplementation>
        <Composite>
            <MemberModules>
                <WriteAction ID="WA"
TypeID="Windows!Microsoft.Windows.PowerShellWriteAction">

<ScriptName>GetDBEngineLongRunningQueries.ps1</ScriptName>

<ScriptBody>$IncludeFileContent/Scripts/GetDBEngineLongRunningQueries
.ps1$</ScriptBody>
                <Parameters>
                    <Parameter>
                        <Name>ConnectionString</Name>
                        <Value>$Config/ConnectionString$</Value>
                    </Parameter>
                    <Parameter>
                        <Name>TCPPort</Name>
                        <Value>$Config/TCPPort$</Value>
                    </Parameter>
                    <Parameter>
                        <Name>TopN</Name>
                        <Value>$Config/TopN$</Value>
                    </Parameter>
                    <Parameter>
                        <Name>SQLQueryTimeoutSeconds</Name>
                        <Value>$Config/SQLQueryTimeoutSeconds$</Value>
                    </Parameter>
                </Parameters>

<TimeoutSeconds>$Config/TimeoutSeconds$</TimeoutSeconds>
                </WriteAction>
            </MemberModules>
            <Composition>
                <Node ID="WA" />
            </Composition>
        </Composite>
    </ModuleImplementation>
    <InputType>System!System.BaseData</InputType>
</WriteActionModuleType>
</ModuleTypes>
</TypeDefinitions>
<LanguagePacks>
    <LanguagePack ID="ENU" IsDefault="true">
        <DisplayStrings>

```

```

        <DisplayString
ElementID="YourName.SquaredUp.VSAE.Workshop.Demo.Get.DB.Engine.Long.R
unning.Queries.WA">
        <Name>Get Database Engine Long Running Queries Write Action
Module</Name>
        </DisplayString>
        <DisplayString
ElementID="YourName.SquaredUp.VSAE.Workshop.Demo.Get.DB.Engine.Long.R
unning.Queries.WA" SubElementID="SQLQueryTimeoutSeconds">
        <Name>SQL Query Timeout Seconds</Name>
        </DisplayString>
        <DisplayString
ElementID="YourName.SquaredUp.VSAE.Workshop.Demo.Get.DB.Engine.Long.R
unning.Queries.WA" SubElementID="TimeoutSeconds">
        <Name>Script Timeout Seconds</Name>
        </DisplayString>
        <DisplayString
ElementID="YourName.SquaredUp.VSAE.Workshop.Demo.Get.DB.Engine.Long.R
unning.Queries.WA" SubElementID="TopN">
        <Name>Number of Top Long Running Queries To Retrieve</Name>
        </DisplayString>
    </DisplayStrings>
</LanguagePack>
</LanguagePacks>

```

4. Replace 'YourName' with your name in the code snippet above.

5.4 Create Agent Task for Pre SQL 2014 Database Engines

Note: When Microsoft developed the SQL 2014 management packs, they have changed the management pack class relationship structures. As the result, we are not able to create a single agent task for all versions of SQL database engines because SQL 2014 Database Engine class is based on a different class than the pre SQL 2014 versions. In this section, we will create the agent task to get long running queries for the database engine based on the PowerShell script and the Write Action custom module we have just defined. In the next section, we will repeat the same process and create a similar agent task for SQL 2014 Database Engines. If you do not have any Pre SQL 2014 Database Engines, please skip this section and move to the next section.

Please follow the following steps to create the 'Get DB Engine Long Running Queries' agent task for Pre SQL 2014 Database Engines:

1. In the management pack solution, create a new folder called 'Tasks' (if you have not done this in the previous section).
2. Create a new Empty Management Pack Fragment and name it 'SQL.DBEngine.GetLongRunningQueries.Task.mpx'
3. Place the following XML code inside the <ManagementPackFragment> tag:

```

<Monitoring>
    <Tasks>
        <Task
ID="YourName.SquaredUp.VSAE.Workshop.Demo.DB.Engine.Get.Long.Runni
ng.Queries.Task" Accessibility="Internal" Timeout="600"
Enabled="true" Remotable="true"
Target="SQL!Microsoft.SQLServer.DBEngine">

```

```

        <Category>Operations</Category>
        <WriteAction ID="WA"
TypeID="YourName.SquaredUp.VSAE.Workshop.Demo.Get.DB.Engine.Long.R
unning.Queries.WA">

<ConnectionString>$Target/Property[Type="SQL!Microsoft.SQLServer.D
BEngine"]/ConnectionString$</ConnectionString>

<TCPPort>$Target/Property[Type="SQL!Microsoft.SQLServer.DBEngine"]
/TcpPort$</TCPPort>
        <TopN>5</TopN>
        <SQLQueryTimeoutSeconds>300</SQLQueryTimeoutSeconds>
        <TimeoutSeconds>600</TimeoutSeconds>
        </WriteAction>
    </Task>
</Tasks>
</Monitoring>
<LanguagePacks>
    <LanguagePack ID="ENU" IsDefault="true">
        <DisplayStrings>
            <DisplayString
ElementID="YourName.SquaredUp.VSAE.Workshop.Demo.DB.Engine.Get.Lon
g.Running.Queries.Task">
                <Name>Get DB Engine Long Running Queries -
YourName</Name>
            </DisplayString>
        </DisplayStrings>
    </LanguagePack>
</LanguagePacks>

```

4. Replace 'YourName' with your name in the code snippet above.

5.5 Create Agent Task for SQL 2014 Database Engines

As explained in the previous section, we must create separate but almost identical tasks for pre SQL 2014 and SQL 2014 database engines. This section will walk through the steps of creating an identical task for SQL 2014 database engine (targeting the SQL 2014 database engine class).

Note: If you do not have any SQL 2014 Database Engines in your environment, you may skip this section.

Please follow the following steps to create the 'Get DB Engine Long Running Queries' agent task for Pre SQL 2014 Database Engines:

1. In the management pack solution, create a new folder called 'Tasks' if you have skipped the last section.
2. Create a new Empty Management Pack Fragment and name it 'SQL2014.DBEngine.GetLongRunningQueries.Task.mpx'
3. Place the following XML code inside the <ManagementPackFragment> tag:

```

<Monitoring>
    <Tasks>
        <Task
ID="YourName.SquaredUp.VSAE.Workshop.Demo.SQL2014.DB.Engine.Get.Lo
ng.Running.Queries.Task" Accessibility="Internal" Timeout="600"
Enabled="true" Remotable="true"
Target="SQL2014Disc!Microsoft.SQLServer.2014.DBEngine">

```



```

        <Category>Operations</Category>
        <WriteAction ID="WA"
TypeID="YourName.SquaredUp.VSAE.Workshop.Demo.Get.DB.Engine.Long.R
unning.Queries.WA">

<ConnectionString>$Target/Property[Type="SQL2014Disc!Microsoft.SQL
Server.2014.DBEngine"]/ConnectionString$</ConnectionString>

<TCPPort>$Target/Property[Type="SQL2014Disc!Microsoft.SQLServer.20
14.DBEngine"]/TcpPort$</TCPPort>
        <TopN>5</TopN>
        <SQLQueryTimeoutSeconds>300</SQLQueryTimeoutSeconds>
        <TimeoutSeconds>600</TimeoutSeconds>
        </WriteAction>
    </Task>
</Tasks>
</Monitoring>
<LanguagePacks>
    <LanguagePack ID="ENU" IsDefault="true">
        <DisplayStrings>
            <DisplayString
ElementID="YourName.SquaredUp.VSAE.Workshop.Demo.SQL2014.DB.Engine
.Get.Long.Running.Queries.Task">
                <Name>Get DB Engine Long Running Queries -
YourName</Name>
            </DisplayString>
        </DisplayStrings>
    </LanguagePack>
</LanguagePacks>

```

4. Replace 'YourName' with your name in the code snippet above.

5.6 Build and Import the Management Pack

Now that we have created the agent tasks, we can build the management pack and import it into the OpsMgr management group for test. Please follow the steps described previously, build the project and import the sealed MP into the management group.

6 Test Agent Tasks

We can now test the agent tasks we have just created. Please follow the following steps to run the 'Get DB Engine Long Running Queries' tasks we have just created:

1. Launch SCOM console and go to the '**Monitoring**' pane.
2. Browse to the Database Engines state view
 - For Pre SQL 2014, go to '**Microsoft SQL Server\SQL Server Database Engines\SQL Server 2005/2008/2012\Database Engines\Database Engines**'
 - For SQL 2014, go to '**Microsoft SQL Server\SQL Server Database Engines\SQL Server 2014\Database Engines\Database Engines**'
3. Click on a Database Engine instance, and then click on the '**Get DB Engine Long Running Queries - <YourName>**' agent task

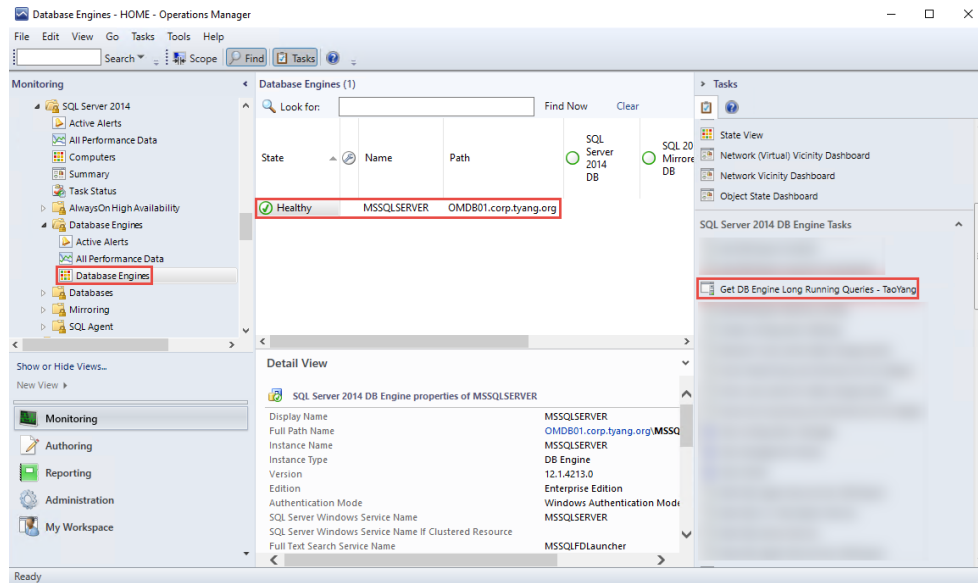


FIGURE 6.1 LOCATE THE AGENT TASK IN SCOM CONSOLE

4. On the Run Task window, click on 'Run'

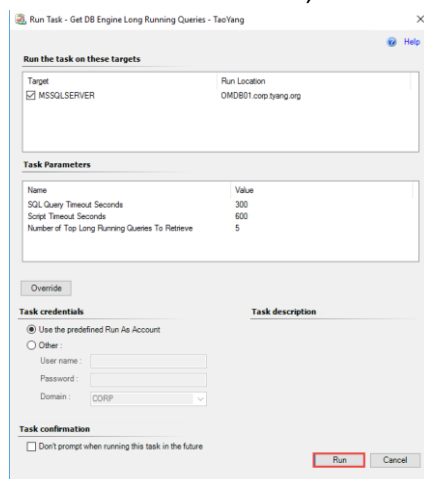


FIGURE 6.2 EXECUTING THE AGENT TASK

Note: As you can see, this task contains 3 overrideable parameters. These parameters were defined in the custom Write Action module that we have created earlier. You may click on **'Override'** and supply different values before clicking on 'Run'. i.e. instead of returning top 5 long running queries by default, you can enter an override value of 10 for the **'Number of Top Long Running Queries To Retrieve'** parameter and the task will retrieve the top 10 long running queries instead of 5.

5. Wait until the task execution finishes. The result will be displayed in the output. You may need to scroll down to examine the entire output.

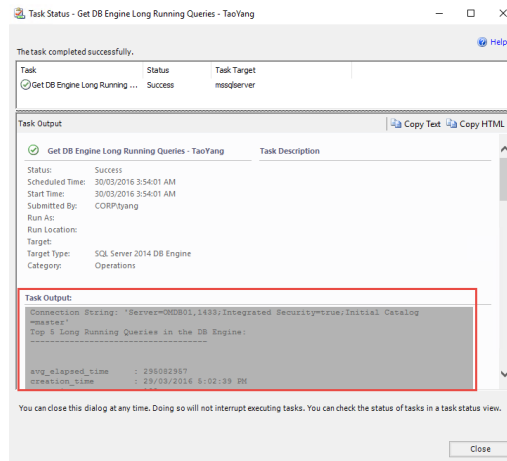


FIGURE 6.3 AGENT TASK EXECUTION RESULT

- The task execution is also logged within SCOM. You can find the past agent task executions under the **'Task Status'** view.

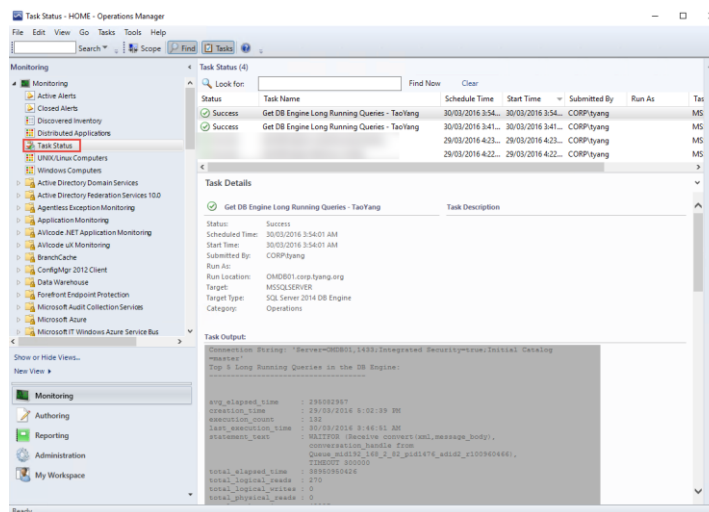


FIGURE 6.4 TASK STAUS - AGENT TASK EXECUTION HISTORY

7 Summary

In this workshop, we firstly demonstrated how to configure your authoring computer with all required and recommended software. We then walked through the process of creating a sealed SCOM 2012 management pack that contains a simple task created using the VSAE agent task template, as well as two more complex agent tasks that can be manually triggered against SQL 2005/2008/2012 and SQL 2014 database engines.

We hope you find this guide useful, and we welcome your feedback.