Microsoft System Center Operations Manager

# Configuration Manager Service Window Maintenance Mode Automation Management Pack

**Version:** 1.0.1.34

**Developed by:** Jason Daggett

Microsoft

Sr. Premier Field Engineer

Without the help, expert advice, insight and technical assistance of these fine individuals, this Management Pack could not have been made possible. Thanks to everyone who assisted & supported these efforts to bring this MP to a reality!

Tom Riddle – System Center Operations Manager & PowerShell Specialist

Geoffrey Guynn – Microsoft Premier Field Engineer

Kevin Holman – Microsoft Principal Premier Field Engineer

James Harper – Microsoft Senior Consultant

Contents

[Configuration Manager Service Window Maintenance Mode Automation Management Pack 1](#_Toc512267904)

[Management Pack Description 3](#_Toc512267905)

[Management Pack Requirements 3](#_Toc512267906)

[Products 3](#_Toc512267907)

[Referenced Management Packs 3](#_Toc512267908)

[Microsoft.SystemCenter.Library 3](#_Toc512267909)

[Microsoft.SystemCenter.Visualization.Library 3](#_Toc512267910)

[Microsoft.Windows.Library 3](#_Toc512267911)

[System.Health.Library 3](#_Toc512267912)

[System.Library 4](#_Toc512267913)

[Management Pack Contents 4](#_Toc512267914)

[Files Included in the Management Pack 4](#_Toc512267915)

[Management Pack Change Log 4](#_Toc512267916)

[Disabling the Rule Overview 5](#_Toc512267917)

[Best Practices 5](#_Toc512267918)

[For Testing Purposes or Phased Rollout approach 5](#_Toc512267919)

[Technical Details 6](#_Toc512267920)

[Data Source Module Types 6](#_Toc512267921)

[Microsoft System Center Configuration Manager Maintenance Mode PowerShell Data Source Module 6](#_Toc512267922)

[Rules 6](#_Toc512267923)

[SCOM Agent Initiated Maintenance Mode Rule for Active SCCM Service Windows 6](#_Toc512267924)

# Management Pack Description

This management pack was designed with two parts. These parts, contained in a single Rule and one Data Source Method, were broken up to provide System Center Operations Manager with the ability of automatically placing a system into Maintenance Mode using System Center Configuration Manager Service Windows.

These two parts enable the ability to automatically trigger the Operations Manager Maintenance Mode function based upon the Configuration Manager Service Windows found in the Configuration Manager Agent’s WMI stack. The scheduled “***SCOM Agent Initiated Maintenance Mode Rule for Active SCCM Service Windows***” rule, accounts for Service Window overlap, calculates the total service window duration and sends property bag items from the Operations Manager Agent directly to the Management Server that the agent is connected to.

# Management Pack Requirements

## Products

System Center Operations Manager

System Center Configuration Manager 2012, 2012 R2 & Current Branch

PowerShell 2.0+

## Referenced Management Packs

### Microsoft.SystemCenter.Library

* **Alias**: SC
* **Version**: 7.0.8433.0

### Microsoft.SystemCenter.Visualization.Library

* **Alias**: Visualization
* **Version**: 7.1.10226.0

### Microsoft.Windows.Library

* **Alias**: Windows
* **Version**: 7.5.8501.0

### System.Health.Library

* **Alias**: Health
* **Version**: 7.0.8433.0

### System.Library

* **Alias**: System
* **Version**: 7.5.8501.0

# Management Pack Contents

## Files Included in the Management Pack

*\*\* Both files are provided for convenience. Only choose to import the sealed (.mp) or unsealed (.xml) Management Pack.*

ConfigurationManager.ServiceWindow.MaintenanceMode.Automation.mp

* Sealed MP contains one Data Source Type and one Rule
  + Rule is enabled by default
    - See “*Rules*” section for information about rules, their use, and when to disable them

ConfigurationManager.ServiceWindow.MaintenanceMode.Automation.xml

* Unsealed MP contains one Data Source Type and one Rules
  + Rule is enabled by default
    - See “*Rules*” section for information about rules, their use, and when to disable them

Configuration.Manager.Service.Window.Maintenance.Mode.Automation.Overrides.xml

* Unsealed MP contains one Group and one rule Override

# Management Pack Change Log

1.0.1.29 – Resolved EventID 22409 warning being run every hour on Management Server. Fixed EventLog entries to display information in a more readable format.

1.0.1.26 – Added logic to write out error on Agent side if Duration variable is null, and added logic to if statement for oBag handling to not run if Duration is null.

1.0.1.24 – Included an Override Management Pack that contains a Group that has all systems in it and disables the rule for all the systems in the group.

1.0.1.35 – Revised PowerShell Script (SetSCOMMaintenanceModeViaSCCMServiceWindow.ps1) to resolve issues with Data Packet Errors on Management Server using overrides.

1.0.1.22 – Enabled Rule by default, automatically excludes Management Servers from Service Windows Maintenance Mode Automation

1.0.1.16 – Added Error Handling for WMI checks and modified Eventlog to include WhoAmI & Script Runtime.

1.0.1.6 – Initial Release

# Disabling the Rule Overview

## Best Practices

To start using this Management Pack a few tasks need to be performed before automation function. By default the Rule in this Management Pack is enabled for all systems with a SCOM Agent installed and reporting to the Management Group. It is highly suggested to:

1. Create an Override Management Pack *\*\*If the Sealed Management Pack is being used*
2. Create a Group or use an existing group that:
   1. Excludes all systems you do not want to automatically put into Maintenance Mode during SCCM Service Windows.
3. Disable the “***SCOM Agent Initiated Maintenance Mode Rule for Active SCCM Service Windows***”
   1. Using an override on the created group to disable the rule

## For Testing Purposes or Phased Rollout approach

Included in the Management Pack download is an Override file that contains a group called “***Disabled CfgMgr Service Window SCOM Maintenance Mode Automation Group***.” This Group contains a wildcard expression to include all systems that is being monitored by SCOM. There is an override that disables the“***SCOM Agent Initiated Maintenance Mode Rule for Active SCCM Service Windows***” rule.

If you want to test the functionality of the rule:

1. Open the “***Disabled CfgMgr Service Window SCOM Maintenance Mode Automation Group”***
2. Click the "***Excluded Members***" tab
3. Click the "***Exclude objects***" button
4. Change the "***Search For:***" to "***Windows Computers***"
5. Type in the system you are looking for in the "***Filter by Part of name***" field
6. Add the selected item and click "**OK**".

***\*\* If you want to enable the Automation functionality for all systems, just remove the override.***

# Technical Details

## Data Source Module Types

### Microsoft System Center Configuration Manager Maintenance Mode PowerShell Data Source Module

Microsoft.SystemCenter.ConfigMgrAgent.Automation.MaintenanceMode.Powershell.DataSourceModule

* Provides Simple Scheduler functionality to the “**SCOM Agent Initiated Maintenance Mode Rule for Active SCCM Service Windows**” rule.
* Provides a Probe Action using the “**PowerShellProbe**” to write back information using a System Center Operations Manager Property Bag.

## Rules

### SCOM Agent Initiated Maintenance Mode Rule for Active SCCM Service Windows

ConfigurationManager.ServiceWindow.MaintenanceMode.Automation.AgentInitiated.Rule

* Dependent on “***Microsoft System Center Configuration Manager Maintenance Mode PowerShell Data Source Module***” to perform the schema and scheduling functions.
* Enabled by default
  + This rule detects System Center Configuration Manager 2012, 2012 R2 or Current Branch Agent Service Windows and places the System Center Operations Manager Agent into Maintenance Mode.
    1. Management Servers are excluded from Automation by default.
    2. If using the Sealed Management Pack, create an override management pack, and store the enable override in the newly created override management pack.

#### Workflow Details



##### PowerShell Data Source Script Name “SetSCOMMaintenanceModeViaSCCMServiceWindow.ps1”

param($ComputerName)

#region Script Global Variables

$ScriptName = "SetSCOMMaintenanceModeViaSCCMServiceWindow.ps1"

$ScriptStartTime = Get-Date

$CurrentTime = Get-Date

$WhoAmI = whoami

$Error.Clear()

$GetSCCMClientVersion = Get-WmiObject -NameSpace Root\CCM -Class Sms\_Client | Select-Object ClientVersion

$ServiceWindows = Get-WmiObject -namespace root\ccm\ClientSDK -class CCM\_ServiceWindow | Where-Object {$\_.Type -lt '6'} | Select-Object ID, StartTime, EndTime, Duration, Type

IF ($Error)

{

$momapi.LogScriptEvent($ScriptName,$EventID,2221,"$ComputerName `n FATAL ERROR: Unable to detect an active SCCM Client on system via WMI Root\CCM. `n Terminating script. `n Running Script as: $WhoAmI `n Error is: ($Error).")

EXIT

}

# This table allows you to map a specific SCOM Maintenance mode reason to the different service window types in SCCM.

# This hash table correlates data between 3 APIs.

# The WMI ServiceWindow.Type (1,2,3,4,5,6) [Integer]

# The String description of that Type ( Reference https://msdn.microsoft.com/en-us/library/jj155419.aspx )

# The Matching Maintenance Mode description used by SCOM. ( Reference https://docs.microsoft.com/en-us/powershell/module/operationsmanager/start-scommaintenancemode?view=systemcenter-ps-2016 )

#Retrieve strings using $ServiceWindowType.($ServiceWindow.Type).CMName or $ServiceWindowType.($ServiceWindow.Type).OMName

$ServiceWindowType = @{

1 = @{ CMName = "ALLPROGRAM\_SERVICEWINDOW";

OMName = "PlannedApplicationMaintenance" }

2 = @{ CMName = "PROGRAM\_SERVICEWINDOW";

OMName = "PlannedApplicationMaintenance" }

3 = @{ CMName = "REBOOTREQUIRED\_SERVICEWINDOW";

OMName = "PlannedOther" }

4 = @{ CMName = "SOFTWAREUPDATE\_SERVICEWINDOW";

OMName = "PlannedOperatingSystemReconfiguration" }

5 = @{ CMName = "OSD\_SERVICEWINDOW";

OMName = "PlannedOperatingSystemReconfiguration" }

6 = @{ CMName = "USER\_DEFINED\_SERVICE\_WINDOW";

OMName = "PlannedApplicationMaintenance" }

}

#endregion Script Global Variables

#region Define SCOM API and Property Bag to send data back to SCOM

$oAPI = new-object -comObject "MOM.ScriptAPI";

$oBag = $oAPI.CreatePropertyBag();

#endregion Define SCOM API and Property Bag to send data back to SCOM

#region Define Global Functions

#Function to Calculate Service Windows Status

#returns [bool]InMaintenance, [int]DurationMinutes (calculates totalduration if there is any overlap in service windows), [DateTime]Start, [DateTime]End

Function Get-ServiceWindowStatus {

param ( $ServiceWindows )

#Templating an empty return object. Properties are set before return is called.

$ReturnObject = New-Object PSCustomObject -Property @{

ServiceWindow = $Null

InMaintenance = $False

DurationMinutes = $Null

Start = $Null

End = $Null

}

#If no service windows are passed, try to get them

if ( -not $ServiceWindows ) {

$ServiceWindows = get-wmiobject -namespace root\ccm\ClientSDK -class CCM\_ServiceWindow -ErrorAction Stop | Where-Object {$\_.Type -lt 6} | Select-Object ID, StartTime, EndTime, Duration, Type

}

#The following select uses what we call "Calculated Properties".

#This select pulls all properties already attached (using \*) then attaches two more calculated properties.

#The full syntax is @{Name="PropName";Expression={Scriptblock that modifies an existing property} }

$ServiceWindows = $ServiceWindows | Select \*,

@{N="Start";E={ [System.Management.ManagementDateTimeConverter]::ToDateTime($\_.StartTime).ToUniversalTime()} },

@{N="End";E={ [System.Management.ManagementDateTimeConverter]::ToDateTime($\_.EndTime).ToUniversalTime()} }

#Select the next service window or the current one if endtime hasn't passed yet.

$ServiceWindow = $ServiceWindows |

Sort start |

Select -First 1

#If the current/next incoming service window has overlap, get the service window(s) that overlap it.

$SW\_Overlap = $ServiceWindows | ? { ( $\_.Start -ge $ServiceWindow.Start ) -and ( $\_.Start -le $ServiceWindow.End ) -and ( $\_.ID -ne $ServiceWindow.ID ) }

if ( $SW\_Overlap ) {

#This is a tricky addition, it takes each element from both arrays are creates a larger arrray with all elements.

#This was done just in case there is more than one overlapping service window.

$StartTime = @($ServiceWindow.Start) + @($sw\_overlap.start) | sort | Select -first 1

$EndTime = @($ServiceWindow.End) + @($SW\_Overlap.End) | sort | Select -Last 1

}

else {

$StartTime = $ServiceWindow.Start

$EndTime = $ServiceWindow.End

}

#Get current time to do some calculations

$CurrentTime = (Get-Date)

#Determine if we're currently in maintenance.

$ReturnObject.InMaintenance = [bool]( ($CurrentTime -ge $ServiceWindow.Start) -and ($CurrentTime -le $ServiceWindow.End) )

#If we're in maintenance, calculate duration based on time left between now and end of maintenance window.

if ( $ReturnObject.InMaintenance -eq $True ) {

[int]$DurationMinutes = [double]( New-TimeSpan -Start $CurrentTime -End $ServiceWindow.End | Select -ExpandProperty TotalMinutes ) + .5 #round up to the next whole minute

}

#If not in maintenance, calculate duration based on start and end of service window.

else {

[int]$DurationMinutes = [double]( New-TimeSpan -Start $ServiceWindow.Start -End $ServiceWindow.End | Select -ExpandProperty TotalMinutes ) + .5 #round up to the next whole minute

}

#Define remaining properties of the returnobject

$ReturnObject.ServiceWindow = $ServiceWindow

$ReturnObject.DurationMinutes = $DurationMinutes

$ReturnObject.Start = $StartTime

$ReturnObject.End = $EndTime

return $ReturnObject

}

#endregion Define Global Functions

#region Main Script Block

$CurrentStatus = Get-ServiceWindowStatus -ServiceWindows $ServiceWindows

# 1. Stop processing if SCCM client isn't version 5 or later.

if ( $GetSCCMClientVersion.ClientVersion -lt '5.' ) {

#Log a warning in SCOM

$oAPI.LogScriptEvent($ScriptName,2222,2,"`n SCCM Service Windows script determined SCCM client is lower than version 5 on $ComputerName. `n Running Script as: $WhoAmI")

return

}

# 2. Stop processing if the system has no service windows.

if ( -not $ServiceWindows ) {

#Log a warning in SCOM

$oAPI.LogScriptEvent($ScriptName,2222,2,"`n SCCM Service Windows script found no defined service windows on $ComputerName. `n Running Script as: $WhoAmI")

return

}

# 3. Check if the duration has a value

if ($CurrentStatus.DurationMinutes -eq $null){

#Log a warning in SCOM

$oAPI.LogScriptEvent($ScriptName,2222,2,"`n SCCM Service Windows script found no defined service window duration on $ComputerName. `n Running Script as: $WhoAmI")

return

}

# 4. Check if the system is in an SCCM service window and set SCOM Maintenance mode if it is.

if ( $CurrentStatus.InMaintenance -and $CurrentStatus.ServiceWindow.Type -le '5' -and $CurrentStatus.DurationMinutes -ne $null)

{

$StartMMTime = $CurrentStatus.Start

$EndMMTime = $CurrentStatus.End

$MMDuration = $CurrentStatus.DurationMinutes

$Reason = $($ServiceWindowType.([INT]$CurrentStatus.ServiceWindow.Type).OMName )

$Comment = "The SCCM Service Window Started for $Reason on $ComputerName at $StartMMTime and will end at $EndMMTime for at total of $MMDuration minutes. This Maintenance Mode Request was processed by $ScriptName on $ComputerName from the 'Configuration Manager Service Window Maintenance Mode Automation' management pack."

$ScriptEndTime = Get-Date

$ScriptTime = ($ScriptEndTime - $ScriptStartTime).TotalSeconds

$oBag.AddValue("Duration",$CurrentStatus.DurationMinutes.ToString());

$oBag.AddValue("Reason", $Reason);

$oBag.AddValue("Comment",$Comment );

$oBag.AddValue("ComputerName",$ComputerName);

$oAPI.LogScriptEvent($ScriptName, 2222, 0, "$Comment `n Running Script as: $WhoAmI `n Script Completed in seconds: $ScriptTime")

$oBag

return

}

Else

{

$StartMMTime = $CurrentStatus.Start

$EndMMTime = $CurrentStatus.End

$MMDuration = $CurrentStatus.DurationMinutes

$ScriptEndTime = Get-Date

$ScriptTime = ($ScriptEndTime - $ScriptStartTime).TotalSeconds

$oAPI.LogScriptEvent($ScriptName,2224,0,"Computer: $ComputerName `n SCCM Service Windows script returned no active service windows. The next SCCM service window will start at: `n `n Start Time: $StartMMTime `n End Time: $EndMMTime `n Duration: $MMDuration `n `n Running Script as: $WhoAmI `n Script Completed in seconds: $ScriptTime")

return

}

#endregion Main Script Block

###### PowerShell Data Source Script Property Bag

Script gathers “Duration, Reason, Comment & ComputerName” and provides them to the PowerShell Write Action Script “StartMaintenanceMode.ps1” that will execute on the Operations Manager Management Server.

$oBag.AddValue("Duration","");

$oBag.AddValue("Reason", "");

$oBag.AddValue("Comment", "");

$oBag.AddValue("ComputerName","");

#### Workflow Details



##### PowerShell Write Action Script Name “StartSCCMServiceWindowMaintenaceMode.ps1”

param($DurationStr, $Reason, $Comment, $ComputerName)

if([string]::IsNullOrEmpty($DurationStr) -or [string]::IsNullOrEmpty($Reason) -or [string]::IsNullOrEmpty($Comment) -or [string]::IsNullOrEmpty($ComputerName))

{

return;

}

try

{

$Instance = Get-SCOMClassInstance -Name $ComputerName;

}

catch

{

$OpsMgrInstalledLoc=(Get-ItemProperty "hklm:\SOFTWARE\Microsoft\Microsoft Operations Manager\3.0\Setup").InstallDirectory|Split-Path;

$OpsMgrMdlLoc="$OpsMgrInstalledLoc\Powershell\OperationsManager";

Import-Module $OpsMgrMdlLoc;

$Instance = Get-SCOMClassInstance -Name $ComputerName;

}

if($Instance -ne $null)

{

$api = new-object -comObject 'MOM.ScriptAPI'

$CurTime=Get-Date;

[double]$Duration = $DurationStr;

if(5.0001 -gt $Duration -and 5.0 -le $Duration)

{

$Duration = 7.0001;

}

$EndTime = ($CurTime.AddMinutes($Duration));

Start-SCOMMaintenanceMode -Instance $Instance -EndTime $EndTime -Reason $Reason -Comment $Comment -ErrorAction SilentlyContinue;

if($? -eq $True)

{

$api.LogScriptEvent('StartSCCMServiceWindowMaintenanceMode.ps1',2223,0,$computerName+":"+$Duration+"|"+$Reason+"|"+$Comment+"|"+$EndTime);

}

}