# SCCM Granular Security

## SCCM Console Security

Your company has two (or more) different groups that utilize SCCM. To keep these groups separate and to ensure that one group can not affect another groups resources the following is needed.

Initial settings:

* The SCCM Central Site has two local users that are used as templates in SCCM Security. The two local users are disabled.
	+ User Name = SCCM-Template-WorkstationGroup
	+ User Name = SCCM-Template-ServerGroup
* Two collections have been created and dynamically populated with the appropriate resources.
	+ WorkstationGroup
		- Contains only workstations
	+ ServerGroup
		- Contains only servers
* Two folders have been created in the Packages node
	+ WorkstationGroup – Packages
	+ ServerGroup – Packages
* Two folders have been created in the Advertisements node
	+ WorkstationGroup – Advertisements
	+ ServerGroup – Advertisements
* You granted the appropriate security rights in the SCCM Console for both users.

Unfortunately the above method doesn’t allow for inherited permissions.

Example: ‘User A’ is a member of the ‘SCCM.WorkstationGroup’ group. The ‘SCCM.WorkstationGroup’ group has rights to a Collection named ‘WorkstationGroup’. If ‘User A’ creates a subcollection, the other members of the ‘SCCM.WorkstationGroup’ group will not have rights to that new subcollection, only UserA will.

That is where these scripts come in. The PowerShell scripts need to be run once manually to apply the correct permissions to already created objects. After that, the SQL Triggers and SQL Agent Job will take care of the newly created objects.

The example below uses a SCCM Site Code of ABC. Replace with your site code

The example below uses two AD Groups named ‘SCCM.WorkstationGroup’ and ‘SCCM.ServerGroup’. Replace with the group you want to have access to the secured objects.

The example below uses an Active Directory domain named ‘ADDomain’. Replace with your AD Domain name.

* Local User SCCM-Template-WorkstationGroup has been added to the SCCM Security Rights node with the following permissions
	+ Permissions
		- Advertisement – All Instances
			* Read, Create, Delegate, Manage Folders
		- Applicable Updates Summary – All Instances
			* Read, Modify, Create, Delegate, Manage Folders
		- Asset Intelligence – All Instances
			* Read, Delegate, Manage Asset Intelligence, View Asset Intelligence
		- Boot Image Package – All Instances
			* Read, Modify, Distribute, Create, Delegate, Manage Folders
		- Collection – [CollectionID of the top level collection – WorkstationGroup]
			* Read, Modify, Use Remote Tools, Advertise, Modify Resource, View Collected Files, Read Resource, Modify Collection Setting, Mange Management Controllers, View Management Controllers
		- Collection – All Instances
			* Create, Delegate
		- Computer Association – All Instances
			* Read, Delete, Create, Delegate, Manage Folders, Recover User State
		- Configuration Items – All Instances
			* Read, Modify, Distribute, Create, Delegate, Manage Folders, Network Access
		- Deployment – All Instances
			* Read, Create
		- Deployment Package – All Instances
			* Read, Modify, Distribute, Create, Manage Folders
		- Deployment Template – All Instances
			* Read, Modify, Create
		- Device Driver – All Instances
			* Read, Modify, Delete, Create, Delegate, Manage Folders
		- Device Setting Item – All Instances
			* Read, Modify, Delete, Create, Delegate
		- Device Setting Package – All Instances
			* Read, Modify, Delete, Distribute, Create, Delegate, Manage Folders
		- Driver Package – All Instances
			* Read, Modify, Delete, Distribute, Create, Delegate, Manage Folders
		- OS Image – All Instances
			* Read, Modify, Delete, Distribute, Create, Delegate, Manage Folders
		- OS Install Package – All Instances
			* Read, Modify, Delete, Distribute, Create, Delegate, Manage Folders
		- Package – All Instances
			* Read, Create, Delegate, Manage Folders
		- Query – All Instances
			* Read, Create, Manage Folders
		- Report – All Instances
			* Read, Create, Mange Folders
		- Site – All Instances
			* Read, Import Computer Entry, Manage OSD and ISV Proxy Certificates
		- Software Metering Rule – All Instances
			* Read, Create, Manage Folders
		- Status Message – All Instance
			* Read
		- Task Sequence Package – All Instances
			* Read, Modify, Delete, Administer, Create, Delegate, Manage Folders, Create Task Sequence Media
* Local User SCCM-Template-ServerGroup has been added to the SCCM Security Rights node with the following permissions
	+ Permissions
		- Advertisement – All Instances
			* Read, Create, Delegate, Manage Folders
		- Applicable Updates Summary – All Instances
			* Read, Modify, Create, Delegate, Manage Folders
		- Asset Intelligence – All Instances
			* Read, Delegate, Manage Asset Intelligence, View Asset Intelligence
		- Boot Image Package – All Instances
			* Read, Modify, Distribute, Create, Delegate, Manage Folders
		- Collection – [CollectionID of the top level collection – ServerGroup]
			* Read, Modify, Use Remote Tools, Advertise, Modify Resource, View Collected Files, Read Resource, Modify Collection Setting, Mange Management Controllers, View Management Controllers
		- Collection – All Instances
			* Create, Delegate
		- Computer Association – All Instances
			* Read, Delete, Create, Delegate, Manage Folders, Recover User State
		- Configuration Items – All Instances
			* Read, Modify, Distribute, Create, Delegate, Manage Folders, Network Access
		- Deployment – All Instances
			* Read, Create
		- Deployment Package – All Instances
			* Read, Modify, Distribute, Create, Manage Folders
		- Deployment Template – All Instances
			* Read, Modify, Create
		- Device Driver – All Instances
			* Read, Modify, Delete, Create, Delegate, Manage Folders
		- Device Setting Item – All Instances
			* Read, Modify, Delete, Create, Delegate
		- Device Setting Package – All Instances
			* Read, Modify, Delete, Distribute, Create, Delegate, Manage Folders
		- Driver Package – All Instances
			* Read, Modify, Delete, Distribute, Create, Delegate, Manage Folders
		- OS Image – All Instances
			* Read, Modify, Delete, Distribute, Create, Delegate, Manage Folders
		- OS Install Package – All Instances
			* Read, Modify, Delete, Distribute, Create, Delegate, Manage Folders
		- Package – All Instances
			* Read, Create, Delegate, Manage Folders
		- Query – All Instances
			* Read, Create, Manage Folders
		- Report – All Instances
			* Read, Create, Mange Folders
		- Site – All Instances
			* Read, Import Computer Entry, Manage OSD and ISV Proxy Certificates
		- Software Metering Rule – All Instances
			* Read, Create, Manage Folders
		- Status Message – All Instance
			* Read
		- Task Sequence Package – All Instances
			* Read, Modify, Delete, Administer, Create, Delegate, Manage Folders, Create Task Sequence Media
* Two active directory groups were created for the two teams
	+ ADDomain\ SCCM.WorkstationGroup
	+ ADDomain\ SCCM.ServerGroup
	+ These two groups were added to the SCCM Security node and the permissions from the respective template were copied to them.

# SCCM Advanced Granular Security

## SCCM Advanced Console Security’

Unfortunately the above method doesn’t allow for inherited permissions.

Example: ‘User A’ is a member of the ‘SCCM.WorkstationGroup’ group. The ‘SCCM.WorkstationGroup’ group has rights to a Collection named ‘WorkstationGroup’. If ‘User A’ creates a subcollection, the other members of the ‘SCCM.WorkstationGroup’ group will not have rights to that new subcollection, only UserA will.

That is where these scripts come in. The PowerShell scripts need to be run once manually to apply the correct permissions to already created objects. After that, the SQL Triggers and SQL Agent Job will take care of the newly created objects.

The example below uses a SCCM Site Code of ABC. Replace with your site code

The example below uses two AD Groups named ‘SCCM.WorkstationGroup’ and ‘SCCM.ServerGroup’. Replace with the group you want to have access to the secured objects.

The example below uses an Active Directory domain named ‘ADDomain’. Replace with your AD Domain name.

### SCCM Collection Security PowerShell script

SCCM-Security-Collection.ps1

# Find the first level collections of the top level collection that is to be secured

$subcollections = **Get-WmiObject** *-namespace* root\SMS\site\_ABC*-Query* "select \* from SMS\_Collection as coll join SMS\_CollectToSubCollect as assoc on coll.CollectionID=assoc.subCollectionID where assoc.parentCollectionID='**[The top level collection you want to start with. Place the CollectionID here without brackets. In the above notes, this would be the WorkstationGroup collection]**'"

# Place all of the subcollection ID's into the variable $collections

$collections = foreach ($i in $subcollections)

 {

 $i.coll.collectionid

 }

# place the collectionID's into a placeholder

$placeholder += $collections

# start while loop to look for second, third, forth, etc. collections

$AreThereMoreCollections = "filler", "for while loop" # Need to have the $aretheremorecollections variable populated with two elements for some reason.

while ($AreThereMoreCollections.count -ge 1)

 {

 # Find the second level collections of the first level collections from the top level collection

 $subcollections = foreach ($i in $collections)

 {

 **get-wmiobject** *-Namespace* root\SMS\Site\_ABC *-Query* "select \* from SMS\_Collection as coll join SMS\_CollectToSubCollect as assoc on coll.CollectionID=assoc.subCollectionID where assoc.parentCollectionID='$i'"

 }

 # Place all of the subcollection ID's into the variable $collections

 $collections = foreach ($i in $subcollections)

 {

 $i.coll.collectionid

 }

 # place the collectionID's into a placeholder

 $placeholder += $collections

 $AreThereMoreCollections = foreach ($i in $subcollections)

 {

 $i.coll.name

 }

 }

foreach ($i in $placeholder)

 {

 if ($i -ne $null) # Need to ensure we are working with an object!

 {

 # Looking to see if the object is already secured with the correct permissions.

 $SecureObject = **get-wmiobject** *-Namespace* root\SMS\site\_ABC*-Query* "Select username From SMS\_UserInstancePermissionNames WHERE InstanceKey='$i'"

 foreach ($item in $SecureObject)

 {

 $AlreadySet = $null

 if ($item.username -eq "ADDOMAIN\SCCM.WorkstationGroup")

 {

 $AlreadySet = $true

 break

 }

 }

 If ($AlreadySet -ne $true)

 {

 #$CollectionSecurity = foreach ($i in $subcollections) {get-wmiobject -Namespace root\SMS\site\_ABC-Query "Select \* From SMS\_UserInstancePermissionNames WHERE ObjectKey=1 AND InstanceKey='$i.coll.name'"}

 $object = ([wmiclass]"\\.\root\SMS\site\_ABC:SMS\_UserInstancePermissions").CreateInstance()

 $object.username = "ADDOMAIN\ SCCM.WorkstationGroup "

 #ObjectKey Value

 #--------- ------

 #1 collections

 #2 Packages

 #3 advertisements

 $object.objectkey = 1

 $object.InstanceKey = $i

 #permission permissionname

 #---------- --------------

 #1 Read

 #2 Modify

 #32 Remote Control

 #64 Advertise

 #128 Modify Resource

 #2048 View Collected File

 #4096 Read Resource

 #2097152 Modify Collection Setting

 #16777216 Manage BMC

 #33554432 View BMC

 $object.InstancePermissions = 1+2+3+32+64+128+2048+4096+2097152+16777216+33554432

 $object.put()

 }

 }

 }

**Remove-Variable** Collections, i, placeholder, AreThereMoreCollections, subcollections, item, alreadyset *-ErrorAction* SilentlyContinue

# Find the first level collections of the top level collection that is to be secured

$subcollections = **Get-WmiObject** *-namespace* root\SMS\site\_ABC*-Query* "select \* from SMS\_Collection as coll join SMS\_CollectToSubCollect as assoc on coll.CollectionID=assoc.subCollectionID where assoc.parentCollectionID='**[The top level collection you want to start with. Place the CollectionID here without brackets. In the above notes, this would be the ServerGroup collection]**'"

# Place all of the subcollection ID's into the variable $collections

$collections = foreach ($i in $subcollections)

 {

 $i.coll.collectionid

 }

# place the collectionID's into a placeholder

$placeholder += $collections

# start while loop to look for second, third, forth, etc. collections

$AreThereMoreCollections = "filler", "for while loop" # Need to have the $aretheremorecollections variable populated with two elements for some reason.

while ($AreThereMoreCollections.count -ge 1)

 {

 # Find the second level collections of the first level collections from the top level collection

 $subcollections = foreach ($i in $collections)

 {

 get-wmiobject -Namespace root\SMS\Site\_ABC-Query "select \* from SMS\_Collection as coll join SMS\_CollectToSubCollect as assoc on coll.CollectionID=assoc.subCollectionID where assoc.parentCollectionID='$i'"

 }

 $collections = foreach ($i in $subcollections)

 {

 $i.coll.collectionid

 }

 # place the collectionID into a placeholder

 $placeholder += $collections

 $AreThereMoreCollections = foreach ($i in $subcollections)

 {

 $i.coll.name

 }

 }

$placeholder = $placeholder | sort

foreach ($i in $placeholder)

 {

 if ($i -ne $null) # Need to ensure we are working with an object!

 {

 # Looking to see if the object is already secured with the correct permissions.

 $SecureObject = **get-wmiobject** *-Namespace* root\SMS\site\_ABC*-Query* "Select username From SMS\_UserInstancePermissionNames WHERE InstanceKey='$i'"

 foreach ($item in $SecureObject)

 {

 $AlreadySet = $null

 if ($item.username -eq "ADDOMAIN\SCCM.ServerGroup")

 {

 $AlreadySet = $true

 break

 }

 }

 If ($AlreadySet -ne $true)

 {

 #$CollectionSecurity = foreach ($i in $subcollections) {get-wmiobject -Namespace root\SMS\site\_ABC-Query "Select \* From SMS\_UserInstancePermissionNames WHERE ObjectKey=1 AND InstanceKey='$i.coll.name'"}

 $object = ([wmiclass]"\\.\root\SMS\site\_ABC:SMS\_UserInstancePermissions").CreateInstance()

 $object.username = "ADDOMAIN\SCCM.ServerGroup"

 #ObjectKey Value

 #--------- ------

 #1 collections

 #2 Packages

 #3 advertisements

 $object.objectkey = 1

 $object.InstanceKey = $i

 #permission permissionname

 #---------- --------------

 #1 Read

 #2 Modify

 #32 Remote Control

 #64 Advertise

 #128 Modify Resource

 #2048 View Collected File

 #4096 Read Resource

 #2097152 Modify Collection Setting

 #16777216 Manage BMC

 #33554432 View BMC

 $object.InstancePermissions = 1+2+3+32+64+128+2048+4096+2097152+16777216+33554432

 $object.put()

 }

 }

 }

**Remove-Variable** Collections, i, placeholder, AreThereMoreCollections, subcollections, alreadyset, item *-ErrorAction* SilentlyContinue

### SCCM Collection SQL Server Agent Job – This is the SQL job that will kick off the script

USE [msdb]

GO

/\*\*\*\*\*\* Object: Job [SCCM-Security-Collections] Script Date: 05/26/2010 15:29:34 \*\*\*\*\*\*/

BEGIN TRANSACTION

DECLARE @ReturnCode INT

SELECT @ReturnCode = 0

/\*\*\*\*\*\* Object: JobCategory [[Uncategorized (Local)]]] Script Date: 05/26/2010 15:29:34 \*\*\*\*\*\*/

IF NOT EXISTS (SELECT name FROM msdb.dbo.syscategories WHERE name=N'[Uncategorized (Local)]' AND category\_class=1)

BEGIN

EXEC @ReturnCode = msdb.dbo.sp\_add\_category @class=N'JOB', @type=N'LOCAL', @name=N'[Uncategorized (Local)]'

IF (@@ERROR <> 0 OR @ReturnCode <> 0) GOTO QuitWithRollback

END

DECLARE @jobId BINARY(16)

EXEC @ReturnCode = msdb.dbo.sp\_add\_job @job\_name=N'SCCM-Security-Collections',

 @enabled=1,

 @notify\_level\_eventlog=0,

 @notify\_level\_email=0,

 @notify\_level\_netsend=0,

 @notify\_level\_page=0,

 @delete\_level=0,

 @description=N'No description available.',

 @category\_name=N'[Uncategorized (Local)]',

 @owner\_login\_name=N'ADDOMAIN\UserID', @job\_id = @jobId OUTPUT

IF (@@ERROR <> 0 OR @ReturnCode <> 0) GOTO QuitWithRollback

/\*\*\*\*\*\* Object: Step [Set SCCM Security] Script Date: 05/26/2010 15:29:36 \*\*\*\*\*\*/

EXEC @ReturnCode = msdb.dbo.sp\_add\_jobstep @job\_id=@jobId, @step\_name=N'Set SCCM Security',

 @step\_id=1,

 @cmdexec\_success\_code=0,

 @on\_success\_action=1,

 @on\_success\_step\_id=0,

 @on\_fail\_action=2,

 @on\_fail\_step\_id=0,

 @retry\_attempts=0,

 @retry\_interval=0,

 @os\_run\_priority=0, @subsystem=N'CmdExec',

 @command=N'C:\WINDOWS\system32\WINDOW~2\v1.0\powershell.exe -command "& ''I:\SCCM Scripts\SCCM Powershell Production Scripts\SCCM-Security-Collection.ps1''"',

 @flags=0,

 @proxy\_name=N'SQL\_Proxy\_ID'

IF (@@ERROR <> 0 OR @ReturnCode <> 0) GOTO QuitWithRollback

EXEC @ReturnCode = msdb.dbo.sp\_update\_job @job\_id = @jobId, @start\_step\_id = 1

IF (@@ERROR <> 0 OR @ReturnCode <> 0) GOTO QuitWithRollback

EXEC @ReturnCode = msdb.dbo.sp\_add\_jobserver @job\_id = @jobId, @server\_name = N'(local)'

IF (@@ERROR <> 0 OR @ReturnCode <> 0) GOTO QuitWithRollback

COMMIT TRANSACTION

GOTO EndSave

QuitWithRollback:

 IF (@@TRANCOUNT > 0) ROLLBACK TRANSACTION

EndSave:

### SCCM Collection SQL Trigger – This is needed to kick of the script whenever a new collection is created

USE [SMS\_ABC]

GO

/\*\*\*\*\*\* Object: DdlTrigger [SCCM-Security-Collections] Script Date: 05/26/2010 15:34:21 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

create trigger [SCCM-Security-Collections]

on database

for create\_view

as EXEC msdb.dbo.sp\_start\_job N'SCCM-Security-Collections'

GO

SET ANSI\_NULLS OFF

GO

SET QUOTED\_IDENTIFIER OFF

GO

ENABLE TRIGGER [SCCM-Security-Collections] ON DATABASE

### SCCM Advertisement Security - PowerShell Script

SCCM-Security-Advertisement.ps1

# SCCM Security - Advertisements

# This will display all advertisements in the root of the desired top level Advertisement folder

# To find the ContainerNodeID run this: Get-WmiObject -Namespace root\SMS\site\_ABC-Query "Select name, containernodeid from SMS\_ObjectContainerNode" |sort name | ft name, containernodeid

$Advertisements = **Get-WmiObject** *-Namespace* root\SMS\site\_ABC*-Query* "Select instancekey from SMS\_ObjectContainerItem where containernodeID = '[Container Node ID of the WorkstationGroup Advertisement Folder without the brackets]'" | **select-object** InstanceKey

# $placeholder will hold all of the Advertisement Instance keys under the top level folder. ie: recursive

$placeholder += $Advertisements

# Look for other folders under the top level folder

# Find the second level nodes of the first level nodes from the top level node

$NodeIDs = **Get-WmiObject** *-Namespace* root\SMS\site\_ABC*-Query* "Select ContainerNodeID from SMS\_ObjectContainerNode where ParentContainerNodeID = '[Container Node ID of the WorkstationGroup Advertisement Folder without the brackets]'"

foreach ($i in $NodeIDs)

 {

 $ContainerNodeID = $i.ContainerNodeID

 $advertisements = **Get-WmiObject** *-Namespace* root\SMS\site\_ABC*-Query* "Select instancekey from SMS\_ObjectContainerItem where containernodeID = '$ContainerNodeID'" | **select-object** InstanceKey

 $placeholder += $Advertisements

 $NodeIDs = **Get-WmiObject** *-Namespace* root\SMS\site\_ABC*-Query* "Select ContainerNodeID from SMS\_ObjectContainerNode where ParentContainerNodeID = '$ContainerNodeID'"

 # start while loop to look for second, third, forth, etc. Advertisements

 $AreThereMoreNodeIDs = (**Get-WmiObject** *-Namespace* root\SMS\site\_ABC*-Query* "Select ContainerNodeID from SMS\_ObjectContainerNode where ParentContainerNodeID = '$ContainerNodeID'").count

 while ($AreThereMoreNodeIDs -ne $null)

 {

 foreach ($i in $NodeIDs)

 {

 $ContainerNodeID = $i.ContainerNodeID

 $advertisements = **Get-WmiObject** *-Namespace* root\SMS\site\_ABC*-Query* "Select instancekey from SMS\_ObjectContainerItem where containernodeID = '$ContainerNodeID'" | **select-object** InstanceKey

 $placeholder += $Advertisements

 $NodeIDs = **Get-WmiObject** *-Namespace* root\SMS\site\_ABC*-Query* "Select ContainerNodeID from SMS\_ObjectContainerNode where ParentContainerNodeID = '$ContainerNodeID'"

 While ($NodeIDs -ne $null)

 {

 foreach ($i in $NodeIDs)

 {

 $ContainerNodeID = $i.ContainerNodeID

 $advertisements = **Get-WmiObject** *-Namespace* root\SMS\site\_ABC*-Query* "Select instancekey from SMS\_ObjectContainerItem where containernodeID = '$ContainerNodeID'" | **select-object** InstanceKey

 $placeholder += $Advertisements

 $NodeIDs = **Get-WmiObject** *-Namespace* root\SMS\site\_ABC*-Query* "Select ContainerNodeID from SMS\_ObjectContainerNode where ParentContainerNodeID = '$ContainerNodeID'"

 }

 }

 }

 IF ($NodeIDs -ne $null)

 {

 $AreThereMoreNodeIDs = "yes"

 }

 else

 {

 $AreThereMoreNodeIDs = $null

 }

 }

 }

# Set the permissions on all the Instance Keys in $Placeholder

$placeholder = $placeholder | **sort**

foreach ($i in $placeholder)

 {

 if ($i -ne $null)

 {

 #ObjectKey Value

 #--------- ------

 #1 collections

 #2 Packages

 #3 advertisements

 #

 # Sample query below

 #$CollectionSecurity = foreach ($i in $subcollections) {get-wmiobject -Namespace root\SMS\site\_ABC-Query "Select \* From SMS\_UserInstancePermissionNames WHERE ObjectKey=3 AND InstanceKey='$i'"}

 $InstanceKey = $i.InstanceKey

 $SecureObject = **get-wmiobject** *-Namespace* root\SMS\site\_ABC*-Query* "Select username From SMS\_UserInstancePermissionNames WHERE InstanceKey='$InstanceKey'"

 foreach ($i in $SecureObject)

 {

 $AlreadySet = $null

 if ($i.username -eq "ADDOMAIN\SCCM.WorkstationGroup")

 {

 $AlreadySet = $true

 break

 }

 }

 If ($AlreadySet -ne $true)

 {

 $object = ([wmiclass]"\\.\root\SMS\site\_ABC:SMS\_UserInstancePermissions").CreateInstance()

 $object.username = "ADDOMAIN\ SCCM.WorkstationGroup "

 $object.objectkey = 3

 $object.InstanceKey = $InstanceKey

 # permission permissionname

 # ---------- --------------

 # 1 Read

 # 2 Modify

 # 4 Delete

 $object.InstancePermissions = 1+2+4

 $object.put()

 }

 }

 }

**Remove-Variable** Advertisements, i, placeholder, NodeIDs, ContainerNodeID, InstanceKey, AreThereMoreNodeIDs, object, AlreadySet, SecureObject *-ErrorAction* SilentlyContinue

# This will display all advertisements in the root of the desired top level Advertisement folder

# To find the ContainerNodeID run this: Get-WmiObject -Namespace root\SMS\site\_ABC-Query "Select name, containernodeid from SMS\_ObjectContainerNode" |sort name | ft name, containernodeid

$Advertisements = **Get-WmiObject** *-Namespace* root\SMS\site\_ABC*-Query* "Select instancekey from SMS\_ObjectContainerItem where containernodeID = '[Container Node ID of the ServerGroup Advertisement Folder without the brackets]'" | **select-object** InstanceKey

# $placeholder will hold all of the Advertisement Instance keys under the top level folder. ie: recursive

$placeholder += $Advertisements

# Look for other folders under the top level folder

# Find the second level collections of the first level collections from the top level collection

$NodeIDs = **Get-WmiObject** *-Namespace* root\SMS\site\_ABC*-Query* "Select ContainerNodeID from SMS\_ObjectContainerNode where ParentContainerNodeID = '[Container Node ID of the ServerGroup Advertisement Folder without the brackets]'"

foreach ($i in $NodeIDs)

 {

 $ContainerNodeID = $i.ContainerNodeID

 $advertisements = **Get-WmiObject** *-Namespace* root\SMS\site\_ABC*-Query* "Select instancekey from SMS\_ObjectContainerItem where containernodeID = '$ContainerNodeID'" | **select-object** InstanceKey

 $placeholder += $Advertisements

 $NodeIDs = **Get-WmiObject** *-Namespace* root\SMS\site\_ABC*-Query* "Select ContainerNodeID from SMS\_ObjectContainerNode where ParentContainerNodeID = '$ContainerNodeID'"

 # start while loop to look for second, third, forth, etc. Advertisements

 $AreThereMoreNodeIDs = (**Get-WmiObject** *-Namespace* root\SMS\site\_ABC*-Query* "Select ContainerNodeID from SMS\_ObjectContainerNode where ParentContainerNodeID = '$ContainerNodeID'").count

 while ($AreThereMoreNodeIDs -ne $null)

 {

 foreach ($i in $NodeIDs)

 {

 $ContainerNodeID = $i.ContainerNodeID

 $advertisements = **Get-WmiObject** *-Namespace* root\SMS\site\_ABC*-Query* "Select instancekey from SMS\_ObjectContainerItem where containernodeID = '$ContainerNodeID'" | **select-object** InstanceKey

 $placeholder += $Advertisements

 $NodeIDs = **Get-WmiObject** *-Namespace* root\SMS\site\_ABC*-Query* "Select ContainerNodeID from SMS\_ObjectContainerNode where ParentContainerNodeID = '$ContainerNodeID'"

 While ($NodeIDs -ne $null)

 {

 foreach ($i in $NodeIDs)

 {

 $ContainerNodeID = $i.ContainerNodeID

 $advertisements = **Get-WmiObject** *-Namespace* root\SMS\site\_ABC*-Query* "Select instancekey from SMS\_ObjectContainerItem where containernodeID = '$ContainerNodeID'" | **select-object** InstanceKey

 $placeholder += $Advertisements

 $NodeIDs = **Get-WmiObject** *-Namespace* root\SMS\site\_ABC*-Query* "Select ContainerNodeID from SMS\_ObjectContainerNode where ParentContainerNodeID = '$ContainerNodeID'"

 }

 }

 }

 IF ($NodeIDs -ne $null)

 {

 $AreThereMoreNodeIDs = "yes"

 }

 else

 {

 $AreThereMoreNodeIDs = $null

 }

 }

 }

# Set the permissions on all the Instance Keys in $Placeholder

$placeholder = $placeholder | **sort**

foreach ($i in $placeholder)

 {

 if ($i -ne $null)

 {

 #ObjectKey Value

 #--------- ------

 #1 collections

 #2 Packages

 #3 advertisements

 #

 # Sample query below

 #$CollectionSecurity = foreach ($i in $subcollections) {get-wmiobject -Namespace root\SMS\site\_ABC-Query "Select \* From SMS\_UserInstancePermissionNames WHERE ObjectKey=3 AND InstanceKey='$i'"}

 $InstanceKey = $i.InstanceKey

 $SecureObject = **get-wmiobject** *-Namespace* root\SMS\site\_ABC*-Query* "Select username From SMS\_UserInstancePermissionNames WHERE InstanceKey='$InstanceKey'"

 foreach ($i in $SecureObject)

 {

 $AlreadySet = $null

 if ($i.username -eq "ADDOMAIN\SCCM.ServerGroup")

 {

 $AlreadySet = $true

 break

 }

 }

 If ($AlreadySet -ne $true)

 {

 $object = ([wmiclass]"\\.\root\SMS\site\_ABC:SMS\_UserInstancePermissions").CreateInstance()

 $object.username = "ADDOMAIN\SCCM.ServerGroup"

 $object.objectkey = 3

 $object.InstanceKey = $InstanceKey

 # permission permissionname

 # ---------- --------------

 # 1 Read

 # 2 Modify

 # 4 Delete

 $object.InstancePermissions = 1+2+4

 $object.put()

 }

 }

 }

**Remove-Variable** Advertisements, i, placeholder, NodeIDs, ContainerNodeID, InstanceKey, AreThereMoreNodeIDs, object, AlreadySet, SecureObject *-ErrorAction* SilentlyContinue

### SCCM Advertisement SQL Server Agent Job – This is the SQL job that will kick off the script

USE [msdb]

GO

/\*\*\*\*\*\* Object: Job [SCCM-Security-Advertisements] Script Date: 05/26/2010 15:35:23 \*\*\*\*\*\*/

BEGIN TRANSACTION

DECLARE @ReturnCode INT

SELECT @ReturnCode = 0

/\*\*\*\*\*\* Object: JobCategory [[Uncategorized (Local)]]] Script Date: 05/26/2010 15:35:23 \*\*\*\*\*\*/

IF NOT EXISTS (SELECT name FROM msdb.dbo.syscategories WHERE name=N'[Uncategorized (Local)]' AND category\_class=1)

BEGIN

EXEC @ReturnCode = msdb.dbo.sp\_add\_category @class=N'JOB', @type=N'LOCAL', @name=N'[Uncategorized (Local)]'

IF (@@ERROR <> 0 OR @ReturnCode <> 0) GOTO QuitWithRollback

END

DECLARE @jobId BINARY(16)

EXEC @ReturnCode = msdb.dbo.sp\_add\_job @job\_name=N'SCCM-Security-Advertisements',

 @enabled=1,

 @notify\_level\_eventlog=0,

 @notify\_level\_email=0,

 @notify\_level\_netsend=0,

 @notify\_level\_page=0,

 @delete\_level=0,

 @description=N'No description available.',

 @category\_name=N'[Uncategorized (Local)]',

 @owner\_login\_name=N'ADDOMAIN\UserID', @job\_id = @jobId OUTPUT

IF (@@ERROR <> 0 OR @ReturnCode <> 0) GOTO QuitWithRollback

/\*\*\*\*\*\* Object: Step [SCCM-Security-Advertisements] Script Date: 05/26/2010 15:35:23 \*\*\*\*\*\*/

EXEC @ReturnCode = msdb.dbo.sp\_add\_jobstep @job\_id=@jobId, @step\_name=N'SCCM-Security-Advertisements',

 @step\_id=1,

 @cmdexec\_success\_code=0,

 @on\_success\_action=1,

 @on\_success\_step\_id=0,

 @on\_fail\_action=2,

 @on\_fail\_step\_id=0,

 @retry\_attempts=0,

 @retry\_interval=0,

 @os\_run\_priority=0, @subsystem=N'CmdExec',

 @command=N'C:\WINDOWS\system32\WINDOW~2\v1.0\powershell.exe -command "& ''I:\SCCM Scripts\SCCM Powershell Production Scripts\SCCM-Security-Advertisement.ps1''"',

 @flags=0,

 @proxy\_name=N'SQL\_Proxy\_ID

IF (@@ERROR <> 0 OR @ReturnCode <> 0) GOTO QuitWithRollback

EXEC @ReturnCode = msdb.dbo.sp\_update\_job @job\_id = @jobId, @start\_step\_id = 1

IF (@@ERROR <> 0 OR @ReturnCode <> 0) GOTO QuitWithRollback

EXEC @ReturnCode = msdb.dbo.sp\_add\_jobserver @job\_id = @jobId, @server\_name = N'(local)'

IF (@@ERROR <> 0 OR @ReturnCode <> 0) GOTO QuitWithRollback

COMMIT TRANSACTION

GOTO EndSave

QuitWithRollback:

 IF (@@TRANCOUNT > 0) ROLLBACK TRANSACTION

EndSave:

### SCCM Advertisement SQL Trigger – This is needed to kick of the script whenever a new collection is created

USE [SMS\_CEN]

GO

/\*\*\*\*\*\* Object: Trigger [dbo].[SCCM-Security-Advertisements] Script Date: 05/26/2010 15:40:19 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

create trigger [dbo].[SCCM-Security-Advertisements]

on [dbo].[ProgramOffers]

after insert

as EXEC msdb.dbo.sp\_start\_job N'SCCM-Security-Advertisements'

### SCCM Package Security - PowerShell Script

SCCM-Security-Package.ps1

# SCCM Security - Packages

# This will display all packages in the root of the desired top level Packages folder

# To find the ContainerNodeID run this: Get-WmiObject -Namespace root\SMS\site\_ABC-Query "Select name, containernodeid from SMS\_ObjectContainerNode" |sort name | ft name, containernodeid

$Packages = **Get-WmiObject** *-Namespace* root\SMS\site\_ABC*-Query* "Select instancekey from SMS\_ObjectContainerItem where containernodeID = '[Container Node ID of the Workstation Package Folder without the brackets]'" | **select-object** InstanceKey

# $placeholder will hold all of the Packages Instance keys under the top level folder. ie: recursive

$placeholder += $Packages

# Look for other folders under the top level folder

# Find the second level nodes of the first level nodes from the top level node

$NodeIDs = **Get-WmiObject** *-Namespace* root\SMS\site\_ABC*-Query* "Select ContainerNodeID from SMS\_ObjectContainerNode where ParentContainerNodeID = '[Container Node ID of the Workstation Package Folder without the brackets]'"

if ($NodeIDs -ne $null)

 {

 foreach ($i in $NodeIDs)

 {

 $ContainerNodeID = $i.ContainerNodeID

 $Packages = **Get-WmiObject** *-Namespace* root\SMS\site\_ABC*-Query* "Select instancekey from SMS\_ObjectContainerItem where containernodeID = '$ContainerNodeID'" | **select-object** InstanceKey

 $placeholder += $Packages

 $NodeIDs = **Get-WmiObject** *-Namespace* root\SMS\site\_ABC*-Query* "Select ContainerNodeID from SMS\_ObjectContainerNode where ParentContainerNodeID = '$ContainerNodeID'"

 # start while loop to look for second, third, forth, etc. packages

 $AreThereMoreNodeIDs = (**Get-WmiObject** *-Namespace* root\SMS\site\_ABC*-Query* "Select ContainerNodeID from SMS\_ObjectContainerNode where ParentContainerNodeID = '$ContainerNodeID'").count

 while ($AreThereMoreNodeIDs -ne $null)

 {

 foreach ($i in $NodeIDs)

 {

 $ContainerNodeID = $i.ContainerNodeID

 $Packages = **Get-WmiObject** *-Namespace* root\SMS\site\_ABC*-Query* "Select instancekey from SMS\_ObjectContainerItem where containernodeID = '$ContainerNodeID'" | **select-object** InstanceKey

 $placeholder += $Packages

 $NodeIDs = **Get-WmiObject** *-Namespace* root\SMS\site\_ABC*-Query* "Select ContainerNodeID from SMS\_ObjectContainerNode where ParentContainerNodeID = '$ContainerNodeID'"

 While ($NodeIDs -ne $null)

 {

 foreach ($i in $NodeIDs)

 {

 $ContainerNodeID = $i.ContainerNodeID

 $Packages = **Get-WmiObject** *-Namespace* root\SMS\site\_ABC*-Query* "Select instancekey from SMS\_ObjectContainerItem where containernodeID = '$ContainerNodeID'" | **select-object** InstanceKey

 $placeholder += $Packages

 $NodeIDs = **Get-WmiObject** *-Namespace* root\SMS\site\_ABC*-Query* "Select ContainerNodeID from SMS\_ObjectContainerNode where ParentContainerNodeID = '$ContainerNodeID'"

 }

 }

 }

 IF ($NodeIDs -ne $null) {$AreThereMoreNodeIDs = "yes"} else {$AreThereMoreNodeIDs = $null}

 }

 }

 }

# Set the permissions on all the Instance Keys in $Placeholder

$placeholder = $placeholder | **sort**

foreach ($i in $placeholder)

 {

 if ($i -ne $null)

 {

 #ObjectKey Value

 #--------- ------

 #1 collections

 #2 Packages

 #3 advertisements

 #

 # Sample query below

 #$CollectionSecurity = foreach ($i in $subcollections) {get-wmiobject -Namespace root\SMS\site\_ABC-Query "Select \* From SMS\_UserInstancePermissionNames WHERE ObjectKey=3 AND InstanceKey='$i'"}

 $InstanceKey = $i.InstanceKey

 $SecureObject = **get-wmiobject** *-Namespace* root\SMS\site\_ABC*-Query* "Select username From SMS\_UserInstancePermissionNames WHERE InstanceKey='$InstanceKey'"

 foreach ($i in $SecureObject)

 {

 $AlreadySet = $null

 if ($i.username -eq "ADDOMAIN\SCCM.Test")

 {

 $AlreadySet = $true

 break

 }

 }

 If ($AlreadySet -ne $true)

 {

 $object = ([wmiclass]"\\.\root\SMS\site\_ABC:SMS\_UserInstancePermissions").CreateInstance()

 $object.username = "ADDOMAIN\SCCM.WorkstationGroup"

 $object.objectkey = 2

 $object.InstanceKey = $InstanceKey

 # permission permissionname

 # ---------- --------------

 # 1 Read

 # 2 Modify

 # 4 Delete

 # 8 Distribute

 $object.InstancePermissions = 1+2+4+8

 $object.put()

 }

 }

 }

**Remove-Variable** Packages, i, placeholder, NodeIDs, ContainerNodeID, InstanceKey, AreThereMoreNodeIDs, object, AlreadySet, SecureObject *-ErrorAction* SilentlyContinue

# SCCM Security - Packages

# This will display all packages in the root of the desired top level Packages folder

# To find the ContainerNodeID run this: Get-WmiObject -Namespace root\SMS\site\_ABC-Query "Select name, containernodeid from SMS\_ObjectContainerNode" |sort name | ft name, containernodeid

$Packages = **Get-WmiObject** *-Namespace* root\SMS\site\_ABC*-Query* "Select instancekey from SMS\_ObjectContainerItem where containernodeID = '[Container Node ID of the ServerGroup Package Folder without the brackets]'" | **select-object** InstanceKey

# $placeholder will hold all of the Packages Instance keys under the top level folder. ie: recursive

$placeholder += $Packages

# Look for other folders under the top level folder

# Find the second level folder of the first level folder from the top level folder

$NodeIDs = **Get-WmiObject** *-Namespace* root\SMS\site\_ABC*-Query* "Select ContainerNodeID from SMS\_ObjectContainerNode where ParentContainerNodeID = '[Container Node ID of the ServerGroup Package Folder without the brackets]'"

if ($NodeIDs -ne $null)

 {

 foreach ($i in $NodeIDs)

 {

 $ContainerNodeID = $i.ContainerNodeID

 $Packages = **Get-WmiObject** *-Namespace* root\SMS\site\_ABC*-Query* "Select instancekey from SMS\_ObjectContainerItem where containernodeID = '$ContainerNodeID'" | **select-object** InstanceKey

 $placeholder += $Packages

 $NodeIDs = **Get-WmiObject** *-Namespace* root\SMS\site\_ABC*-Query* "Select ContainerNodeID from SMS\_ObjectContainerNode where ParentContainerNodeID = '$ContainerNodeID'"

 # start while loop to look for second, third, forth, etc. Advertisements

 $AreThereMoreNodeIDs = (**Get-WmiObject** *-Namespace* root\SMS\site\_ABC*-Query* "Select ContainerNodeID from SMS\_ObjectContainerNode where ParentContainerNodeID = '$ContainerNodeID'").count

 while ($AreThereMoreNodeIDs -ne $null)

 {

 foreach ($i in $NodeIDs)

 {

 $ContainerNodeID = $i.ContainerNodeID

 $Packages = **Get-WmiObject** *-Namespace* root\SMS\site\_ABC*-Query* "Select instancekey from SMS\_ObjectContainerItem where containernodeID = '$ContainerNodeID'" | **select-object** InstanceKey

 $placeholder += $Packages

 $NodeIDs = **Get-WmiObject** *-Namespace* root\SMS\site\_ABC*-Query* "Select ContainerNodeID from SMS\_ObjectContainerNode where ParentContainerNodeID = '$ContainerNodeID'"

 While ($NodeIDs -ne $null)

 {

 foreach ($i in $NodeIDs)

 {

 $ContainerNodeID = $i.ContainerNodeID

 $Packages = **Get-WmiObject** *-Namespace* root\SMS\site\_ABC*-Query* "Select instancekey from SMS\_ObjectContainerItem where containernodeID = '$ContainerNodeID'" | **select-object** InstanceKey

 $placeholder += $Packages

 $NodeIDs = **Get-WmiObject** *-Namespace* root\SMS\site\_ABC*-Query* "Select ContainerNodeID from SMS\_ObjectContainerNode where ParentContainerNodeID = '$ContainerNodeID'"

 }

 }

 }

 IF ($NodeIDs -ne $null) {$AreThereMoreNodeIDs = "yes"} else {$AreThereMoreNodeIDs = $null}

 }

 }

 }

# Set the permissions on all the Instance Keys in $Placeholder

$placeholder = $placeholder | **sort**

foreach ($i in $placeholder)

 {

 if ($i -ne $null)

 {

 #ObjectKey Value

 #--------- ------

 #1 collections

 #2 Packages

 #3 advertisements

 #

 # Sample query below

 #$CollectionSecurity = foreach ($i in $subcollections) {get-wmiobject -Namespace root\SMS\site\_ABC-Query "Select \* From SMS\_UserInstancePermissionNames WHERE ObjectKey=3 AND InstanceKey='$i'"}

 $InstanceKey = $i.InstanceKey

 $SecureObject = **get-wmiobject** *-Namespace* root\SMS\site\_ABC*-Query* "Select username From SMS\_UserInstancePermissionNames WHERE InstanceKey='$InstanceKey'"

 foreach ($i in $SecureObject)

 {

 $AlreadySet = $null

 if ($i.username -eq "ADDOMAIN\SCCM.ServerGroup")

 {

 $AlreadySet = $true

 break

 }

 }

 If ($AlreadySet -ne $true)

 {

 $object = ([wmiclass]"\\.\root\SMS\site\_ABC:SMS\_UserInstancePermissions").CreateInstance()

 $object.username = "ADDOMAIN\SCCM.WorkstationGroup"

 $object.objectkey = 2

 $object.InstanceKey = $InstanceKey

 # permission permissionname

 # ---------- --------------

 # 1 Read

 # 2 Modify

 # 4 Delete

 # 8 Distribute

 $object.InstancePermissions = 1+2+4+8

 $object.put()

 }

 }

 }

**Remove-Variable** Packages, i, placeholder, NodeIDs, ContainerNodeID, InstanceKey, AreThereMoreNodeIDs, object, AlreadySet, SecureObject *-ErrorAction* SilentlyContinue

### SCCM Package SQL Server Agent Job – This is the SQL job that will kick off the script

USE [msdb]

GO

/\*\*\*\*\*\* Object: Job [SCCM-Security-Packages] Script Date: 05/26/2010 15:57:42 \*\*\*\*\*\*/

BEGIN TRANSACTION

DECLARE @ReturnCode INT

SELECT @ReturnCode = 0

/\*\*\*\*\*\* Object: JobCategory [[Uncategorized (Local)]]] Script Date: 05/26/2010 15:57:42 \*\*\*\*\*\*/

IF NOT EXISTS (SELECT name FROM msdb.dbo.syscategories WHERE name=N'[Uncategorized (Local)]' AND category\_class=1)

BEGIN

EXEC @ReturnCode = msdb.dbo.sp\_add\_category @class=N'JOB', @type=N'LOCAL', @name=N'[Uncategorized (Local)]'

IF (@@ERROR <> 0 OR @ReturnCode <> 0) GOTO QuitWithRollback

END

DECLARE @jobId BINARY(16)

EXEC @ReturnCode = msdb.dbo.sp\_add\_job @job\_name=N'SCCM-Security-Packages',

 @enabled=1,

 @notify\_level\_eventlog=0,

 @notify\_level\_email=0,

 @notify\_level\_netsend=0,

 @notify\_level\_page=0,

 @delete\_level=0,

 @description=N'No description available.',

 @category\_name=N'[Uncategorized (Local)]',

 @owner\_login\_name=N'ADDOMAIN\UserID', @job\_id = @jobId OUTPUT

IF (@@ERROR <> 0 OR @ReturnCode <> 0) GOTO QuitWithRollback

/\*\*\*\*\*\* Object: Step [SCCM-Security-Advertisements] Script Date: 05/26/2010 15:57:42 \*\*\*\*\*\*/

EXEC @ReturnCode = msdb.dbo.sp\_add\_jobstep @job\_id=@jobId, @step\_name=N'SCCM-Security-Advertisements',

 @step\_id=1,

 @cmdexec\_success\_code=0,

 @on\_success\_action=1,

 @on\_success\_step\_id=0,

 @on\_fail\_action=2,

 @on\_fail\_step\_id=0,

 @retry\_attempts=0,

 @retry\_interval=0,

 @os\_run\_priority=0, @subsystem=N'CmdExec',

 @command=N'C:\WINDOWS\system32\WINDOW~2\v1.0\powershell.exe -command "& ''I:\SCCM Scripts\SCCM Powershell Production Scripts\SCCM-Security-Package.ps1''"',

 @flags=0,

 @proxy\_name=N'SQL\_Proxy\_ID'

IF (@@ERROR <> 0 OR @ReturnCode <> 0) GOTO QuitWithRollback

EXEC @ReturnCode = msdb.dbo.sp\_update\_job @job\_id = @jobId, @start\_step\_id = 1

IF (@@ERROR <> 0 OR @ReturnCode <> 0) GOTO QuitWithRollback

EXEC @ReturnCode = msdb.dbo.sp\_add\_jobserver @job\_id = @jobId, @server\_name = N'(local)'

IF (@@ERROR <> 0 OR @ReturnCode <> 0) GOTO QuitWithRollback

COMMIT TRANSACTION

GOTO EndSave

QuitWithRollback:

 IF (@@TRANCOUNT > 0) ROLLBACK TRANSACTION

EndSave:

### SCCM Package SQL Trigger – This is needed to kick of the script whenever a new collection is created

USE [SMS\_CEN]

GO

/\*\*\*\*\*\* Object: Trigger [dbo].[SCCM-Security-Packages] Script Date: 05/26/2010 16:00:21 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

create trigger [dbo].[SCCM-Security-Packages]

on [dbo].[SMSPackages]

after insert

as EXEC msdb.dbo.sp\_start\_job N'SCCM-Security-Packages'