Inventory Mapped Network Printers and Drives With SCCM

A common issue we have is knowing what drives and printers are actively mapped and being used. There are several ways to get usage data via the file server and print servers, but these are not always adequate and fail to produce a complete picture of the user environment. You can currently return local printers as the system account has rights to that data, but anything mapped under the user account will not be accessible.

What I have done is modify some of the work done at https://social.technet.microsoft.com/Forums/en-US/c08c393d-1ea4-4f6b-8f07-affc0f743193/network-printer-inventory-in-system-centre-configuration-manager-sccm-2012 and extend it to allow for multiple users on a machine to have data reported as well as cleaning out anything that is no longer current. This allows an administrator to see how many users and how many machines are connecting to the network drive and printer resources and also report on users attempting to connect to resources that have been decommissioned. When a machine tries connecting to a resource that is no longer available, this impacts the user's environment and causes degraded performance.

I am setting this up in a SCCM 2012 R2 SP1 environment. This setup should account for both 32 and 64 systems as well as multiple users on a system, such as a Citrix or terminal server environment. All files needed are attached to this posting here, but I will be putting the code

blocks inline in case you want to copy and paste or your environment does not permit file downloads.

I did discover during testing, that since this runs as the user, the powershell script will not run unless signed due to the default permissions of restricted. You could of course change powershell's execution policy to unrestricted, but this is no good for an enterprise, so we'll check out Scott and Ed's posts at http://www.hanselman.com/blog/SigningPowerShellScripts.aspx and http://blogs.technet.com/b/heyscriptingguy/archive/2010/06/16/hey-scripting-guy-how-can-i-sign-windows-powershell-scripts-with-an-enterprise-windows-pki-part-1-of-2.aspx and Part 2 on how to sign code, which is the proper way. I also setup a GPO to enable running local and remote signed code to allow the scripts to execute using the information from http://www.techrepublic.com/blog/the-enterprise-cloud/set-the-powershell-execution-policy-via-group-policy/

First you need to extend the configuration.mof file. We just need to add the following code to your configuration.mof and run the command (modify to match your install location at the primary or central site server) (see file **Custom_Printer_Drives_Configuration.mof**)

> *%WINDIR%\System32\WBEM\Mofcomp.exe "C:\Program Files\Microsoft Configuration Manager\inboxes\clifiles.src\hinv\configuration.mof"*

Paste the codeblock between these lines. You may have material in there already, just add this after your own custom inventory information.

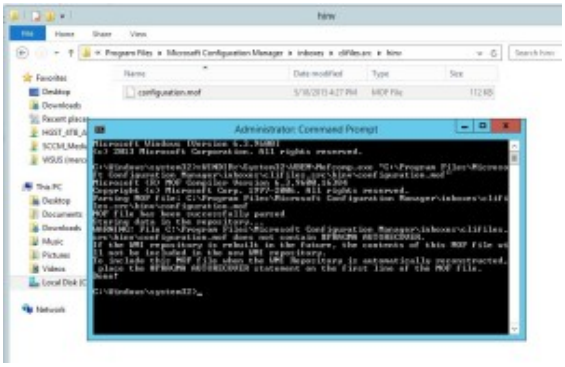
//=========================
// Added extensions start
//=========================

//=========================
// Added extensions end
//=========================

```
1.  //==== Start custom printer and drive reporting ====
2.
3.  #pragma namespace ("\\\\.\\root\\cimv2")
4.  #pragma deleteclass("MAPPEDDRIVES", NOFAIL)
5.  [dynamic, provider("RegProv"),
    ClassContext("Local|HKEY_LOCAL_MACHINE\\SOFTWARE\\Wow6432Node\\SCCMINVENTORY\\MAPPEDDR
6.  Class MAPPEDDRIVES
7.  {
8.  [key] string KeyName;
9.  [PropertyContext("UserDomain")] String UserDomain;
10. [PropertyContext("UserName")] String UserName;
11. [PropertyContext("ShareName")] String ShareName;
12. [PropertyContext("DriveLetter")] String DriveLetter;
```

```
 [PropertyContext("Size")] Uint32 Size;
 [PropertyContext("FreeSpace")] Uint32 FreeSpace;
 [PropertyContext("System")] String System;
 [PropertyContext("FileSystem")] String FileSystem;
 [PropertyContext("DateInventoried")] String DateInventoried;
 };

 #pragma namespace ("\\\\.\\root\\cimv2")
 #pragma deleteclass("MAPPEDDRIVES_64", NOFAIL)
 [dynamic, provider("RegProv"),
ClassContext("Local|HKEY_LOCAL_MACHINE\\SOFTWARE\\Wow6432Node\\SCCMINVENTORY\\MAPPEDDR
 Class MAPPEDDRIVES_64
 {
 [key] string KeyName;
 [PropertyContext("UserDomain")] String UserDomain;
 [PropertyContext("UserName")] String UserName;
 [PropertyContext("ShareName")] String ShareName;
 [PropertyContext("DriveLetter")] String DriveLetter;
 [PropertyContext("Size")] Uint32 Size;
 [PropertyContext("FreeSpace")] Uint32 FreeSpace;
 [PropertyContext("System")] String System;
 [PropertyContext("FileSystem")] String FileSystem;
 [PropertyContext("DateInventoried")] String DateInventoried;
 };

 #pragma namespace ("\\\\.\\root\\cimv2")
 #pragma deleteclass("NETWORKPRINTERS", NOFAIL)
 [dynamic, provider("RegProv"),
ClassContext("Local|HKEY_LOCAL_MACHINE\\SOFTWARE\\Wow6432Node\\SCCMINVENTORY\\NETWORKP
 Class NETWORKPRINTERS
 {
 [key] string KeyName;
 [PropertyContext("UserDomain")] String UserDomain;
 [PropertyContext("UserName")] String UserName;
 [PropertyContext("PrintServer")] String PrintServer;
 [PropertyContext("PrinterQueue")] String PrinterQueue;
 [PropertyContext("PrinterLocation")] String PrinterLocation;
 [PropertyContext("PrinterDriver")] String PrinterDriver;
 [PropertyContext("PrintProcessor")] String PrintProcessor;
 [PropertyContext("PrinterPortName")] String PrinterPortName;
 [PropertyContext("DateInventoried")] String DateInventoried;
 };

 #pragma namespace ("\\\\.\\root\\cimv2")
 #pragma deleteclass("NETWORKPRINTERS_64", NOFAIL)
 [dynamic, provider("RegProv"),
ClassContext("Local|HKEY_LOCAL_MACHINE\\SOFTWARE\\Wow6432Node\\SCCMINVENTORY\\NETWORKP
 Class NETWORKPRINTERS_64
 {
 [key] string KeyName;
 [PropertyContext("UserDomain")] String UserDomain;
 [PropertyContext("UserName")] String UserName;
 [PropertyContext("PrintServer")] String PrintServer;
 [PropertyContext("PrinterQueue")] String PrinterQueue;
 [PropertyContext("PrinterLocation")] String PrinterLocation;
 [PropertyContext("PrinterDriver")] String PrinterDriver;
 [PropertyContext("PrintProcessor")] String PrintProcessor;
 [PropertyContext("PrinterPortName")] String PrinterPortName;
 [PropertyContext("DateInventoried")] String DateInventoried;
 };

 //===== End custom printer and drive reporting ======
```

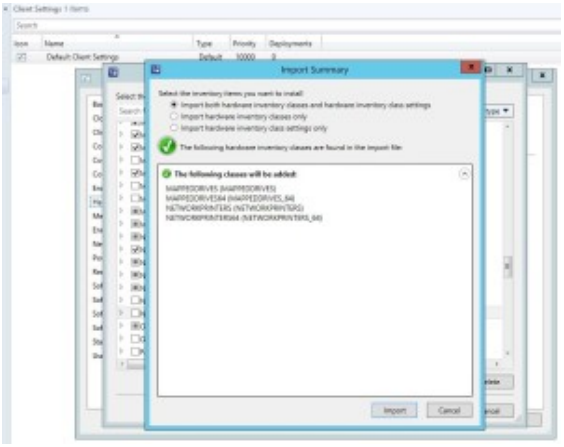
It should look something like this

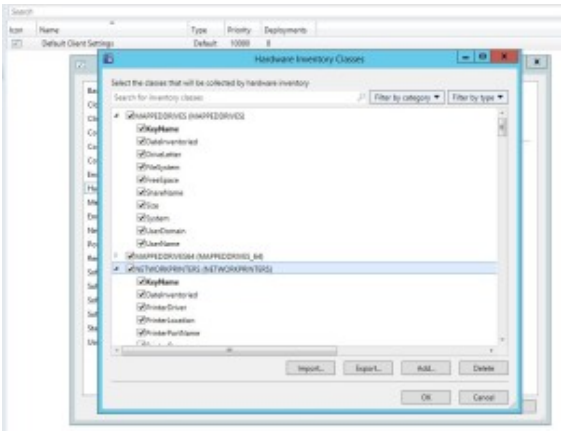


Then we need to add this to the hardware inventory. This should be as simple as importing the mof file **Printer_and_Drive_Inventory_To_Import.mof** and making sure everything is checked

```
#pragma namespace ("\\\\.\\root\\cimv2\\SMS")
#pragma deleteclass("NETWORKPRINTERS", NOFAIL)
[SMS_Report(TRUE),SMS_Group_Name("NETWORKPRINTERS"),SMS_Class_ID("NETWORKPRINTERS"),
SMS_Context_1("__ProviderArchitecture=32|uint32"),
SMS_Context_2("__RequiredArchitecture=true|boolean")]
Class NETWORKPRINTERS: SMS_Class_Template
{
[SMS_Report(TRUE),key] string KeyName;
[SMS_Report(TRUE)] String UserDomain;
[SMS_Report(TRUE)] String UserName;
[SMS_Report(TRUE)] String PrintServer;
[SMS_Report(TRUE)] String PrinterQueue;
[SMS_Report(TRUE)] String PrinterLocation;
[SMS_Report(TRUE)] String PrinterDriver;
[SMS_Report(TRUE)] String PrintProcessor;
[SMS_Report(TRUE)] String PrinterPortName;
[SMS_Report(TRUE)] String DateInventoried;
};

#pragma namespace ("\\\\.\\root\\cimv2\\SMS")
#pragma deleteclass("NETWORKPRINTERS_64", NOFAIL)

[SMS_Report(TRUE),SMS_Group_Name("NETWORKPRINTERS64"),SMS_Class_ID("NETWORKPRINTERS64"
SMS_Context_1("__ProviderArchitecture=64|uint32"),
SMS_Context_2("__RequiredArchitecture=true|boolean")]
```

```
Class NETWORKPRINTERS_64 : SMS_Class_Template
{
[SMS_Report(TRUE),key] string KeyName;
[SMS_Report(TRUE)] String UserDomain;
[SMS_Report(TRUE)] String UserName;
[SMS_Report(TRUE)] String PrintServer;
[SMS_Report(TRUE)] String PrinterQueue;
[SMS_Report(TRUE)] String PrinterLocation;
[SMS_Report(TRUE)] String PrinterDriver;
[SMS_Report(TRUE)] String PrintProcessor;
[SMS_Report(TRUE)] String PrinterPortName;
[SMS_Report(TRUE)] String DateInventoried;
};

#pragma namespace ("\\\\.\\root\\cimv2\\SMS")
#pragma deleteclass("MAPPEDDRIVES", NOFAIL)
[SMS_Report(TRUE),SMS_Group_Name("MAPPEDDRIVES"),SMS_Class_ID("MAPPEDDRIVES"),
SMS_Context_1("__ProviderArchitecture=32|uint32"),
SMS_Context_2("__RequiredArchitecture=true|boolean")]
Class MAPPEDDRIVES: SMS_Class_Template
{
[SMS_Report(TRUE),key] string KeyName;
[SMS_Report(TRUE)] String UserDomain;
[SMS_Report(TRUE)] String UserName;
[SMS_Report(TRUE)] String ShareName;
[SMS_Report(TRUE)] String DriveLetter;
[SMS_Report(TRUE)] Uint32 Size;
[SMS_Report(TRUE)] Uint32 FreeSpace;
[SMS_Report(TRUE)] String System;
[SMS_Report(TRUE)] String FileSystem;
[SMS_Report(TRUE)] String DateInventoried;
};

#pragma namespace ("\\\\.\\root\\cimv2\\SMS")
#pragma deleteclass("MAPPEDDRIVES_64", NOFAIL)
[SMS_Report(TRUE),SMS_Group_Name("MAPPEDDRIVES64"),SMS_Class_ID("MAPPEDDRIVES64"),
SMS_Context_1("__ProviderArchitecture=64|uint32"),
SMS_Context_2("__RequiredArchitecture=true|boolean")]
Class MAPPEDDRIVES_64 : SMS_Class_Template
{
[SMS_Report(TRUE),key] string KeyName;
[SMS_Report(TRUE)] String UserDomain;
[SMS_Report(TRUE)] String UserName;
[SMS_Report(TRUE)] String ShareName;
[SMS_Report(TRUE)] String DriveLetter;
[SMS_Report(TRUE)] Uint32 Size;
[SMS_Report(TRUE)] Uint32 FreeSpace;
[SMS_Report(TRUE)] String System;
[SMS_Report(TRUE)] String FileSystem;
[SMS_Report(TRUE)] String DateInventoried;
};
```





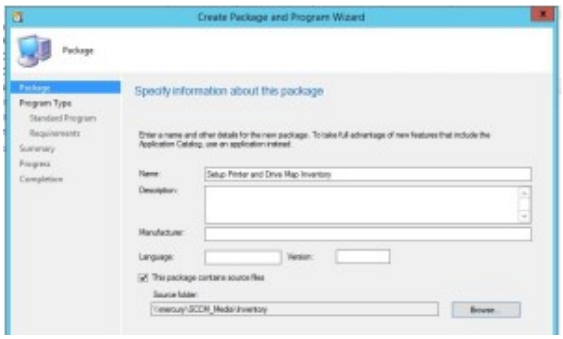
Then we need to create a script to generate a registry setting to write the data to and assign permissions that will allow users to write this location (see file: **Create_Registry_Inventory.ps1**)

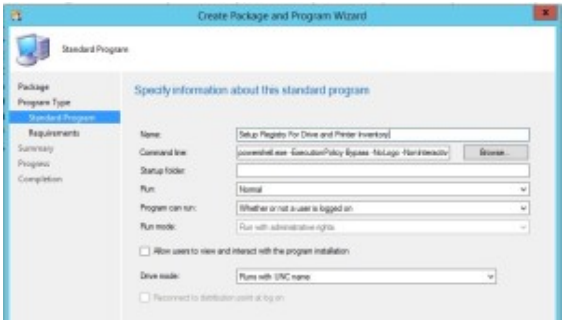
```
1.  if (!(Test-Path HKLM:\SOFTWARE\SCCMINVENTORY)) {new-item
    HKLM:\SOFTWARE\SCCMINVENTORY  -ErrorAction SilentlyContinue}
2.  $perm = get-acl HKLM:\SOFTWARE\SCCMINVENTORY  -ErrorAction SilentlyContinue
3.  $rule = New-Object System.Security.AccessControl.RegistryAccessRule("Authenticated
    Users","FullControl", "ContainerInherit, ObjectInherit", "InheritOnly", "Allow")  -
    ErrorAction SilentlyContinue
4.  $perm.SetAccessRule($rule)
5.  Set-Acl -Path HKLM:\SOFTWARE\SCCMINVENTORY $perm  -ErrorAction SilentlyContinue
6.  if (!(Test-Path HKLM:\SOFTWARE\SCCMINVENTORY\NETWORKPRINTERS)) {new-item
    HKLM:\SOFTWARE\SCCMINVENTORY\NETWORKPRINTERS -ErrorAction SilentlyContinue}
7.  if (!(Test-Path HKLM:\SOFTWARE\SCCMINVENTORY\MAPPEDDRIVES)) {new-item
    HKLM:\SOFTWARE\SCCMINVENTORY\MAPPEDDRIVES -ErrorAction SilentlyContinue}
```

You can setup a package to run this file on all machines in your environment once or limit to a smaller collection.

I am using the command line as follows and all files for this project will be in the same folder as they are small, but you can split them into multiple folders or create multiple programs and advertise each individually.

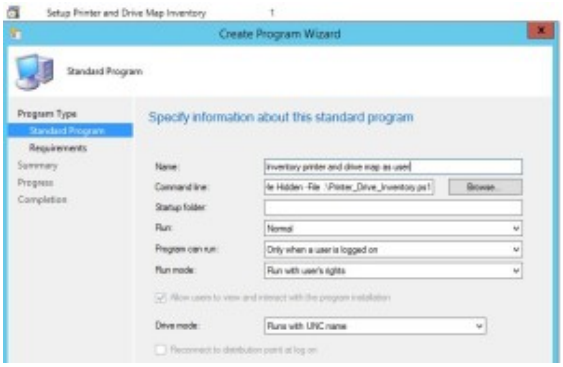
> *powershell.exe -ExecutionPolicy Bypass -NoLogo -NonInteractive -NoProfile -WindowStyle Hidden -File .\Create_Registry_Inventory.ps1*



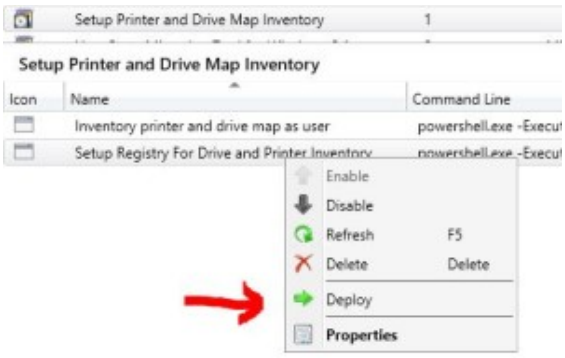


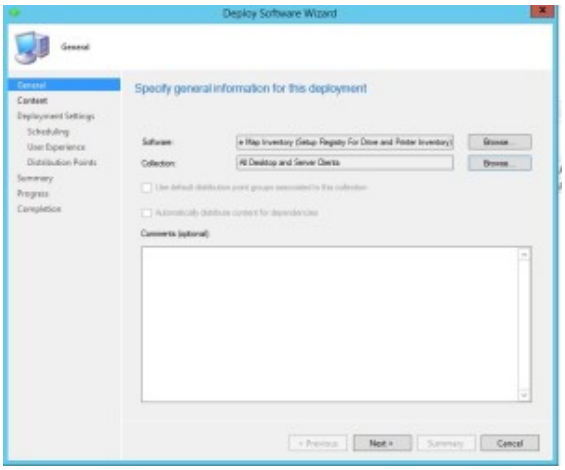
Then setup a program for the user piece using the following command line (see file **Printer_Drive_Inventory.ps1**):

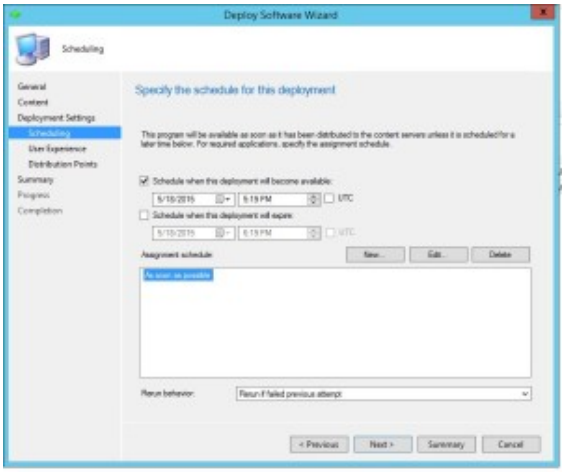
> *powershell.exe -ExecutionPolicy Bypass -NoLogo -NonInteractive -NoProfile -WindowStyle Hidden -File .\Printer_Drive_Inventory.ps1*

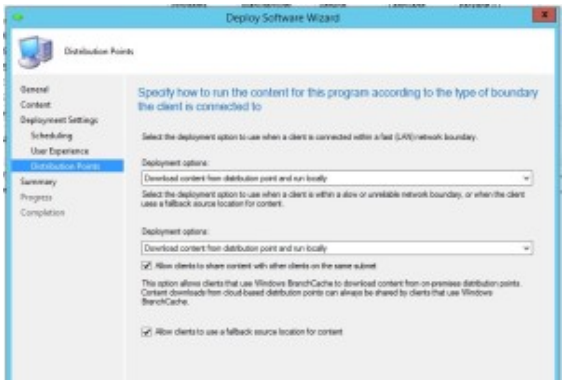


Then setup the deployment and schedule to run only if previously failed





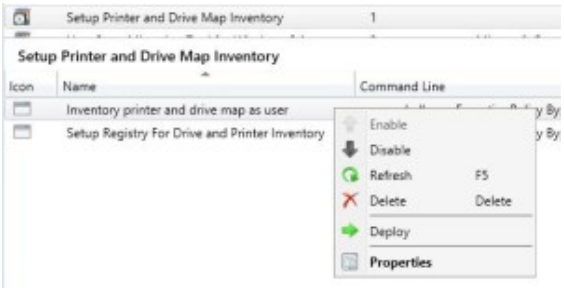




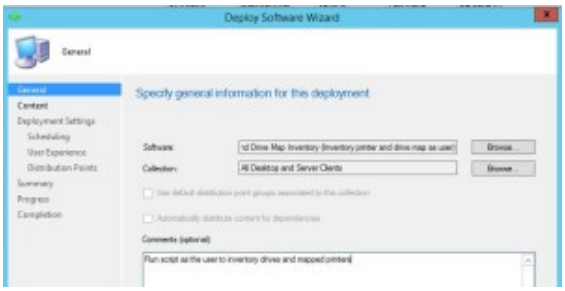
Then setup the deployment for the user piece. This will run the following code

```
 # https://social.technet.microsoft.com/Forums/en-US/c08c393d-1ea4-4f6b-8f07-affc0f743193/network-printer-inventory-in-system-centre-configuration-manager-sccm-2012?forum=configmanagergeneral#c08c393d-1ea4-4f6b-8f07-affc0f743193
 # http://blogs.technet.com/b/breben/archive/2013/08/26/inventory-mapped-drives-in-configmgr-2012.aspx

 # run with user rights
 # PowerShell.exe -NonInteractive -WindowStyle Hidden -noprofile -ExecutionPolicy Bypass -file .\Printer_Drive_Inventory.ps1

 $printers = Get-WMIObject -class Win32_Printer -ErrorAction SilentlyContinue|select-Object -Property ServerName,ShareName,Location,DriverName,PrintProcessor,PortName,Local |Where-Object {$_.Local -ne $true}|Where-Object {$_.ServerName.length -gt 2} -ErrorAction SilentlyContinue
 $user = $([System.Security.Principal.WindowsIdentity]::GetCurrent().Name).Replace('\','-')

 #Remove previous entries
 Get-ChildItem -Path HKLM:\SOFTWARE\SCCMINVENTORY\NETWORKPRINTERS\ -Recurse -Include $user* -ErrorAction SilentlyContinue | Remove-Item

 ForEach($printer in $printers){

 Try {
     $PServerName= $printer.ServerName -replace ('\\','')
     $PShareName = $printer.ShareName
     $PLocation = $printer.Location
     $PDriverName = $printer.DriverName
     $PPrintProcessor = $printer.PrintProcessor
     $PPortName = $printer.PortName

     if ((Test-Path HKLM:\SOFTWARE\SCCMINVENTORY\NETWORKPRINTERS)) {
         if ((Test-Path "HKLM:\SOFTWARE\SCCMINVENTORY\NETWORKPRINTERS\$user $PShareName on $PServerName")) {
             Remove-item "HKLM:\SOFTWARE\SCCMINVENTORY\NETWORKPRINTERS\$user $PShareName on $PServerName" -Force -ErrorAction SilentlyContinue
         }
         New-item "HKLM:\SOFTWARE\SCCMINVENTORY\NETWORKPRINTERS\$user $PShareName on $PServerName" -ErrorAction SilentlyContinue
         New-ItemProperty "HKLM:\SOFTWARE\SCCMINVENTORY\NETWORKPRINTERS\$user $PShareName on $PServerName" -Name "UserDomain" -Value $user.Split('-')[0] -PropertyType "String" -ErrorAction SilentlyContinue
         New-ItemProperty "HKLM:\SOFTWARE\SCCMINVENTORY\NETWORKPRINTERS\$user $PShareName on $PServerName" -Name "UserName" -Value $user.Split('-')[1] -PropertyType "String" -ErrorAction SilentlyContinue
         New-ItemProperty "HKLM:\SOFTWARE\SCCMINVENTORY\NETWORKPRINTERS\$user $PShareName on $PServerName" -Name "PrintServer" -Value $PServerName -PropertyType "String" -ErrorAction SilentlyContinue
         New-ItemProperty "HKLM:\SOFTWARE\SCCMINVENTORY\NETWORKPRINTERS\$user $PShareName on $PServerName" -Name "PrinterQueue" -Value $PShareName -PropertyType "String" -ErrorAction SilentlyContinue
         New-ItemProperty "HKLM:\SOFTWARE\SCCMINVENTORY\NETWORKPRINTERS\$user $PShareName on $PServerName" -Name "PrinterLocation" -Value $PLocation -PropertyType "String" -ErrorAction SilentlyContinue
         New-ItemProperty "HKLM:\SOFTWARE\SCCMINVENTORY\NETWORKPRINTERS\$user $PShareName on $PServerName" -Name "PrinterDriver" -Value $PDriverName -PropertyType "String" -ErrorAction SilentlyContinue
         New-ItemProperty "HKLM:\SOFTWARE\SCCMINVENTORY\NETWORKPRINTERS\$user $PShareName on $PServerName" -Name "PrintProcessor" -Value $PPrintProcessor -PropertyType "String" -ErrorAction SilentlyContinue
         New-ItemProperty "HKLM:\SOFTWARE\SCCMINVENTORY\NETWORKPRINTERS\$user $PShareName on $PServerName" -Name "PrinterPortName" -Value $PPortName -PropertyType "String" -ErrorAction SilentlyContinue
```

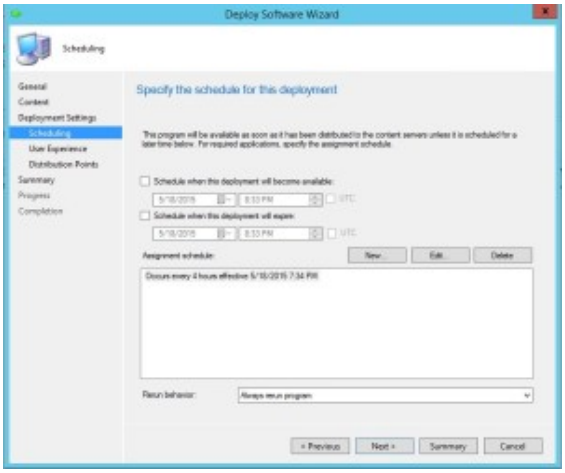
```
         New-ItemProperty "HKLM:\SOFTWARE\SCCMINVENTORY\NETWORKPRINTERS\$user $PShareName on $PServerName" -Name "DateInventoried" -Value $(get-date) -PropertyType "String" -ErrorAction SilentlyContinue
     } # End If
     } # End Try
 Catch  {}
 } #End For Each


 #now inventory drives
 $drives = Get-WMIObject -class Win32_MappedLogicalDisk -ErrorAction SilentlyContinue|select-Object -Property Caption,Name,FreeSpace,ProviderName,Size,SystemName,FileSystem |Where-Object {$_.Local -ne $true}|Where-Object {$_.ProviderName.length -gt 3} -ErrorAction SilentlyContinue

 #Remove previous entries
 Get-ChildItem -Path HKLM:\SOFTWARE\SCCMINVENTORY\MAPPEDDRIVES\ -Recurse -Include $user* -ErrorAction SilentlyContinue | Remove-Item

 ForEach($drive in $drives){
 Try {
     $DShareName = $drive.ProviderName -Replace ('\\','\')
     $DName = $drive.Name
     #convert to GB
     $DSize = $drive.Size/1000000000
     $DFreeSpace = $drive.FreeSpace/1000000000
     $DSystem = $drive.SystemName
     $DFileSystem = $drive.FileSystem

     if ((Test-Path HKLM:\SOFTWARE\SCCMINVENTORY\MAPPEDDRIVES)) {
         if ((Test-Path "HKLM:\SOFTWARE\SCCMINVENTORY\MAPPEDDRIVES\$user $DName")) {
             Remove-item "HKLM:\SOFTWARE\SCCMINVENTORY\MAPPEDDRIVES\$user $DName" -Force -ErrorAction SilentlyContinue
         }
         New-item "HKLM:\SOFTWARE\SCCMINVENTORY\MAPPEDDRIVES\$user $DName" -ErrorAction SilentlyContinue
         New-ItemProperty "HKLM:\SOFTWARE\SCCMINVENTORY\MAPPEDDRIVES\$user $DName" -Name "UserDomain" -Value $user.Split('-')[0] -PropertyType "String" -ErrorAction SilentlyContinue
         New-ItemProperty "HKLM:\SOFTWARE\SCCMINVENTORY\MAPPEDDRIVES\$user $DName" -Name "UserName" -Value $user.Split('-')[1] -PropertyType "String" -ErrorAction SilentlyContinue
         New-ItemProperty "HKLM:\SOFTWARE\SCCMINVENTORY\MAPPEDDRIVES\$user $DName" -Name "ShareName" -Value $DShareName -PropertyType "String" -ErrorAction SilentlyContinue
         New-ItemProperty "HKLM:\SOFTWARE\SCCMINVENTORY\MAPPEDDRIVES\$user $DName" -Name "DriveLetter" -Value $DName -PropertyType "String" -ErrorAction SilentlyContinue
         New-ItemProperty "HKLM:\SOFTWARE\SCCMINVENTORY\MAPPEDDRIVES\$user $DName" -Name "Size" -Value $DSize -PropertyType "DWord" -ErrorAction SilentlyContinue
         New-ItemProperty "HKLM:\SOFTWARE\SCCMINVENTORY\MAPPEDDRIVES\$user $DName" -Name "FreeSpace" -Value $DFreeSpace -PropertyType "DWord" -ErrorAction SilentlyContinue
         New-ItemProperty "HKLM:\SOFTWARE\SCCMINVENTORY\MAPPEDDRIVES\$user $DName" -Name "System" -Value $DSystem -PropertyType "String" -ErrorAction SilentlyContinue
         New-ItemProperty "HKLM:\SOFTWARE\SCCMINVENTORY\MAPPEDDRIVES\$user $DName" -Name "FileSystem" -Value $DFileSystem -PropertyType "String" -ErrorAction SilentlyContinue
         New-ItemProperty "HKLM:\SOFTWARE\SCCMINVENTORY\MAPPEDDRIVES\$user $DName" -Name "DateInventoried" -Value $(get-date) -PropertyType "String" -ErrorAction SilentlyContinue
     } # End If
     } # End Try
 Catch {}
 } #End For Each
```

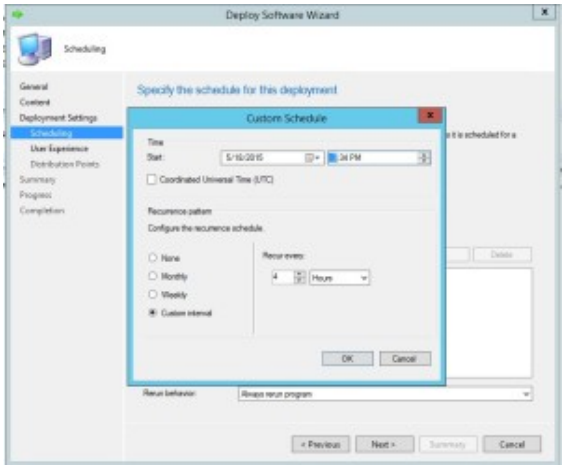


Assign a collection

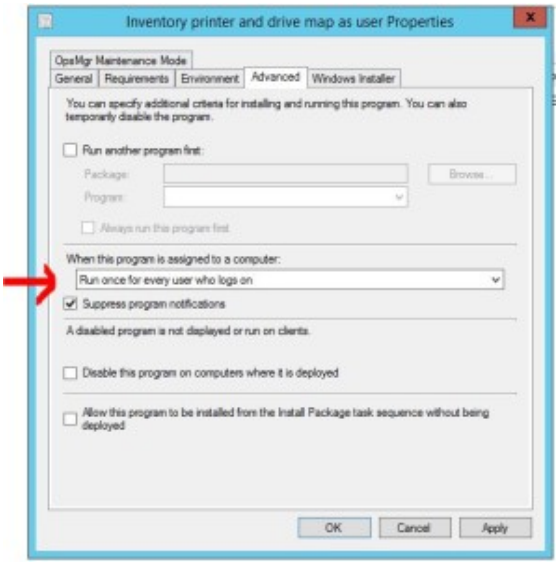


Setting the schedule to run every 4 hours should capture most use cases. You can accomplish this with a login script to call this file also

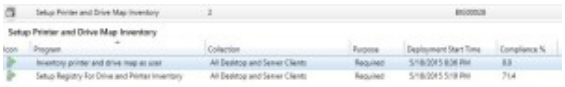




Also go back in and make sure to set it to run once for each user that is logged in and that the **Run** setting under the **General** tab is set to **Hidden**



So now we have 2 deployments



Then, provided everything is working, you should start to see data come in. Remember this is setup to run every 4 hours, but we're patient and looking for environmental data for a large group of people.

You can then query the data in SQL and write SSRS reports to show this data

Mapped drives can be uncovered here

```
SELECT [MachineID]
      ,[InstanceKey]
      ,[TimeKey]
      ,[RevisionID]
      ,[AgentID]
      ,[rowversion]
      ,[DateInventoried00]
      ,[DriveLetter00]
      ,[FileSystem00]
      ,[FreeSpace00]
      ,[KeyName00]
      ,[ShareName00]
      ,[Size00]
      ,[System00]
      ,[UserDomain00]
      ,[UserName00]
  FROM [MAPPEDDRIVES_DATA]

  Union

  SELECT [MachineID]
```

```
        ,[InstanceKey]
        ,[TimeKey]
        ,[RevisionID]
        ,[AgentID]
        ,[rowversion]
        ,[DateInventoried00]
        ,[DriveLetter00]
        ,[FileSystem00]
        ,[FreeSpace00]
        ,[KeyName00]
        ,[ShareName00]
        ,[Size00]
        ,[System00]
        ,[UserDomain00]
        ,[UserName00]
    FROM [MAPPEDDRIVES64_DATA]
```

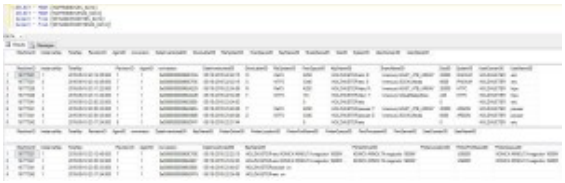
And printers can be found here

```
SELECT [MachineID]
      ,[InstanceKey]
      ,[TimeKey]
      ,[RevisionID]
      ,[AgentID]
      ,[rowversion]
      ,[DateInventoried00]
      ,[KeyName00]
      ,[PrinterDriver00]
      ,[PrinterLocation00]
      ,[PrinterPortName00]
      ,[PrinterQueue00]
      ,[PrintProcessor00]
      ,[PrintServer00]
      ,[UserDomain00]
      ,[UserName00]
  FROM [NETWORKPRINTERS_DATA]

  Union

  SELECT [MachineID]
      ,[InstanceKey]
      ,[TimeKey]
      ,[RevisionID]
      ,[AgentID]
      ,[rowversion]
      ,[DateInventoried00]
      ,[KeyName00]
      ,[PrinterDriver00]
      ,[PrinterLocation00]
      ,[PrinterPortName00]
      ,[PrinterQueue00]
      ,[PrintProcessor00]
```

```
        ,[PrintServer00]
        ,[UserDomain00]
        ,[UserName00]
    FROM [NETWORKPRINTERS64_DATA]
```

Or the even easier queries of the following which just throws everything on the screen

```
SELECT * FROM [MAPPEDDRIVES_DATA]
SELECT * FROM [MAPPEDDRIVES64_DATA]
Select * From [NETWORKPRINTERS_DATA]
Select * From [NETWORKPRINTERS64_DATA]
```



This is obviously a small environment. Use at your own caution and test before deployment into production. I do need to add in some filtering so it does not populate some obviously bad data. There are a lot of moving parts here, so I would expect issues, but I hope this helps you get started.

Let me know if you have any questions or suggestions. Please email **eric@holzhueter.us**

Files if you missed the link in the document:
https://plainlytechnical.com/files/Printer_And_Drive_Inventory.zip

Edited script text for Printer_Drive_Inventory.ps1 on 5/19/15 to a filter share and server information that is blank as this does no help in reporting. There is also a reg key to increase the MIF file size to 50MB if needed.

Edited script test for Printer_Drive_Inventory.ps1 on 5/21 to add a try/catch into the logic. This will require Windows 7 and Server 2008 as minimum requirements with this change. Those lines can be removed for older environments.

PUBLISHED BY

Eric

I tinker, I fix, I improve. I do these things in all aspects of life, managing computers is one of them. I am also fascinated with data; lots and lots of data. Finding the patterns where none seem to exist is a joy of mine.
View all posts by Eric →

May 18, 2015 Eric SCCM drives, Inventory, printers, SCCM

18 thoughts on “Inventory Mapped Network Printers and Drives With SCCM”

Tien

October 21, 2015 at 11:12 am

This is exactly what I’d like to do. Thanks for spending the time to test and write this out. I’m going to test this and hopefully it will do what I want.

Fred

July 14, 2016 at 7:36 am

Can i Query from inside SCCM 2012? what would be the syntax?

Eric

March 13, 2019 at 3:36 pm

> You have to query the database directly. You can use WQL queries to build collections, but they usually aren’t used for this type of inventory.

Tim

September 26, 2016 at 11:41 am

Love the scripts, it will help us tremendously.

I could not get any information to pull via “Win32_MappedLogicalDisk”, but got the info from “Win32_LogicalDisk”.
Thanks again!

Brett

February 6, 2017 at 4:38 pm

I can't get the Win32_MappedLogicalDisk piece to work in SCCM 2012. The scripts work flawlessly on local testing, but when I push it from SCCM I get no results from Win32_MappedLogicalDisk. I know I'm running in user context because I get the Win32_Printer as expected. I even recreated the entire script as a VBscript and got the same results – Win32_MappedLogicalDisk query gives me results on local run but not from SCCM. I then tried similar query with Win32_LogicalDisk (where driveType = 4 for network drives) and got exactly the same results. Local run enumerates fine, run from SCCM generates nothing. I am really scratching my head why the Win32_Printer query works while the other does not.

Eric

March 13, 2019 at 3:35 pm

Did you figure this out? It may be the issue of not being able to see mapped drives from an elevated prompt. EnableLinkedConnections is needed

https://support.microsoft.com/en-us/help/3035277/mapped-drives-are-not-available-from-an-elevated-prompt-when-uac-is-co

Cory

August 26, 2021 at 1:37 pm

I know this is too late to help you, but maybe it will help a googler like me. I had the same problem and it was two things. First I had to change the script to use Win32_LogicalDisk but even then when pushed via SCCM it wouldn't populate the registry with drives. It worked fine locally however. The issue was if a user is a local admin SCCM seems to run elevated regardless of the package settings.

Now I just wish I could figure out why I can't get the hardware class to show up in resource explorer for machines it is applied to.

Linda B

November 2, 2017 at 1:05 pm

This is the greatest information on this I have found. Very detailed and concise. Exactly what I was looking for. In process of testing.

Thank You

Eric

March 13, 2019 at 3:34 pm

Thank you!

Terry Stringer

January 4, 2018 at 7:55 am

Certainly appreciate this writeup! It's clear, concise and works well.
One question however, it seems that the data being collected may only tend to grow as there's no particular purge process for aged data. For example, if a user logs onto a workstation infrequently, the data records become stale & of little value. Is there a purge process? Or am I overthinking what might happen in the future?

Eric

March 13, 2019 at 3:34 pm

It can grow. The records will delete when machines are cleared from SCCM as they are linked to the machine resourceID. I think I modified some code to clear out anything older than 6 months in a cleanup script that I run on all the machines, but its been a while. It was never an issue even with lots of people logging into machines, like training computers. Most of the time they will only have one or 2 mapped drives. The data can be filtered when queried via SQL and is small in size compared to other inventory data like software installations.

AnilKK

February 1, 2019 at 9:21 am

Hi,
I applied this article to my sccm environment but it is returning only one network drive to console.

Eric

March 13, 2019 at 3:31 pm

Did you figure this out? It may be the issue of not being able to see mapped drives from an elevated prompt. EnableLinkedConnections is needed

https://support.microsoft.com/en-us/help/3035277/mapped-drives-are-not-available-from-an-elevated-prompt-when-uac-is-co

AnilKK

March 31, 2019 at 1:18 pm

Hi Eric,
When you said extending configuration.mof, you mentioned below registry path
“HKEY_LOCAL_MACHINE\\SOFTWARE\\Wow6432Node\\SCCMINVENTORY\\”
but in script i noticed that you we are adding data to
HKEY_LOCAL_MACHINE\\SOFTWARE\\SCCMINVENTORY\\”.

I didn’t get these lines. please help

BradP

September 28, 2021 at 12:15 pm

In a clients InventoryAgent.log I was seeing:
Unknown error encountered processing an instance of class MAPPEDDRIVES64: 80041001
Unknown error encountered processing an instance of class MAPPEDDRIVES: 80041001
Unknown error encountered processing an instance of class NETWORKPRINTERS64: 80041001
Unknown error encountered processing an instance of class NETWORKPRINTERS: 80041001

I adjusted the MOF extension code for x64 to use HKEY_LOCAL_MACHINE\\SOFTWARE\\SCCMINVENTORY\\ and now MAPPEDDRIVES64 and NETWORKPRINTERS64 are populating.

Eric, You may want to review the code as to where regkeys are created and stored for x86/x64 or maybe there is no need to separate them.

Anil Kushwaha

March 13, 2019 at 2:52 pm

Hi Eric,
This blog is exactly what I was looking for ? I applied in my environment, it took little bit time to understand the blog post. Now it is working fine for me. Thanks a lot.

Sarah P.

April 23, 2019 at 7:00 am

Hi, thank you so much for information. It was super helpful and I was able to get this running in my enviornment. I do have a question for you though. Is there a way to configure this to show the default printer? Also, how would I configure that in the SQL database?

Thank you for your time.

Cory

August 26, 2021 at 10:54 am

The scripts seem to be working fine; data is being correctly written to the registry keys. But, I don’t see anything in resource explorer for the machines. I have double checked the hardware classes are setup and the configuration.mof was loaded, so I am really at a loss.
