Extending the ITMS CMDB using Custom Inventory

Description	This exercise is designed to provide a deep understanding of some of the more complex features within Inventory Solution. To extend the type of inventory data that you gather, custom inventory lets you create the new data classes that are not included by default and populate them with gathered data from scripts. There may be special circumstances where data you want is in locations not collected by Inventory Solution. This could be data located in text files, WMI Data, File Information or even performing special calls on hardware or running processes This lab assumes a basic and general understanding of the Symantec Management Platform Basic Microsoft Windows Scripting skills are recommended.

At the end of this lab, you should be able to	▪ Understand the components that make up a Custom Inventory task.
	▪ Understand the inner workings of the following custom inventory examples that:
	  o Executes a WMI Service read that gathers the status of the Windows 'spooler' service.
	  o Performs a Text file read that reads the settings in a file generated by a OS Imaging process.
	  o Advanced WMI calls that gathers the Bitlocker settings before a Windows migration.
	  o Performs a Security forensics analysis that captures file hashes of files in a directory for IOC hunting.
	▪ Custom Inventory Reports.

Notes	▪ Please complete the lab using the instructions on the following pages. ▪ Be sure to ask your instructor any questions you may have. ▪ Thank you for coming to our lab session.

Table of Contents

Introduction to Custom Inventory

Custom Inventory is designed to extend the functionality of Inventory Solution by allowing an administrator to collect data points outside of what is included by default with Inventory Solution. First, a custom data class is created to store your information in the database. Then some methodology for collecting the data point (typically a script of sorts) is executed on the endpoint(s) to gather the data. Once the data is on the Notification Server it is managed like any other piece of information.

The approach to creating custom inventory for an environment follows some simple steps from data class creation, script testing to validating that the data flows through the system and associates with the appropriate machine. These exercises step you through the initial creation of a Custom Inventory and demonstrates how to create the data classes along with four simple scripting examples;

- WMI Service read example that gathers the status of the Windows 'spooler' service.
- Text file read example that reads the settings in a file generated by a OS Imaging process.
- Advanced WMI example that gathers the Bitlocker settings before a Windows migration.
- Security forensics example that captures file hashes of files in a directory for IOC hunting.

We will demonstrate what we believe are best practices for the creation of these script items including the process to follow for creating them. It is important to realize that many functions are available directly within Inventory Solution itself. Please review the Inventory Solution documentation so you are clear what you are able to do from the solution before starting with custom inventory.

Custom Inventory is cross-platform and script-based. The same console screens are used to create and configure custom inventory across all platforms - Windows, Unix, Linux and Macintosh. However, the code is not cross-platform. Custom inventory scripts must be written in a language that can be executed on each OS. The following scripting languages are supported:

- JavaScript
- Perl
- PowerShell
- Python
- Unix Shell script
- VBScript

Lab Exercise 1: Querying the State of a Windows Service

In this scenario, the Administrator has been tasked with gathering the service state of the Windows 'spooler' service on specific print servers that handle the day to day operations of the organizations enterprise printer. It is critical that this service is in a running state as it would seriously affect productivity in the digital print department.

In order to create a Custom Inventory for this requirement it is essential that you complete the following tasks:

- Create the Data Class to hold the gathered data
- Create and test the script that will be used to gather the data
- Create the Custom Inventory script
- Create the Custom Inventory Task
- Target and Schedule the Custom Inventory task

Step 1: Create the Data Class

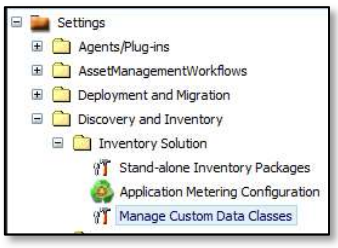
Custom Inventory has a specific area were Data Classes are to be created, these are located through the Symantec Management Console from the console menu select **Settings->Discovery and Inventory->Manage Custom Data Classes**

Data Classes are simply tables that are created within the Symantec CMDB. When referring to Data Classes we are talking about the fields, the data type and actual information created in the database. This data is usually associated with a computer or another resource type. Custom data types are generally mentioned only within Inventory Solution and Asset Management Suite documentation.

The first step is to create a Data Class called Monitor Print Spooler, and within that data class we will create a field called 'Status' where we will store information about the Microsoft Print Spooler, and send information such as is it "Running" or is it "Stopped".

To get this data into the CMDB we will need to create a custom data class for Custom Inventory to place the collected data into.

1. Go to the **SMP** Virtual Machine
2. Open the **Symantec Management Console** by clicking the Icon on the Desktop.
3. Select **Settings | All Settings**
4. Expand **Settings > Discovery & Inventory > Inventory Solution**
5. Select **Manage Custom Data Classes**.

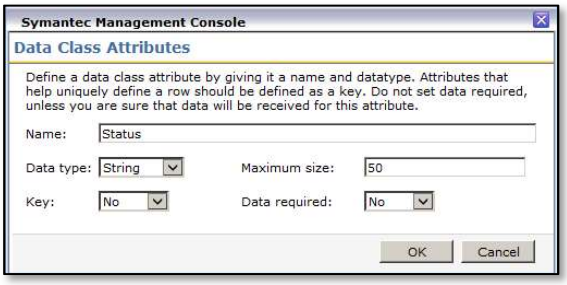


6. Press the **New Data Class** button in the middle pane

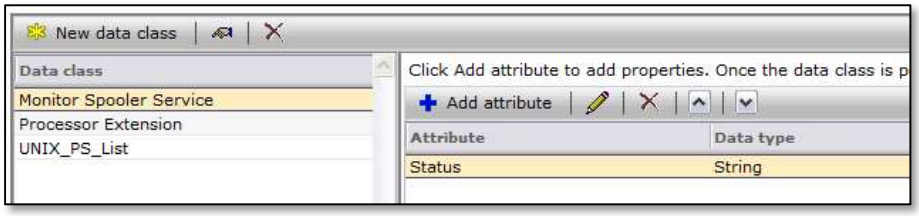
7. Name it **Monitor Spooler Service**
8. Press **OK**
9. Make sure the **Monitor Spooler Service** data class is selected in the left pane
10. Press the **+ Add Attribute** button on the right pane
11. Type **Status** in the Name: field

We do not need a key as we are not storing multiple variants of this service, in this case we only want a one to one relationship, meaning that we only want to know when the item is running or when it is stopped.

12. Ensure that you select **NO** to Key, and **NO** to required like the following:



13. Click **OK**
14. Your managed custom data classes should look like the following now:



BEST PRACTICE: It is very important at this stage to check that every entry is set exactly to what you want – the names, data types, order, etc., cannot be edited once you save the changes. If you make a mistake you will have to delete it off the SQL Server manually and re-enter the entire data class.

15. If you are sure of your Custom Data Class, Press **Save Changes**

    You have now successfully created a data class that you can use for collecting data. Let's look at some of the details behind this data class we just created.

16. You can see details about your data class is by clicking on the properties icon (Beside *New data class)

    This will open a screen showing you the GUID, display name and SQL Table name along with the fields you just created

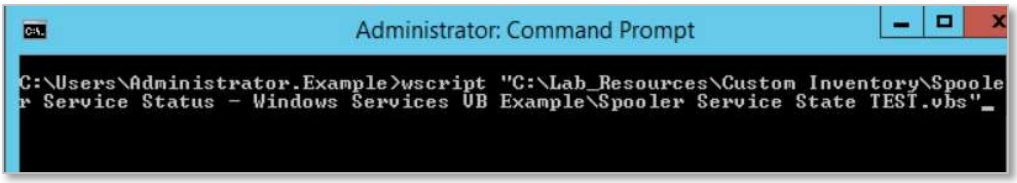
Step 2: Create and Test the Data Gathering Script

Now that all the pieces are in place for storing the data you now need to write a script to collect the information you want, then modify the script to work with the Symantec Management Agent and send data back to the Notification Server.

Please note that initial script testing does not need to be run through a task server or any other scheduled tasks, this should be a manual exercise so that you can slowly watch the data flow through the system and identify coding mistakes.

In this example, we will use VB Script to gather the information we need. A script has been created that will use WMI to query the state of the Windows 'spooler' service.

1. Go to the **SMP** Virtual Machine
2. Open File Explorer and browse to **C:\Lab_Resources\Custom Inventory\Spooler Service Status…**
3. Right Click on the **Spooler Service State TEST.vbs** file and choose **Edit**
4. Read the script, and you can see that this script completes the following:
   - It Opens the WMI DB
   - It Queries WMI for all services Where the service = 'Print Spooler'
   - It Returns the Service State to the screen (wscript.echo …)
5. Close Notepad.
6. You may run the **Spooler Service State TEST.vbs** in a command prompt if you wish to test the script.



7. You should see it return "The Spooler Service is: Running"

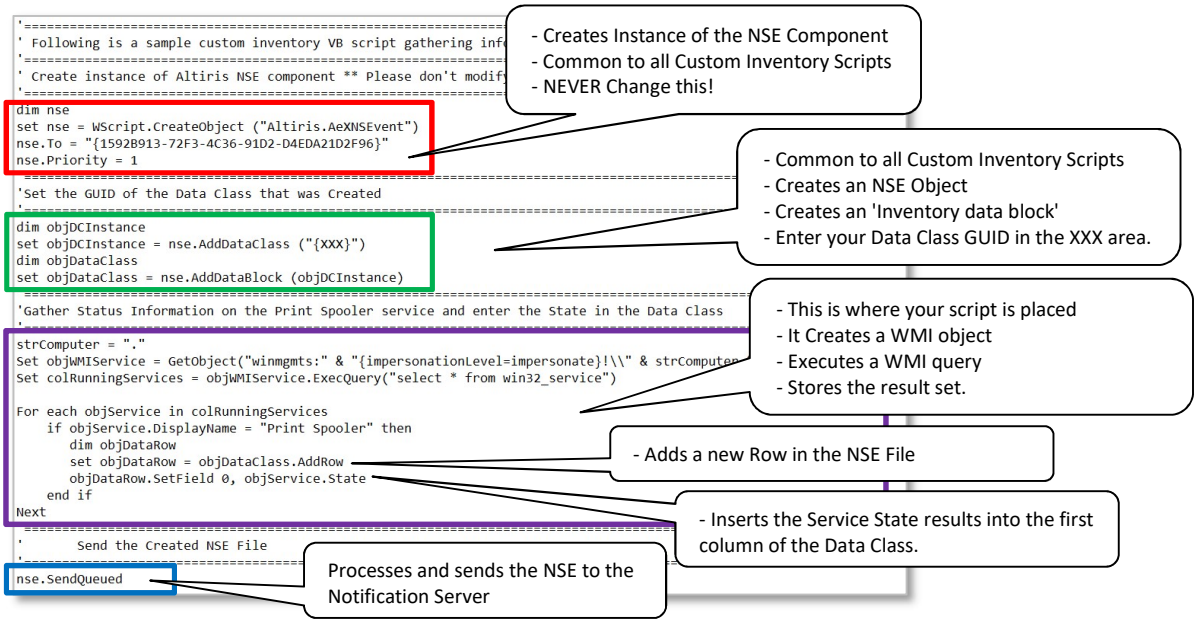
Step 3: Create the Custom Inventory Script

Now we need to create an initial VBScript Task that will be executed on a computer resource. But before we do that you will need to configure a premade script that has been written and modify it so it will collect data and place it in the data class we just created.

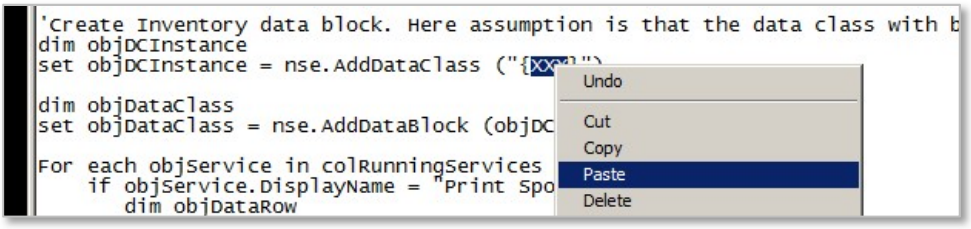
This script was created by taking the base script in the Processor Extension example provided in the default installation of the Symantec Management Platform.

1. In File Explorer, browse to **C:\Lab_Resources\Custom Inventory\Spooler Service Status…**
2. In Notepad, Open **Monitor Spooler Service.txt**

Let's look at the script in Detail:



3. Look for the line with **("{XXX}")** in the script. It will need to be updated with the GUID of the dataclass we just created.
4. In the console, Select **Settings | All Settings** in the main menu
5. In the settings tree, expand **Settings > Notification Server > Data Classes > Inventory > Custom**
6. Right click on the **Monitor Spooler Service** data class and select **Properties**.
7. Highlight the **GUID** on the right pane when it appears, and select **copy**
8. Return to the **Monitor Spooler Service.txt** file and replace the **XXX** letters with the copied GUID. Make sure there are no leading or trailing spaces between the { and }.

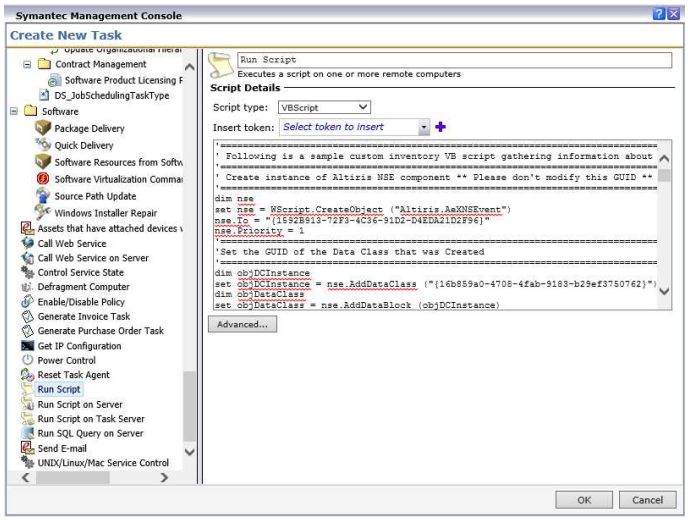


9. Save the script, but keep it open.

Step 4: Create the Custom Inventory Task

1. You are ready to create the task and associate a schedule to it to collect this information
2. From the Symantec Management Console select **Manage | Jobs & Tasks**
3. Expand the **Samples > Discovery & Inventory Inventory Samples > Custom folders**
4. Right Click on the **Custom** folder and select **New > Task**. The Create New Task window appears

5. Scroll down the list on the left pane and select **Run Script**
6. Name the task **Monitor Spooler Service**
7. Select **VBScript** for the Script Type
8. Return to the **Monitor Spooler Service.txt** and copy the contents of the file
9. Return to the **Monitor Spooler Service** task and **Paste** the contents into the blank area.



10. Press **OK**
11. ***Keep the task Open***
12. **We will now stop the Spooler Service**.
    a) Open a Command Prompt and type `net stop spooler` then press enter.
    b) The following will appear indicating that your command executed correctly

```
The Print Spooler service is stopping.
The Print Spooler service was stopped successfully.
```

13. You can test the script on the task against the SMP virutual machine by selecting the **Quick Run** button under **Task Status** and selecting **SMP** as the Resource you want to run this on.
14. Press the **Run** button. It should only take a few seconds for this task to run.
15. Double click on the **Symantec Management Agent** in the tray
16. Select the **Task tab** and notice that the "**Monitor Spooler Service**" task Ran.
17. In the console main menu Select **Manage | Computers**
18. In the Left Pane, Select **Favorites > Lab Computers**
19. Find **SMP** in the list and Double Click on it. The ***Resource Manager*** will open
20. Select **View > Inventory** on the top menu
21. In the middle pane, expand the **Data Classes > Inventory > Custom** folder
22. Select the **Monitor Spooler Service** data class and note that the status indicates **"Stopped"**

23. Close the **Resource Manager** window.

Step 5: Schedule the Custom Inventory Task

Once you have verified that everything is working you can now schedule the script for daily/hourly runs a targeted group of computers. In this case, the Administrator has been asked to target all Windows Servers.

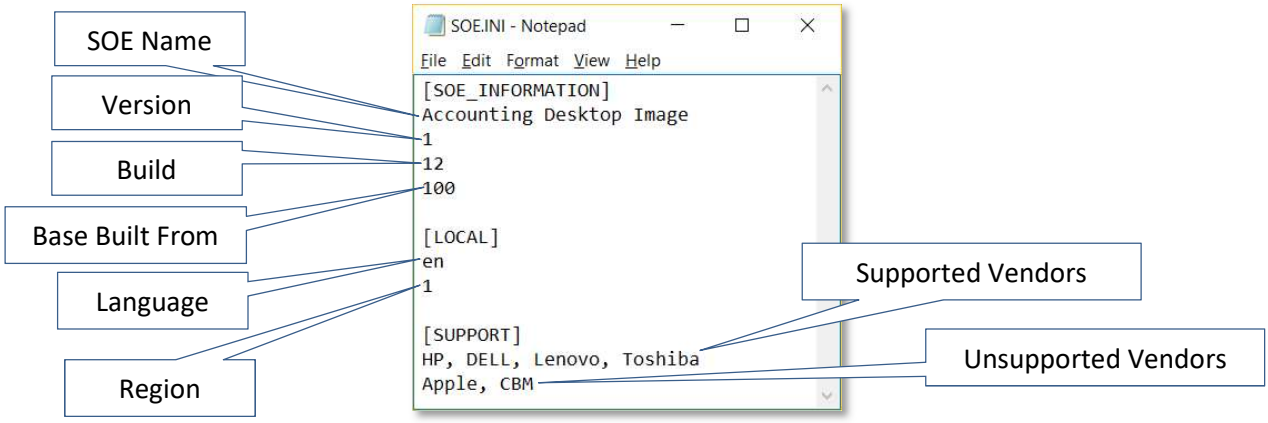
1. Return to the console and select **Manage | Jobs and Tasks**
2. Highlight the **Monitor Spooler Service** task
3. On the right pane, select the **New Schedule** button on the bottom of the pane.
4. In the New Schedule Window select the **Schedule** radio button and select **At date/time** in the dropdown
5. Check the **Repeat every**: box and select **30 Minutes**
6. In the **Selected Devices** area, press the **Add** dropdown list and select **Target**
7. Type **All Windows Servers** in the search field of ***Available Targets*** button and search for and select **All Windows Servers**
8. Press the **'>'** button to add the **"All Windows Servers"** target to the selected targets area.
9. Press **OK**
10. Press **Schedule**

This has demonstrated to you how to create a custom inventory in a windows environment and that it provides a mechanism for collecting custom inventory from a machine and the ability to view the information.

Lab Exercise 2: Reading Values from a TEXT File

In this scenario, the Administrator has been tasked with gathering the OS Image Information from a SOE Image Information file named SOE.INI that is placed on every computer during the staging process. This information will be used to create reports that list this information for future OS Disk Image updates.

The SOE.INI file Information looks like this:



In order to create a Custom Inventory for this requirement it is essential that you complete the following tasks:

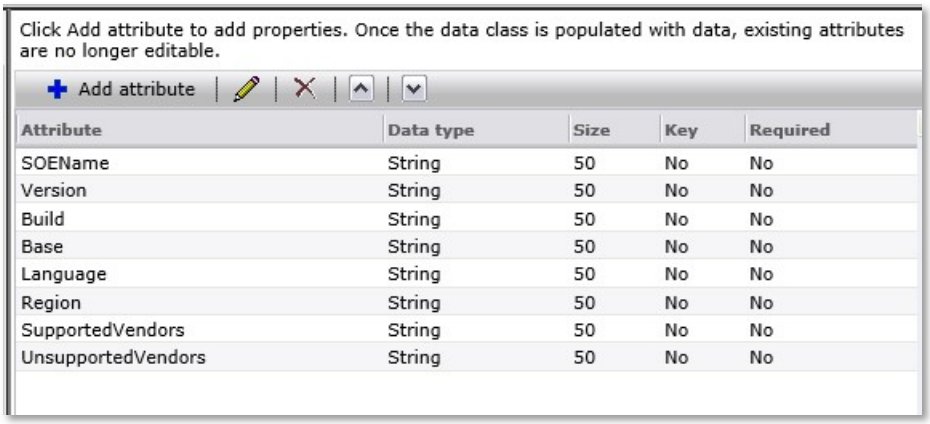
- Create the Data Class to hold the gathered data
- Create and test the script that will be used to gather the data
- Create the Custom Inventory script
- Create the Custom Inventory Task
- Target and Schedule the Custom Inventory task

Step 1: Create the Data Class

To get this data into the CMDB we will need to create a custom data class for Custom Inventory to place the collected data into.

1. Go to the **SMP** Virtual Machine
2. Open the **Symantec Management Console** by clicking the Icon on the Desktop.
3. Select **Settings | All Settings**
4. Expand **Settings > Discovery & Inventory > Inventory Solution**
5. Select **Manage Custom Data Classes**.
6. Press the **New Data Class** button in the middle pane
7. Name it **SOE OS Image Information**
8. Press **OK**
9. Make sure the **SOE OS Image Information** data class is selected

10. Press the **+ Add Attribute** button on the right pane
11. Type **SOEName** in the **Name:** field
12. Keep all the fields as default and ensure that you select **NO** to **Key,** and **NO** to **Required**.
13. Press **OK**
14. Repeat Steps 10 to 12 for the following Fields:
    a) Version
    b) Build
    c) Base
    d) Language
    e) Region
    f) SupportedVendors
    g) UnsupportedVendors
15. Your managed custom data classes should look like the following now:


Click Add attribute to add properties. Once the data class is populated with data, existing attributes are no longer editable.

Add attribute

Attribute	Data type	Size	Key	Required
SOEName	String	50	No	No
Version	String	50	No	No
Build	String	50	No	No
Base	String	50	No	No
Language	String	50	No	No
Region	String	50	No	No
SupportedVendors	String	50	No	No
UnsupportedVendors	String	50	No	No

16. If you are sure of your Custom Data Class, Press **Save Changes**
17. You can look at details about your data class is by clicking on the properties icon

Step 2: Create and Test the Data Gathering Script

In this example, we will use VB Script to gather the information we need. A script has been created that will Open the C:\Windows\SOE.INI on each computer and return the values it finds.

1. Go to the **SMP** Virtual Machine
2. Open File Explorer and browse to **C:\Lab_Resources\Custom Inventory\OS Image Revision - TXT File VB Example…**
3. Right Click on the **READ SOE FILE.vbs** file and choose **Edit**
4. Read the script, and you can see that this script completes the following:
    a) It Opens the C:\Windows\SOE.INI file and creates an object

b) It loops through every line and stores the information

c) It Returns the stored lines requested and shows them on the screen (wscript.echo …)

d) Notice that Lines 1,2,3,4,7,8,11 and 12 are returned, as we don't need the other lines.

5. Close Notepad.
6. You may run **`WSCRIPT "C:\Lab_Resources\Custom Inventory\OS Image Revision - TXT File VB Example READ SOE FILE.vbs"`** in a command prompt if you wish to test the script.
7. You should see it return the 8 lines of data.

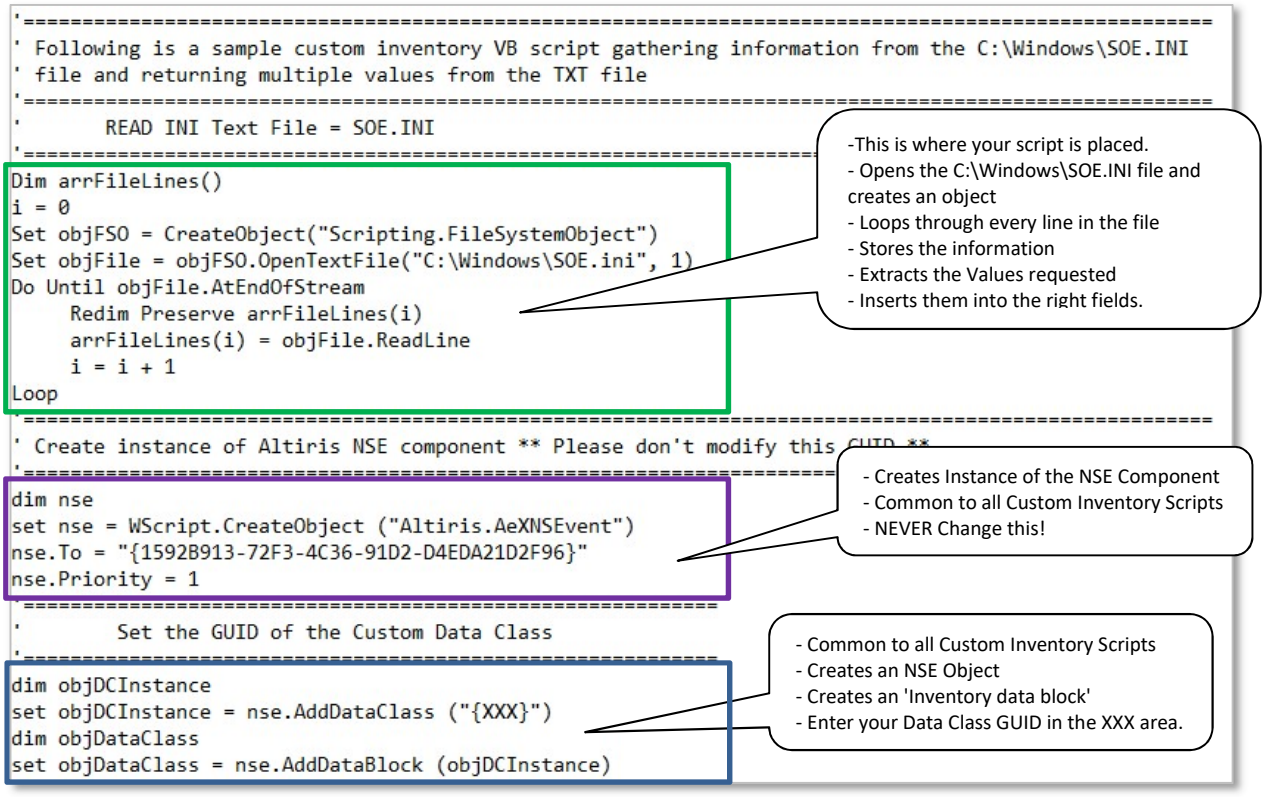
Step 3: Create the Custom Inventory Script

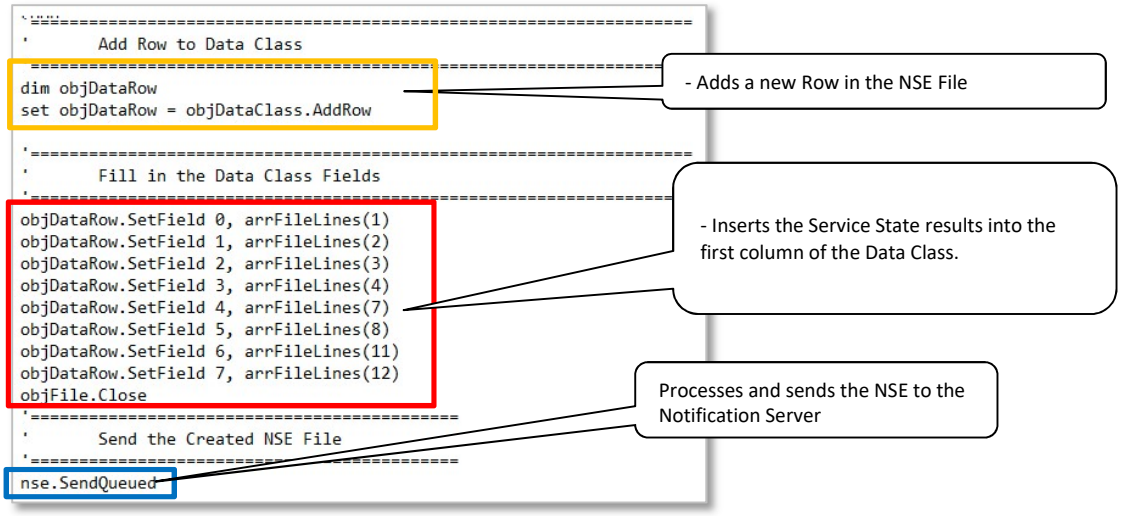
Now we need to create an initial Script Task that will be executed on a computer resource. But before we do that you will need to configure a premade script that has been written and modify it so it will collect data and place it in the data class we just created.

This script was created by taking the base script in the Processor Extension example provided in the default installation of the Symantec Management Platform.

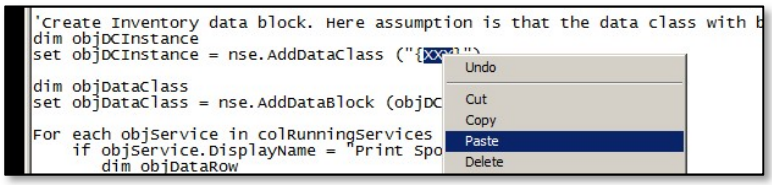
1. In File Explorer, browse to **C:\Lab_Resources\Custom Inventory\OS Image Revision - TXT File VB Example**
2. In Notepad, Open **SOE Image Information - Script.txt**

Let's look at the script in Detail:





3. Look for the line with **("{XXX}")** in the script. It will need to be updated with the GUID of the dataclass we just created.
4. In the console, Select **Settings | All Settings** in the main menu
5. In the settings tree, expand **Settings > Notification Server > Data Classes > Inventory > Custom**
6. Right click on the **SOE OS Image Information** data class and select **Properties**.
7. Highlight the **GUID** on the right pane when it appears, and select **copy**
8. Return to the **SOE Image Information - Script.txt** file and replace the **XXX** letters with the copied GUID. Make sure there are no spaces between the { and }.



9. Save the script, but keep it open.

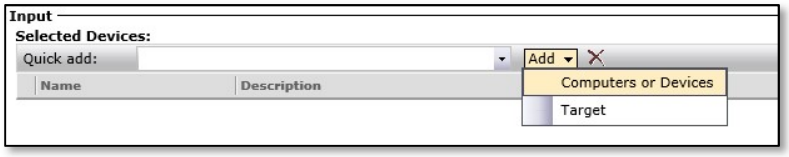
Step 4: Create the Custom Inventory Task

1. You are ready to create the task and associate a schedule to it to collect this information
2. From the Symantec Management Console select **Manage | Jobs & Tasks**
3. Expand the **Samples > Discovery & Inventory Inventory Samples > Custom** folders
4. Right Click on the **Custom** folder and select **New Task**. The Create New Task window appears
5. Scroll down the list on the left pane and select **Run Script**
6. Name the task **SOE Image Information**
7. Select **VBScript** for the **Script Type**
8. Return to the **SOE Image Information - Script.txt** and copy the contents of the file

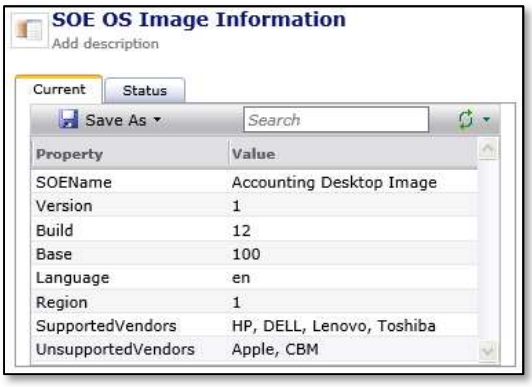
9. Return to the **SOE Image Information** task and **Paste** the contents into the blank area.
10. Press **OK**
11. Keep the task Open

Step 5: Distribute the Custom Inventory Task

1. Return to the console and select **Manage | Jobs and Tasks**
2. Highlight the **SOE Image Information** task
3. On the right pane, select the **New Schedule** button on the bottom of the pane.
4. In the New Schedule Window select the **Now** radio button.
5. In the **Selected Devices** area, press the **Add** dropdown list and select **Computers & Devices**



6. Select **SMP** and **Win8** from the available computers list (Using CTRL – Left Mouse) and press the > symbol to insert them into the Selected Computers List.
7. Press **OK**
8. Press the **Schedule** button
9. The task should run on SMP and Win8 in less than a minute.
10. You may verify this by opening the Symantec Management Console and selecting **Manage | Computers**
11. Double Click on **SMP** or **Win8** in the middle (list) pane
12. In the Resource Manager, Select **View > Inventory**
13. In the data class view, expand **Data Classes > Inventory > Custom** and select **SOE OS Image Information**
14. You should see data returned to the CMDB that looks something like this…



15. If you do not see data, go to the target computer and open the Symantec Management Agent and press the Update Configuration button to start the task.

Lab Exercise 3: Reading Multiple Values from WMI

In this scenario, the Desktop Engineering Team has been tasked with the Windows 10 Migration Project. Before beginning a migration process, it is very important to gather the encryption state of the Hard Drives that will be migrated from Windows 7 and Windows 8.1 as the OS Imaging process will not capture an encrypted drive.

They know that Custom inventory can be used to detect the encryption status of each drive on a computer by querying WMI. They also need to capture not only the encryption state of every drive, but some of the additional properties available to report on the conversion status and encryption percentage if it is in the process of encrypting the drive(s).

BEST PRACTICE: It is very advantageous to use a tool like WMI Explorer that will walk you through WMI and even provide you with sample scripts to Query WMI with in VB Script and PowerShell.

In order to create a Custom Inventory for this requirement it is essential that you complete the following tasks:

- Create the Data Class to hold the gathered data
- Create the Custom Inventory script
- Create the Custom Inventory Task
- Target and Schedule the Custom Inventory task
- Create a Report that will be used to seek out computers with encrypted drives.

Step 1: Create the Data Class

To get this data into the CMDB we will need to create a custom data class for Custom Inventory to place the collected data into.

1. Go to the **SMP** Virtual Machine
2. Open the **Symantec Management Console** by clicking the Icon on the Desktop.
3. Select **Settings | All Settings**
4. Expand **Settings > Discovery & Inventory > Inventory Solution**
5. Select **Manage Custom Data Classes**.
6. Press the **New Data Class** button in the middle pane
7. Name it **Bitlocker Status**
8. Press **OK**
9. Make sure the **Bitlocker Status** data class is selected
10. Press the **+ Add Attribute** button on the right pane
11. Type **Drive** in the **Name:** field
12. Keep all the fields as default and ensure that you select **NO** to **Key,** and **NO** to **Required**.
13. Press **OK**
14. Repeat Steps 10 to 12 for the following items:
    a) EncryptionMethod

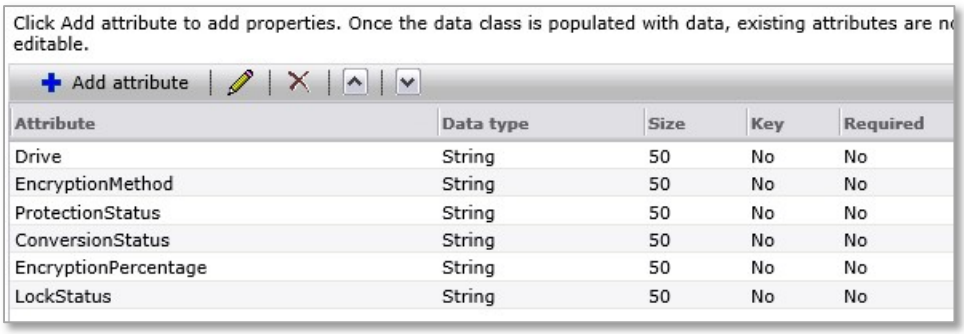
b) ProtectionStatus

c) ConversionStatus

d) EncryptionPercentage

e) LockStatus

15. Your managed custom data classes should look like the following now:



Attribute	Data type	Size	Key	Required
Drive	String	50	No	No
EncryptionMethod	String	50	No	No
ProtectionStatus	String	50	No	No
ConversionStatus	String	50	No	No
EncryptionPercentage	String	50	No	No
LockStatus	String	50	No	No

16. **IMPORTANT STEP!:** Scroll to the very bottom of the left pane and enable (check) the "**Allow multiple rows from a single computer resource"**

This setting is enabled as this Custom Inventory can return more than one row of data (Drive C:\, D:\...)

17. If you are sure of your Custom Data Class, Press **Save Changes**

Step 2: Create the Custom Inventory Script

Now we need to create an initial Script Task that will be executed on a computer resource. But before we do that you will need to configure a premade script that has been written and modify it so it will collect data and place it in the data class we just created.

This script was created by taking the base script in the Processor Extension example provided in the default installation of the Symantec Management Platform.

1. In File Explorer, browse to **C:\Lab_Resources\Custom Inventory\Bitlocker Status - WMI Call VB Example**
2. In Notepad, Open **Bitlocker Status Script.txt**

   As you are now familiar with the standard parts of a Custom inventory, we will concentrate on the WMI scripting portion. This portion calls WMI to gather the following information from each Drive:

   - **Encryption Method**
   - **Protection Status**
   - **Conversion Status**
   - **Lock Status**

```
'======================================================================
' The Following is a sample custom inventory VB script gathering
' information from WMI regarding the status of Bitlocker in Windows
' and returning multiple values from the query
'======================================================================
'Call WMI for encryption information
'======================================================================
On Error Resume Next

 strComputer = "."
 Set objWMIService = GetObject("winmgmts:\\" & strComputer & "\root\CIMV2\Security\MicrosoftVolumeEncryption")
 Set colItems = objWMIService.ExecQuery("SELECT * FROM Win32_EncryptableVolume",,48)

 Dim arEncryptionMethod
 Dim arProtectionStatus
 Dim arConversionStatus
 Dim arLockStatus

 arEncryptionMethod = Array("None", "AES 128 With Diffuser", "AES 256 With Diffuser", "AES 128", "AES 256")
 arProtectionStatus = Array("Protection Off", "Protection On", "Protection Unknown")
 arConversionStatus = Array("Fully Decrypted", "Fully Encrypted", "Encryption In Progress", "Decryption In Progress"
 arLockStatus = Array("Unlocked", "Locked")
```

This portion Adds the results from WMI and for each row returned places the following information into the NSE file:

- **The Drive Letter in to "Drive"**
- **The Encryption Method to "EncryptionMethod"**
- **Protection Status to "ProtectionStatus"**
- **Conversion Status to "ConversionStatus"**
- **Encryption Percentage (if in progress) to "EncryptionPercentage"**
- **Lock Status to "LockStatus"**

```
'======================================================================
'Populate the NSE file with desired Bitlocker data
'======================================================================
 For Each objItem in colItems
 'Add a new row for each drive on the computer
 Dim objDataRow
 set objDataRow = objDataClass.AddRow

 Dim EncryptionMethod
 Dim ProtectionStatus
 Dim ConversionStatus
 Dim EncryptionPercentage
 Dim VolumeKeyProtectorID
 Dim LockStatus

 objItem.GetEncryptionMethod EncryptionMethod
 objItem.GetProtectionStatus ProtectionStatus
 objItem.GetConversionStatus ConversionStatus, EncryptionPercentage
 objItem.GetKeyProtectors 0,VolumeKeyProtectorID
 objItem.GetLockStatus LockStatus

 objDataRow.SetField 0, objItem.DriveLetter
 objDataRow.SetField 1, arEncryptionMethod(EncryptionMethod)
 If arProtectionStatus(ProtectionStatus) = "Protection On" then
 objDataRow.SetField 2, "1"
 ElseIf arProtectionStatus(ProtectionStatus) = "Protection Off" then
 objDataRow.SetField 2, "0"
 End If
 objDataRow.SetField 3, arConversionStatus(ConversionStatus)
 objDataRow.SetField 4, arEncryptionPercentage(EncryptionPercentage)
 objDataRow.SetField 5, arLockStatus(LockStatus)
```

3. Look for the line of text that is below the "Enter the guid shown in the properties…" line
4. This line will have a GUID that was previously entered ({***4bf741…***). It will need to be updated with the GUID of the dataclass we just created.

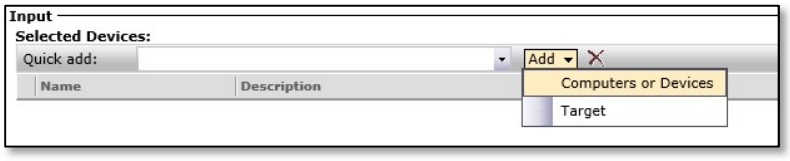
5. In the console, Select **Settings | All Settings** in the main menu
6. In the settings tree, expand **Settings > Notification Server > Data Classes > Inventory > Custom**
7. Right click on the **Bitlocker Status** data class and select **Properties**.
8. Highlight the **GUID** on the right pane when it appears, and select **copy**
9. Return to the **Bitlocker Status Script.txt** file and replace the GUID in the line that was previously entered ({***4bf741…***) with the copied GUID. Make sure there are no spaces between the { and }.
10. Save the script, but keep it open.

Step 3: Create the Custom Inventory Task

1. You are ready to create the task and associate a schedule to it to collect this information
2. From the Symantec Management Console select **Manage | Jobs & Tasks**
3. Expand the **Samples > Discovery & Inventory Inventory Samples > Custom** folders
4. Right Click on the **Custom** folder and select **New Task**. The Create New Task window appears
5. Scroll down the list on the left pane and select **Run Script**
6. Name the task **Bitlocker Status for All Drives**
7. Select **VBScript** for the Script Type
8. Return to the **Bitlocker Status Script.txt** and copy the contents of the file
9. Return to the **Bitlocker Status for All Drives** task and **Paste** the contents into the blank area.
10. Press **OK**
11. Keep the task Open

Step 4: Distribute the Custom Inventory Task

1. Return to the console and select **Manage | Jobs and Tasks**
2. Highlight the **Bitlocker Status for All Drives** task
3. On the right pane, select the **New Schedule** button on the bottom of the pane.
4. In the New Schedule Window select the **Now** radio button.
5. In the **Selected Devices** area, press the **Add** dropdown list and select **Computers & Devices**



6. Select **Win8** from the available computers list and press the **>** symbol to insert them into the Selected Computers List. You can also creat a target that lists all windows workstations (as bitlocker only applies to non-server Windows operating systems.
7. Press **OK**

8. Press the **Schedule** button
9. The task should run on **Win8** in less than a minute.
10. You may verify this by opening the Symantec Management Console and selecting **Manage | Computers**
11. Double Click on **Win8** in the middle (list) pane
12. In the Resource Manager, Select **View > Inventory**
13. In the data class view, expand **Data Classes > Inventory > Custom** and select **Bitlocker Status**
14. You should see data returned to the CMDB that looks something like this…

15. If you do not see data, go to the target computer and open the Symantec Management Agent and press the Update Configuration button to start the task. You can also check that the GUID you replaced in the Run Script is correct.

Step 5: Reporting the Custom Inventory Data

The Administrator was asked to create a report that shows the results of gathering the Bitlocker Status Information from all computers. This report has been completed and is in txt format.

1. Return to the console and select **Reports | All Reports** in the **Main Menu**
2. Right Click on the **Reports** item at the top of the report tree and Select **New > Folder**
3. Name the folder **Custom Inventory Reports**
4. Right Click on the **Custom Inventory Reports** folder and select **Import**
5. Press the **Browse** button and browse to **C \Lab_Resources\Custom Inventory\Bitlocker Status - WMI Call VB Example*Bitlocker Status Report.xml***
6. Press **Open**
7. Press **OK**
8. You will now see the **Bitlocker Status Report** in the **Custom Inventory Reports** folder
9. Select the **Bitlocker Status Report** to see the results.
10. Notice that Operating System Name, BIOS Manufacturer and BIOS version exist in additon to the Custom Inventory values that were returned.

11. Press the EDIT button on the top right of the report. You may investigate the SQL Query to see how it returns the data:

The Query starts by setting up the columns to return

```
SELECT vComputer.Name, BLS.Drive, BLS.ProtectionStatus,
BLS.EncryptionMethod, BLS.ConversionStatus, BLS.EncryptionPercentage,
BLS.LockStatus, vComputer.[OS Name],
Inv_SW_BIOS_Element.Manufacturer AS [BIOS MFR],
Inv_SW_BIOS_Element.Version AS [BIOS Version]
```

The Query joins the BOLDED data classes to return the columns

```
FROM Inv_Bitlocker_Status AS BLS
LEFT OUTER JOIN vComputer ON vComputer.Guid = BLS._ResourceGuid
LEFT OUTER JOIN Inv_SW_BIOS_Element ON Inv_SW_BIOS_Element._ResourceGuid
= BLS._ResourceGuid
```

Then it excludes any Server class machine from the Report (In case the Custom Inventory ran on them)

```
WHERE  (NOT (vComputer.[OS Name] LIKE N'%Server%'))
```

NOTE: This Custom Inventory report results will show a 1 for each encrypted drive and a 0 for not-encrypted drives. If the drive shows 'NULL' then Bitlocker has not been installed on that computer and thus the WMI query returned no data.

Lab Exercise 4: Using PowerShell Script as a IOC Hunter for File Hashes

THIS EXERCISE IS A FICTITIOUS EXAMPLE OF A VULNERABLE SOFTWARE INSTALLATION AND APPLICATION – WE ARE NOT IMPLYING THAT BGINFO IS IN ANYWAY A VULNERABLE APPLICATION.

Preface:

Simply put, an IOC (indicator of compromise) a list of threat data (e.g., strings defining file paths or registry keys) which can be used to detect a threat in the infrastructure using automated software-based analysis.

Simple IOC usage scenarios involve searching the system for specific files using a variety of search criteria: hashes, file names, creation dates, sizes and other attributes. Additionally, memory can be searched for various signs specific to the threat and the Windows registry can be searched for specific records.

Scenario:

A recent IOC report was released after an investigation of the BGInfo Utility. Page 36 of the report lists the MD5 and other File hashes of all malware components that may be present in the installation as a result of this infection. Organizations need to inventory the software found in the installation directory and compare it with the known infected file hashes in this report.

In this scenario, the InfoSec Team has been tasked with the job of finding multiple File Hashes for the contents of the BGInfo Installation Directory so that they can be compared to known infected hashes using various forensics and security tools. They are also tasked with creating a report that will show different File Hash algorithm types for each file in the installation directory:

They know that Custom inventory can be used to find and return data on the files in a specific directory using a PowerShell script, and can gather many hash types of specific files using the Get-File Hash command. Their task is to return the following hashes of each file in the C:\SES_Temp\utilities\BGinfo\ directory where it was placed during staging. They must gather the following hash algorithm types for each file:

- SHA1
- SHA256
- SHA384
- SHA512
- MACTripleDES
- MD5
- RIPEMD160

In order to create a Custom Inventory for this requirement it is essential that you complete the following tasks:

- Create the Data Class to hold the gathered data
- Create the Custom Inventory script
- Create the Custom Inventory Task
- Target and Schedule the Custom Inventory task
- Create a Report that will be used to seek out computers with encrypted drives.

Step 1: Create the Data Class

The InfoSec team has decided that the required data for the Data Class would be to return the Algorithm Type, Hash and Full Path of the File for use in their search for infected files. They have already created the data class on their Test Notification Server and have provided an Import File to save time.

To get this data into the CMDB we will need to Import the previously created data class so Custom Inventory will have something to put data into.

1. Go to the **SMP** Virtual Machine
2. Open the **Symantec Management Console** by clicking the Icon on the Desktop.
3. Open the Symantec Management Console and select **Settings > All Settings**
4. Navigate to the **Settings > Notification Server > Resource and Data Class Settings > Data Classes > Inventory > Custom**
5. Right-click on the **Custom** folder and select **Import**
6. Press the **Browse** button, and browse to **C:\Lab_Resources\Custom Inventory\IOC Hunter - PowerShell Example**
7. Select **IOC File Hashes – Data Class.xml**, then press **Open**
8. Press **OK**

   The Data Class is imported as an "Editable Data Class". This type will not be available in the **"Manage Custom Data Classes"** settings page, and can be treated as a standard data class in the CMDB. Since it is an editable data clsss, it must be added to a resource type. In this case the "Computer" resource type is appropriate.
9. To add the data class to the Computer resource type:
   a) Navigate to the **Settings > Notification Server > Resource and Data Class Settings > Resource Types > Asset Types > IT**
   b) Select the **Computer** icon. Wait until the settings appear
   c) Scroll down the page and Press the **Add Data Classes** link
   d) Select the **IOC File Hashes** data class under the **Inventory > Custom** folder, and press **OK**

   e) Scroll to the bottom of this window and press the **Save Changes** button

Step 2: Create the Custom Inventory Script

Now we need to create an initial Script Task that will be executed on a computer resource. But before we do that you will need to configure a premade script that has been written and modify it so it will collect data and place it in the data class we just created.

This script was created by taking the base script in the Processor Extension example provided in the default installation of the Symantec Management Platform.

1. In File Explorer, browse to **C:\Lab_Resources\Custom Inventory\IOC Hunter - PowerShell Example**
2. In Notepad, Open **IOC Hunter - BGInfo Script.txt**

   As you are now familiar with the standard parts of a Custom inventory, we will concentrate on the Powershell Hash Data scripting portion.

```
#=============================================================================================================
#Gather Hash Information on all files in the specified directory
#=============================================================================================================
$Path = 'C:\SES_Temp\utilities\BGinfo\*.*'
$HashDataSet = Gci $Path | % { Get-FileHash $_.Fullname -Algorithm MD5; Get-FileHash $_.Fullname -Algorithm SHA1; Get-FileHash $_.Fullname
-Algorithm SHA256; Get-FileHash $_.Fullname -Algorithm SHA384; Get-FileHash $_.Fullname -Algorithm SHA512; Get-FileHash $_.Fullname -
Algorithm MACTripleDES; Get-FileHash $_.Fullname -Algorithm RIPEMD160}
```

   This portion calls Powershell to go to the file path indicated in the **$Path** variable (**C:\SES_Temp\utilities\BGInfo*.***) and generate many hash types then return the following information:

   - **Algorithm Type**
   - **Resultant Hash**
   - **Full Path of the File**

   Running this Script portion will return data in this format:

Algorithm	Hash	Path
MD5	0D70B3218B3B810B769A4A45A4E93700	C:\SES_Temp\utilities\BGinfo\Bginfo.exe
SHA1	D654CA14D258886E70D64346A6CC211B4B1839B4	C:\SES_Temp\utilities\BGinfo\Bginfo.exe
SHA256	112ECA16A54474AB97D5DF2C23C3AEE9760978A8355C8B2EE92706B2248ABEB4	C:\SES_Temp\utilities\BGinfo\Bginfo.exe
SHA384	6579DECB44432C8FBD0678D089A182F9BC5E2B644AB82BF8A2DCD33456B4AD42776...	C:\SES_Temp\utilities\BGinfo\Bginfo.exe
SHA512	F9ECD306DFDD27593772A7A5B5CE58C682EFAB62756788829A889CB22D8F806AC02...	C:\SES_Temp\utilities\BGinfo\Bginfo.exe
MACTRIPLEDES	60EA5093254C9788	C:\SES_Temp\utilities\BGinfo\Bginfo.exe
RIPEMD160	B9E8F92CECE1A1844E0ED2077F94A194D15694C5	C:\SES_Temp\utilities\BGinfo\Bginfo.exe
MD5	71C73451CB0242B4776B84623330119A	C:\SES_Temp\utilities\BGinfo\BG_Info.bat
SHA1	944B8F0E4458CA97C829B0B8370E71C446696C27	C:\SES_Temp\utilities\BGinfo\BG_Info.bat
SHA256	9A9B1A7B2E81CB6B4A8B99C17C0A8F540348BE11F0A7D276F3D65F701D6F2BB6	C:\SES_Temp\utilities\BGinfo\BG_Info.bat
SHA384	E1E71C903B5F5BB37DD3A091A605A0A04B111D9F4EAB33C0765AF0E3C7E65399D1D...	C:\SES_Temp\utilities\BGinfo\BG_Info.bat
SHA512	0DAEED54630A99740185CA7AD50F781341B39796194E065330204879AD56FAB4A65...	C:\SES_Temp\utilities\BGinfo\BG_Info.bat
MACTRIPLEDES	3FDE7542CC0ED45F	C:\SES_Temp\utilities\BGinfo\BG_Info.bat
RIPEMD160	E7752C8F9242B8445ED9839F7A32395654569FB0	C:\SES_Temp\utilities\BGinfo\BG_Info.bat
MD5	08B0B80DC7C8F89CCB53EA0342DE04BA	C:\SES_Temp\utilities\BGinfo\Enable_BGinfo_...

   This portion adds the results from the above script and for each row returned and places the following information into the NSE file:

   - **Algorithm to "Algorithm" (MD5, SHA1, SHA256…)**
   - **Hash to 'Hash'**
   - **Path to "Path'**

```
#=============================================================================================================================
#Process each row returned and enter the Algorithm, Hash and Path for each file specified in the directory
#=============================================================================================================================
ForEach ($Hash in $HashDataSet)
{
        #Add new row of data
        $objDataRow = $objDataClass.AddRow()

        #popluate columns
        $objDataRow.SetField(1,$Hash.Algorithm)
        $objDataRow.SetField(2,$Hash.Hash)
        $objDataRow.SetField(3,$Hash.Path)
}
```

3. Look for the line with **("{XXX}")** in the script. It will need to be updated with the GUID of the dataclass we just created.
4. In the console, Select **Settings | All Settings** in the main menu
5. In the settings tree, expand **Settings > Notification Server > Data Classes > Inventory > Custom**
6. Right click on the **IOC File Hashes** data class and select **Properties**.
7. Highlight the **GUID** on the right pane when it appears, and select **copy**
8. Return to the **IOC Hunter - BGInfo Script.txt** file and replace the **XXX** letters with the copied GUID. Make sure there are no spaces between the { and }.
9. Save the script, but keep it open.

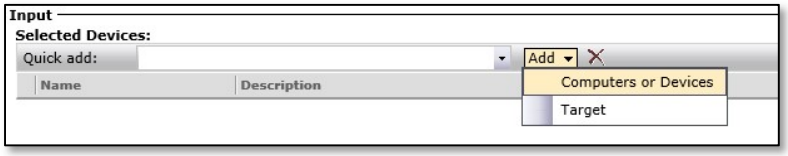
Step 3: Create the Custom Inventory Task

1. You are ready to create the task and associate a schedule to it to collect this information
2. From the Symantec Management Console select **Manage | Jobs & Tasks**
3. Expand the **Samples > Discovery & Inventory Inventory Samples > Custom** folders
4. Right Click on the **Custom** folder and select **New Task**. The Create New Task window appears
5. Scroll down the list on the left pane and select **Run Script**
6. Name the task **IOC: Capture Hash of Files in a Directory**
7. ***IMPORTANT STEP!:*** Select **PowerShell** for the Script Type
8. Return to the **IOC Hunter - BGInfo Script.txt** and copy the contents of the file
9. Return to the **IOC: Capture Hash of Files in a Directory** task and **Paste** the contents into the blank area.
10. Press **OK**
11. Keep the task Open

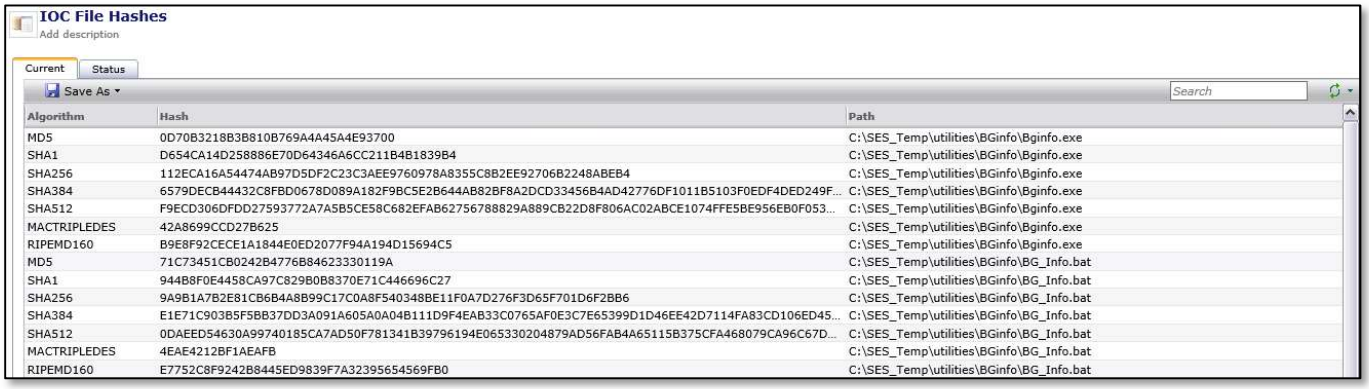
Step 4: Distribute the Custom Inventory Task

1. Return to the console and select **Manage | Jobs and Tasks**
2. Highlight the **IOC: Capture Hash of Files in a Directory** task
3. On the right pane, select the **New Schedule** button on the bottom of the pane.

4. In the New Schedule Window select the **Now** radio button.
5. In the **Selected Devices** area, press the **Add** dropdown list and select **Computers & Devices**



6. Select **SMP and Win8** from the available computers list and press the **>** symbol to insert them into the Selected Computers List. You can also create a target that lists all windows computers if you wanted to run this on many computers.
7. Press **OK**
8. Press the **Schedule** button
9. The task should run on SMP and Win8 in less than a minute.
10. You may verify this by opening the Symantec Management Console and selecting **Manage | Computers**
11. Double Click on **SMP or Win8** in the middle (list) pane
12. In the Resource Manager, Select **View > Inventory**
13. In the data class view, expand **Data Classes > Inventory > Custom** and select **IOC File Hashes**
14. You should see data returned to the CMDB that looks something like this…



Algorithm	Hash	Path
MD5	0D70B3218B3B810B769A4A45A4E93700	C:\SES_Temp\utilities\BGinfo\Bginfo.exe
SHA1	D654CA14D258886E70D64346A6CC211B4B1839B4	C:\SES_Temp\utilities\BGinfo\Bginfo.exe
SHA256	112ECA16A54474AB97D5DF2C23C3AEE9760978A8355C8B2EE92706B2248ABEB4	C:\SES_Temp\utilities\BGinfo\Bginfo.exe
SHA384	6579DECB44432C8FBD0678D089A182F9BC5E2B644AB82BF8A2DCD33456B4AD42776DF1011B5103F0EDF4DED249F...	C:\SES_Temp\utilities\BGinfo\Bginfo.exe
SHA512	F9ECD306DFDD27593772A7A5B5CE58C682EFAB62756788829A889CB22D8F806AC02ABCE1074FFE5BE956EB0F053...	C:\SES_Temp\utilities\BGinfo\Bginfo.exe
MACTRIPLEDES	42A8699CCD27B625	C:\SES_Temp\utilities\BGinfo\Bginfo.exe
RIPEMD160	B9E8F92CECE1A1844E0ED2077F94A194D15694C5	C:\SES_Temp\utilities\BGinfo\Bginfo.exe
MD5	71C73451CB0242B4776B84623330119A	C:\SES_Temp\utilities\BGinfo\BG_Info.bat
SHA1	944B8F0E4458CA97C829B0B8370E71C446696C27	C:\SES_Temp\utilities\BGinfo\BG_Info.bat
SHA256	9A9B1A7B2E81CB6B4A8B99C17C0A8F540348BE11F0A7D276F3D65F701D6F2BB6	C:\SES_Temp\utilities\BGinfo\BG_Info.bat
SHA384	E1E71C903B5F5BB37DD3A091A605A0A04B111D9F4EAB33C0765AF0E3C7E65399D1D46EE42D7114FA83CD106ED45...	C:\SES_Temp\utilities\BGinfo\BG_Info.bat
SHA512	0DAEED54630A99740185CA7AD50F781341B39796194E065330204879AD56FAB4A65115B375CFA468079CA96C67D...	C:\SES_Temp\utilities\BGinfo\BG_Info.bat
MACTRIPLEDES	4EAE4212BF1AEAFB	C:\SES_Temp\utilities\BGinfo\BG_Info.bat
RIPEMD160	E7752C8F9242B8445ED9839F7A32395654569FB0	C:\SES_Temp\utilities\BGinfo\BG_Info.bat

15. If you do not see data, go to the target computer and open the Symantec Management Agent and press the **Update Configuration** button to start the task. You may also want to investigate that the GUID is correct in the script and that the { and } surrond it…

Step 5: Reporting the Custom Inventory Data

The InfoSec team was asked to create a report that shows the results of the file hashes for all of the files in the BGInfo directory. This report has been created and is in xml format to be imported into the NS.

1. Return to the console and select **Reports | All Reports**
2. If you have already created a "**Custom Inventory Reports**" Folder, skip to step 5
3. Right Click on the **Reports** item at the top of the report tree and Select **New > Folder**
4. Name the folder **Custom Inventory Reports**
5. Right Click on the **Custom Inventory Reports** folder and select Import
6. Press the Browse button and browse to **C:\Lab_Resources\Custom Inventory\IOC Hunter - PowerShell Example\IOC Hunter Report - BGInfo.xml**
7. Press **Open**
8. Press **OK**
9. You will now see the **IOC Hunter Report - BGInfo** report in the Custom Inventory Reports folder
10. Select the **IOC Hunter Report - BGInfo** Report to see the results. Press **Run** report if there is no data in the report.
11. Notice that Name of the Computer is in this report as well as the Algorithm, Hash and Path of the file(s)
12. Press the EDIT button on the top right of the report. You may investigate the SQL Query to see how it returns the data. This is a complex report that includes the syntax for resource scoping and advanced grouping to display the contents for use in a secure environment.
13. This report can be used to compare the hashes against the entire resultant set. For example, if you want the IOC report to show you all MD5 Hashes for BGInfo.EXE so you can look for anomolies, then you can do the following on the report that is displayed:
    a) Select **Path** in the **Group By** dropdown list on the top right of the report
    b) Type **MD5** in the search field on the top right of the report
    c) Expand the **Path: C:\SES_Temp\BGInfo\BGInfo.exe** group on the left side. This will return a list of all Computers with BGInfo.exe and its calculated MD5 File Hash. You notice that the MD5 Hashes are the same and there are no anomolies.

In the picture below, you can see an example of an anomaly that shows that SMP8 has returned an invalid hash for the BGInfo.exe file and should be investigated immediately.