EY Script documentation

Software Inventory v4.0.8

October 2017

Building a better working world

Table of contents

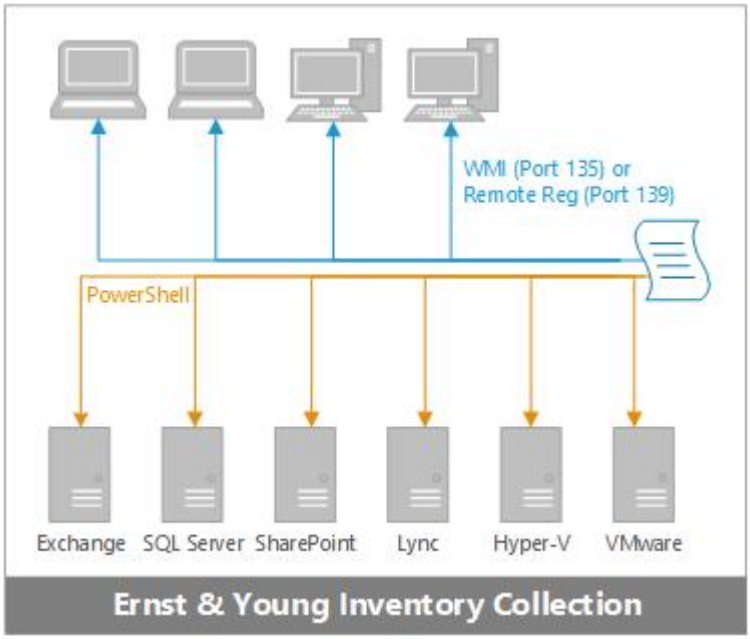
1. Executive summary

As part of an EY software review, we use a small software tool (a tool) to obtain information about installed software on one or more systems.

This document describes the detailed functionality of this tool and should be read before using the tool in your computer environment.

The tool functions by querying systems for installed software and configuration information though Windows Management Instrumentation (WMI), Windows Remote Registry service, PowerShell Commands and SQL queries.

The tool does not scan the file system nor does it write anything on the remote systems. By default, the tool will not scan systems under heavy load, to prevent performance degradation of heavily utilized systems.



Data is only read from the remote systems, writing only takes place on the central system from which the tool is executed, and the data collected is less than 100KB per system, the total network traffic is ~1MB-4MB per scanned system on average.

The main inventory script (EYInventoryScript.vbs) is written in Visual Basic Script. This language was chosen to provide maximum compatibility with all versions of Windows operating systems. The script may call other PowerShell scripts when VB Script is unable to query certain inventory and configuration information. To start inventory collection, simply double click EYInventoryScript.vbs. A menu of scanning options is presented to the user.

The script(s) are provided "as is", and none of EY or any other party involved in the creation, production or delivery of any script(s) makes any warranties, express or implied regarding same. Notably we cannot guarantee the operation of any script(s) will be uninterrupted, error free or that it will be compatible with any hardware or software used by you. Accordingly you are encouraged to submit the script(s) and supporting documentation for review and approvals through your Change Advisory Board, or similar, before operation in your production environment. If you have any questions regarding the script(s) or supporting documentation please revert to your nominated EY contact.

2. Prerequisites for running the tool

2.1 Supported platforms

Windows Server: Windows 2003 server to and including Windows server 2016.
Windows Client: Windows XP to and including Windows 10.

Windows NT4 (with WMI installed) & 2000 (workstation & server) are supported with certain limitations. Limitations include:

- Certain data fields cannot be collected
- Only 'LocalMode' is supported. Data can only be collected for the local device

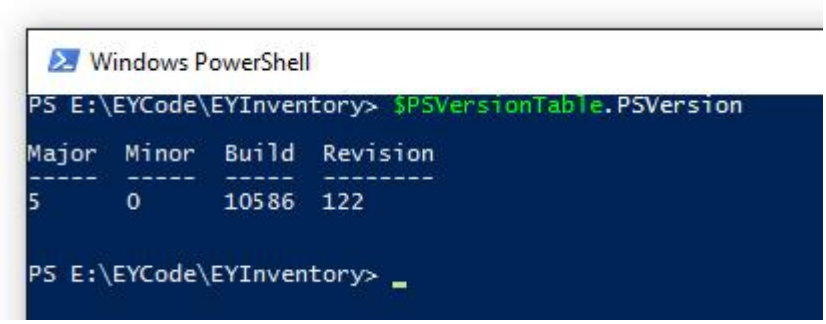
When running scans from a central location, it is recommended to do this from a test server, a dedicated server or a server which is providing a redundant service. It is recommended not to run the scan from c: if possible.

2.2 PowerShell

The primary inventory script is written in Visual Basic Script. In certain circumstances, it is necessary to launch PowerShell scripts to query inventory and configuration information that cannot be gathered through VB script. PowerShell v2 or higher is required to support this integration.

To determine the version of PowerShell installed

1. Launch a PowerShell console from the start menu
2. Type "$PSVersionTable.PSVersion" at the prompt



3. Ensure the Major Version is at least 2. If it is not, download and install the latest "Windows Management Framework" from the Microsoft site.

2.3 Privileges

In order to use the tool, it needs to run under credentials which have administrator privileges on the remote systems. This could be a domain admin account or an account which is in the local administrator group on the remote systems.

2.4 Network

The systems scanned need to be accessible from the host and the host needs to be able to resolve their names to IP addresses.

The tool will use a network ping to determine of the server/workstation is offline or online, so if ICMP traffic is blocked all systems will appear as offline.

Network utilization is approximately 1 MB – 4 MB for each system. The throughput in a LAN environment is on average 50-80 Kilobytes, with bursts around 400-800 Kilobytes. If there is high latency the burst and average throughput will decrease.

Bandwidth can be controlled using Netlimiter or similar software, this will not prevent the tool form running, but will decrease execution speed. Tested with 100Kbit.

2.5 Known issues

Issues that might prevent the tool from running as expected. The errors described does not cause any harm, but will prevent the tool from running correctly.

2.5.1 Windows firewall

In some cases, depending on configuration, an active Windows firewall on the remote system may prevent the tool from gathering data. In this case, data can be collected from the system by running the tool locally via a USB stick. See 3.1 for more information.

For more information about firewall settings and WMI please check:
http://blogs.msdn.com/b/john_daskalakis/archive/2009/02/05/9397926.aspx &
http://msdn.microsoft.com/en-us/library/aa389286.aspx

Example allowing the tool to run through a Windows firewall (Windows 7)

2.5.2 Windows Server 2008/Windows Vista - User Account Control (UAC)

If UAC is enabled, running WMI remotely may fail and the tool will have to be run locally.

See http://msdn.microsoft.com/en-us/library/aa826699(VS.85).aspx for details. The article is related to Vista but the issue is identical on Windows Server 2008.

2.5.3 Aliases and servers/workstations with multiple names

When using a devicelist.txt or /SERVER:<servername>, using a name that's different from the Windows servers/workstation "correct" name, might cause errors and incomplete data. It will not damage the server/workstation, but there is a risk that the output may be incomplete. This can also be caused by a mistyped IP in the host file on the system running the tool. It may be possible to locate a servers or workstations "real name" in the hostname_EY_SYSTEM.TXT or the hostname_EY_ADDevices.csv files.

2.5.4 Systems under heavy load

In default configuration the tool will not scan systems under heavy load.

If a system is under heavy CPU load, the tool will skip the system, and try to scan the next one in queue.

On Windows 2000, a system which is not being used might return 98% cpu utilization, in this case there is no other option but to disable the cpu load check, by setting maxCPULoadHost =100.

See http://support.microsoft.com/kb/888086/en-us for details.

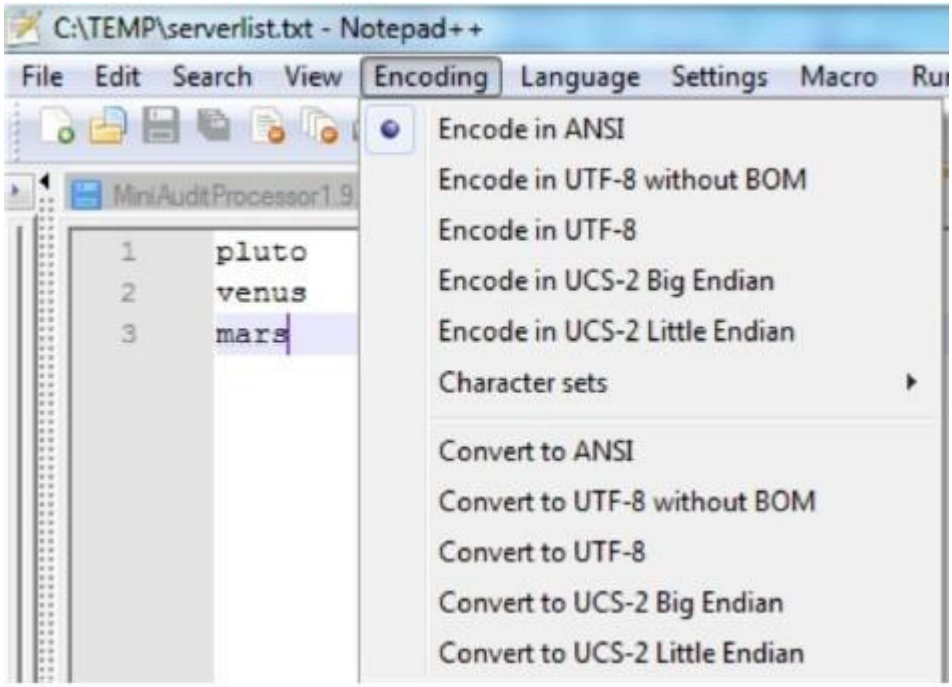
Eventually, when all systems have been scanned, the tool will try to scan systems which were not scanned in the first pass. The number of passes the tool will need to complete depends on the specific review and is part of the configuration settings set by the project manager

2.5.5 Using FAT32 file system for large scans

When scanning more than 7000 systems it is recommended to use a file system formatted using NTFS and not FAT/FAT32. This is due to limitations in the number of files, using FAT32/FAT can cause issues when file count exceeds 65000.

2.5.6 Error when using certain file formats in DeviceList.txt

There are many ways to detect the text file format, one way is to open it in notepad++ (http://notepad-plus-plus.org/) and select encoding from the dropdown menu:



The script will only work correctly when encoded with ANSI or UCS-2 Little Endian. Notepad++ can convert the current format. The recommended encoding is UCS-2 Little Endian.

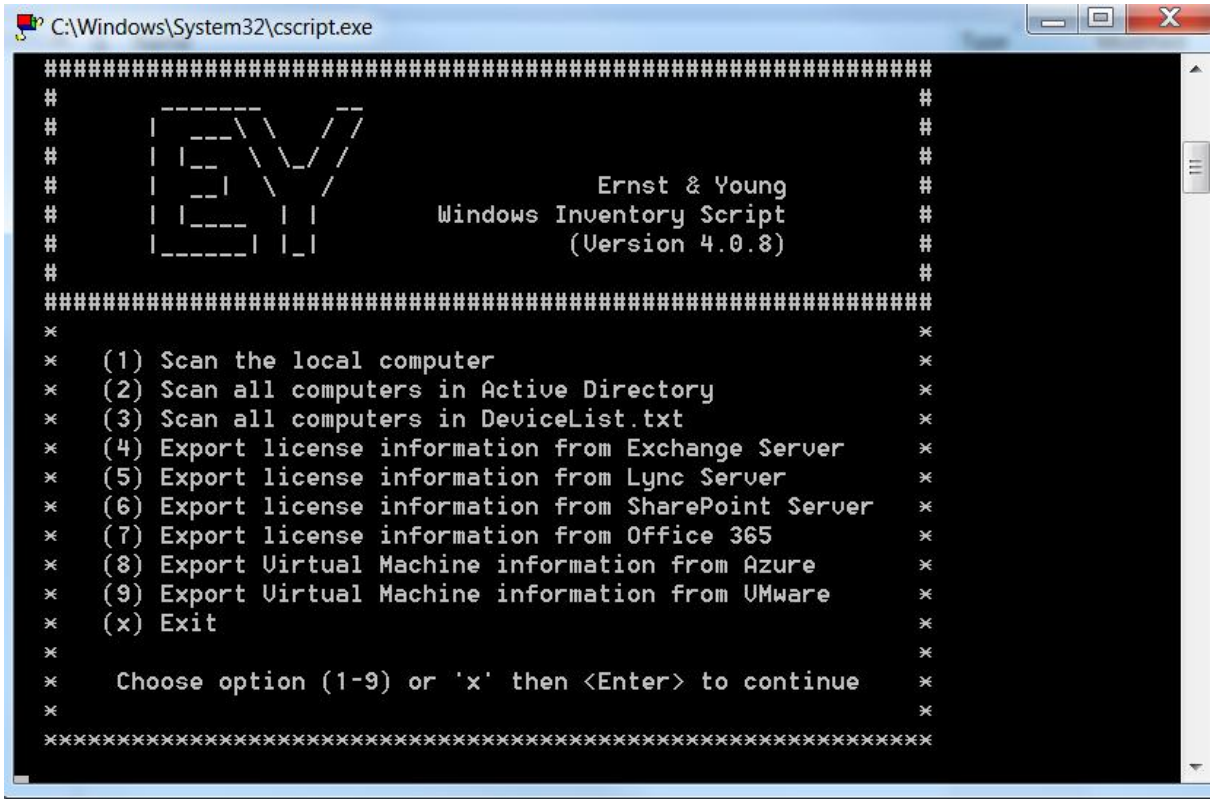
The error when using wrongly formatted text files looks like this:

3. Usage examples

The inventory tool can

- Scan the local device for software and hardware inventory information
- Scan a named device for software and hardware inventory information
- Query Active Directory for all devices on the domain and query each individual device for inventory information
- Read a list of device names from a local file (DeviceList.txt) and query each individual device for inventory information
- Query Exchange Server, Skype for Business / Lync Server or Office 365 for licensing relevant information
- Query Azure IaaS (Classic & RM) or VMware for licensing relevant virtual machine information

The tool can be launched from the command line where certain options can be specified or by simply double clicking EYInventoryScript.vbs.



3.1 Local Scan

To scan the local device

- Launch the Inventory Script menu by double clicking EYInventoryScript.vbs, type 1 then <Enter> Or
- Type the following at the command line

```
cscript EYInventoryScript.vbs /local
```

Several output files containing inventory data are written to a subdirectory of the current working directory.

```
E:\EY>cscript EYInventoryScript.vbs /local
Microsoft (R) Windows Script Host Version 5.812
Copyright (C) Microsoft Corporation. All rights reserved.

------- Installed Application from PICARD -------
------- Exchange Server on PICARD
------- System information from PICARD -------
------- PICARD is a workstation
------- Software license information from PICARD -------
------- Service Information from PICARD -------
------- SQL Edition info from instance SQLEXPRESS on PICARD [Running]
------- Processor information from PICARD -------
------- System Date information from PICARD -------
------- Server Feature information from PICARD -------
------- MSIInstaller information from PICARD -------
------- Process Information from PICARD -------
------- IP address information from PICARD -------
------- User logon Information from PICARD -------
------- User Profile Information from PICARD -------
------- Access group Information from PICARD -------

E:\EY>
```

Sample console output when scanning the local device.

It may be necessary to scan one or more devices locally if they are inaccessible from the central scanning host e.g. When scanning devices in a DMZ or devices where WMI is blocked by firewall. When scanning multiple devices locally, it may be useful to save the script to a central ‘file share’ with write access or a USB key which can be attached to the devices to be scanned.

To scan one or more devices locally using a USB key the following guidelines apply

1. Copy the tool to a USB stick
2. Log on to the system with an administrator account
3. Open a command prompt

4. Change drive to the removable media on which the tool is stored
5. Run cscript EYInventoryScript.vbs /local
6. Wait for the tool to finish
7. Remove the media from the server.
8. Ensure that files on the removable media are deleted after they have been processed.

This way no data is ever stored on the server running the tool. After running the tool, collect the data files on the media and ensure files on the media related to this tool are deleted.

3.2 Remote Scan – Single Device

To scan a single remote device, type the following at the command line, making sure to replace 'DEVICE_NAME' with the name of the device to be scanned.

```
cscript EYInventoryScript.vbs /server:DEVICE_NAME
```

Note that there is no space between '/server:' and the name of the device to be scanned.

In order for this method to work

- The current user must have administrator privileges on the remote device
- The local computer must be able to resolve the remote device name to an IP address and must be able to connect to the WMI service on that device. The WMI service generally listens on port 135.

If the current user does not have administrator privileges on the device to be scanned, it is possible to provide credentials of an account that does.

```
cscript EYInventoryScript.vbs /server:DEVICE_NAME /user:USER_NAME
/domain:DOMAIN_NAME /password:PASSWORD
```

Again, make sure to replace 'USER_NAME', 'DOMAIN_NAME' & 'PASSWORD' with the credentials of the user with administrator privileges on the device to be scanned.

3.3 Scan All Devices in Active Directory

To scan all devices in Active Directory

- Launch the Inventory Script menu by double clicking EYInventoryScript.vbs, type 2 then <Enter> Or
- Type the following

```
cscript EYInventoryScript.vbs /AD
```

This option queries the local Active Directory for all devices, users, groups, forest domains, trusted domains, Exchange servers & ActiveSync devices by calling an Active Directory PowerShell script. The queried data is stored in the script output folder using the following name format – <DeviceName>-EY-AD<DataType>.csv. The script output folder is a sub folder of the current directory (i.e. The directory where the tool was started from) It is call 'EY_Output'

The tool then scans all devices registered in AD, except devices that have not been used for an extended period of time.

The tool will cycle through the list of devices a number of times but it will only check devices which are online and accessible once. The next will only check systems which were offline or did not deliver any data on previous attempts.

The tool will only run against one domain at a time and will run against the same domain as the account executing it. This means that it will only query the root domain (in a forest setup), if the account executing it is in the root domain.

3.3.1 Concurrent Scanning

When running the tool against multiple systems, it will start multiple processes to scan devices concurrently. This prevents a single system with high latency from delaying the entire scan and ensures all scans are completed quickly.

The amount of simultaneous scans is configurable. The constant used to hold the maximum number of scan processes is called MaxThreads and is found near the top of the VBS script.

3.3.2 Status Files

The tool will create scan status files in the output folder for each device that has been queried. There are different status files:

<DeviceName>_EY_online = Ping succeeded and/or WMI connection established.

< DeviceName >_ EY _offline = Device is offline; host was unable to connect to the device. If both online and offline are present for a device, there was a problem connecting to a live host.

< DeviceName >_ EY _Error = One of the output is missing required data or does not exist.

< DeviceName >_ EY _noadmin = The current credentials do not have administrator rights on the tested system.

< DeviceName >_ EY _Load = System is under heavy load, the tool skipped the system and will retry at next pass.

< DeviceName >_ EY _Done = Data was collected from the device, and everything is ok.

If the tool is stopped for some reason, it can be restarted and it will pick up where it left off. The tool uses the status files to figure out which of the systems have been polled and which are still left to be processed.

3.3.3 Tracking Progress

When scanning multiple devices, scan progress is displayed on screen

```
E:\EY>cscript EYInventoryScript.vbs /AD
Microsoft (R) Windows Script Host Version 5.812
Copyright (C) Microsoft Corporation. All rights reserved.

*** DeviceList.txt and EY_ADComputers.txt Created ***

 Scanning multiple devices for software & hardware inventory

Read 10 systems into memory
*** Processing [    1/10] "DC1"
*** Processing [    2/10] "RELIANT"
*** Processing [    3/10] "DEFIANT"
*** Processing [    4/10] "OMAHA"
*** Processing [    5/10] "TRIDENT"
*** Processing [    6/10] "PICARD"
*** Processing [    7/10] "RIKER"
*** Processing [    8/10] "YAR"
*** Processing [    9/10] "SCOTTY"
*** Processing [   10/10] "ARCHER"
```

After each pass, progress is printed to the screen.

```
*** Status     [    8/10] DONE
*** Status     [    0/10] LOAD
*** Status     [    0/10] ERROR
*** Status     [    2/10] OFFLINE
```

3.3.4 Scan duration

The duration of the scan has been set b, default is 3 days, the scan will complete immediately if all systems have been successfully scanned.

3.4 Scan All Devices in DeviceList.txt

To scan all devices listed in DeviceList.txt

- Launch the Inventory Script menu by double clicking EYInventoryScript.vbs, type 3 then <Enter> Or
- Type the following

```
cscript EYInventoryScript.vbs /DeviceList
```

DeviceList.txt should be

- Saved in the same folder as the tool
- Contain one device name per line
- Saved in ANSI or UCS-2 Little Endian format

The ‘Scan All Devices in DeviceList.txt’ feature differs from the ‘Scan All Devices in AD’ in only one way - The devices to be queried are listed in the text file rather than queried from Active Directory. Concurrent scanning, status files, scan progress and scan duration are all the same in both features.

3.5 Export License Information from Exchange Server

To export licensing relevant information from an Exchange environment

- Launch the Inventory Script menu by double clicking EYInventoryScript.vbs, type 4 then <Enter> Or
- Type the following, making sure to replace 'EXCHANGE_SERVER_NAME' with the name of the Exchange server to be scanned

```
cscript EYInventoryScript.vbs /Exchange:EXCHANGE_SERVER_NAME
```

Note that there is no space between '/Exchange:' and the name of the device to be queried.

The tool will launch a PowerShell script (Get-ExchangeDetails.ps1) that queries a single Exchange server and produces up to 5 CSV files

1. <DeviceName>_EY_ExchangeServerDetails.csv – A list of Exchange Servers in the site including role & edition
2. <DeviceName>_EY_ExchangeMailBoxes.csv – List of MailBoxes
3. <DeviceName>_EY_ExchangeDevices.csv – List of ActiveSync devices
4. <DeviceName>_EY_ExchangeCALs.csv - General CAL requirement details
5. <DeviceName>_EY_ExchangeCALDetails.csv - Lists all servers and MailBoxes that require a CAL and the type of license required

Files are written to the standard output directory. The script should be run once per Exchange site. It should not be necessary to run the script for multiple Exchange Servers in a single Exchange site.

The Exchange PowerShell script will query the user for credentials for an account with permission to administer the Exchange server.

```
PS E:\EY\PSScripts> .\Get-ExchangeDetails.ps1

  ################################################################
  #       ________      __                                       #
  #      |  ___\ \    / /                                       #
  #      | |__  \ \_/ /                                         #
  #      |  __|  \    /                  Ernst & Young          #
  #      | |____  | |        Windows Inventory Script           #
  #      |______| |_|                                           #
  #                                                              #
  ################################################################

 Get-ExchangeDetails.ps1

Computer Name:        PICARD
User Name:            user@EY.LOCAL
Windows Version:      Microsoft Windows 10 Enterprise(10.0.10586)
PowerShell Host:      5
PowerShell Version:   5.0.10586.122
PowerShell Word size: 64 bit
CLR Version:          4.0.30319.42000
Current Date Time:    2016-01-01 10:00:00
Username Parameter:
Server Parameter:
Required Data:        AllData
Connection Method:    Both
CAL Script Version:
Output File 1:        Picard_EY_ExchangeServerDetails.csv
Output File 2:        Picard_EY_ExchangeMailBoxes.csv
Output File 3:        Picard_EY_ExchangeDevices.csv
Output File 4:        Picard_EY_ExchangeCALs.csv
Output File 5:        Picard_EY_ExchangeCALDetails.csv
Log File:             Picard_EY_ExchangeQueryLog.txt

Exchange Server Name (Default [PICARD]): defiant
Exchange Server Administrator Credentials Required
Username: domadmin
Password: *********
```

The script only reads data from the Exchange server – It does not make any configuration changes to the Exchange environment.

The script outputs progress to screen

```
Exchange Server Name (Default [PICARD]): defiant
Exchange Server Administrator Credentials Required
Username: domadmin
Password: *********
10:00:01.00 - Connecting...
10:00:05.00 - Importing Session
10:00:10.00 - Getting server details
10:00:12.00 - Querying Mailboxes
10:00:25.00 - Querying Device Data
Report Exchange 2010 client access licenses (CALs) in use in the
organization
It will take some time if there are a large amount of users......

Progress:
Total Standard CALs calculated:                 100
Info Leakage Protection Enabled:                False
Unified Messaging Users calculated:             0
Managed Custom Folder Users calculated:         0
Advanced ActiveSync Policy Users calculated:    0
Archived Mailbox Users calculated:              0
Retention Policy Users calculated:              0

Searchable Users calculated:                    0
Journaling Users calculated:                    0
Total Enterprise CALs calculated:               0

=========================
Exchange CAL Usage Report
=========================

Total Users:                                    100
Total Standard CALs:                            100
Total Enterprise CALs:                          0
00:07:05.96 - Cleaning Session
00:07:06.06 - Complete
```

A log file of the script activity is saved in the standard output folder with the name <DeviceName>_EY_ExchangeQueryLog.txt

This script supports Exchange Server 2007 onwards. There are some limitations on what data can be collected from different versions of Exchange Server remotely. If the script fails to connect to the Exchange Server

- Ensure that Exchange client tools including the Exchange PowerShell module are installed on the local device and/or
- Execute the script locally on the Exchange Server

3.6 Export license information from Skype for Business or Lync Server

To export licensing relevant information from a Skype for Business or Lync environment

- Launch the Inventory Script menu by double clicking EYInventoryScript.vbs, type 5 then <Enter> Or
- Type the following, making sure to replace 'LYNC_SERVER_NAME' with the name of the Skype for Business / Lync server to be scanned.

```
cscript EYInventoryScript.vbs /Lync:LYNC_SERVER_NAME
```

Note that there is no space between '/Lync:' and the name of the device to be queried.

N.B. The Skype for Business / Lync server name must be specified in FQDN (fully qualified domain name) format. E.g. LyncServer01.EY.local

The tool will launch a PowerShell script (Get-LyncUsers.ps1) that queries a single Skype for Business / Lync server and produces a single CSV

- <DeviceName>_EY_LyncUsers.csv – A list of Lync users including enabled features and required CALs

The CSV file is written to the standard output directory. The script should be run once per Skype for Business / Lync site. It should not be necessary to run the script for multiple Skype for Business / Lync Servers in a single Skype for Business / Lync site.

The Skype for Business / Lync PowerShell script will query the user for credentials for an account with permission to administer the Lync server.

The script only reads data from the Skype for Business / Lync server – It does not make any configuration changes to the Skype for Business / Lync environment.

A log file of the script activity is saved in the standard output folder with the name <DeviceName>_EY_LyncQueryLog.txt

This script supports Skype for Business / Lync Server 2010 onwards. There are some limitations on what data can be collected from different versions of Skype for Business / Lync Server remotely. If the script fails to connect to the Skype for Business / Lync Server

- Ensure that Skype for Business / Lync client tools including the Skype for Business / Lync PowerShell module are installed on the local device and/or
- Execute the script locally on the Skype for Business / Lync Server

3.7 Export License Information from SharePoint Server

To export licensing relevant information from a SharePoint farm environment

- Launch the Inventory Script menu by double clicking EYInventoryScript.vbs, type 6 then <Enter>

Or

- Type the following, making sure to replace 'SHAREPOINT_SERVER_NAME' with the name of the SharePoint server to be queried.

```
cscript EYInventoryScript.vbs /SharePoint:SHAREPOINT_SERVER_NAME
```

Note that there is no space between '/SharePoint:' and the name of the device to be queried.

The tool will launch a PowerShell script (Get-SharePointLicenseDetails.ps1) that queries a single SharePoint server and produces 3 CSV files

- <DeviceName>_EY_SharePointSites.csv – A list of SharePoint sites including whether premium features are enabled
- <DeviceName>_EY_SharePointUserGroups.csv – A list of SharePoint groups and group members
- <DeviceName>_EY_ShareUserCALs.csv – A list of users and SharePoint CAL requirements

The CSV file is written to the standard output directory. The script should be run once per SharePoint farm. It should not be necessary to run the script for multiple SharePoint Servers in a single SharePoint farm.

The script only reads data from the SharePoint server – It does not make any configuration changes to the SharePoint environment.

A log file of the script activity is saved in the standard output folder with the name <DeviceName>_EY_ SharePoint QueryLog.txt

This script supports SharePoint Server 2010 onwards.

3.7.1 Querying Remote SharePoint Servers

If the server to be queried is not the local device, the script will use the Microsoft component PSExec to execute the script on the remote server. It is possible to disable this functionality by simply deleting PSExec from the PSScripts folder. Remote script execution also requires PowerShell Remoting to be enabled on the remote server. If this is not already configured, it can be enabled by starting a PowerShell session on the SharePoint server (in elevated mode) and calling the 'Enable-PSRemoting' command.

When connecting to remote servers, the script will prompt the user for credentials for an account with permission to administer the SharePoint server. The user name should be specified in Domain\UserName format.

If the script fails to connect to the SharePoint Server remotely

- Ensure that the Microsoft component PSexec.exe is in the PSScripts folder

- Enable PowerShell Remoting on the SharePoint server

And/or

- Execute the script locally on the SharePoint Server

3.8 Export license information from Office365

To export license entitlement and assignment information from Office 365

- Launch the Inventory Script menu by double clicking EYInventoryScript.vbs, type 6 then <Enter> Or
- Type the following at the command line

```
cscript EYInventoryScript.vbs /Office365
```

The tool will launch a PowerShell script (Get-Office365LicenseDetails.ps1) that queries Office 365 and produces a single CSV file - <DeviceName>_EY_Office365Licenses.csv. The file contains details of Office365 users and Office365 licenses that have been allocated to those users in the current Office 365 tenant.

The Office 365 PowerShell script will query the user for credentials for an account with permission to administer the Office 365 tenant. The credentials are not stored on the device after the script has completed. The script only reads data from the Office 365 environment – It does not make any configuration changes.

The script will need to be executed once per Office 365 tenant. The script only reads data from the Office 365 service – It does not make any configuration changes to the environment.

A log file of the script activity is saved in the standard output folder with the name <DeviceName>_EY_Office365QueryLog.txt

3.8.1 Prerequisites

```
 Get-Office365Details.ps1

Computer Name:        PICARD
User Name:            user@EY.LOCAL
Windows Version:      Microsoft Windows 10 Enterprise(10.0.10586)
PowerShell Host:      5
PowerShell Version:   5.0.10586.122
PowerShell Word size: 64 bit
CLR Version:          4.0.30319.42000
Current Date Time:    2016-01-01 10:00:00
Username Parameter:

10:00:00.50 - Started
The required PowerShell Office 365 components are not installed on this
device.
The following components are required
 1) Microsoft Online Services Sign-In Assistant -
http://www.microsoft.com/en-us/download/details.aspx?id=41950
 2) Windows Azure Active Directory Module for Windows PowerShell (64-bit
version) - http://go.microsoft.com/fwlink/p/?linkid=236297

Download and install required components (Y/N):
```

This script requires the following PowerShell components to run.

- Microsoft Online Services Sign-In Assistant for IT Professionals
- Windows Azure Active Directory Module for PowerShell

These components are compatible with Windows 7 (64 bit) and Windows Server 2008 (64 bit) or higher. They can be downloaded and installed using the links provided.

Alternatively, if required, the script can download and install the necessary components. The script needs to be run in 'Elevated Mode' for this installation to succeed. To automatically install the required components

1. Launch Command Prompt 'As Administrator'
2. Type the command line

```
cscript EYInventoryScript.vbs /Office365
```

3. Type Y, then <Enter>

The installation may require a system reboot to complete the process. If the installation process does not successfully install the components, they can be installed manually using the links provided.

3.9 Export Virtual Machine information from Azure

To export virtual machine information from Azure

- Launch the Inventory Script menu by double clicking EYInventoryScript.vbs, type 7 then <Enter> Or
- Type the following at the command line

```
cscript EYInventoryScript.vbs /Azure
```

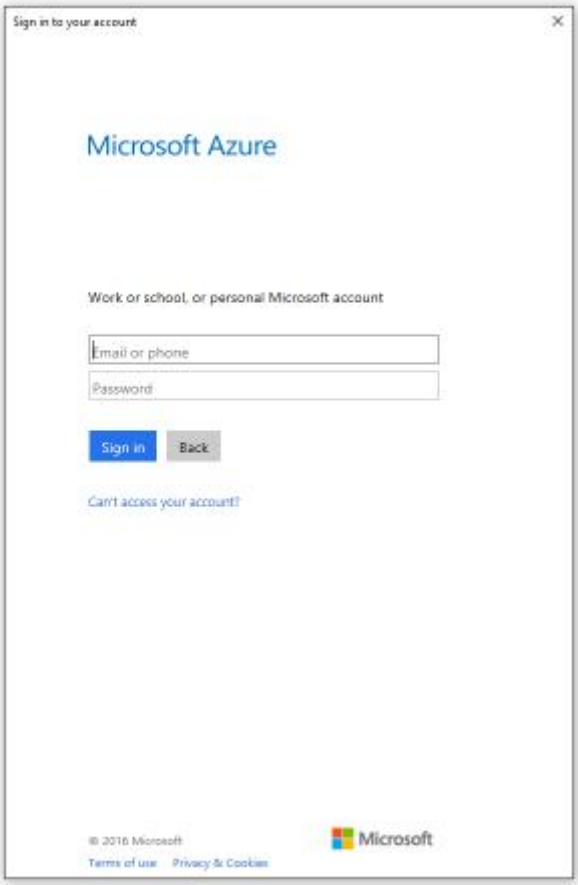
The tool will launch a PowerShell script (Get-AzureVMList.ps1) that queries Azure and produces 2 CSV files

1. <DeviceName>_EY_AzureClassicVMList.csv – A list of 'Classic' Azure VMs
2. <DeviceName>_EY_AzureRMVMList.csv – A list of 'Resource Manager' Azure VMs

The PowerShell script will query the user for credentials for an account that has administrative rights on one or more Azure subscriptions. The output files include details of all virtual machines contained within those Azure subscriptions. The script uses Azure's Service Management API to get a list of 'Classic' VMs and Azure's Resource Manager API to get a list of 'Resource Manager' VMs. For this reason, the script prompts the user for credentials twice. The same credentials should be used both times.

A log file of the script activity is saved in the standard output folder with the name <DeviceName>_EY_AzureQueryLog.txt

The script only reads data from the Azure service – It does not make any configuration changes to the Azure environment.



3.9.1 Prerequisites

This script requires the following PowerShell components to run.

- Web Platform Installer
- Windows Management Framework version 3.0 or higher
- .Net Framework version 4.0 or higher

These components are compatible with Windows 7 (64 bit) and Windows Server 2008 (64 bit) or higher. They can be downloaded and installed using the links provided.

3.10 Export Virtual Machine information from VMware

To export licensing relevant virtual machine information from VMware

- Launch the Inventory Script menu by double clicking EYInventoryScript.vbs, type 8 then <Enter> Or
- Type the following at the command line

```
cscript EYInventoryScript.vbs /VMware
```

The tool will launch a PowerShell script (Get-VMwareVMList.ps1) that queries a VMware vSphere or vCenter server and produces 6 CSV files

1. <DeviceName>_EY_VMwareVMs.csv – A list of VMware VMs, including hosts & cluster information etc.
2. <DeviceName>_EY_VMwareHosts.csv – A list of VMware Hosts
3. <DeviceName>_EY_VMwareClusters.csv – A list of VMware clusters etc.
4. <DeviceName>_EY_ VMwareVMDRSRules.csv – A list of DRS rules that determine potential hosts for VMs
5. <DeviceName>_EY_ VMwareVMDRSGroups.csv –DRS group and group membership information
6. <DeviceName>_EY_ VMwareVMotionData.csv – Recent vMotion history information

Files are written to the standard output directory.

The script will query the user for a vCenter or a vSphere server name plus the credentials for an account with permission to administer the environment. These are the same credentials that might be used with the popular VMware reporting tool, RVTools, or PowerCLI based VMware management scripts. When querying a vSphere server, the account will often be one of the following – root, administrator, root@vpshere.local or administrator@vsphere.local. When querying a vCenter server, the account will usually be a Windows user account.

If vCenter is configured, it should be possible to gather all VM data from the VMware data center by querying a single vCenter server. Otherwise it is necessary to query each individual vSphere server.

The script only reads data from the VMware server(s) – It does not make any configuration changes to the VMware environment

3.10.1 Prerequisites

The PowerShell VMware script requires the VMware PowerCLI component to run. If not already installed, this component can be downloaded from the VMware support site - https://www.vmware.com/support/developer/PowerCLI/

3.11 Other Features

Query AD for Active Devices

```
cscript EYInventoryScript.vbs /ListActiveComputers
```

A list of computers registered in Active Directory is written to EY_ADComputers.txt

Query AD for Users

```
cscript EYInventoryScript.vbs /ListAdUsers
```

A list of users registered in Active Directory is written to #EY_ADUsers.txt

Query AD for Groups

```
cscript EYInventoryScript.vbs /ListAdGroups
```

A list of Active Directory groups is written to #EY_ADGroups.txt

Query AD for User & Groups

```
cscript EYInventoryScript.vbs /ListAdUsersAndGroups
```

A list of users registered in Active Directory is written to #EY_ADUsers.txt and a list of Active Directory groups is written to #EY_ADGroups.txt

4. Appendix

The following table outlines the various data points that can be collected by the EY Inventory script

File Name	Fields	Method	Source	Conditions (if any)	Remarks
ADDomains.csv	Forest	PowerShell script	[System.DirectoryServices.ActiveDirectory.Domain]::GetCurrentDomain()		
	DomainControllers				
	Children				
	DomainMode				
	DomainModeLevel				Available since .NET Framework 4.6
	Parent				
	PdcRoleOwner				
	RidRoleOwner				
	InfrastructureRoleOwner				
	Name				
ADDomainTrusts.csv	name	PowerShell script	SearchAD	(objectClass=trustedDomain)	
	trustpartner				
	flatname				
	distinguishedname				
	adspath				
	trustdirection				
	trustattributes				
	trusttype				
	trustposixoffset				
	instancetype				
	whencreated				
	whenchanged				
ADDomainNETBIOS.csv	name	PowerShell script	SearchAD	(NetBIOSName=*)	
	netbiosname				
	ncname				
	adspath				

File Name	Fields	Method	Source	Conditions (if any)	Remarks
	dnsroot				
	objectguid				
	whencreated				
	whenchanged				
ADDomainControllers.csv	Forest	PowerShell script	[System.DirectoryServices.ActiveDirectory.Domain]::GetCurrentDomain()		
	CurrentTime				
	HighestCommittedUsn				
	OSVersion				
	Roles				
	Domain				
	IPAddress				
	SiteName				
	SyncFromAllServersCallback				
	InboundConnections				
	OutboundConnections				
	Name				
	Partitions				
ADUsers.csv	samaccountname	PowerShell script	SearchAD	(&(objectCategory=person)(objectClass=user))	
	userPrincipalName				
	objectsid				
	objectguid				
	displayname				
	departmentNumber				
	company				
	department				
	distinguishedname				
	lastlogon				

File Name	Fields	Method	Source	Conditions (if any)	Remarks
	lastlogontimestamp				
	logoncount				
	mail				
	telephoneNumber				
	physicalDeliveryOfficeName				
	description				
	whenchanged				
	whencreated				
	msexchmailboxguid				
	useraccountcontrol				
ADGroups.csv	name	PowerShell script	SearchAD	(objectClass=group)	
	description				
	distinguishedname				
	whenchanged				
	whencreated				
	Members				
ADDevices.csv	name	PowerShell script	SearchAD	(objectClass=computer)	
	objectsid				
	objectguid				
	operatingsystem				
	operatingsystemversion				
	operatingSystemServicePack				
	lastlogon				
	lastlogontimestamp				
	adspath				
	location				
	dnshostname				

File Name	Fields	Method	Source	Conditions (if any)	Remarks
	description				
	whenchanged				
	whencreated				
	serviceprincipalname				
ADExchangeServers.csv	name	PowerShell script	SearchAD	(objectCategory=msExchExchangeServer)	
	objectguid				
	msexchproductid				
	msexchcurrentserverroles				
	type				
	msexchserversite				
	usncreated				
	adspath				
	msexchversion				
	serialnumber				
	msexchserverrole				
	ExchangeEdition				
	ExchangeCurrentRoles				
ADActiveSyncDevices.csv	name	PowerShell script	SearchAD	(objectClass=msExchActiveSyncDevice)	
	objectguid				
	adspath				
	description				
	whenchanged				
	whencreated				
	msexchdeviceeasversion				
	msExchDeviceFriendlyName				
	msexchdeviceid				
	msExchDeviceIMEI				

File Name	Fields	Method	Source	Conditions (if any)	Remarks
	msExchDeviceMobileOperator				
	msexchdevicemodel				
	msExchDeviceOS				
	msExchDeviceOSLanguage				
	msExchDeviceTelephoneNumber				
	msexchdevicetype				
	msExchLastExchangeChangedTime				
	msExchLastUpdateTime				
ADSettings.csv	distinguishedName	PowerShell script	SearchAD	(objectCategory=domainDNS) OR (objectClass=domainPolicy)	
	name				
	whenCreated				
	whenChanged				
	minPwdLength				
	minPwdAge				
	maxPwdAge				
	pwdProperties				
	lockoutThreshold				
	lockoutDuration				
	pwdHistoryLength				
	msDS-Behavior-Version				
	msDS-NcType				
HyperVExportBasic.csv	FullyQualifiedDomainName	WMI to {'root\virtualization', 'root\virtualization\v2'}	Msvm_ComputerSystem		Using Powershell
	OSName				
	OSVersion				
	CSDVersion				
	ProductType				
	NetworkAddressIPv4				

File Name	Fields	Method	Source	Conditions (if any)	Remarks
	NetworkAddressIPv6				
	OSEditionId				
	ProcessorArchitecture				
	SuiteMask				
HyperVExportGuests.csv	Name	PowerShell script			
	GUID				Renamed from VMId
	State				
	ReplicationState				
	MemoryAssigned				
	IntegrationServicesVersion				
	Host				Renamed from ComputerName
	CreationTime				
	IsClustered				
	ProcessorCount				
	IPAddresses		NetworkAdapters class		
	MACAddresses		NetworkAdapters class		
HyperVExportHosts.csv	Name	PowerShell script			
	FullyQualifiedDomainName				
	LogicalProcessorCount				
	MemoryCapacity				
	MacAddressMinimum				
	MacAddressMaximum				
	VirtualMachineMigrationEnabled				
HyperVExportVMStarts.csv	ID	PowerShell script			
	Time				
	Description				

File Name	Fields	Method	Source	Conditions (if any)	Remarks
	UserID				
	MachineName				
	MachineGUID				
Applications.csv	ComputerName	PowerShell script			
	DisplayName				
	InstalledDate				
	VersionMajor				
	DisplayVersion				
	Publisher				
SharePointSites.csv	WebApplicationUrl	PowerShell script	Remote Session/Load SnapIn		
	WebApplicationName				
	WebApplicationDisplayName				
	WebApplicationId				
	WebApplicationFarmName				
	WebApplicationStatus				
	WebApplicationVersion				
	WebApplicationApplicationPoolName				
	SCUrl				
	SCHostName				
	SCWebApplication				
	SCID				
	SCSchemaVersion				
	SCArchived				
	SCCreation Date				
	SCExpirationDate				
	SCLastContentModifiedDate				

File Name	Fields	Method	Source	Conditions (if any)	Remarks
	SCLastSecurityModifiedDate				
	SiteUrl				
	SiteTitle				
	SiteName				
	SiteID				
	SiteDescription				
	SiteAuthor				
	SiteParentWeb				
	SiteParentWebId				
	SiteIsAppWeb				
	SiteIsRootWeb				
	SiteHasUniqueRoleDefinitions				
	SiteAllowAnonymousAccess				
	SiteWebTemplate				
	SiteUIVersion				
	SiteCreationDate				
	SiteLastItemModifiedDate				
	IsPremiumFeatureEnabled				
	WebApplicationFeatureNames				
	SCFeatureNames				
	SiteFeatureNames				
	UsersList				
	SPGroups				
	ADGroups				
SharePointUserCALs.csv	User	PowerShell script	Remote Session/Load SnapIn		
	SPCALRequired				
	Name	PowerShell script			

File Name	Fields	Method	Source	Conditions (if any)	Remarks
SharePointUserGroups.csv	LoginName		Remote Session/Load SnapIn		
	Owner				
	Description				
	ParentWeb				
	DistributionGroupEmail				
	ADGroups				
	UsersList				
ExchangeServerDetails.csv	PSComputerkName	PowerShell script			
	RunspaceId				
	PSShowComputerName				
	Name				
	DataPath				
	Domain				
	Edition				
	ExchangeLegacyDN				
	ExchangeLegacyServerRole				
	Fqdn				
	CustomerFeedbackEnabled				
	InternetWebProxy				
	IsHubTransportServer				
	IsClientAccessServer				
	IsExchange2007OrLater				
	IsEdgeServer				
	IsMailboxServer				
	IsE14OrLater				
	IsE15OrLater				
	IsProvisionedServer				

File Name	Fields	Method	Source	Conditions (if any)	Remarks
	IsUnifiedMessagingServer				
	IsFrontendTransportServer				
	NetworkAddress				
	OrganizationalUnit				
	AdminDisplayVersion				
	Site				
	ServerRole				
	ErrorReportingEnabled				
	StaticDomainControllers				
	StaticGlobalCatalogs				
	StaticConfigDomainController				
	StaticExcludedDomainControllers				
	MonitoringGroup				
	CurrentDomainControllers				
	CurrentGlobalCatalogs				
	CurrentConfigDomainController				
	ProductID				
	IsExchange2007TrialEdition				
	IsExpiredExchange2007TrialEditio n				
	MailboxProvisioningAttributes				
	RemainingTrialPeriod				
	Identity				
	IsValid				
	ExchangeVersion				
	DistinguishedName				
	Guid				

File Name	Fields	Method	Source	Conditions (if any)	Remarks
	ObjectCategory				
	ObjectClass				
	WhenChanged				
	WhenCreated				
	WhenChangedUTC				
	WhenCreatedUTC				
	OrganizationId				
	Id				
	OriginatingServer				
	ObjectState				
ExchangeMailBoxes.csv	UserPrincipalName	PowerShell script			
	SamAccountName				
	DisplayName				
	WindowsLiveID				
	ExchangeGuid				
	PrimarySmtpAddress				
	ExternalDirectoryObjectId				
	EmailAddresses				
	DistinguishedName				
	Guid				
	RecipientType				
	IsMailboxEnabled				
	WhenMailboxCreated				
	WhenCreatedUTC				
	WhenChangedUTC				
	LastLogonTime				
	LastLogoffTime				

File Name	Fields	Method	Source	Conditions (if any)	Remarks
	ProtocolSettings				
	ExchangeVersion				
	ObjectCategory				
	ServerName				
	Office				
	ActiveSyncMailboxPolicy				
	ActiveSyncEnabled				
	OwaMailboxPolicy				
	OWAEnabled				
	ECPEnabled				
	EmwsEnabled				
	PopEnabled				
	ImapEnabled				
	MAPIEnabled				
	EwsEnabled				
ExchangeDevices.csv	Identity	PowerShell script			
	FriendlyName				
	Name				
	DeviceId				
	Guid				
	DeviceImei				
	DeviceTelephoneNumber				
	DeviceMobileOperator				
	DeviceOS				
	DeviceOSLanguage				
	DeviceType				
	DeviceUserAgent				

File Name	Fields	Method	Source	Conditions (if any)	Remarks
	DeviceModel				
	UserDisplayName				
	OrganizationId				
	DeviceActiveSyncVersion				
	FirstSyncTime				
	WhenCreatedUTC				
	WhenChangedUTC				
	LastPingHeartbeat				
	LastSyncAttemptTime				
	LastSuccessSync				
	LastPolicyUpdateTime				
	DevicePolicyApplied				
	DevicePolicyApplicationStatus				
	Status				
	StatusNote				
	IsRemoteWipeSupported				
	DeviceWipeSentTime				
	DeviceWipeRequestTime				
	DeviceWipeAckTime				
ExchangeCALs.csv	TotalMailboxes	PowerShell script			
	TotalStandardCALs				
	TotalEnterpriseCALs				
	UnifiedMessagingUserCount				
	ManagedCustomFolderUserCount				
	AdvancedActiveSyncPolicyUserCount				
	ArchivedMailboxUserCount				

File Name	Fields	Method	Source	Conditions (if any)	Remarks
	RetentionPolicyUserCount				
	SearchableUserCount				
	JournalingUserCount				
	InfoLeakageProtectionEnabled				
	AdvancedAntispamEnabled				
ExchangeCALDetails.csv	PSComputerName	PowerShell script			
	RunspaceId				
	PSShowComputerName				
	Name				
	LicenseName				
LyncUsers.csv	SamAccountName	PowerShell script	Remote Session/Load SnapIn		
	UserPrincipalName				
	FirstName				
	LastName				
	WindowsEmailAddress				
	Sid				
	LineServerURI				
	OriginatorSid				
	AudioVideoDisabled				
	IPPBXSoftPhoneRoutingEnabled				
	RemoteCallControlTelephonyEnabled				
	PrivateLine				
	HostedVoiceMail				
	DisplayName				
	HomeServer				
	TargetServerIfMoving				

File Name	Fields	Method	Source	Conditions (if any)	Remarks
	EnabledForFederation				
	EnabledForInternetAccess				
	PublicNetworkEnabled				
	EnterpriseVoiceEnabled				
	EnabledForRichPresence				
	LineURI				
	SipAddress				
	Enabled				
	TenantId				
	TargetRegistrarPool				
	VoicePolicy				
	MobilityPolicy				
	ConferencingPolicy				
	PresencePolicy				
	RegistrarPool				
	DialPlan				
	LocationPolicy				
	ClientPolicy				
	ClientVersionPolicy				
	ArchivingPolicy				
	PinPolicy				
	ExternalAccessPolicy				
	HostedVoicemailPolicy				
	HostingProvider				
	Name				
	DistinguishedName				
	Identity				

File Name	Fields	Method	Source	Conditions (if any)	Remarks
	Guid				
	ObjectCategory				
	WhenChanged				
	WhenCreated				
	OriginatingServer				
	IsValid				
	ObjectState				
	AllowIPAudio				
	AllowIPVideo				
	AllowUserToScheduleMeetingsWithAppSharing				
	EnableDataCollaboration				
	AllowSimulRing				
	AllowCallForwarding				
	AllowPSTNReRouting				
	EnableDelegation				
	EnableTeamCall				
	EnableCallTransfer				
	EnableCallPark				
	EnableMaliciousCallTracing				
	EnableBWPolicyOverride				
	PreventPSTNTollBypass				
	CALRequired				
VMwareVMs.csv	VM	PowerShell script	Get-VM		Renamed from Name
	VMCPUCount				Calculated (ExtensionData.Config.Hardware.NumCPU/ExtensionData.Config.Hardware.NumCoresPerSocket)

File Name	Fields	Method	Source	Conditions (if any)	Remarks
	VMCPUCoreCount				Renamed from NumCPU
	VMCPUCorePerCPUCount				Renamed from ExtensionData.Config.Hardware.NumCoresPerSocket
	PowerState				
	FaultToleranceState				
	OnlineStandby				
	Version				
	Description				
	Notes				
	MemoryMB				
	ResourcePool				
	ResourcePoolID				
	PersistentID				
	ID				
	UID				
	UUID				
	IPs				
	MACs				
	NetworkNames				
	OS				Renamed from Guest.OSFullName
	FQDN				Renamed from Guest.HostName
	ScreenDimensions				
	OS2				Renamed from ExtensionData.Guest.OSFullName

File Name	Fields	Method	Source	Conditions (if any)	Remarks
	FQDN2				Renamed from ExtensionData.Guest.HostName
	GuestHostname				Renamed from ExtensionData.Summary.Guest.HostName
	GuestId				
	GuestFullName				
	GuestIP				Renamed from ExtensionData.Summary.Guest.IpAddress
	Datacenter		Get-Datacenter		Renamed from Name
	Cluster		Get-VM		Renamed from VMHost.Parent.Name
	ClusterID				Renamed from VMHost.Parent.ID
	HostName1				Renamed from VMHost.Name
	HostName2				Renamed from VMHost.ExtensionData.Summary.Config.Name
	HostName3				Renamed from VMHost.NetworkInfo.HostName
	HostDomainName				Renamed from VMHost.NetworkInfo.DomainName
	HostCPUCount				Renamed from VMHost.ExtensionData.Summary.Hardware.NumCpuPkgs
	HostCPUCoreCount				Calculated (VMHost.ExtensionData.Summary.Hardware.NumC

File Name	Fields	Method	Source	Conditions (if any)	Remarks
					puCores/VMHost.ExtensionData.Summary.Hardware.NumCpuPkgs)
	HyperthreadingActive				
	HostManufacturer				Renamed from VMHost.Manufacturer
	HostID				Renamed from VMHost.Id
	HostUID				Renamed from VMHost.Uid
	HostPowerState				Renamed from VMHost.PowerState
	HostHyperThreading				Renamed from VMHost.HyperthreadingActive
	HostvMotionEnabled				Renamed from MHost.ExtensionData.Summary.Hardware.*
	HostVendor				
	HostModel				
	HostRAM				
	HostCPUModel				
	HostThreadCount				
	HostCPUSpeed				
	HostProductName				Renamed from VMHost.ExtensionData.Summary.Config.Product.*
	HostProductVersion				
	HostProductBuild				
	HostProductOS				
	HostProductLicenseName				
	HostProductLicense Version				
	HostProductLicenseKey				Renamed from VMHost.LicenseKey
	HostvMotionIPAddress				

File Name	Fields	Method	Source	Conditions (if any)	Remarks
	HostvMotionSubnetMask				Renamed from VMHost.ExtensionData.Config.vMotion.IPConfig.*
VMwareHosts.csv	VMHost	PowerShell script	Get-VM		Renamed from Name
	Name2				Renamed from ExtensionData.Summary.Config.Name
	Name3				Renamed from NetworkInfo.HostName
	ID				
	UID				
	HardwareUUID				Renamed from ExtensionData.Hardware.SystemInfo.Uuid
	VMwareUUID				
	Datacenter		Get-Datacenter		Renamed from Name
	Cluster		Get-VM		Renamed from Parent.Name
	ClusterID				Renamed from Parent.ID
	IsStandalone				
	DomainName				
	CPUCount				Renamed from ExtensionData.Summary.Hardware.NumCpuPkgs
	CPUCoreCount				Calculated (ExtensionData.Summary.Hardware.NumCpuCores/ExtensionData.Summary.Hardware.NumCpuPkgs)
	HyperthreadingActive				
	Manufacturer				
	PowerState				

File Name	Fields	Method	Source	Conditions (if any)	Remarks
	vMotionEnabled				
	Vendor				
	Model				
	RAM				Renamed from ExtensionData.Summary.Hardware.MemorySize
	CPUModel				
	ThreadCount				Renamed from ExtensionData.Summary.Hardware.NumCpuThreads
	CPUSpeed				Renamed from ExtensionData.Summary.Hardware.CpuMhz
	ProductName				Renamed from ExtensionData.Summary.Config.Product.*
	ProductVersion				
	ProductBuild				
	ProductOS				
	InstallDate				
	ProductLicenseKey				Renamed from LicenseKey
	ProductLicenseName				Renamed from ExtensionData.Summary.Config.Product.*
	ProductLicense Version				
	vMotionIPAddress				Renamed from ExtensionData.Config.vMotion.IPConfig.*
	vMotionSubnetMask				
VMwareClusters.csv	Name	PowerShell script	Get-Cluster		
	ID				
	UID				
	ParentID				

File Name	Fields	Method	Source	Conditions (if any)	Remarks
	HAEnabled				
	DrsEnabled				
	DrsMode				
	DrsAutomationLevel				
VMwareVMDRS Rules.csv	Name	PowerShell script	Get-DrsRule		
	Key				
	Uid				
	Cluster				
	ClusterId				
	ClusterUid				
	Enabled				
	Mandatory				
	Type				
	VMIDs				
	VMIDs2				
	VMUIDs				
	VMUUIDs				
	AffineHostGroupName				
	AntiAffineHostGroupName				
	VmGroupName				
VMwareVMDRS Groups.csv	Name	PowerShell script	Get-Cluster		
	Type				
	ClusterName				
	ClusterID				
	VMIDs				
	VMUIDs				
	VMNames				

File Name	Fields	Method	Source	Conditions (if any)	Remarks
	HostIDs				
	HostUIDs				
	HostNames				
	VMs				
	Hosts				
VMwareVMotionData.csv	CreatedTime	PowerShell script	Get-VMotionRecords		
	Type				
	TypeId				
	ID				
	UserName				
	VM				
	VMId				
	VMUid				
	SourceHost				
	DestinationHost				
	Description				
	ResourcePool				
Office365Licenses.csv	AccountName	PowerShell Script	Office 365 Account		
	SkuPartNumber				
	SkuName				
	SkuId				
	TargetClass				
	ActiveUnits				
	ConsumedUnits				
	LockedOutUnits				
	SuspendedUnits				
	WarningUnits				

File Name	Fields	Method	Source	Conditions (if any)	Remarks
Office365LicenseAssignments.csv	UserPrincipalName	PowerShell script	Office 365 Account		
	DisplayName				
	SignInName				
	Title				
	MobilePhone				
	PhoneNumber				
	ObjectId				
	UserType				
	Department				
	Office				
	StreetAddress				
	City				
	State				
	PostalCode				
	Country				
	UsageLocation				
	WhenCreated				
	LastPasswordChangeTimestamp				
	PasswordNeverExpires				
	IsBlackberryUser				
	LicenseReconciliationNeeded				
	PreferredLanguage				
	OverallProvisioningStatus				
	AlternateEmailAddresses				
	ProxyAddresses				
	LicenseSkus				
	LicenseNames				

File Name	Fields	Method	Source	Conditions (if any)	Remarks
	[Office365 Product SKUs Expanded]				
AzureClassicVM List.csv	SubscriptionId	PowerShell script	Azure Account Get-AzureVM		
	SubscriptionName				
	DefaultAccount				
	Environment				
	Deployment Name				
	VM Name				Renamed from Name
	Label				
	Host Name				
	Service Name				
	Availability Set				
	DNS Name				
	Instance Name				
	Instance Size				
	Power State				
	VM Status				
	VM IP Address				
	Public IP Address				
	Public IP Name				
	Virtual Network Name				
	OS				VM.OSVirtualHardDisk.OS
	VM Image Name				VM.OSVirtualHardDisk.SourceImageName
	Endpoints				
	Location				
	SubscriptionId	PowerShell script	Azure Account		

File Name	Fields	Method	Source	Conditions (if any)	Remarks
AzureRMVMList .csv	SubscriptionName				
	Resource Group Name				
	VM Name				
	License Type				
	Location				
	Availability Set				
	Instance Size				
	Admin Username				
	VM Provisioning State				
	Creation Method				
	Publisher				
	OS				
	VM Image Name				
	VM Image Version				
	VM Private IP Address				
	VM Private IP Allocation Method				
CPU.txt	Device	WMI Query			
	ProcessorName				
	MaxSpeed				
	ProcessorCount				
	CoreCount				Total Count of Cores
	LogicalProcessorCount				Total Count of LCores
	ProcessorCountStatusEnabled				
	ProcessorCountStatusDisabled				
	ProcessorCountStatusOther				
	Status				
Applications.txt	ComputerName				

File Name	Fields	Method	Source	Conditions (if any)	Remarks
	DisplayName	Remote Registry over WMI	*For installed applications:* <HKLM, HKU>, {'SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall\', 'SOFTWARE\Wow6432Node\Microsoft\Windows\CurrentVersion\Uninstall\'} *For Internet Explorer:* <HKLM, HKU>, {SOFTWARE\Microsoft\Internet Explorer}		
	InstallDate				
	VersionMajor				
	DisplayVersion				
	Publisher				
DNS.txt	ComputerName	WMI Query	{'MicrosoftDNS_Atype', 'MicrosoftDNS_CNAMEType' }		
	OwnerName				
	Primary Name				
	DomainName				
IP.txt	ComputerName	WMI Query	Win32_NetworkAdapterConfiguration	Where IPEnabled=TRUE	
	IPAddressCount				
	IPAddress				
	MACAddress				
EventLog.txt	ComputerName	WMI Query	Win32_NTLogEvent	Software Management events from Event Log	
	EventCode				
	Message				
	TimeWritten				
Processes.txt	ComputerName	WMI Query	Win32_Process		
	Name				
	ExecutablePath				
Services.txt	ComputerName	WMI Query	Win32_Service		
	DisplayName			Service Description	
	Name			Service Name	

File Name	Fields	Method	Source	Conditions (if any)	Remarks
	State				
	StartMode				
	Path				
System.txt	ComputerName				
	Name	WMI Query	Win32_OperatingSystem		
	Caption				
	TypeDescription				
	ComputerDomain				
	TotalPhysicalMemory				
	Model				
	Manufacturer				Name of a computer manufacturer
	BIOSManufacturer		Win32_BIOS		This value comes from the Vendor member of the BIOS Information structure in the SMBIOS information
	ScriptName				
UserLogons.txt	ComputerName				Remote Registry over WMI is used to get path and drive for every user - <HKLM>, {'SOFTWARE\MICROSOFT\WINDOWS NT\CURRENTVERSION\PROFILELIST', 'PROFILESDIRECTORY'}
	FileName	WMI Query	Win32_Directory	Hidden=false And Path= <path> And drive=<drive>	
	CreationDate				
	LastModified				
	LastAccessed				

File Name	Fields	Method	Source	Conditions (if any)	Remarks
SoftwareLicenseInfo.txt	ComputerName				
	Name	WMI Query	SoftwareLicensingProduct, OfficeSoftwareProtectionProduct	PartialProductKey<>null AND LicenseStatus<>0	
	Description				
	LicenseFamily				
	PartialProductKey				
	LicenseStatus				
	LicenseStatusReason				
	EvaluationEndDate				
	ExtendedGrace				
	GenuineStatus				
	GracePeriodRemaining				
	DiscoveredKeyManagementServiceMachineName				
	DiscoveredKeyManagementServiceMachinePort				
	KeyManagementServiceMachine				
	KeyManagementServicePort				
	ApplicationID				
	ProductKeyID				
SystemDates.txt	ComputerName	WMI Query	Win32_BIOS, Win32_OperatingSystem		
	BIOSDate				
	InstallDate				
	LastBootUpTime				
	LocalDateTime				
HyperV.txt	ComputerName	WMI or WbemScripting.SWbemLocator to	Msvm_VirtualSystemManagementService		
	Name				call to method ‘GetSummaryInformation’

File Name	Fields	Method	Source	Conditions (if any)	Remarks
	ElementName	root\virtualization			
	NumberofProcessors				
	CreationTime				
	EnabledState				Values On or Off
	GuestOperatingSystem				
AccessGroup.txt	ComputerName	WinNT query			
	GroupName				
	Member				
UserProfiles.txt	Computer				
	SID	WMI Query	Win32_UserProfile		
	LocalPath				
	Special				
	RoamingConfigured				
	LastUseTime				
EY_ADComputers.txt	Name				
	operatingSystem	ADODB	Active Directory Provider		
	operatingSystemVersion				
	whenCreated				
	distinguishedName				
	lastLogon				
	lastLogonTimestamp				
#EY_ADUsers.txt	UserName				
	DisplayName	ADODB	Active Directory Provider		
	GivenName				
	SirName				
	PwdLastSet				
	LastLogonTimeStamp				

File Name	Fields	Method	Source	Conditions (if any)	Remarks
	WhenCreated				
	WhenChanged				
#EY_ADGroups.txt	GroupName				
	GroupDescription	ADODB	Active Directory Provider		
	GroupWhenCreated				
	GroupWhenChanged				
	UserName				
	DisplayName				
	GivenName				
	SirName				
	PwdLastSet				
	LastLogonTimeStamp				
	WhenCreated				
	WhenChanged				
Cluster.txt	ComputerName				
	ClusterAlias	WMI or WbemScripting.SWbemLocator to root\MSCluster	MSCluster_Cluster, MSCluster_NodeToActiveResource, MSCluster_ResourceToPossibleOwner,		
	ClusterMode	{'Active'/'Passive'}			Static Value
	MSCluster_Node	WMI or WbemScripting.SWbemLocator to root\MSCluster	MSCluster_Cluster, MSCluster_NodeToActiveResource, MSCluster_ResourceToPossibleOwner,		
	MSCluster_Resource				
BizTalk.txt	Computer				
	InstallDate	Remote Registry over WMI	HKLM, 'SOFTWARE\Microsoft\BizTalk Server\3.0'		
	ProductEdition				
	ProductVersion				
	ProductName				

File Name	Fields	Method	Source	Conditions (if any)	Remarks
	FoundInRegistry	{'Found'/'Not Found'}	HKLM, {'SOFTWARE\Microsoft\BizTalk Server\3.0\ConfigFramework\ConfigModules\MSEBiz.Rule EngineConfig', 'SOFTWARE\Wow6432Node\Microsoft\BizTalk Server\3.0\ConfigFramework\ConfigModules\MSEBiz.Rule EngineConfig'}		Static Value
SharePoint.txt	ComputerName				
	Data Source	Remote Registry over WMI	HKLM, 'SOFTWARE\Microsoft\Shared Tools\Web Server Extensions\<key>\Secure\ConfigDB'		
	Initial Catalog		HKLM, 'SOFTWARE\Microsoft\Shared Tools\Web Server Extensions\<key>\Secure\ConfigDB'		
ISA.txt	ComputerName				
	Name	Remote Registry over WMI	HKU, '<path supplied if the application is present>'		
	Edition				
	Version				
DiskSpace.txt	Device	WMI Query	Win32_BIOS, Win32_OperatingSystem		
	Description				
	DeviceID				
	FileSystem				
	FreeSpace GB				
	Size GB				
	Status				
Exchange.txt	ComputerName				
	Name	Based on path and Static Array			
	Edition				
	Version				
SQL_Sharepoint_SiteFeatures.txt	ComputerName				
	Instance	Variable passed to function			

File Name	Fields	Method	Source	Conditions (if any)	Remarks
	DB	Variable passed to function			
	Host	Variable passed to function			
	FeatureId	ADODB			
	TimeActivated				
	Title				
	FullUrl				
	SiteId				
SQL_Sharepoint_Sites.txt	ComputerName				
	Instance	Variable passed to function			
	DB	Variable passed to function			
	Host	Variable passed to function			
	FullUrl	ADODB			
	Title				
	SiteId				
	Id				
	ParentWebId				
SQL_Sharepoint_Users.txt	ComputerName				
	Instance	Variable passed to function			
	DB	Variable passed to function			
	Host	Variable passed to function			
	SiteID	ADODB			
	ID				

File Name	Fields	Method	Source	Conditions (if any)	Remarks
	Deleted				
	SiteAdmin				
	IsActive				
	Login				
	Title				
SQL_Sharepoint_Config.txt	ComputerName				
	Instance	Variable passed to function			
	DB	Variable passed to function			
	Host	Variable passed to function			
	ServerId	ADODB			
	Product				
	Version				
SQL.txt	ComputerName				
	State	Variable passed to function			
	Source of Info	{'Service', 'Registry', 'WMI', 'Login'}			
	Instance Name	Variable passed to function			
	Edition	Remote Registry over WMI	HKLM, 'SOFTWARE\Microsoft\Microsoft SQL Server\Instance Names\SQL\Setup','Edition'		
	Version		HKLM, 'SOFTWARE\Microsoft\Microsoft SQL Server\Instance Names\SQL\Setup','Version'		

EY | Assurance | Tax | Transactions | Advisory

About EY
EY is a global leader in assurance, tax, transaction and advisory services. The insights and quality services we deliver help build trust and confidence in the capital markets and in economies the world over. We develop outstanding leaders who team to deliver on our promises to all of our stakeholders. In so doing, we play a critical role in building a better working world for our people, for our clients and for our communities.

EY refers to the global organisation and may refer to one or more of the member firms of Ernst & Young Global Limited, each of which is a separate legal entity. Ernst & Young Global Limited, a UK company limited by guarantee, does not provide services to clients. For more information about our organisation, please visit ey.com.

The Irish firm Ernst & Young is a member practice of Ernst & Young Global Limited. It is authorised by the Institute of Chartered Accountants in Ireland to carry on investment business in the Republic of Ireland.

Ernst & Young, Harcourt Centre, Harcourt Street, Dublin 2, Ireland.

ey.com