

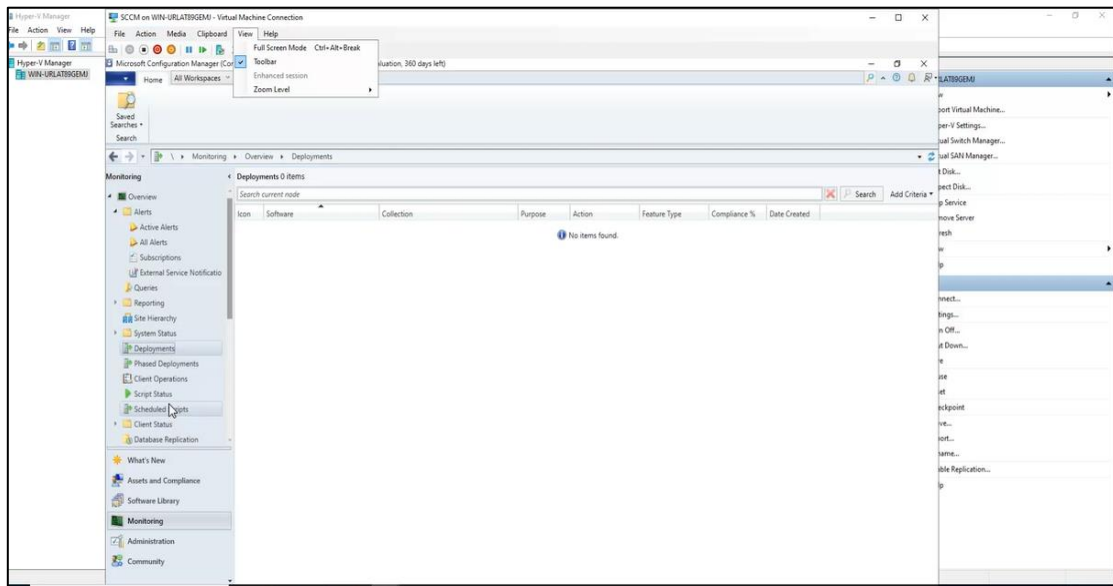
## Managing the Configuration Manager Client

### Discovery and Deployment:

**Discovery:** SCCM uses **build-in methods** like Active Directory Discovery to **locate devices** in your network **that don't have the client installed**.

**Deployment:** Install the client software using various methods, including **client push installation, software update-based installation, Group Policy, manual installation, etc.**

### Deployment:



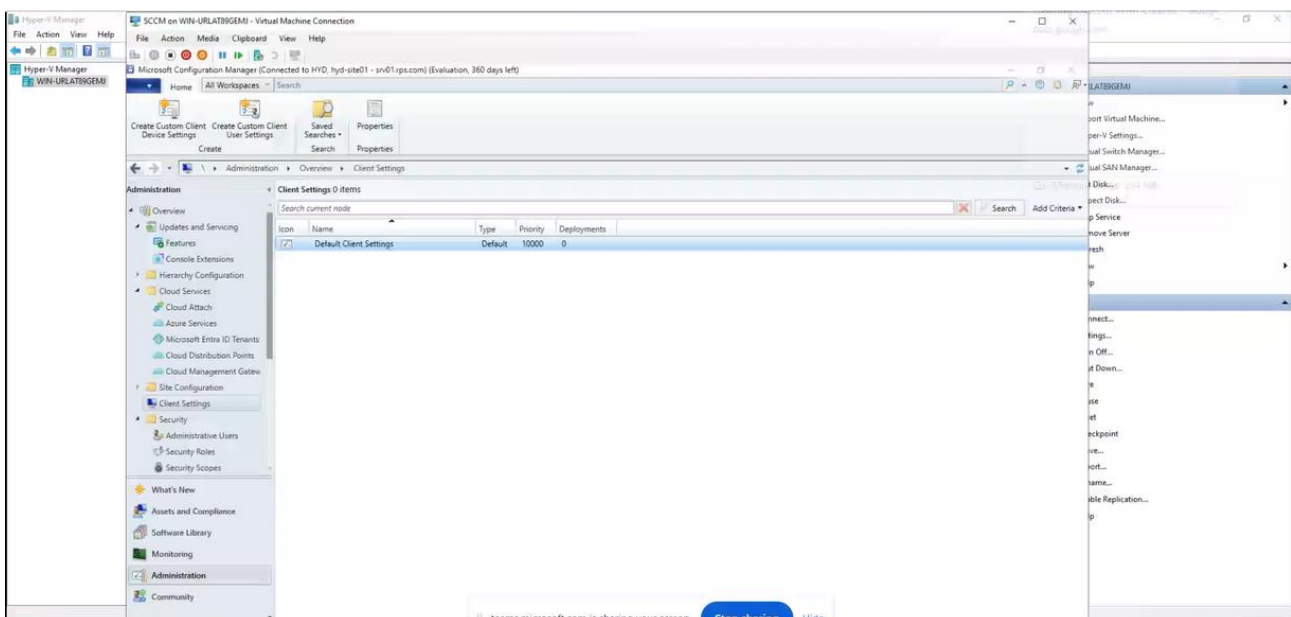
### Configuring Client Settings

**Access:** Manage all client settings through the Client Settings node (in the Administration)

**Default vs. Custom:** Default settings (apply to all clients), Custom settings (targeted to specific collections).

**Client Settings:** Includes Software Updates, Client Activity, and Client Check, to manage client behavior and compliance.

### Client Settings:



- **Managing the Client Cache**

The client cache stores files needed for deployments, like software updates and packages. Manage the space used to temporarily store downloaded files on the client.

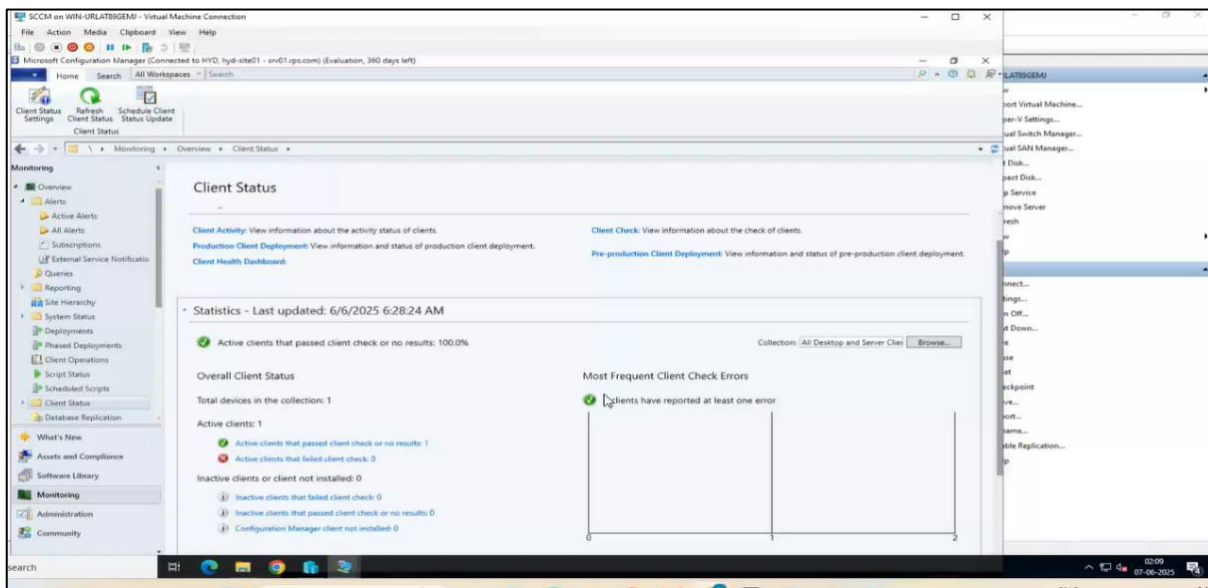
The **client cache** stores: Software packages, Updates, Scripts and also we can use "Delete Files" option in the control panel to remove files in the cache when needed.

- **Monitoring Client Status**

In Monitoring we check the status, View health, and alerts for all clients.

**Alerts** can notify you if a certain percentage of clients go inactive or fail health checks

## Client Status:

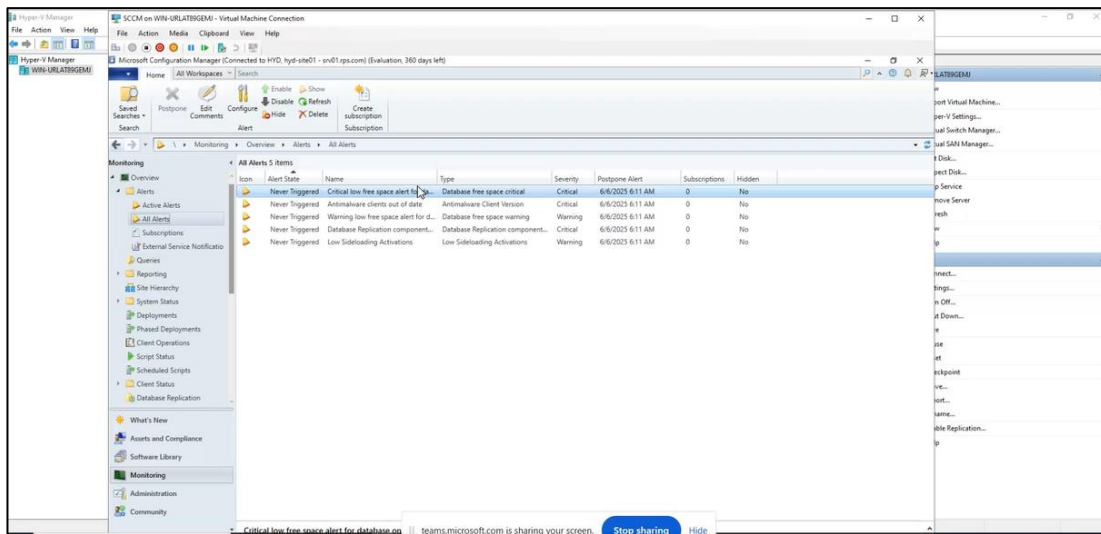


## Health:

The screenshot shows the 'Scenario Health' page in the Microsoft Configuration Manager console. The left sidebar is the same as the previous screenshot. The main pane displays 'Scenario Health 2 items' in a table. The table has columns for 'Icon', 'Scenario', 'Enabled', 'Run interval (minute)', 'Timeout (minute)', 'Start time', 'State', 'Duration (millisecond)', 'Last update time', 'Job id', and 'Next'. The table contains two rows: 'Client action health' and 'SQL Server service broker health'. The 'Client action health' row shows a green checkmark icon, 'Enabled' status, a 30-minute run interval, a 60-minute timeout, a start time of 6/9/2022, a 'Success' state, a duration of 5,320 milliseconds, a last update time of 6/9/2023 4:05 AM, and a job ID of 106488. The 'SQL Server service broker health' row shows a green checkmark icon, 'Enabled' status, a 30-minute run interval, a 60-minute timeout, a start time of 6/9/2022, a 'Success' state, a duration of 26 milliseconds, a last update time of 6/9/2023 4:05 AM, and a job ID of ED7A6.

Icon	Scenario	Enabled	Run interval (minute)	Timeout (minute)	Start time	State	Duration (millisecond)	Last update time	Job id	Next
✓	Client action health	Yes	30	60	6/9/2022	Success	5,320	6/9/2023 4:05 AM	106488	6/9/2023
✓	SQL Server service broker health	Yes	30	60	6/9/2022	Success	26	6/9/2023 4:05 AM	ED7A6	6/9/2023

## Alert:



- **Troubleshooting**

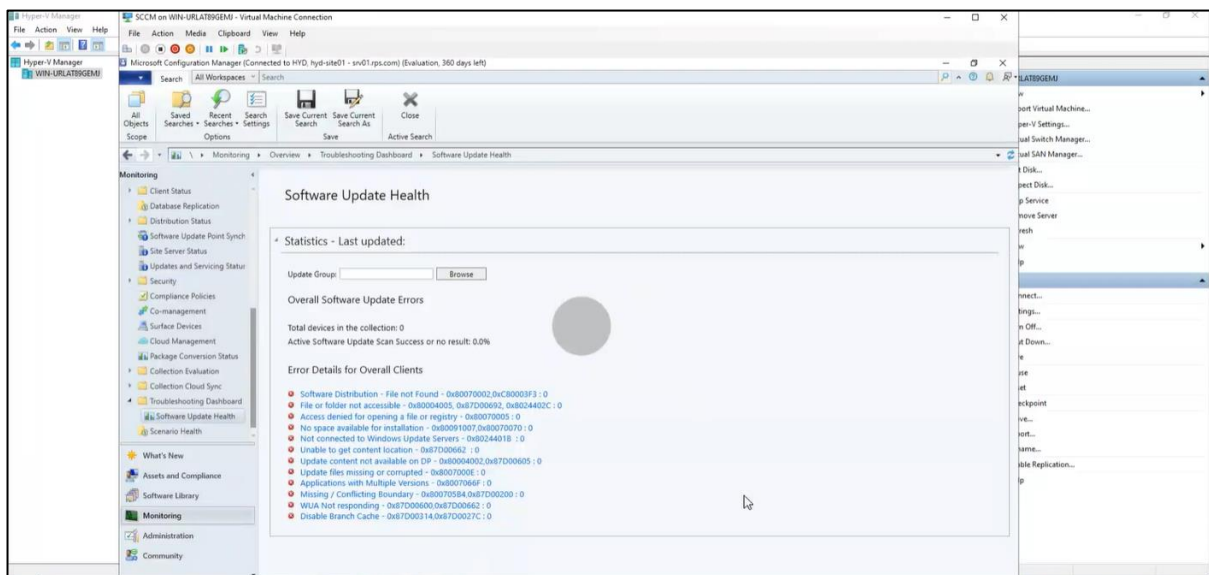
In Troubleshooting, fix issues using Software Center, Control Panel, and command-line tools. Client issues like failed installations, updates, or communication.

**Software Center:** Installed on each client, allows users to install apps, view updates, and see device compliance.

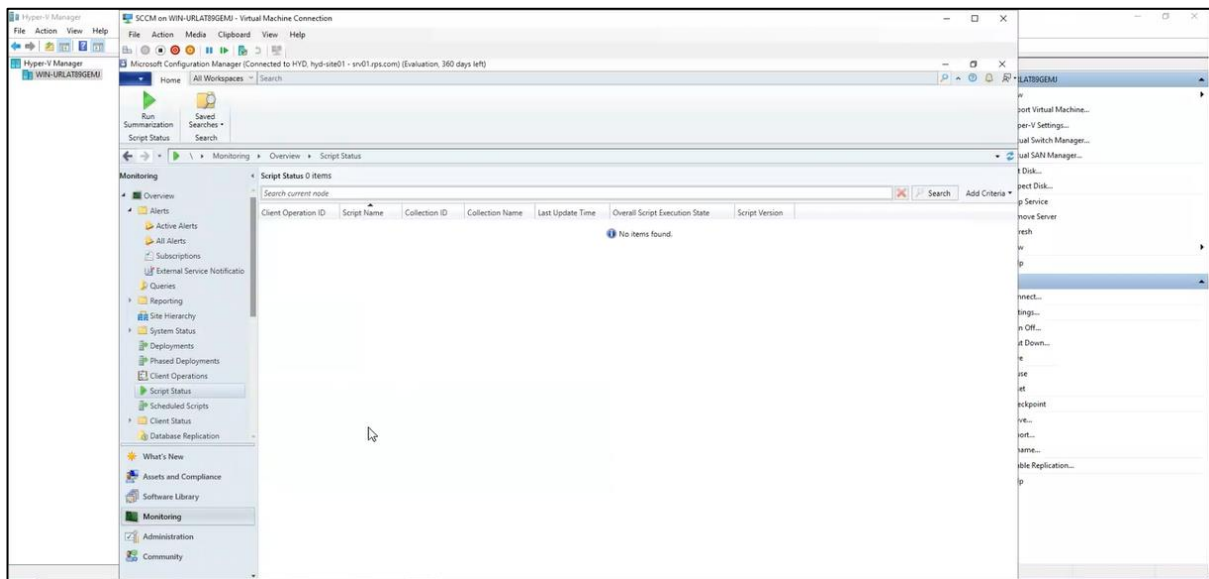
**ConfigMgr Control Panel Applet (smscfgrc):** Use the ConfigMgr client applet to check the client's core configuration, troubleshoot client-related issues.

**Command Line Tools:** Use tools like ccmrepair, cmsetup, and control smscfgrc for fixing and accessing client settings

## Troubleshooting Dashboard:



**Script Status:** Here the scripts are present and on the top we have **Run Summarization** to run the script



## Configure Software Metering

**Software Metering** in SCCM helps you **track software usage** on client computers.

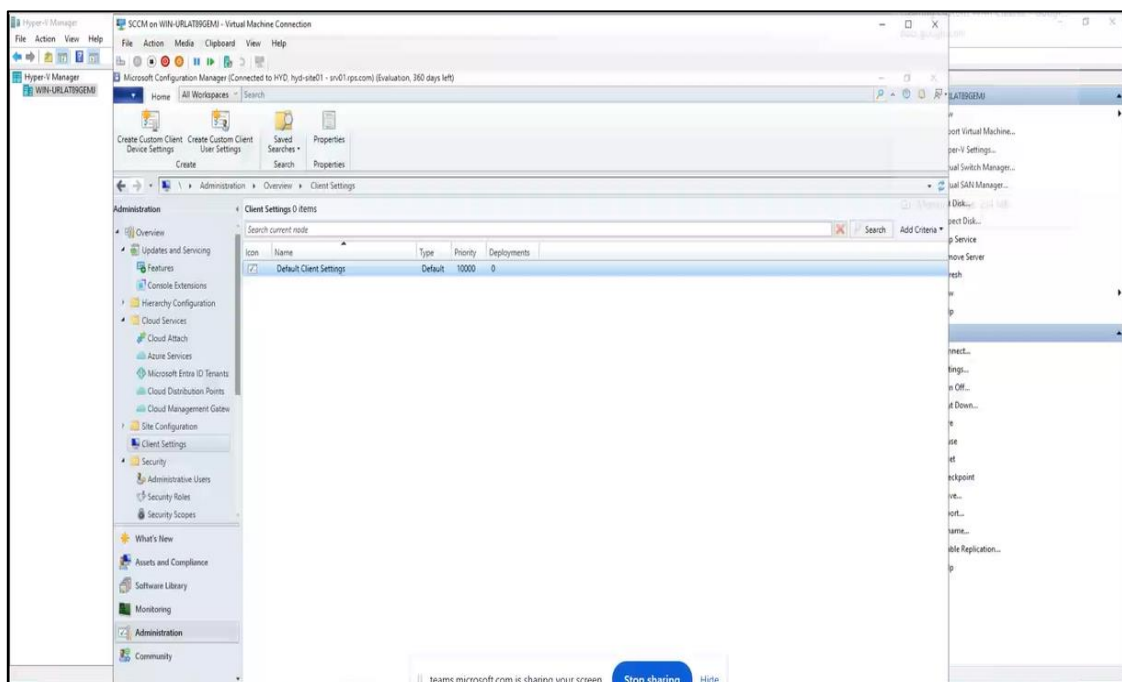
It tells us: Which applications are being used, How frequently they are used, Helps make decisions about software licenses (keep/remove).

This procedure configures the default client settings for software metering and applies to all computers in your hierarchy. If you want these settings to apply to only some computers, create a custom device client setting and deploy it to a collection that contains the computers on which you want to use software metering.

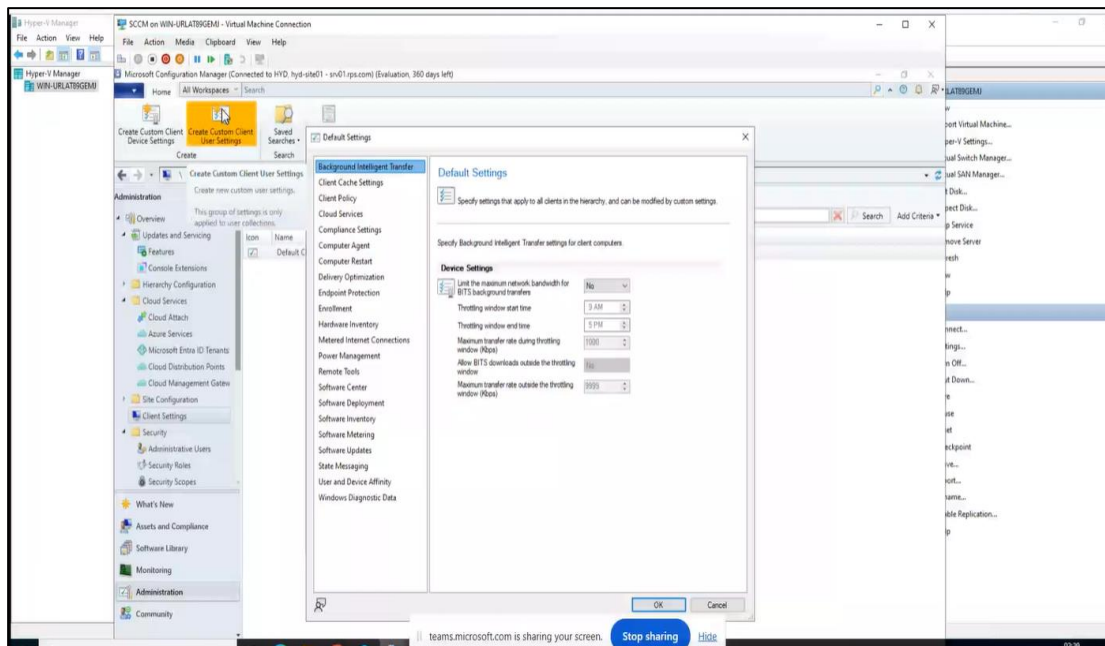
### Steps:

1. In the Configuration Manager console, click **Administration > Client Settings > Default Client Settings**.
2. On the **Home** tab, in the **Properties** group, click **Properties**.
3. In the **Default Settings** dialog box, click **Software Metering**.
4. In the **Device Settings** list, configure the following:
  - **Enable software metering on clients:** Select **True** to enable software metering.
  - **Schedule data collection:** Configure how often software metering data is collected from client computers. Use the default value of every **7 days** or click **Schedule** to specify a custom schedule.
5. Click **OK** to close the **Default Settings** dialog box.

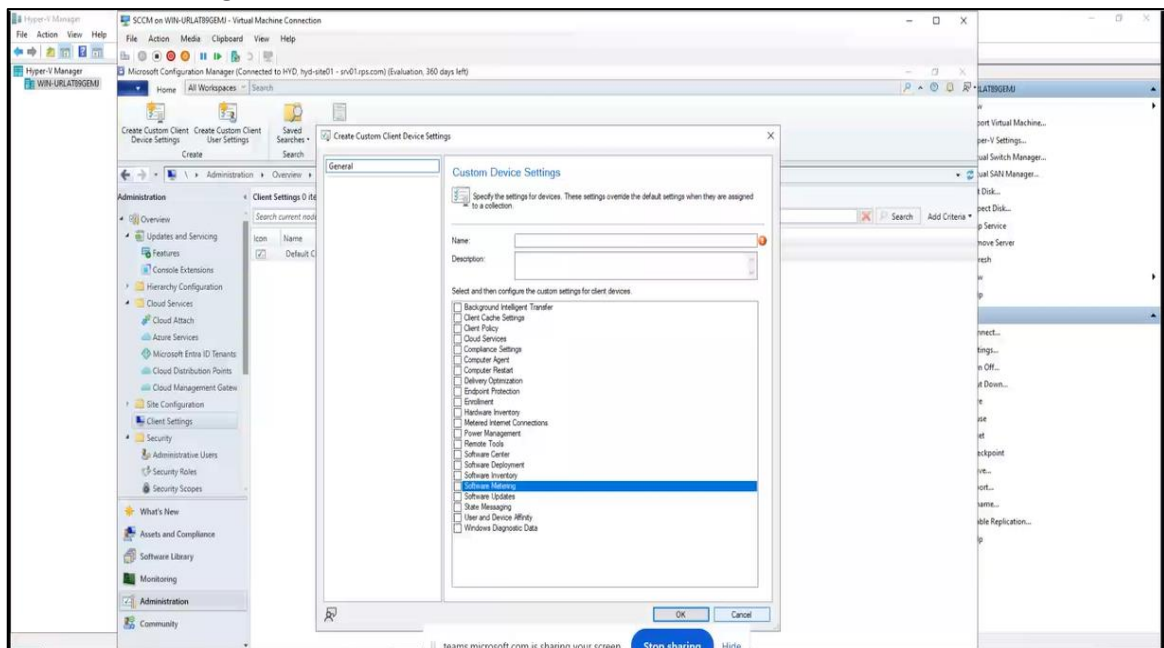
### Client Setting:



## Default Settings:



## Custom device settings :

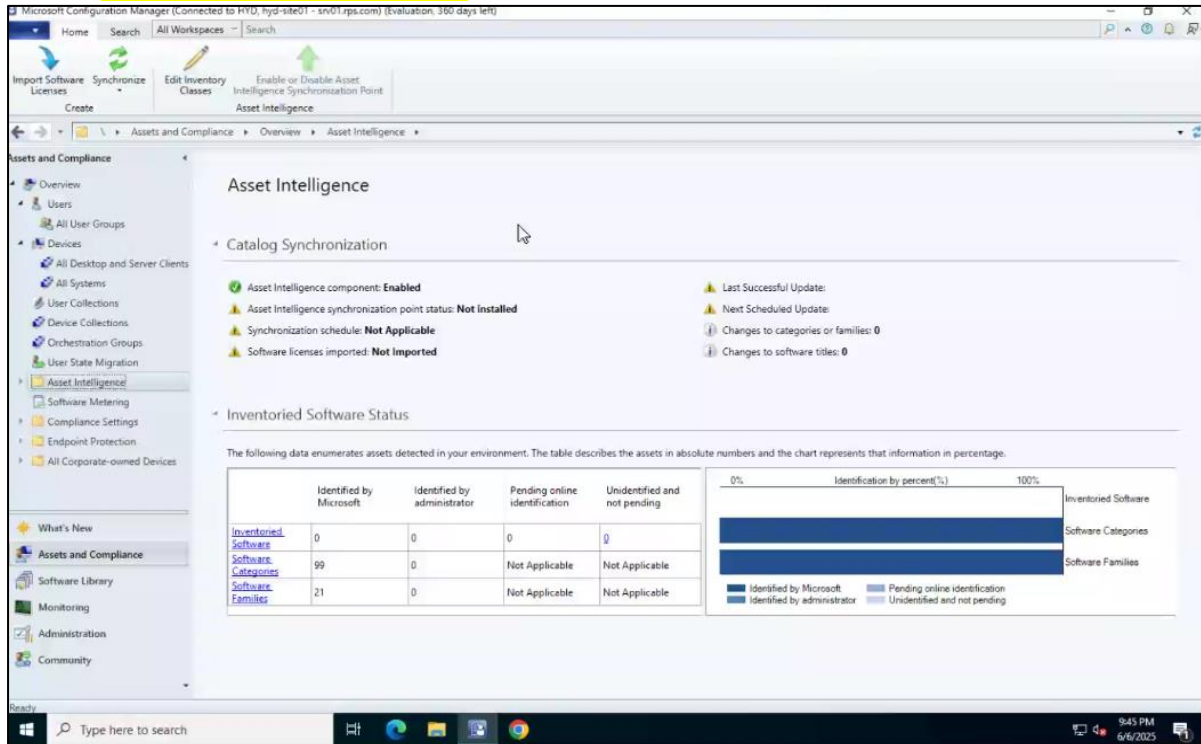




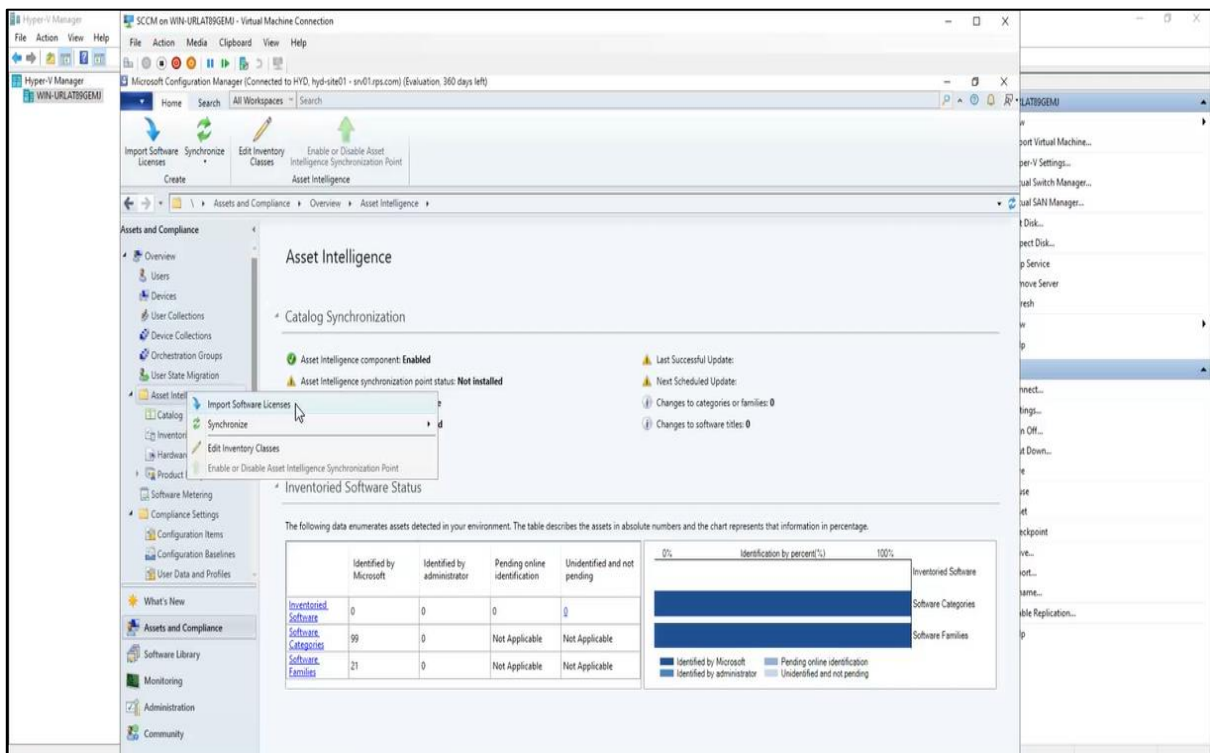
# Importing Software Licenses in Asset Intelligence

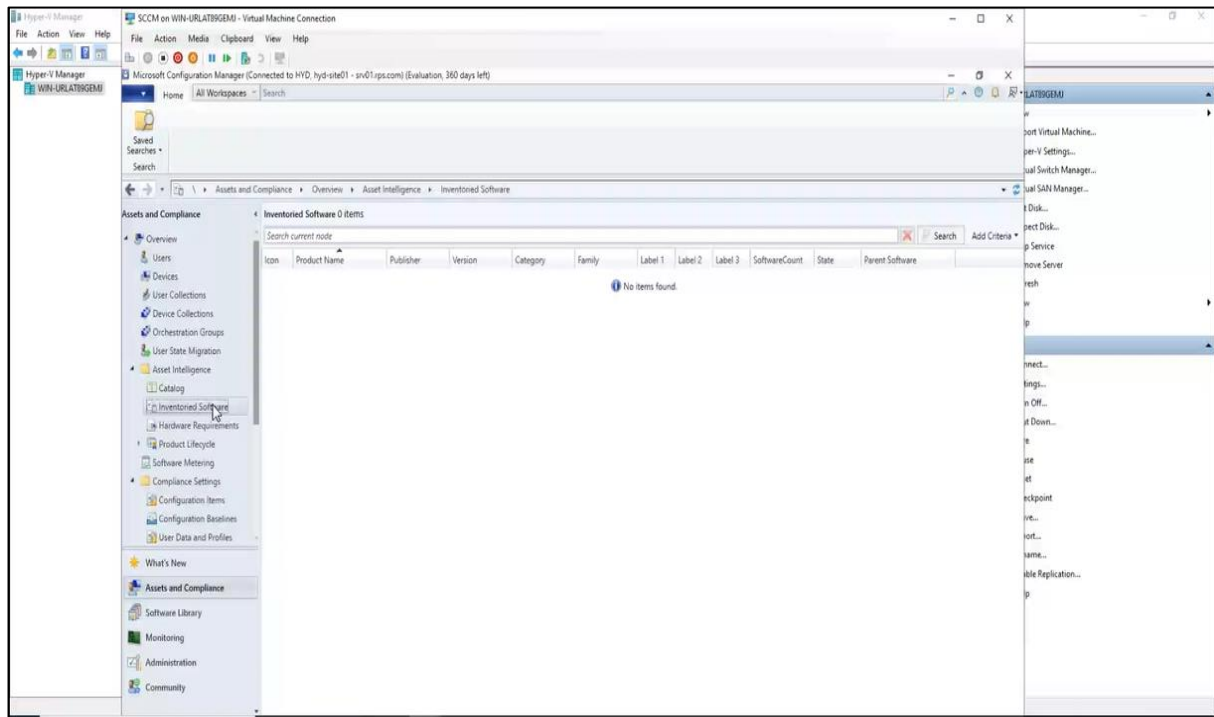
## Steps:

1. **Navigate to Asset Intelligence:** Go to Assets and Compliance -> Asset Intelligence



2. **Initiate the import :** Right click on **Asset Intelligence** and select “**Import Software Lincenses**”





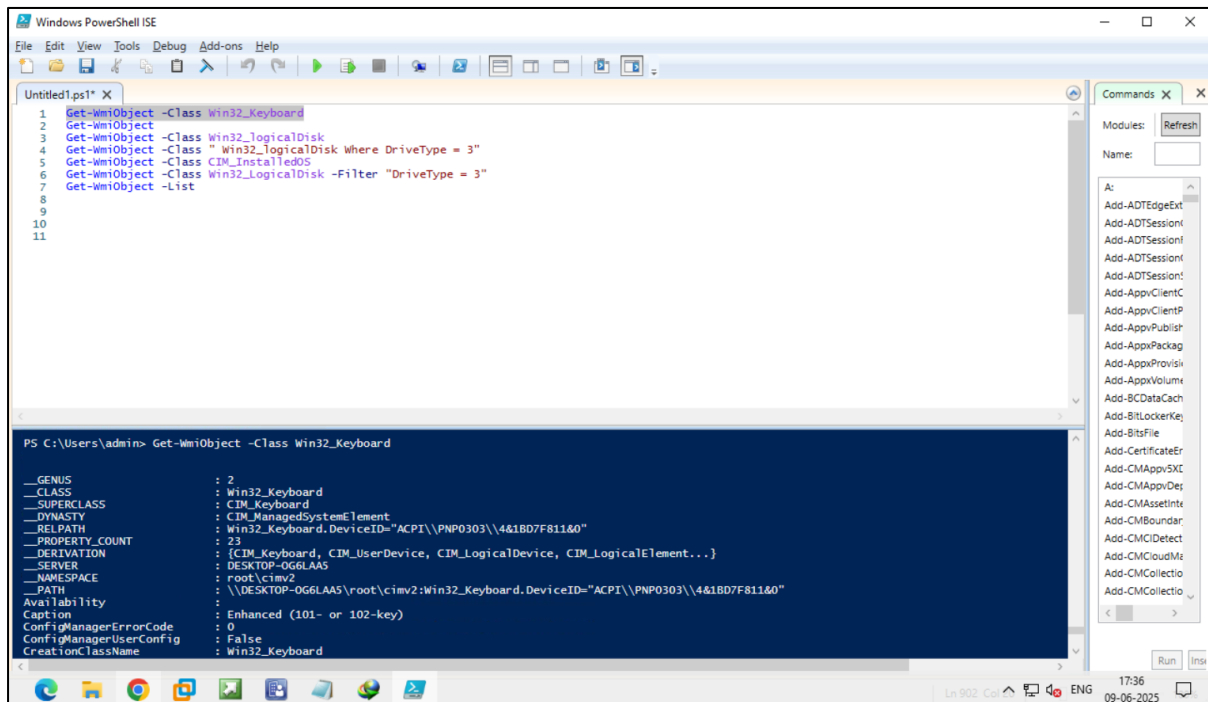
3. **Specify the License File Type:** Select an **MVLS file (.xml or .csv)** or a **General License Statement file (.csv)**
4. **Provide the License File Path:** Enter the **UNC** path to the license file or browse to select it from a **network share**.
5. **Complete the Wizard:** Follow the prompts of the **Import Software License Wizard** to complete the process.
6. **Verify Permissions:** Ensure the shared folder where the license file is located is properly secured and that the computer account running the wizard has **“Full Control”** permissions to the share.



## Get-WmiObject

**Get-WmiObject** is a PowerShell cmdlet used to **retrieve management information** from local and remote Windows computers using **Windows Management Instrumentation (WMI)**. WMI provides **access to information** about the operating system, hardware, and installed software.

- `Get-WmiObject -Class Win32_Keyboard`: it gets the information of the windows keyboard



The screenshot shows the Windows PowerShell ISE interface. The script editor contains the following code:

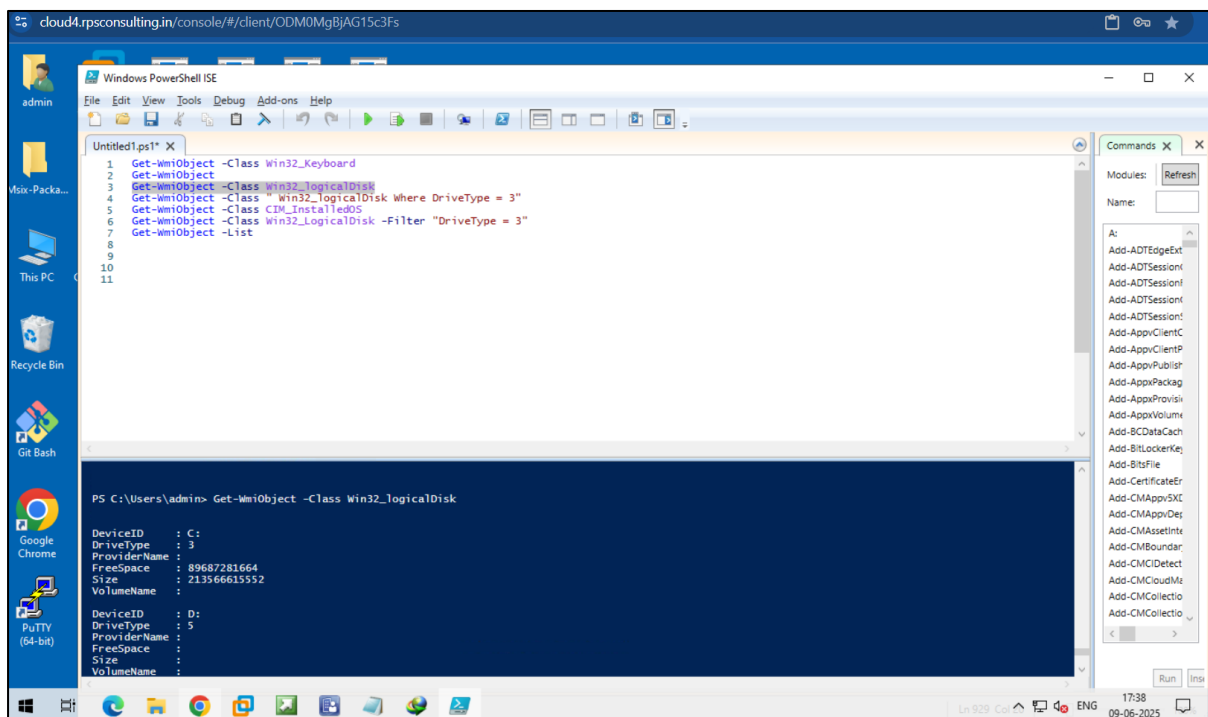
```
1 Get-WmiObject -Class Win32_Keyboard
2 Get-WmiObject
3 Get-WmiObject -Class Win32_LogicalDisk
4 Get-WmiObject -Class "Win32_LogicalDisk Where DriveType = 3"
5 Get-WmiObject -Class CIM_InstalledOS
6 Get-WmiObject -Class Win32_LogicalDisk -Filter "DriveType = 3"
7 Get-WmiObject -List
8
9
10
11
```

The console output shows the result of the command `Get-WmiObject -Class Win32_Keyboard`:

```
PS C:\Users\admin> Get-WmiObject -Class Win32_Keyboard

__GENUS              : 2
__CLASS              : Win32_Keyboard
__SUPERCLASS         : CIM_Keyboard
__DYNASTY             : CIM_ManagedSystemElement
__RELPATH             : Win32_Keyboard.DeviceID="ACPI\PNP0303\4&1BD7F811&0"
__PROPERTY_COUNT     : 23
__DERIVATION         : {CIM_Keyboard, CIM_UserDevice, CIM_LogicalDevice, CIM_LogicalElement...}
SERVER               : DESKTOP-OG6LA45
NAMESPACE            : root\cimv2
__PATH               : \\DESKTOP-OG6LA45\root\cimv2:Win32_Keyboard.DeviceID="ACPI\PNP0303\4&1BD7F811&0"
Availability         : Enhanced (101- or 102-key)
Caption              :
ConfigManagerErrorCode : 0
ConfigManagerUserConfig : False
CreationClassName     : Win32_Keyboard
```

- `Get-WmiObject -Class Win32_LogicalDisk`: Gives the information of the disk drive



The screenshot shows the Windows PowerShell ISE interface. The script editor contains the same code as the previous screenshot. The console output shows the result of the command `Get-WmiObject -Class Win32_LogicalDisk`:

```
PS C:\Users\admin> Get-WmiObject -Class Win32_LogicalDisk

DeviceID      : C:
DriveType     : 3
ProviderName  : 89687281664
FreeSpace     : 21356661552
VolumeName    :

DeviceID      : D:
DriveType     : 5
ProviderName  :
FreeSpace     :
Size         :
VolumeName    :
```

- `Get-WmiObject -Class "Win32_logicalDisk where DriveType=3"` : This filters the drive

```

1 Get-WmiObject -Class Win32_Keyboard
2 Get-WmiObject
3 Get-WmiObject -Class Win32_logicalDisk
4 Get-WmiObject -Class "Win32_logicalDisk where DriveType = 3"
5 Get-WmiObject -Class CIM_InstalledOS
6 Get-WmiObject -Class Win32_LogicalDisk -Filter "DriveType = 3"
7 Get-WmiObject -List
8
9
10
11

```

```

PS C:\Users\admin> Get-WmiObject -Class "Win32_logicalDisk where DriveType = 3"

DeviceID      : C:
DriveType     : 3
ProviderName  :
FreeSpace     : 89687261184
Size          : 213566615552
VolumeName    :

DeviceID      : E:
DriveType     : 3
ProviderName  :
FreeSpace     : 170848186368
Size          : 214729486336
VolumeName    : New Volume

```

- `Get-WmiObject -Class CIM_InstalledOS`: Gives the information about the Operating System which is installed in the System.

```

1 Get-WmiObject -Class Win32_Keyboard
2 Get-WmiObject
3 Get-WmiObject -Class Win32_logicalDisk
4 Get-WmiObject -Class "Win32_logicalDisk where DriveType = 3"
5 Get-WmiObject -Class CIM_InstalledOS
6 Get-WmiObject -Class Win32_LogicalDisk -Filter "DriveType = 3"
7 Get-WmiObject -List
8
9
10
11

```

```

PS C:\Users\admin> Get-WmiObject -Class CIM_InstalledOS

--GENUS      : 2
--CLASS      : Win32_SystemOperatingSystem
--SUPERCLASS : CIM_InstalledOS
--DYNASTY     : CIM_Component
--RELPATH    : Win32_SystemOperatingSystem.GroupComponent="\\\\DESKTOP-OG6LAAS\\root\\cimv2:Win32_ComputerSystem.Name='DESKTOP-OG6LAAS'",PartComponent="\\\\DESKTOP-OG6LAAS\\root\\cimv2:Win32_OperatingSystem=e"
--PROPERTY_COUNT : 3
--DERIVATION : {CIM_InstalledOS, CIM_SystemComponent, CIM_Component}
--SERVER     : DESKTOP-OG6LAAS
--NAMESPACE  : root\\cimv2
--PATH       : \\DESKTOP-OG6LAAS\\root\\cimv2:Win32_SystemOperatingSystem.GroupComponent="\\\\DESKTOP-OG6LAAS\\root\\cimv2:Win32_ComputerSystem.Name='DESKTOP-OG6LAAS'",PartComponent="\\\\DESKTOP-OG6LAAS\\root\\cimv2:Win32_OperatingSystem=e"
GroupComponent : \\DESKTOP-OG6LAAS\\root\\cimv2:Win32_ComputerSystem.Name="DESKTOP-OG6LAAS"
PartComponent  : \\DESKTOP-OG6LAAS\\root\\cimv2:Win32_OperatingSystem=e
PrimaryOS      : True

```

- `Get-WmiObject -Class Win32_LogicalDisk -Filter "DriverType=3"`: This filters the driverType 3 from the Disk Driver

The screenshot shows the Windows PowerShell ISE interface. The script editor contains the following commands:

```

1 Get-WmiObject -Class Win32_Keyboard
2 Get-WmiObject -Class Win32_LogicalDisk
3 Get-WmiObject -Class Win32_LogicalDisk Where DriveType = 3
4 Get-WmiObject -Class Win32_LogicalDisk Where DriveType = 3
5 Get-WmiObject -Class Win32_LogicalDisk -Filter "DriveType = 3"
6 Get-WmiObject -Class Win32_LogicalDisk -Filter "DriveType = 3"
7 Get-WmiObject -List
8
9
10
11

```

The console output shows the results of the command executed at line 5:

```

PS C:\Users\admin> Get-WmiObject -Class Win32_LogicalDisk -Filter "DriveType = 3"

DeviceID : C:
DriveType : 3
ProviderName : 89687670784
FreeSpace : 21356661552
Size : 21356661552
VolumeName :

DeviceID : E:
DriveType : 3
ProviderName : 170848186368
FreeSpace : 214729486336
Size : 214729486336

```

- `Get-WmiObject -List`: This gives us the class names which are present.

The screenshot shows the Windows PowerShell ISE interface. The script editor contains the following commands:

```

1 Get-WmiObject -Class Win32_Keyboard
2 Get-WmiObject -Class Win32_LogicalDisk
3 Get-WmiObject -Class Win32_LogicalDisk Where DriveType = 3
4 Get-WmiObject -Class Win32_LogicalDisk Where DriveType = 3
5 Get-WmiObject -Class Win32_LogicalDisk -Filter "DriveType = 3"
6 Get-WmiObject -Class Win32_LogicalDisk -Filter "DriveType = 3"
7 Get-WmiObject -List
8
9
10
11

```

The console output shows the results of the command executed at line 7:

```

PS C:\Users\admin> Get-WmiObject -List

Namespace: ROOT\WMI
Name      Methods      Properties
-----
SystemClass {}
__thisNAMESPACE
__Provider {}
__Win32Provider {}
__ProviderRegistration {}
__EventProviderRegistration {}
__ObjectProviderRegistration {}
__ClassProviderRegistration {}
__InstanceProviderRegistration {}
__MethodProviderRegistration {}
__PropertyProviderRegistration {}

```

- Get-WmiObject -Class Win32\_NetworkAdapter

The screenshot shows the Windows PowerShell ISE interface. The script editor contains a list of WMI classes to be queried. The console window shows the output of the command `Get-WmiObject -Class Win32_NetworkAdapter`, which returns details for two network adapters: 'kdnic' and 'RasSstp'.

```

1 Get-WmiObject -Class Win32_Keyboard
2 Get-WmiObject -Class Win32_LogicalDisk
3 Get-WmiObject -Class Win32_LogicalDisk
4 Get-WmiObject -Class "Win32_LogicalDisk Where DriveType = 3"
5 Get-WmiObject -Class CIM_InstalledOS
6 Get-WmiObject -Class Win32_LogicalDisk -Filter "DriveType = 3"
7 Get-WmiObject -List
8 Get-WmiObject -Class Win32_NetworkAdapter
9
10
11
12
13
14
15

```

```

PS C:\Users\admin> Get-WmiObject -Class Win32_NetworkAdapter

ServiceName      : kdnic
MACAddress       :
AdapterType      :
DeviceID         : 0
Name             : Microsoft Kernel Debug Network Adapter
NetworkAddresses :
Speed           :

ServiceName      : RasSstp
MACAddress       :
AdapterType      :
DeviceID         : 1
Name             : WAN Miniport (SSTP)

```

## All Commands

The screenshot shows a remote desktop session of a Windows 10 desktop. A Windows PowerShell ISE window is open, displaying a list of WMI classes. The console window shows the output of the command `Get-WmiObject -Class Win32_Keyboard`, which returns detailed information about the keyboard device, including its class, superclass, and various properties.

```

1 Get-WmiObject -Class Win32_Keyboard
2 Get-WmiObject -Class Win32_LogicalDisk
3 Get-WmiObject -Class Win32_LogicalDisk
4 Get-WmiObject -Class "Win32_LogicalDisk Where DriveType = 3"
5 Get-WmiObject -Class CIM_InstalledOS
6 Get-WmiObject -Class Win32_LogicalDisk -Filter "DriveType = 3"
7 Get-WmiObject -List
8 Get-WmiObject -Class Win32_NetworkAdapter
9
10
11
12
13
14
15

```

```

PS C:\Users\admin> Get-WmiObject -Class Win32_Keyboard

___GENUS          : 2
___CLASS          : Win32_Keyboard
___SUPERCLASS     : CIM_Keyboard
___DYNASTY        : CIM_ManagedSystemElement
___RELPATH        : Win32_Keyboard.DeviceID="ACPI\PNP0303\4&1B07F811&0"
___PROPERTY_COUNT : 23
___DERIVATION     : {CIM_Keyboard, CIM_UserDevice, CIM_LogicalDevice, CIM_LogicalElement...}
___SERVER         : DESKTOP-OG6LAAS
___NAMESPACE     : root\cimv2

```