**Autopilot Background Logging Solution Scripting Requirement**

Submitted to



**Ernst & Young L.L.P**

By



**Loadset Delivery Services Team**

**Wipro Technologies**

**25 October 2022**

Contents

[Document Details 2](#_Toc117887688)

[Document Version History 2](#_Toc117887689)

[Requirement Approval 3](#_Toc117887690)

[Process Flow 3](#_Toc117887691)

[Script Checklist 3](#_Toc117887692)

[1. Purpose of Script: 3](#_Toc117887693)

[2. Programming Language to be Used 4](#_Toc117887694)

[3. Script for PCB or Autopilot 4](#_Toc117887695)

[4. System Requirement 4](#_Toc117887696)

[5. New or Modification Request 4](#_Toc117887697)

[6. Detailed Requirement 4](#_Toc117887698)

[A. Items Section 4](#_Toc117887699)

[B. Scripting Details 14](#_Toc117887700)

[7. Deployment Tool 18](#_Toc117887701)

[8. Template Available 18](#_Toc117887702)

[9. Requestor 18](#_Toc117887703)

[10. Developer 19](#_Toc117887704)

[11. Target Completion Date 19](#_Toc117887705)

[12. Risk, if any 19](#_Toc117887706)

[13. Testing in Test Environment 19](#_Toc117887707)

[14. Testing in QA 19](#_Toc117887708)

[15. Production SR raised 20](#_Toc117887709)

[16. Attachments 20](#_Toc117887710)

# Document Details

|  |  |
| --- | --- |
| Project Name | Ernst & Young L.L.P |
| Tower | Loadset Delivery Services |
| Current Version | 14.0 |

# Document Version History

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Version | Date of Revision | Description | Author | Reviewed By | Approved By |
| 1.0 | 27/07/2022 | First Draft | Adarsh Shetty | Jaynendra C/Rohit Verma |  |
| 11.0 | 13/10/2022 |  | Sravanthi Gundavelly | Gyorgy |  |
| 12.0 | 18/10/2022 |  | Sravanthi Gundavelly | Gyorgy |  |
| 13.0 | 25/10/2022 |  | Sravanthi Gundavelly |  |  |
| 14.0 | 04/10/2022 |  | Sravanthi Gundavelly |  |  |

# Requirement Approval

This document is reviewed and approved by

| **Name** | **Role** | **Date of Approval** | **Remarks** |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# Process Flow

# Script Checklist

## Purpose of Script:

When troubleshooting Autopilot issues, it can be difficult to try to piece together a comprehensive story of what happened when and what state the PC was in at a certain point in time. To help with that, we need to design, implement, and test a solution to understand the state of the machine and its environment at regular intervals throughout the autopilot provisioning.

With this solution, we should be able to analyze what happened, when and what state the PC was in at a certain point in time.

## Programming Language to be Used

* PowerShell v5.1

## Script for PCB or Autopilot

* Autopilot

## System Requirement

* Any autopilot provisioned computer.

## New or Modification Request

* This is a new request to supplement the existing logging.

## Detailed Requirement

The solution should create two PowerShell scripts to record the state of various components in regular intervals for the whole duration of the provisioning process.

1. The first script is logging script , is responsible to create the log file with the below mentioned items.
2. The second script which is the deployment script is responsible to create the scheduled task which runs this logging script. We will configure the interval time in scheduled task to record the items in regular intervals . The scheduled task runs the logging script based on the specified interval time. If we want to change this interval time, we should update the deployment script which creates the scheduled task with repetition interval time.

The PowerShell script file has UTF-8 with BOM encoding.

### Items Section

**Below are the items/stages to be monitored**

|  |  |
| --- | --- |
| List Of Items to be Logged | |
|  | Computer name |
|  | Logged in username |
|  | Collect below network information |
|  | Power status of computer (Plugged in vs. Battery, Power Scheme). |
|  | Device state (AzureAdJoined, Domain Name) |
|  | Computer serial number |
|  | Device details (Device Id, Thumbprint of the Device Certificate) |
|  | Tenant details (Tenant Name, Tenant Id) |
|  | TPM details (TPM Version, TPM Ready) |
|  | Bitlocker status (Encryption Method, Protection Status, Conversion Status) |
|  | Deployment profile name |
|  | Enrollment start time |
| List of items related to PC Setup Assistant Application | |
|  | Initial setup start & end time |
|  | Critical applications start & end |
|  | Post setup start & end time |

**More details on how to & What to get about all the items.**

1. **Computer name**

Example: IN310010XYZ

WMI Class: Win32\_ComputerSystem

Property: Name

Log as: Computer Name

1. **Logged in username**

Example: US\URXXXYY

WMI Class: Win32\_ComputerSystem

Property: Username

Log as: Username

1. **Computer Power Status**

* Battery percentage

Example: 37

Namespace: root\cimv2

WMI Class: Win32\_Battery

Property: EstimatedChargeRemaining

Log as: Battery Percentage

* Plugged-in

Example: True

Namespace: root\wmi

WMI Class: BatteryStatus

Property: PowerOnLine

Log as: Plugged-In

* Power Scheme

Example: Balanced

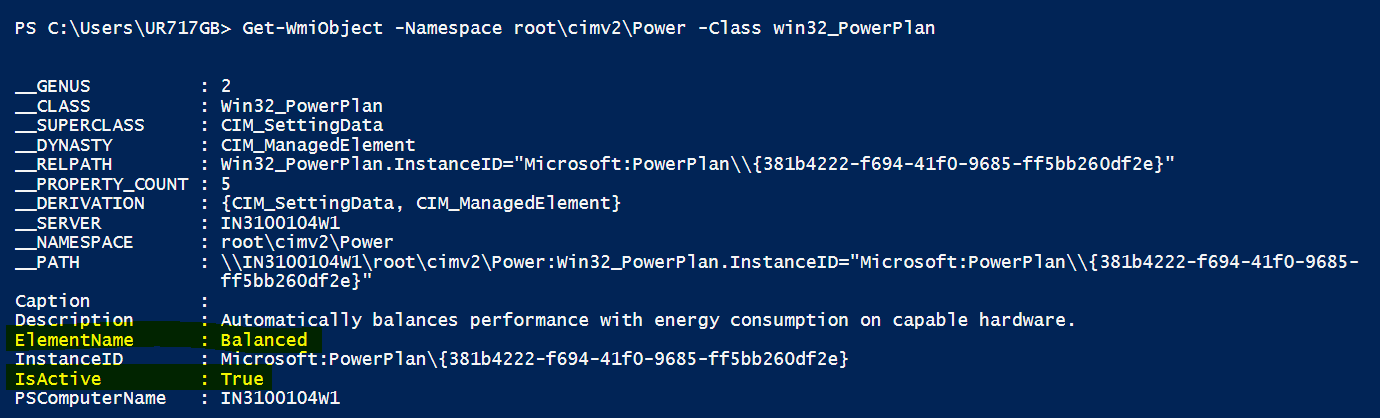
Namespace: root\cimv2\Power

WMI Class: Win32\_PowerPlan

Property: ElementName

Log as: Power Scheme

Please check the attached screenshot.



1. **Collect below network information**

To get the below details, we can use Get-NetAdapter command.

|  |  |  |
| --- | --- | --- |
| Interface Name | InterfaceDescription | Status |
| WI-FI | xxx | Up/disconnected |
| Ethernet | xxx | Up/disconnected |

Along with it use ipconfig /all command to get more network information. Print the output of this command to the log file. Give the header as Network information and print the output of both.

Log as **:** Network information

Output:

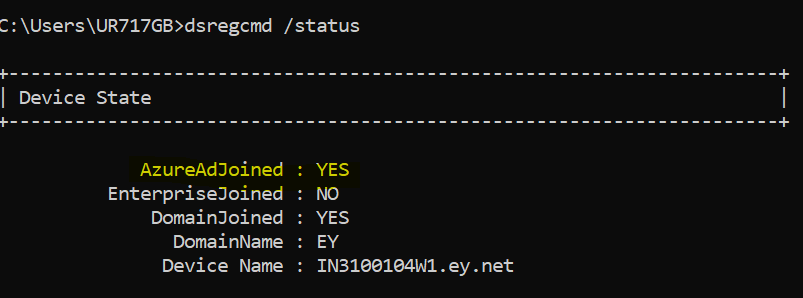
* + Add the above WIFI, Ethernet information as a table.
  + Add the output of ipconfig /all command. Please check the output of this command in attached [ipconfig log](#ipconfiglog) file.

1. **Device State**

* AzureAdJoined (Set the state to YES if the device is joined to Azure AD. Otherwise, set the state to NO.)

We can run the following command in command prompt:

dsregcmd /status to get the AzureAdJoined status. Please check the output of this command in [dsregcmd output](#dsregcmdoutput) file. Please check the screenshot as well.



Example:

Property: AzureAdJoined

Log as: AzureAdJoined: Yes

1. **Computer Serial Number**

Example: GTH3XYZ

WMI Class: win32\_bios

Property: SerialNumber

Log as: Computer Serial Number

1. **Device Details**

* Device Id: (The unique ID of the device in the Azure AD tenant.)

Get the “AadDeviceId” as Device Id from below json file. (C:\Windows\ServiceState\wmansvc\AutopilotDDSZTDFile.json)

Attached file contains below information. Collect the AadDeviceId from this file. Please check the attached file [AutopilotDDSZTDFile.json](#azureadjoinjson).

{"AutopilotServiceCorrelationId":"37c6b380-4970-44aa-a3d3-c42c3f21ac3a","ZtdRegistrationId":"a6969a7f-baa2-48f1-a338-670f04d7673d","AadDeviceId":"aab4e4ff-b23e-4e11-b852-f2d10b7fc0a6","CloudAssignedOobeConfig":286,"CloudAssignedDomainJoinMethod":0,"CloudAssignedForcedEnrollment":1,"CloudAssignedTenantDomain":"EYGS.onmicrosoft.com","CloudAssignedTenantId":"5b973f99-77df-4beb-b27d-aa0c70b8482c","CloudAssignedMdmId":"9cb77803-d937-493e-9a3b-4b49de3f5a74","CloudAssignedDeviceName":"XW%SERIAL%","DeploymentProfileName":"2100\_G\_StandardUserProfile","IsExplicitProfileAssignment":true,"CloudAssignedAutopilotUpdateDisabled":1,"CloudAssignedPrivacyDiagnostics":0,"HybridJoinSkipDCConnectivityCheck":0,"CloudAssignedAutopilotUpdateTimeout":1800000,"PolicyDownloadDate":"2022-09-13T15:14:02Z"}

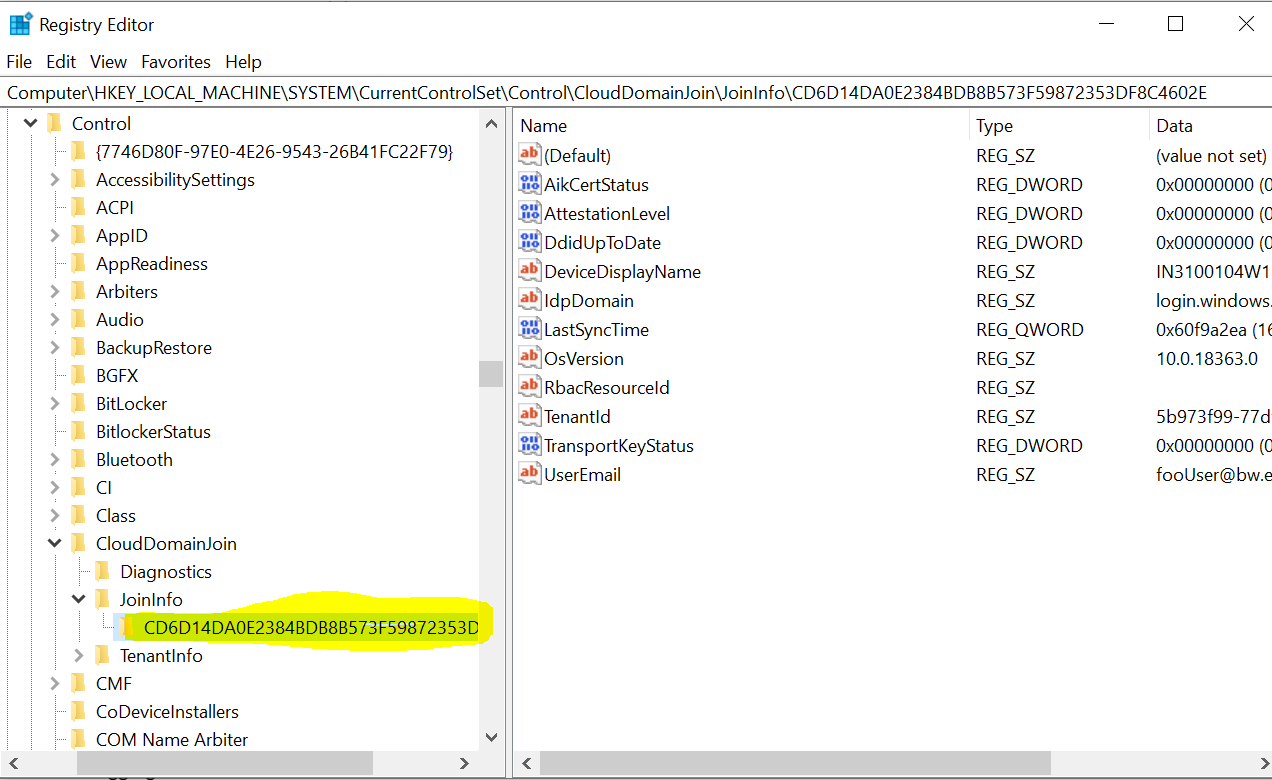
* The thumbprint of the device certificate

Get the below registry path key name as thumbprint or we can get it from dsregcmd command. I suggest using dsregcmd, because already we are using this command to get the tenant information.

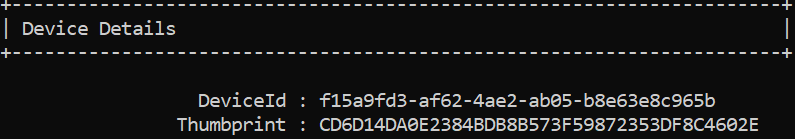
HKEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\Control\CloudDomainJoin\JoinInfo

Please check the attached registry export file here as [thumbprint reg](#thumbprintreg).

Attached screenshot of thumbprint value from registry path.



If we want to get it from dsregcmd command, the output looks like attached screenshot and check the attached [dsregcmd output](#dsregcmdoutput) file.



Example:

Property: thumbprint

Log as: The thumbprint of the device certificate.

1. **Tenant Details**

* Tenant Id

My suggestion is to use dsregcmd command, because to get the Tenant Name we are already using this command. So, we can retrieve both the values by using this command.

We can get this value from both the ways.

Use below file or use the dsregcmd command.

Get the CloudAssignedTenantId as tenant Id from below json file. (C:\Windows\ServiceState\wmansvc\AutopilotDDSZTDFile.json)

**Example**:

Attached file contains below info. Collect the CloudAssignedTenantId from this file. Check the attached file [AutopilotDDSZTDFile.json](#azureadjoinjson).

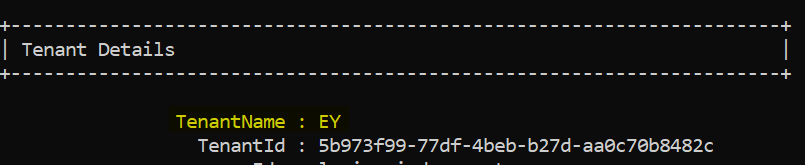
{"AutopilotServiceCorrelationId":"37c6b380-4970-44aa-a3d3-c42c3f21ac3a","ZtdRegistrationId":"a6969a7f-baa2-48f1-a338-670f04d7673d","AadDeviceId":"aab4e4ff-b23e-4e11-b852-f2d10b7fc0a6","CloudAssignedOobeConfig":286,"CloudAssignedDomainJoinMethod":0,"CloudAssignedForcedEnrollment":1,"CloudAssignedTenantDomain":"EYGS.onmicrosoft.com","CloudAssignedTenantId":"5b973f99-77df-4beb-b27d-aa0c70b8482c","CloudAssignedMdmId":"9cb77803-d937-493e-9a3b-4b49de3f5a74","CloudAssignedDeviceName":"XW%SERIAL%","DeploymentProfileName":"2100\_G\_StandardUserProfile","IsExplicitProfileAssignment":true,"CloudAssignedAutopilotUpdateDisabled":1,"CloudAssignedPrivacyDiagnostics":0,"HybridJoinSkipDCConnectivityCheck":0,"CloudAssignedAutopilotUpdateTimeout":1800000,"PolicyDownloadDate":"2022-09-13T15:14:02Z"}

* Tenant Name

Get the Tenant name by running the dsregcmd command. Check the attached [dsregcmd output](#dsregcmdoutput) file to see know the output of this command.

Example: Run the command like attached screenshot& collect the Tenant Name.





1. **TPM Details**

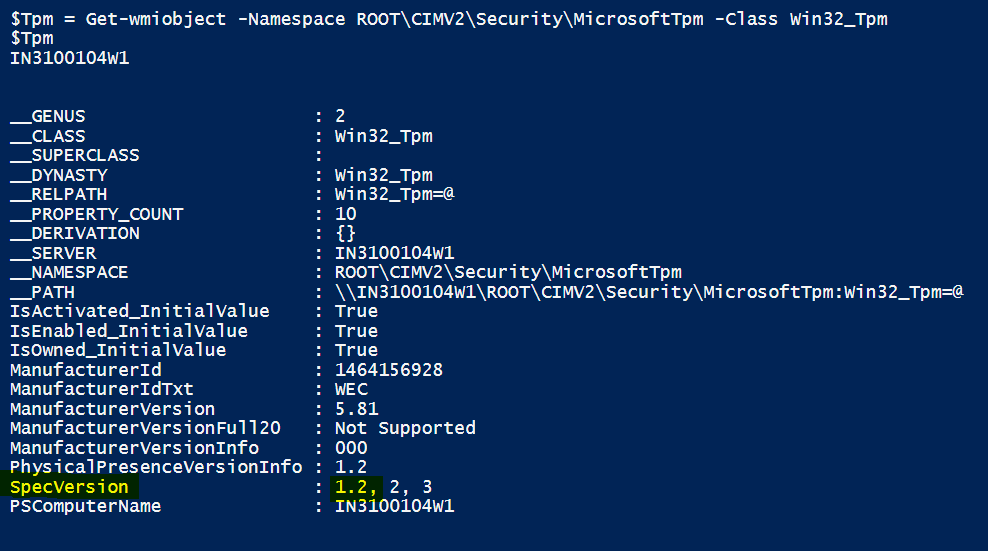
* TPM Version

WMI Class: Win32\_TPM

Property: SpecVerison (Collect major version from SpecVerison value as TPM version.)

Log as: TPM Version

Example: Here the TPM Version is 1.2. please check the screenshot.



(Note: The version of the Trusted Computing Group (TCG) specification that the TPM supports. This value includes the major and minor TCG specification version, the specification revision level, and the errata revision level. All values are in hexadecimal. For example, a version information of "1.2, 2, 0" indicates that the device was implemented to TCG specification version 1.2, revision level 2, and with no errata. When the data is unavailable, "Not Supported" is returned.)

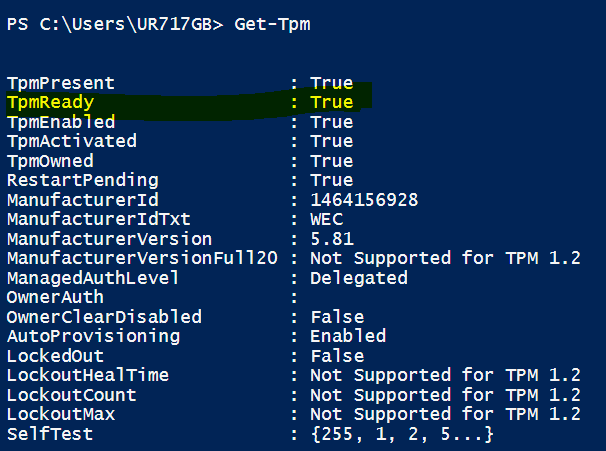
* TPM Ready

Command: Get-Tpm

Property: TPMReady

Log as: TPM Ready

Example: Check the attached screenshot.



1. **Bit locker status**

We can use Get-BitLockerVolume to get the below Bitlocker details.

* Encryption Method

Example: Xtsaes256

Property: EncryptionMethod

Log as: Encryption Method

* Protection Status

Example: On

Property: ProtectionStatus

Log as: Protection Status

* Conversion Status

Example: Fully Encrypted

Property: VolumeStatus

Log as: Conversion Status

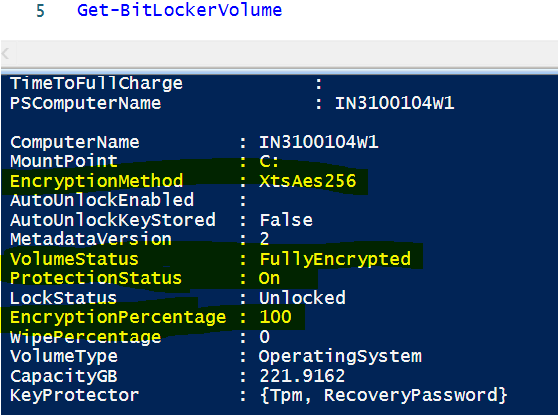
* Encryption Percentage

Example: 100

Property: EncryptionPercentage

Log as: Encryption Percentage

Please check the attached screenshot for more clarity.



1. **Deployment profile name**

Get the DeploymentProfilename from json file. (C:\Windows\ServiceState\wmansvc\AutopilotDDSZTDFile.json)

Attached file contains below info. Collect the DeploymentProfilename from this file. Please check the attached file [AutopilotDDSZTDFile.json](#azureadjoinjson).

{"AutopilotServiceCorrelationId":"37c6b380-4970-44aa-a3d3-c42c3f21ac3a","ZtdRegistrationId":"a6969a7f-baa2-48f1-a338-670f04d7673d","AadDeviceId":"aab4e4ff-b23e-4e11-b852-f2d10b7fc0a6","CloudAssignedOobeConfig":286,"CloudAssignedDomainJoinMethod":0,"CloudAssignedForcedEnrollment":1,"CloudAssignedTenantDomain":"EYGS.onmicrosoft.com","CloudAssignedTenantId":"5b973f99-77df-4beb-b27d-aa0c70b8482c","CloudAssignedMdmId":"9cb77803-d937-493e-9a3b-4b49de3f5a74","CloudAssignedDeviceName":"XW%SERIAL%","DeploymentProfileName":"2100\_G\_StandardUserProfile","IsExplicitProfileAssignment":true,"CloudAssignedAutopilotUpdateDisabled":1,"CloudAssignedPrivacyDiagnostics":0,"HybridJoinSkipDCConnectivityCheck":0,"CloudAssignedAutopilotUpdateTimeout":1800000,"PolicyDownloadDate":"2022-09-13T15:14:02Z"}

1. **Enrollment start time**

To get the Enrollment start time follow below steps.

* Check for the “DeviceEnroller” key name in all the subkeys with in the below Registry.

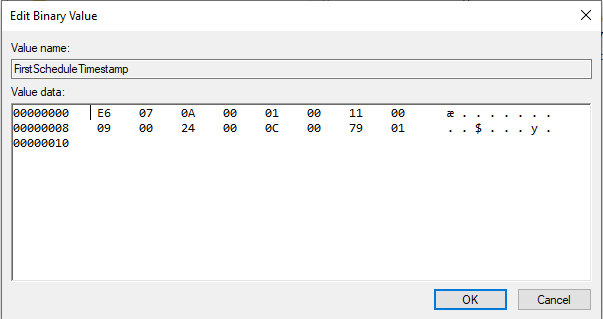
HKEY\_LOCAL\_MACHINE\SOFTWARE\Microsoft\Enrollments

* If DeviceEnroller key founds check for the value name “FirstScheduleTimestamp”. If it found, get the value data. The DeviceEnroller key looks like below image (Highlighted in yellow color).
* Convert this binary timestamp data to local Date format.

The date format is dd/mm/YYYY hh:mm:ss

**How to convert this reg\_binary value to date format?**

* The input date is in reg\_binary with the hexadecimal value like below image. Convert this to specified date format.
* Read the input data from the given path, and here the output type, we will get it as object and convert it to the byte array. Check the length of this byte array is 16 or not. If yes, convert this byte array to date.



Please check the below example implementation script and check the attached example script file [enrollmentdate example script](#enrollmentscript) and check this [URL](https://www.anoopcnair.com/intune-app-ps-script-based-enrollment-date/) to get more details about script.

Function GetRegDate ($path, $key){

function GVl ($ar){

return [uint32]('0x'+(($ar|ForEach-Object ToString X2) -join ''))

}

$ar=Get-ItemPropertyValue $path $key

[array]::reverse($ar)

$time = New-Object DateTime (GVl $ar[14..15]),(GVl $ar[12..13]),(GVl $ar[8..9]),(GVl $ar[6..7]),(GVl $ar[4..5]),(GVl $ar[2..3]),(GVl $ar[0..1])

return $time

}

$RegKey = (@(Get-ChildItem "HKLM:\SOFTWARE\Microsoft\Enrollments" -recurse | Where-Object {$\_.PSChildName -like 'DeviceEnroller'}))

$RegPath = $($RegKey.name).TrimStart("HKEY\_LOCAL\_MACHINE")

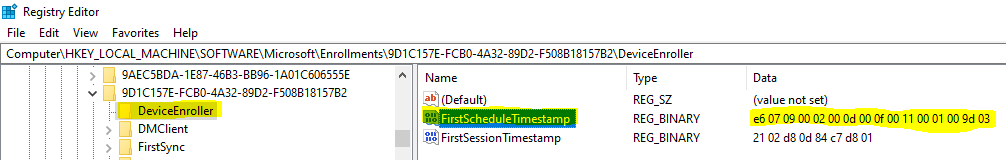
$RegDate = GetRegDate HKLM:\$RegPath "FirstScheduleTimestamp"

$DeviceEnrolmentDate = Get-Date $RegDate

$DeviceEnrolmentDate

Example: Below we search the DeviceEnroller in the given registry path. So, it is existed under the subkey which is starts with (9D1C57E……). Here get the FirstScheduleTimestamp value and convert it to the date.

Check the registry export file [DeviceEnroller Key reg.txt](#deviceenrollerregkey)



1. **PC Setup Assistant Application Information**

Get Initial setup start & end time, Critical applications start & end time, Post setup start & end time from below registry path. Get the date format as it is from below registry.

HKEY\_LOCAL\_MACHINE\SOFTWARE\WOW6432Node\Ernst & Young\GlobalPackageID

Check PCSetupAssistant-Step-1 Key for Initial Setup start & end time

Capture GlobPackIDVal1 for Initial Setup start time

Capture GlobPackIDVal2 for Initial Setup end.

Check PCSetupAssistant-Step-2 Key for Critical Setup start & end time

Capture GlobPackIDVal1 for Critical Setup start time

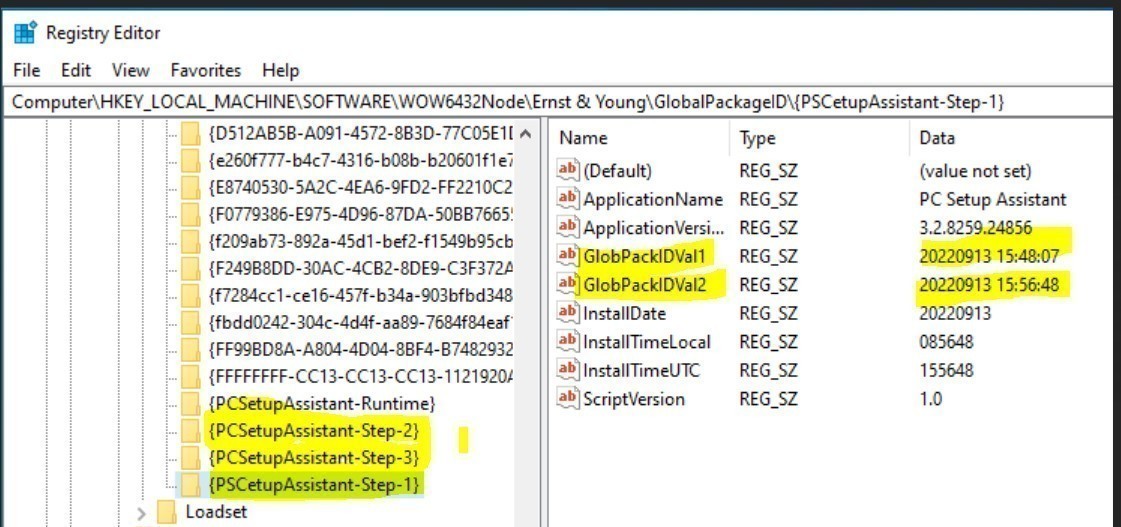
Capture GlobPackIDVal2 for Critical Setup end.

Check PCSetupAssistant-Step-3 Key for Post Setup start & end time

Capture GlobPackIDVal1 for Post Setup start time

Capture GlobPackIDVal2 for Post Setup end.

Please check the attached screenshot of registry and export file as [PC Setup Assistant reg.txt](#pcsetupreg).



How to log? – Give header as “PC Setup Assistant Application Information” and display the phase name, start & end time. If value does not exist, keep it as n/a.

Example:

PC Setup Assistant Application Information

Phase: Initial Phase

Start time:

End time:

### Scripting Details

We required 2 scripts to fulfill the solution requirements.

The script should handle the errors by using try/catch block. And it should not stop the script execution, if any error occurred, the script should be able to record the errors and continue the script execution. Please check this sample script file [EYCollectLogs.txt](#eycollectlogstxt)  to know, how the error handling is implemented.

1. Logging script:

* Create the logging script. Name it as “Autopilot\_Diagnostics\_Provider.ps1”.
* The script should be copied to the below location.

C:\MAINTENANCE\Installers\Autopilot Troubleshooting\Autopilot\_Diagnostics\_Provider.ps1

* This script is responsible to create two logs.

1. One log file is responsible to capture the script workflow (log the details like, is the script file copied to the mentioned location etc.). Create this log file with below details.

Log Filename: Autopilot\_Diagnostics\_PS\_Flow.Log

Date: local

Date Format: dd/MM/yyyy HH:mm:ss.ms (Example:14-10-2022 21:25:47.362)

Log Location: C:\Maintenance\Logs\Autopilot Troubleshooting

File Encoding: UTF-8

Added sample workflow example log file [EYLogGather-x64IsTrue.log](#workflowsamplelog).

* The second log file should record the above-mentioned items.

Log Filename: Autopilot\_Diagnostics\_Provider.Log

Date: local

Date Format: dd/MM/yyyy HH:mm:ss.ms (Example:14-10-2022 21:25:47.362)

Log Location: C:\Maintenance\Logs\Autopilot Troubleshooting

File Encoding: UTF-8

List of Static properties:

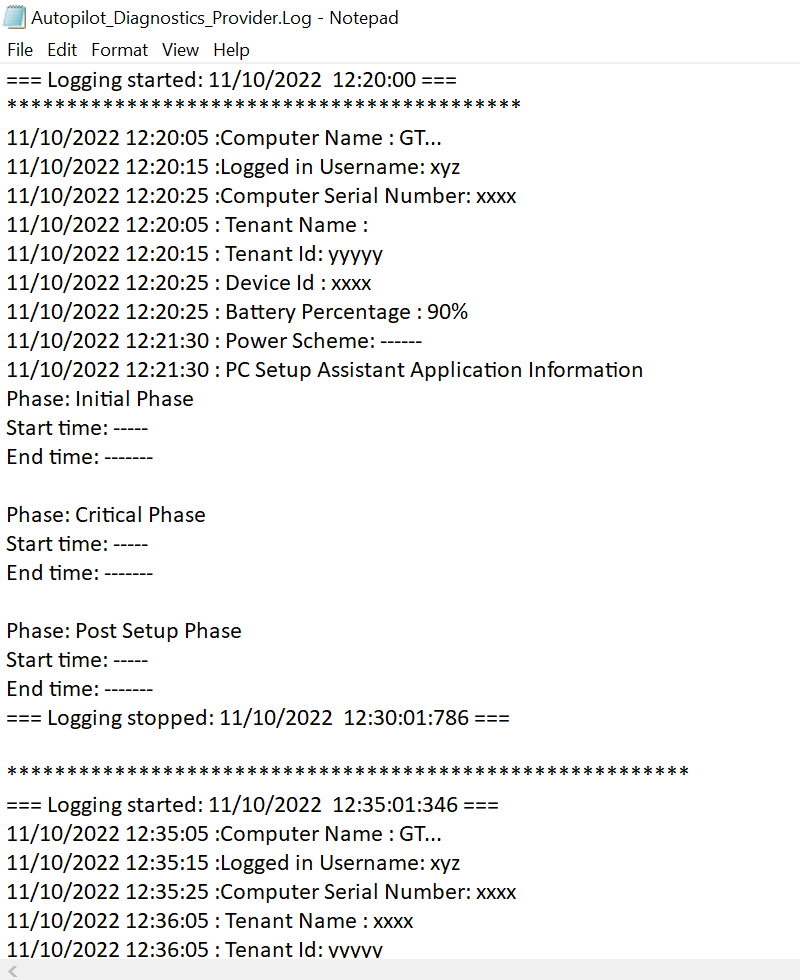
1. Computer Name
2. Username
3. Computer Serial Number

Note:

If we want to log the static properties only one time for every reboot, we need to configure the interval in script instead of scheduler.

* If we configure interval in script, we can capture the static properties only once on every reboot, but if the script fails for any reason, it won’t run the script again until the pc is rebooted.
* If we configure the interval in scheduler, it records the static items every time on script run. But if the script fails for any reason, the scheduler starts the script again, based on the interval time we configured in scheduler.
* So, its better to configure the interval in scheduler to make sure the script runs again, in case of script failure.

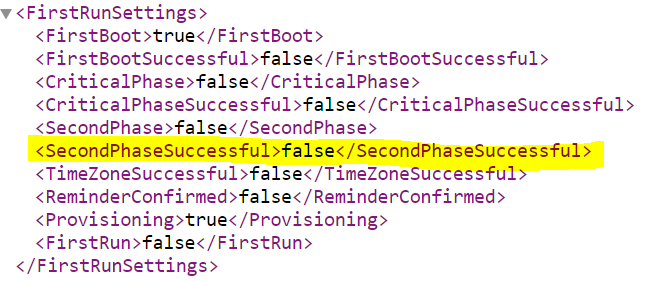
Please check the attached screen shot of sample log file. Also added [Autopilot\_Diagnostics\_Provider.Log](#auopilotitemslog)



* Along with the recording the items, the script should check the provisioning status of the PC Setup Assistant application. Based on this, the script should delete the scheduled task which runs this logging script.
* Check the status of the PC Setup Assistant application every time when the logging script is running. To check the PC Setup Assistant application provisioning status, go to the below path

C:\Users\Public\Documents\DoNotDelete

From this path find the EYGetStartedWPFData.xml file. In this file find the SecondPhaseSuccessful tag. If the SecondPhaseSuccessful status is true, delete the scheduled task, which runs the logging script.



**Alternate Condition to stop the script.**

The security identifier (SID) of the logged-in user. This subkey is created in the account setup phase. It won't be created if the device setup phase fails.

We can check this key is exists or not . If it exists, then we can check the active setup key of PC Setup Assistant application key existence. If key does not exist, we can remove the task.

1. Another is Intune deployment script:

* Create the deployment script name it as “Run\_Autopilot\_Diagnostics Script.ps1”
* This script is responsible to create the scheduled task. The scheduled task should run the logging script from below location

C:\MAINTENANCE\Installers\Autopilot Troubleshooting \Autopilot\_Diagnostics\_Provider.ps1

* The deployment script is also responsible to start the scheduled task (EY\_Autopilot\_Troubleshooting\_Task) initially to log the details during the enrollment.
* Create the Scheduled task with the following details.

1. Name of the scheduled task is “EY\_Autopilot\_Troubleshooting\_Task”.
2. This scheduled task should run the logging script

C:\MAINTENANCE\Installers\Autopilot Troubleshooting \Autopilot\_Diagnostics\_Provider.ps1

which should trigger on system startup and add the repetition interval to run the script periodically.

1. For this we can use AtStartup trigger to run the script on system start and use the repetition interval as “5” minute for a duration of 1 day. If the script is already in running state, then apply the rule as “Do not start a new instance”.
2. If the script does not complete in specified time, or if it runs longer than expected time (1 day), we can stop the task.
3. Create the scheduled task with the Action as Start a Program, this program should run the logging script.
4. Run the task under system user account.
5. The script requires elevation rights to create this scheduled task.

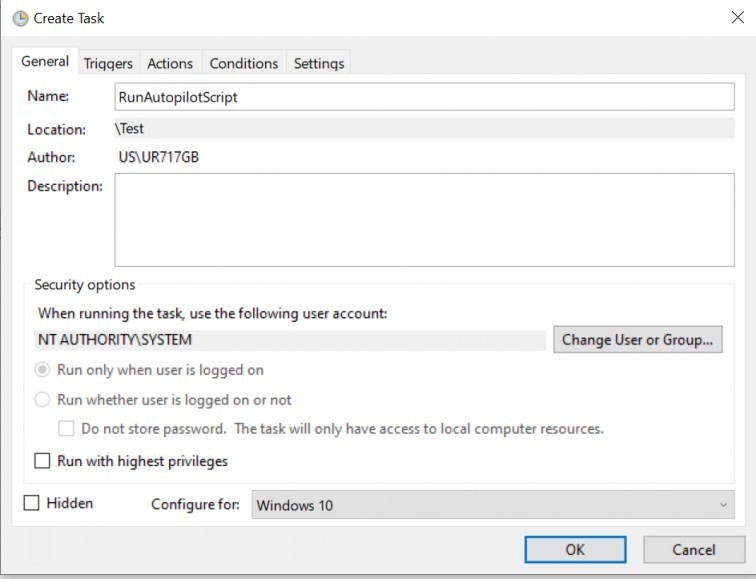
Example:

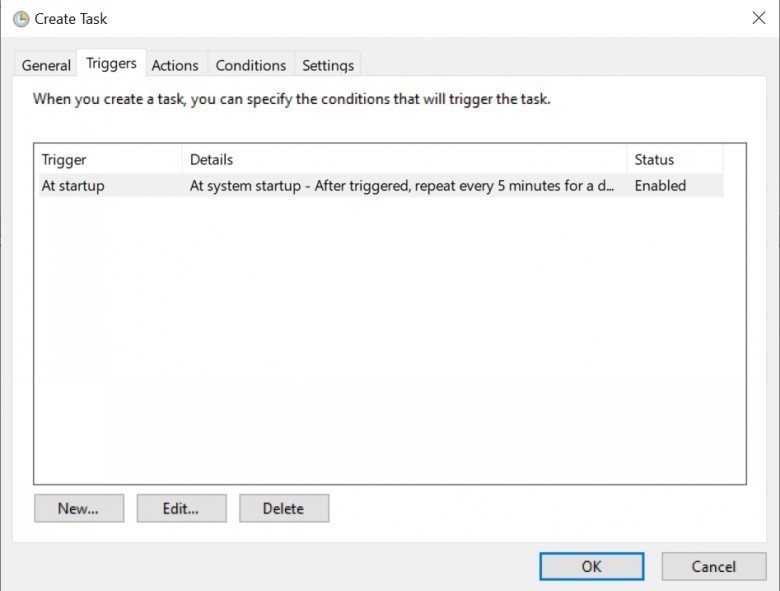
Scheduled task Name: EY\_Autopilot\_Troubleshooting\_Task

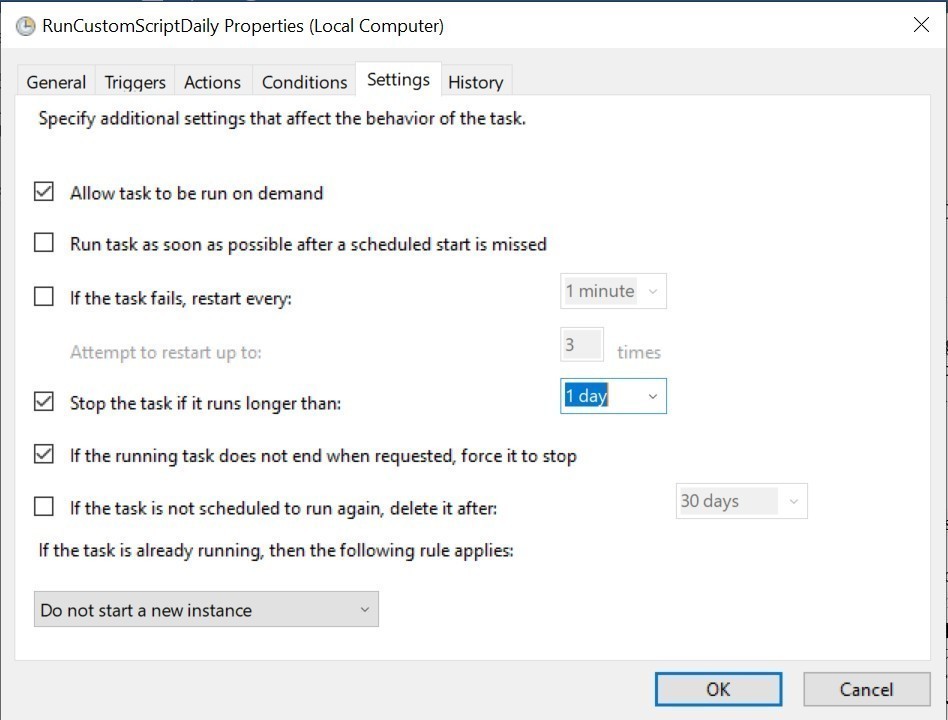
Trigger: At Startup, repeat every 5minutes for a duration of 1 day, stop the task if it runs longer than 1 day.

Action: Start a program

Program: Run the logging script







* Below Coding standard should be followed
  + Revision history
  + Comments
  + Documentation

## Deployment Tool

* The script will be deployed through Intune with below configuration
  + Run this script using the logged-on credentials - No
  + Enforce script signature check - No
  + Run script in 64-bit PowerShell Host - Yes

## Template Available

* Yes, attached PowerShell template in the document.
* Once the script is finalized, tested, and implemented it will be uploaded in Azure Devops Repo.
* How to test?
* Test the scripts. Run the logging script manually and check that the script is copied to the specified location. That is “C:\MAINTENANCE\Installers\Autopilot Troubleshooting\Autopilot\_Diagnostics\_Provider.ps1” .
* Test the deployment script. Run the script manually with elevation and check whether the task scheduler is created with the given details. And check, whether the task scheduler runs the logging script and creates the log files as expected or not
* During the build and after build check whether the log files are created and check above mentioned items are logged or not.

## Requestor

* Jan Gutjahr <jan.gutjahr@ch.ey.com>

## Developer

* Gyorgy Nemesmagasi <Gyorgy.Nemesmagasi@hu.ey.com>

## Target Completion Date

## Risk, if any

* If the script (Run\_Autopilot\_Diagnostics Script.ps1)fails after three retry, no additional attempts will be made to run the script. It may fail to run the script for different reasons. Those are
* Be sure devices are [joined to Azure AD](https://learn.microsoft.com/en-us/azure/active-directory/user-help/user-help-join-device-on-network). Devices that are only joined to your workplace or organization ([registered](https://learn.microsoft.com/en-us/azure/active-directory/user-help/user-help-register-device-on-network) in Azure AD) won't receive the scripts.
* Confirm the Intune management extension is downloaded to %ProgramFiles(x86)%\Microsoft Intune Management Extension.

**Troubleshooting Steps:**

* Review the logs for any errors.

C:\ProgramData\Microsoft\IntuneManagementExtension\Logs

## Testing in Test Environment

* The script will be tested in Tenant 6 during a full Autopilot build,

How to Test?

* Deploy the scripts in Intune in tenant6 environment.
* Add the device hash to Tenant6 environment.
* Take the Autopilot build.
* During the build and after build, verify whether the log files are created with the listed items or not.
* Test whether the script removed the scheduled task, once the PCSA is completed.

## Testing in QA

* Create a change request to deploy the scripts in QA environment.
* Test the scripts in QA during a full Autopilot build,

How to Test?

* After the script deployed in the QA tenant, add the device hash to the QA environment.
* Take the Autopilot build.
* During the build and after build, verify whether the log files are created with the listed items or not.
* Test whether the script removed the scheduled task, once the PCSA is completed.

## Production SR raised

## Attachments

* PowerShell Template



****





* Registry export file







* PowerShell script workflow log template



* Autopilot Json template



* Ipconfig command output



* Dsregcmd command output



* Sample script file

