# V3400 Series Win 11 LTSC 24H2 User Manual

**Version 1.0, July 2025**

**www.moxa.com/products**

© 2025 Moxa Inc. All rights reserved.

# V3400 Series Win 11 LTSC 24H2 User Manual

The software described in this manual is furnished under a license agreement and may be used only in accordance with the terms of that agreement.

## Copyright Notice

© 2025 Moxa Inc. All rights reserved.

## Trademarks

The MOXA logo is a registered trademark of Moxa Inc.
All other trademarks or registered marks in this manual belong to their respective manufacturers.

## Disclaimer

- Information in this document is subject to change without notice and does not represent a commitment on the part of Moxa.
- Moxa provides this document as is, without warranty of any kind, either expressed or implied, including, but not limited to, its particular purpose. Moxa reserves the right to make improvements and/or changes to this manual, or to the products and/or the programs described in this manual, at any time.
- Information provided in this manual is intended to be accurate and reliable. However, Moxa assumes no responsibility for its use, or for any infringements on the rights of third parties that may result from its use.
- This product might include unintentional technical or typographical errors. Changes are periodically made to the information herein to correct such errors, and these changes are incorporated into new editions of the publication.

## Technical Support Contact Information

**www.moxa.com/support**

# Table of Contents


**1. Introduction ..... 6**
Moxa Computers and Windows ..... 6
**2. System Initialization ..... 7**
Initializing User Settings ..... 7
Initializing System ..... 10
**3. BitLocker ..... 11**
Enabling the BitLocker ..... 11
Disabling the BitLocker ..... 14
**4. RAID ..... 16**
Intel® RAID: Changing the RAID Mode ..... 16
Intel® RAID: Creating a RAID Disk in BIOS ..... 19
Intel® RAID: Removing a RAID Volume From the BIOS ..... 23
SW RAID: Creating the RAID 0 or RAID 1 From Disk Management ..... 25
**5. Network ..... 31**
Ethernet Setting ..... 31
WiFi Setting ..... 33
**6. Intel® Active Management Technology ..... 35**
Turning on Intel® AMT on PC ..... 35
Access the Intel® AMT From Website ..... 42
Access the Intel® AMT by Intel® Manageability Commander ..... 44
Remote Desktop ..... 46
Advanced Power Control ..... 49
Remote RAID Configuration ..... 50
Remote Boot Manager ..... 50
Remote Mount/Recovery Image ..... 51
Reset to PXE ..... 53
**7. Unified Write Filter ..... 55**
Turning on UWF on a Running PC ..... 55
Installing UWF Using WMI ..... 57
**8. Driver ..... 58**
**9. Utility ..... 59**
Where to Find the Utility ..... 59
Dependent Packages ..... 61
Moxa IO Controller ..... 61
Setting the DIO Status ..... 62
Setting the UART Mode ..... 63
Setting the LED Status ..... 64
Setting the PCIE Slot Power Status ..... 65
Setting the PCIE Reset Pin Status ..... 66
Setting the M.2 B Key Socket 5G Module Power Status ..... 66
Moxa Serial Interface Utility ..... 67
Setting the Serial Port Mode ..... 67
Moxa Sort Net Name ..... 68
Moxa Disk Hot Swap Application ..... 69
Setting Up the Notification ..... 70
Setting Up the Disk Removal Policy ..... 71
Setting Up the Drive Actions ..... 73
Default Actions ..... 74
Unmounting and Mounting Disks ..... 74
Hot Swap Function Notifications ..... 76
Moxa LANBypass Utility ..... 76
Relay Mode ..... 77
WDT Reset Mode ..... 78
WDT Relay Mode ..... 78
App WDT Mode ..... 79
App WDT Reset Mode ..... 79
App WDT Relay Mode ..... 80
Power Off Relay Mode ..... 80

Checking the Firmware Version of the MCU ..... 81
Moxa MCU Firmware Upgrade ..... 81
Upgrade the MCU Firmware ..... 82
Check MCU Firmware Version ..... 82
Moxa Tca9535 Initial Service ..... 83
Moxa Cellular Assistant ..... 84
Install Moxa Cellular Assistant ..... 84
Command Line Tool ..... 85
GUI Utility ..... 88
Moxa WiFi AP Service ..... 91
Install WiFi AP Service ..... 91
Configuring the WiFi AP Service ..... 92
Moxa Battery Detect ..... 96
**10. IO Control API ..... 99**
Downloading the API ..... 99
mxdgio ..... 100
GetDinCount ..... 100
GetDoutCount ..... 101
GetDinStatus ..... 101
GetDoutStatus ..... 102
SetDoutStatus ..... 102
mxsp ..... 103
GetUartCount ..... 103
GetUartMode ..... 104
SetUartMode ..... 104
mxled ..... 105
GetLedData ..... 105
SetLedData ..... 106
mxwdg ..... 106
mxwdg_open ..... 106
mxwdg_refresh ..... 107
mxwdg_close ..... 107
mxpcie ..... 108
GetPCIESlotStatus ..... 108
SetPCIESlotStatus ..... 109
SetPCIESlotStatusWithReset ..... 110
mxpciereset ..... 111
GetRESETSlotStatus ..... 111
SetRESETSlotStatus ..... 112
mxm2bpower ..... 113
GetPowerStatus ..... 113
SetPowerStatus ..... 114
MxLBPLibrary ..... 115
Relay Mode ..... 116
GetRelayMode ..... 116
SetRelayMode ..... 116
WDTResetMode ..... 117
GetWDTResetMode ..... 117
SetWDTResetMode ..... 117
WDTRelayMode ..... 118
GetWDTRelayMode ..... 118
SetWDTRelayMode ..... 118
AppWDTMode ..... 119
GetAppWDTMode ..... 119
SetAppWDTMode ..... 119
AppWDTKick ..... 120
SendAppWDTKick ..... 120
AppWDTResetMode ..... 120
GetAppWDTResetMode ..... 120
SetAppWDTResetMode ..... 121

AppWDTRelayMode ..... 121
GetAppWDTRelayMode ..... 121
SetAppWDTRelayMode ..... 122
PowerOffRelayMode ..... 122
GetPowerOffRelayMode ..... 122
SetPowerOffRelayMode ..... 123
MCUFirmwareVersion ..... 123
GetMCUFirmwareVersion ..... 123
**11. System Backup Restore ..... 124**
WindowsRecovery ..... 124
Preparing the USB Device ..... 124
Booting From the USB Disk ..... 126
System Image Backup ..... 127
Restoring the System From a Backup ..... 130

# 1. Introduction

This Windows 11IoT Enterprise LTSC 2024(24H2) user manual is applicable to Moxa's x86-based computers listed below and covers the complete set of instructions for these series. Detailed instructions on configuring advanced settings are covered in the following chapters of the manual. Before referring to sections in these chapters, confirm that the hardware specification of your computer model supports the functions/settings covered therein.

## Moxa Computers and Windows

Moxa computers are integrated with Windows drivers and I/O controller utilities based on the Microsoft Windows up-to-date version so that you can use the most compatible hardware-software combinations in your application field.

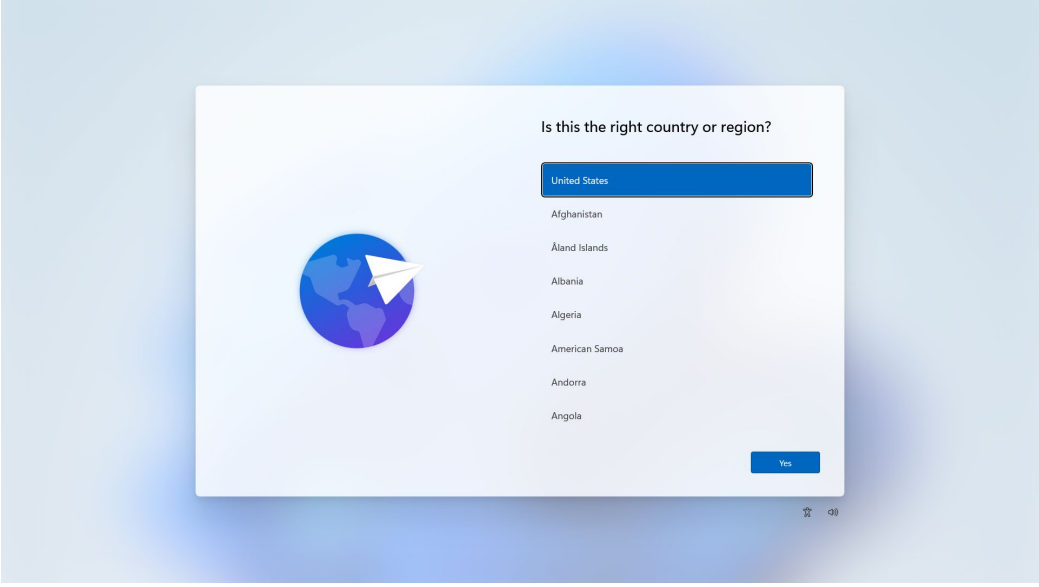
# 2. System Initialization

In this chapter, we describe how to initialize the system settings when you boot up the computer for the first time. When you turn on the computer, you will see the Windows Out of Box Experience (OOBE) wizard. OOBE consists of a series of screens that require customers to accept the license agreement, connect to the internet, log in with or sign up for a Microsoft Account, and share information with the OEM.

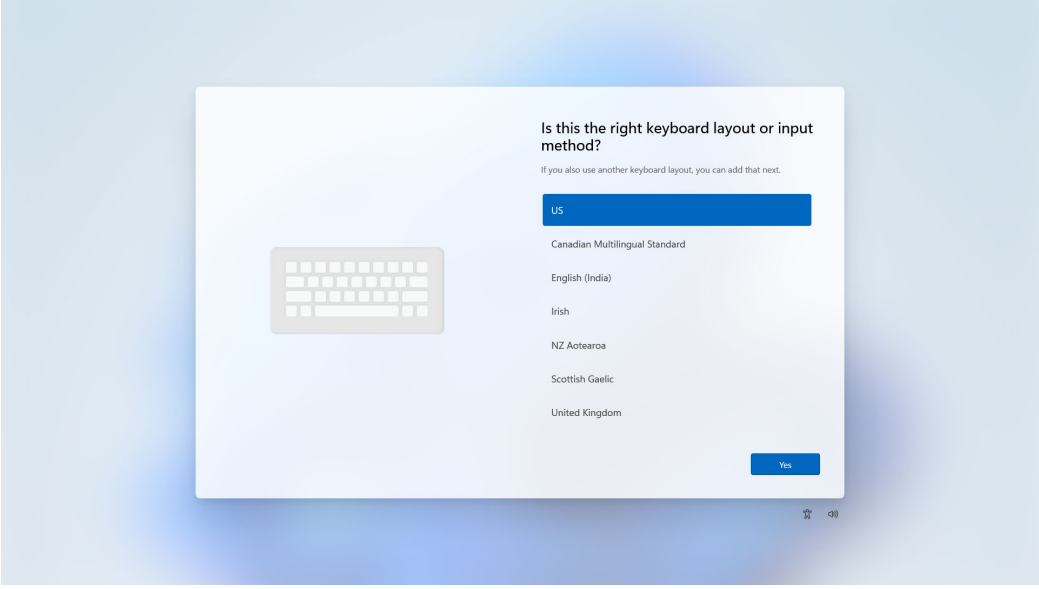
## Initializing User Settings

The following is a non-exhaustive list of OOBE screens that you will see in the order that they are listed here:

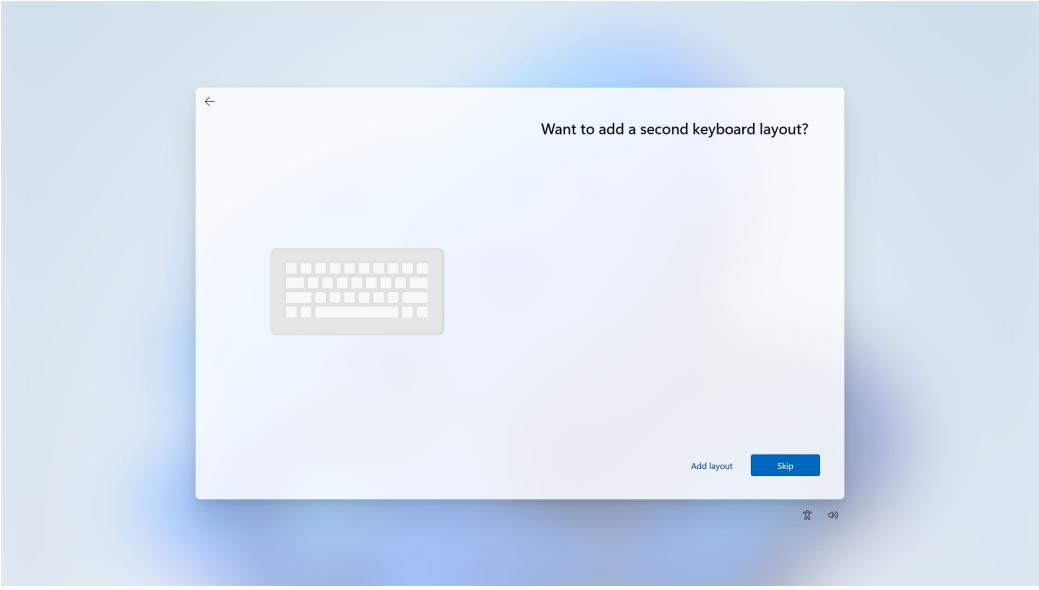
1. Select a region.



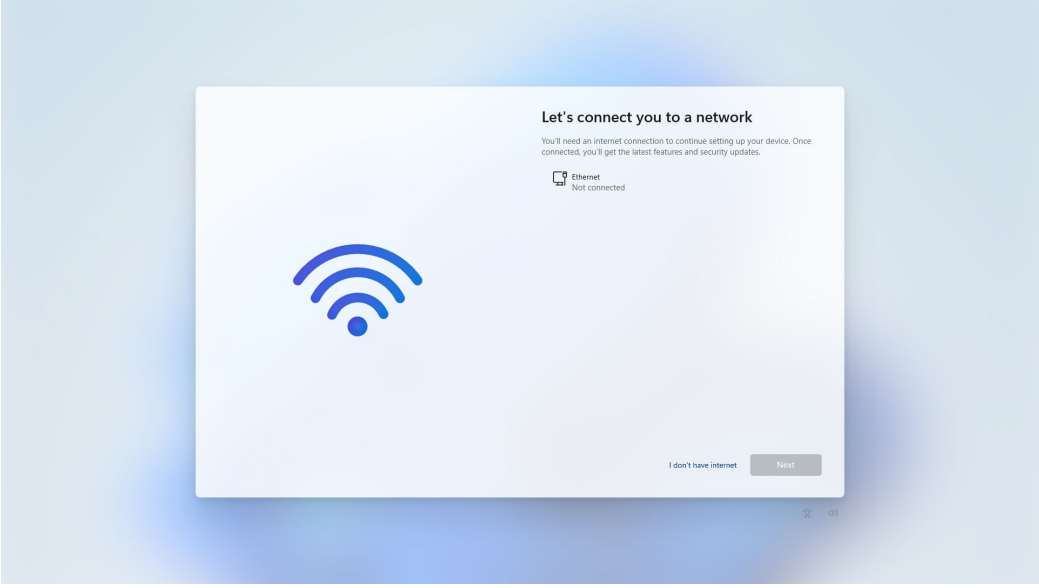

2. Select a keyboard.

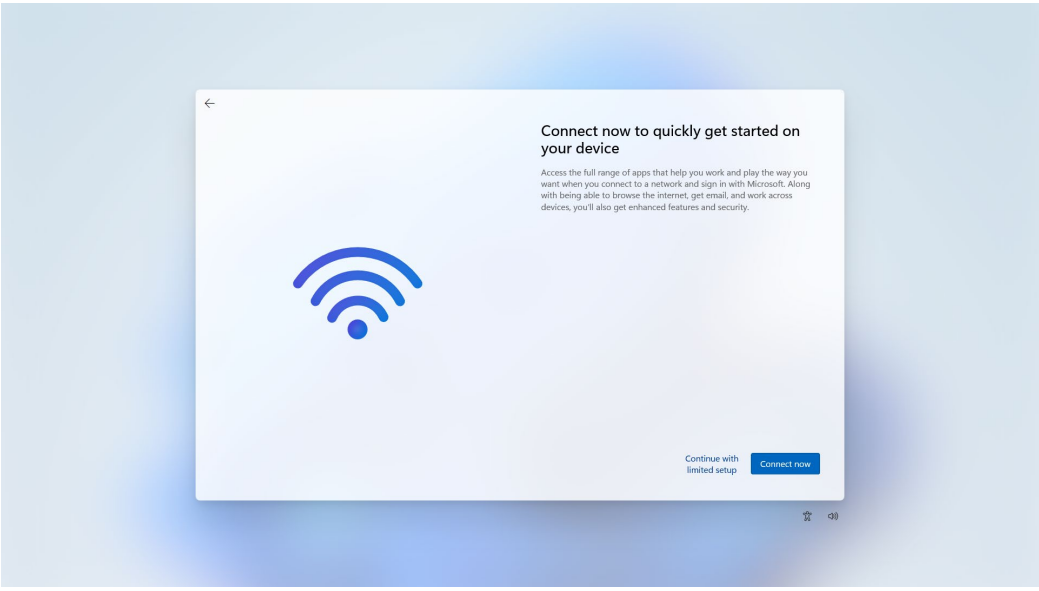


3. Select a second keyboard.

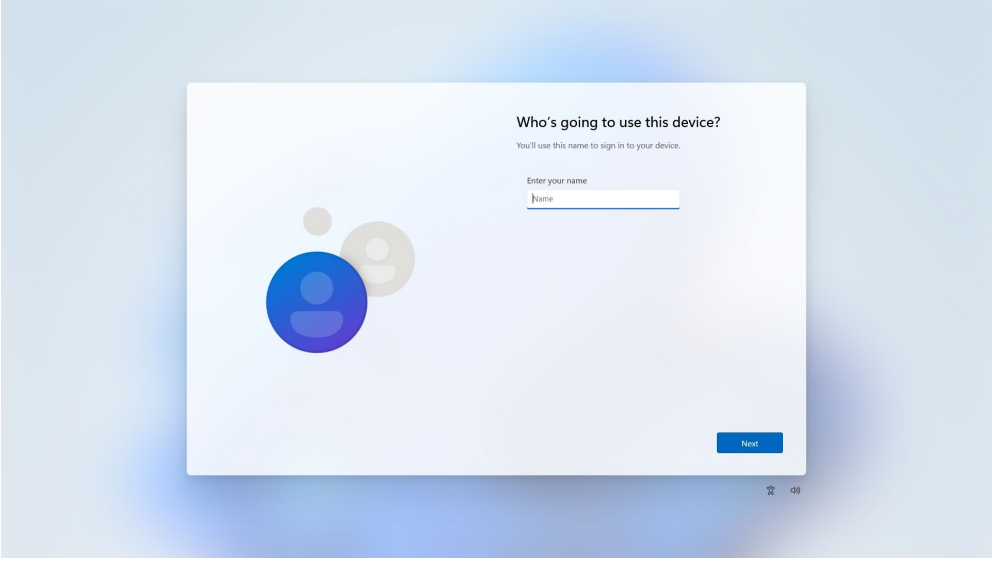



4. Connect to a network or continue with limited setup.



5. Sign in to or create a local account or a Microsoft account (MSA).

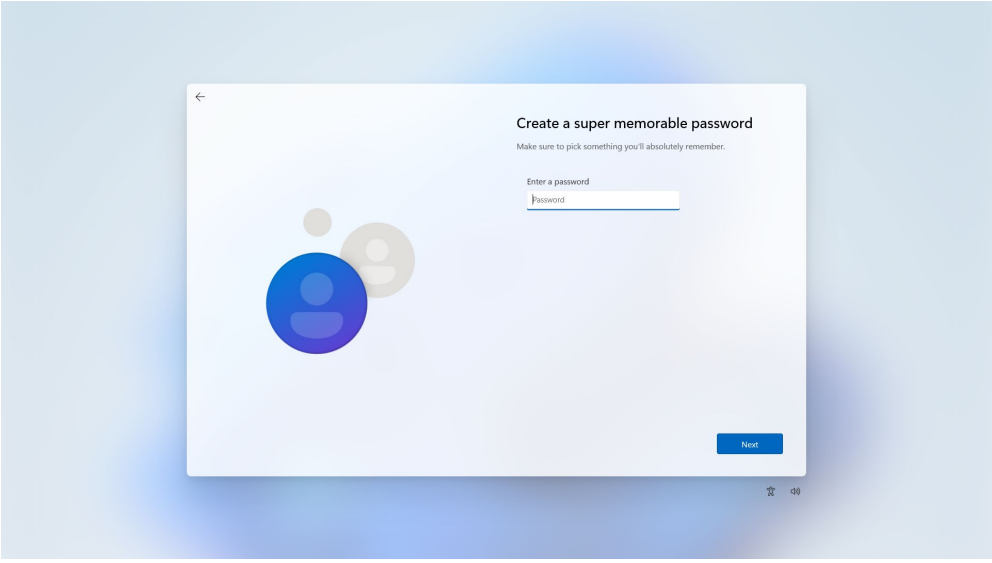



6. Set a password.

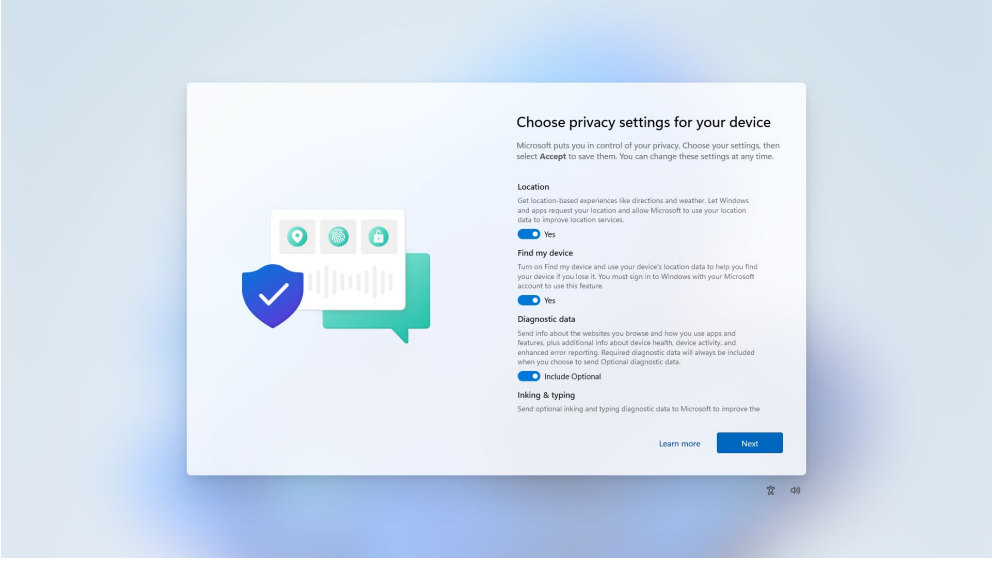
**NOTE**

**Local account password policy:**

The password for any newly created local account must be **at least 8 characters** in length.

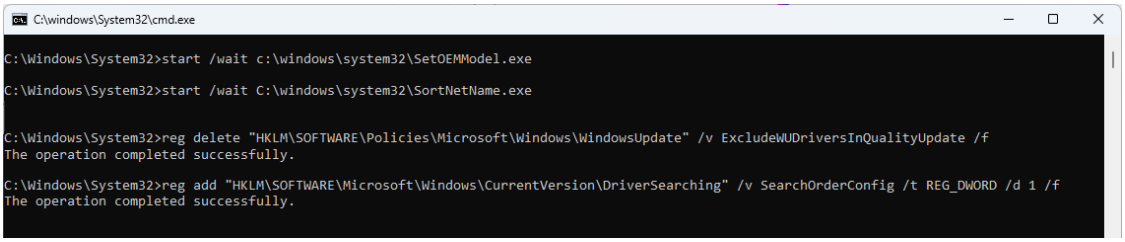


7. Choose your privacy settings.




# Initializing System

- After the OOBE settings, you will be redirected to the device desktop. Wait until the process is complete.




- The device will reboot, and the new settings will take effect after the system restarts.

# 3. BitLocker

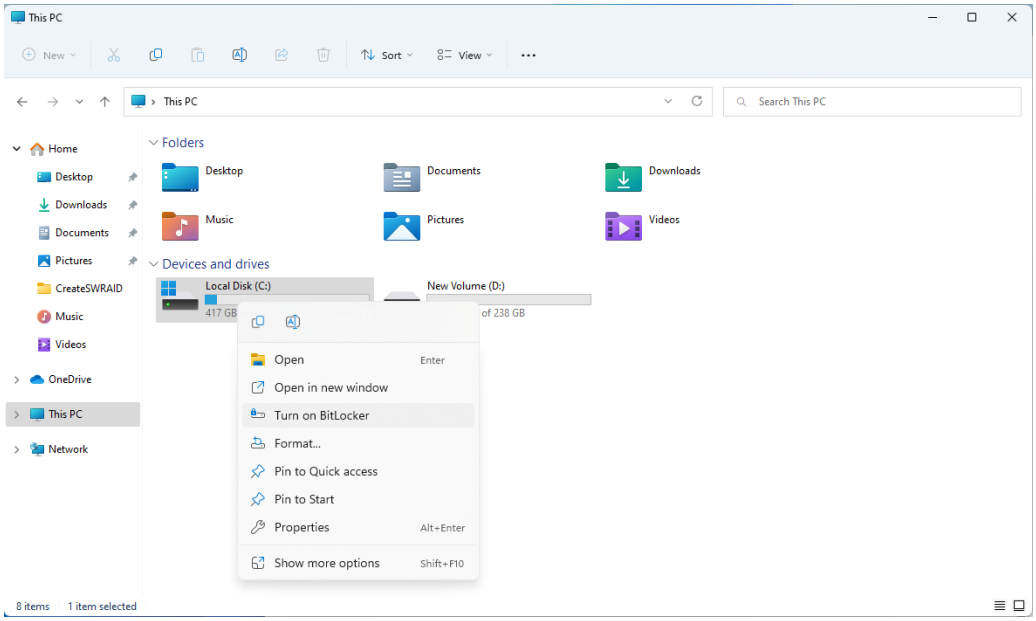
BitLocker is a Windows disk encryption feature, designed to protect data by providing encryption for entire volumes. BitLocker addresses the threats of data theft or exposure from lost, stolen, or inappropriately decommissioned devices.

For more information about BitLocker:
https://learn.microsoft.com/en-us/windows/security/operating-system-security/data-protection/bitlocker/

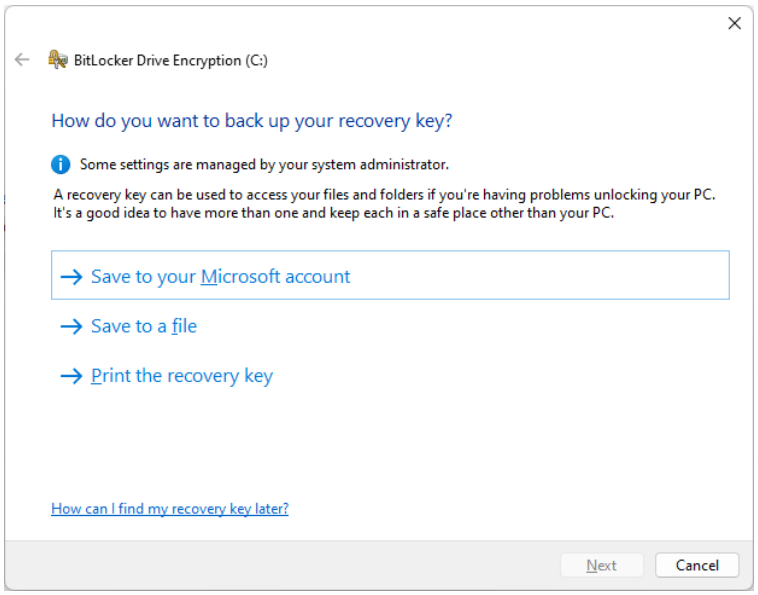
This chapter describes the BitLocker setup process.

# Enabling the BitLocker

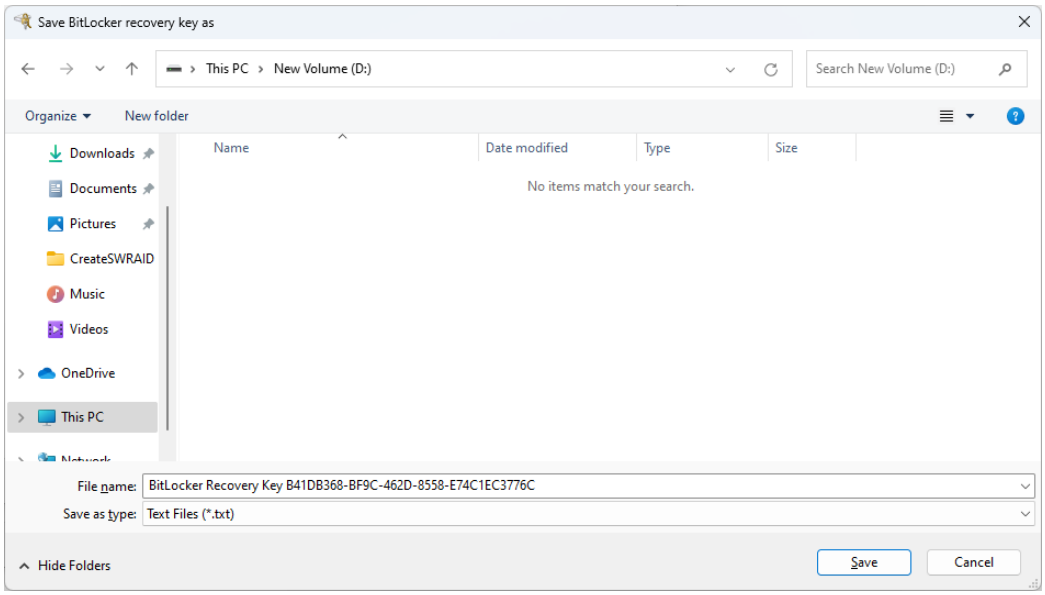
In the **Windows Devices and drives**, right-click on the drive and select **Turn on BitLocker**.



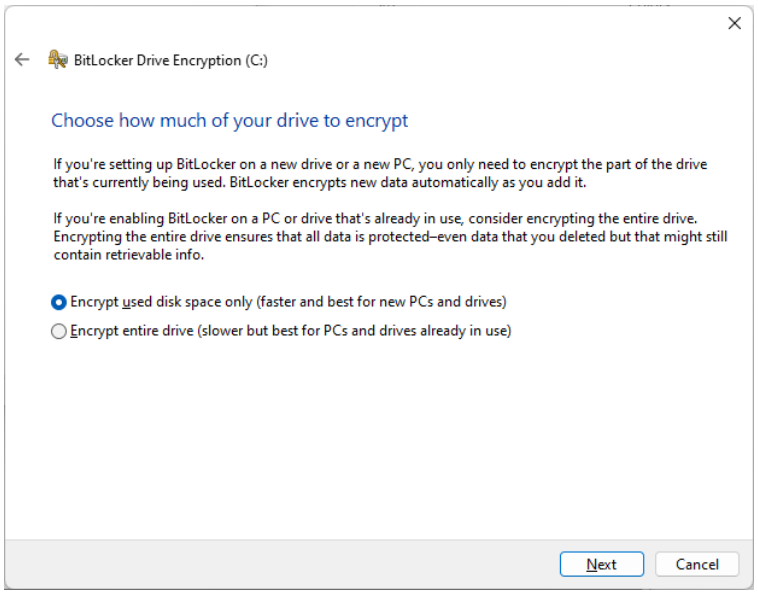

1. Select an option to back up the recovery key. For example, select **Save to a file**.

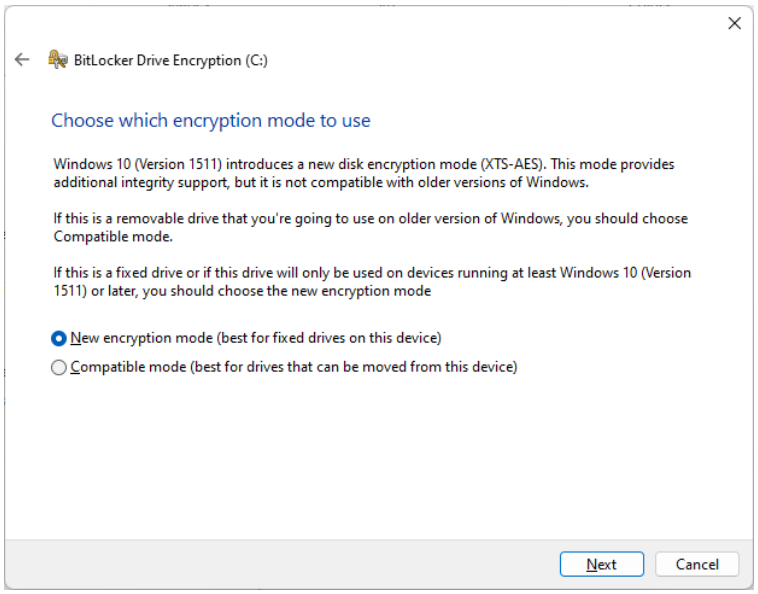


2. Select the path to store the file in.

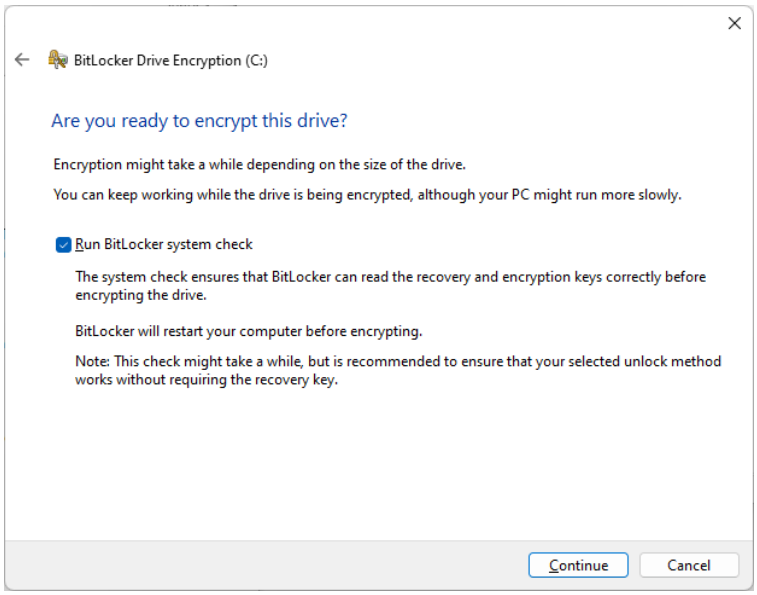



3. Follow the onscreen instructions to specify the drive encryption options.

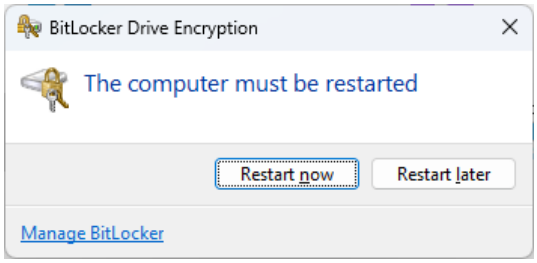


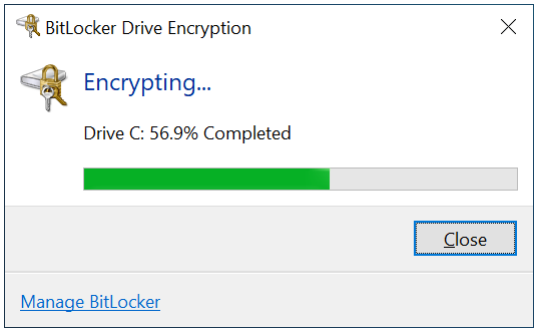
4. Click **Continue**.




5. Restart the computer.

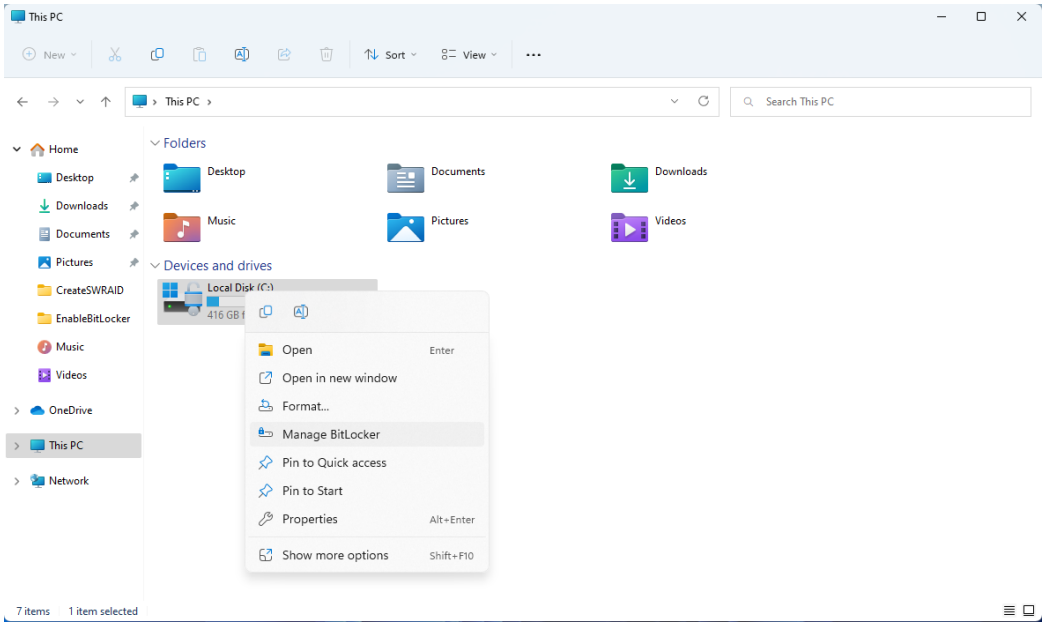



6. Wait for the encryption process to complete and then click **Close**.

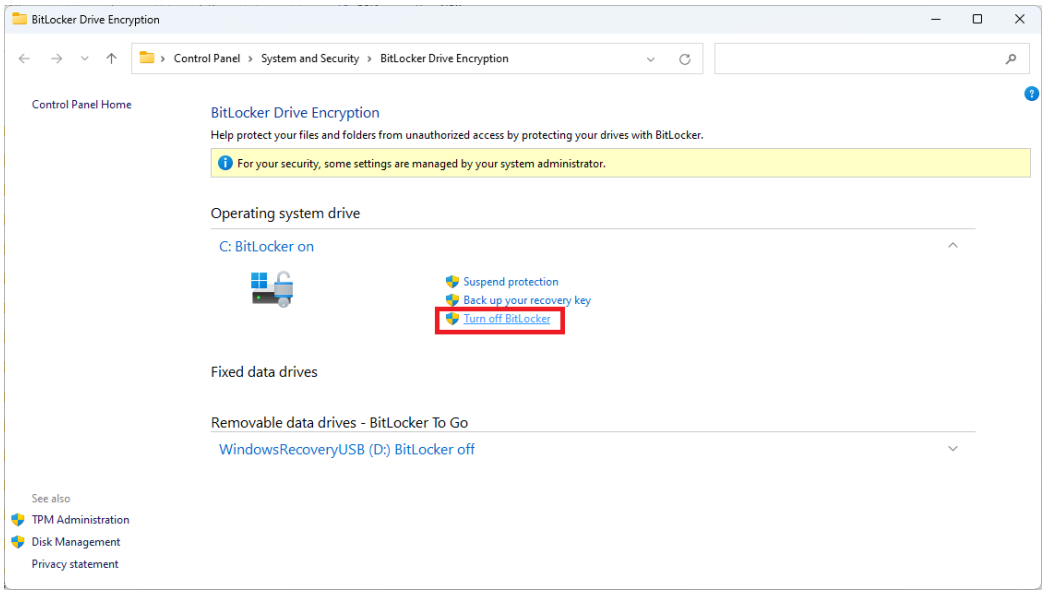


# Disabling the BitLocker

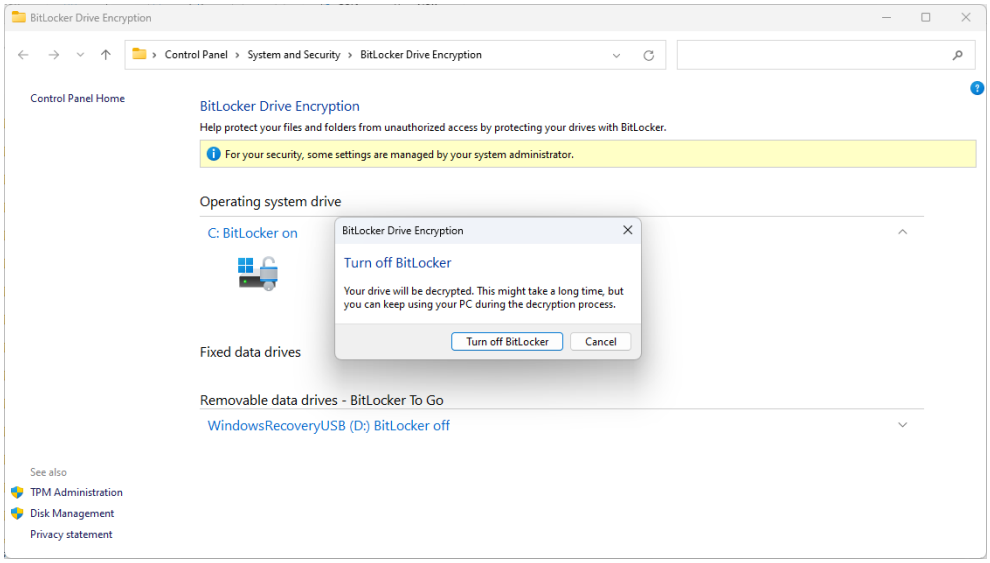
1. In the **Windows Devices and drives**, right-click on the drive and select **Manage BitLocker**.

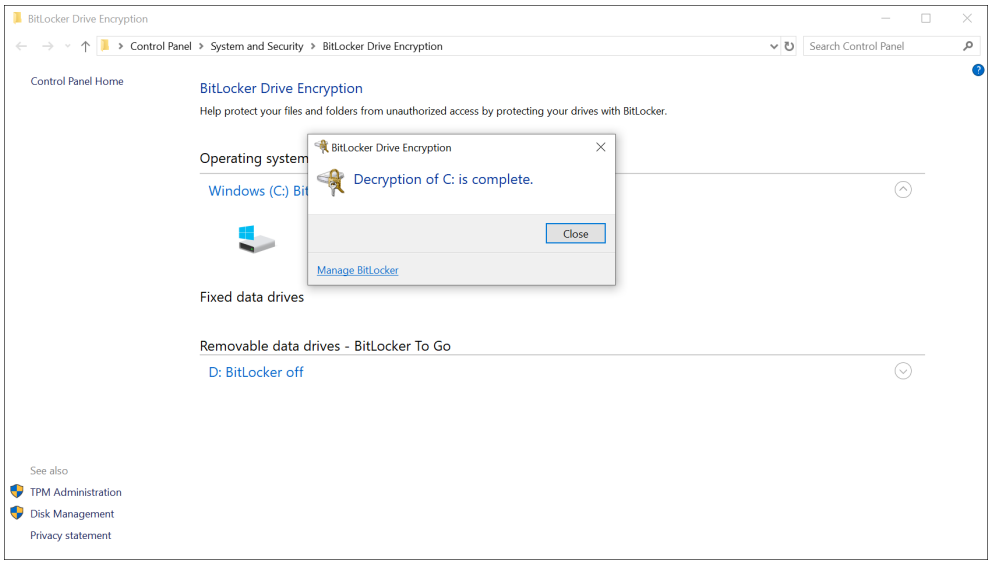



2. Click on **Turn off BitLocker**.

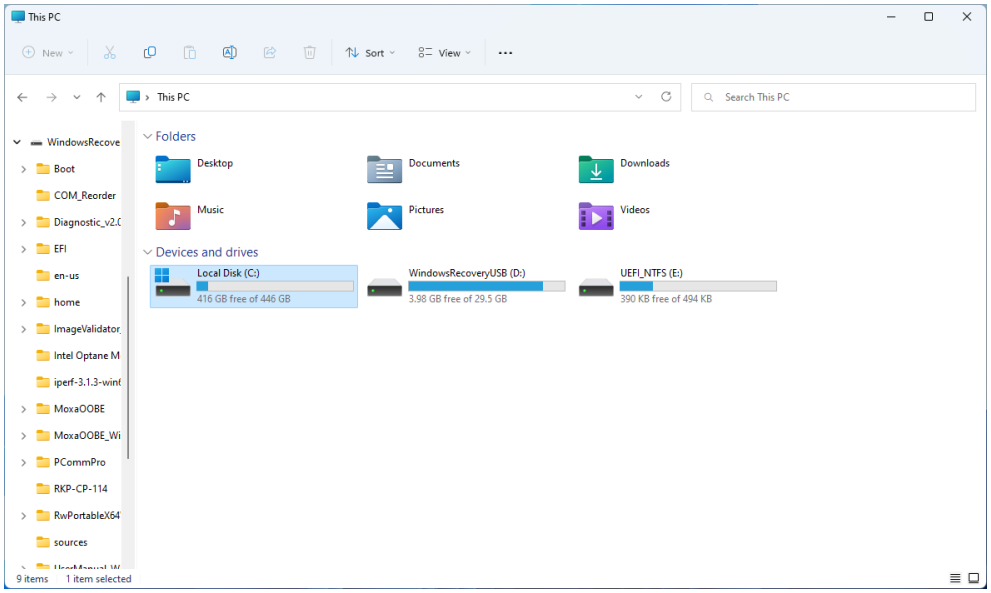


3. Wait for the decryption process to complete and click **Close** to exit the program.




4. Check the disk status after the decryption process is completed.



# 4. RAID

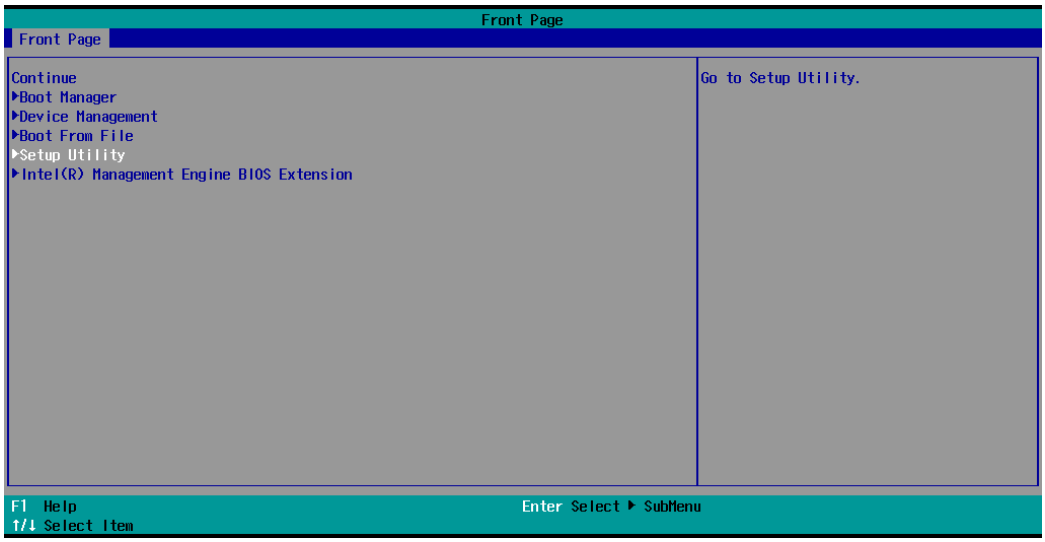
RAID is the acronym for **Redundant Array of Independent Disk** which indicates the use of combining multiple disks into one or more logical units for data redundancy, performance improvement, or both. This chapter describes the setup process for Intel® RAID (Intel® RST) and SW RAID.

**NOTE**

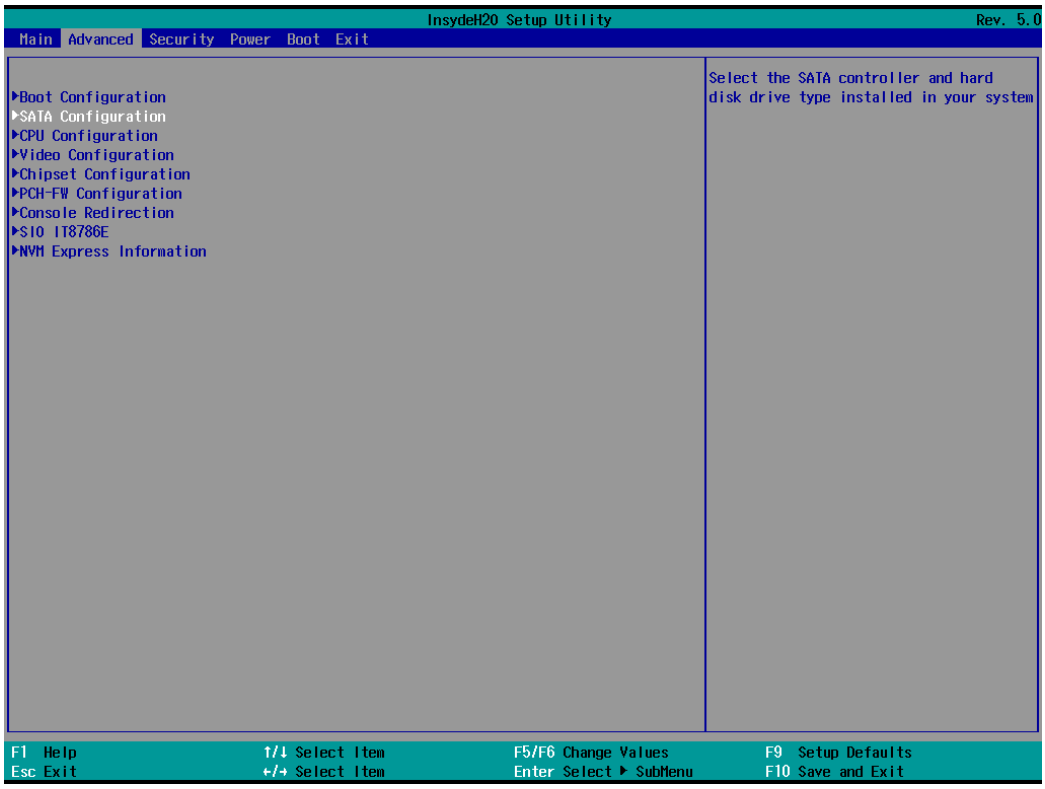
Use hard disks of the same brand, same model, and same capacity to create a RAID for best performance.

# Intel® RAID: Changing the RAID Mode

1. Power on the computer and press F2 to enter the BIOS menu.
2. Select the Setup Utility option.

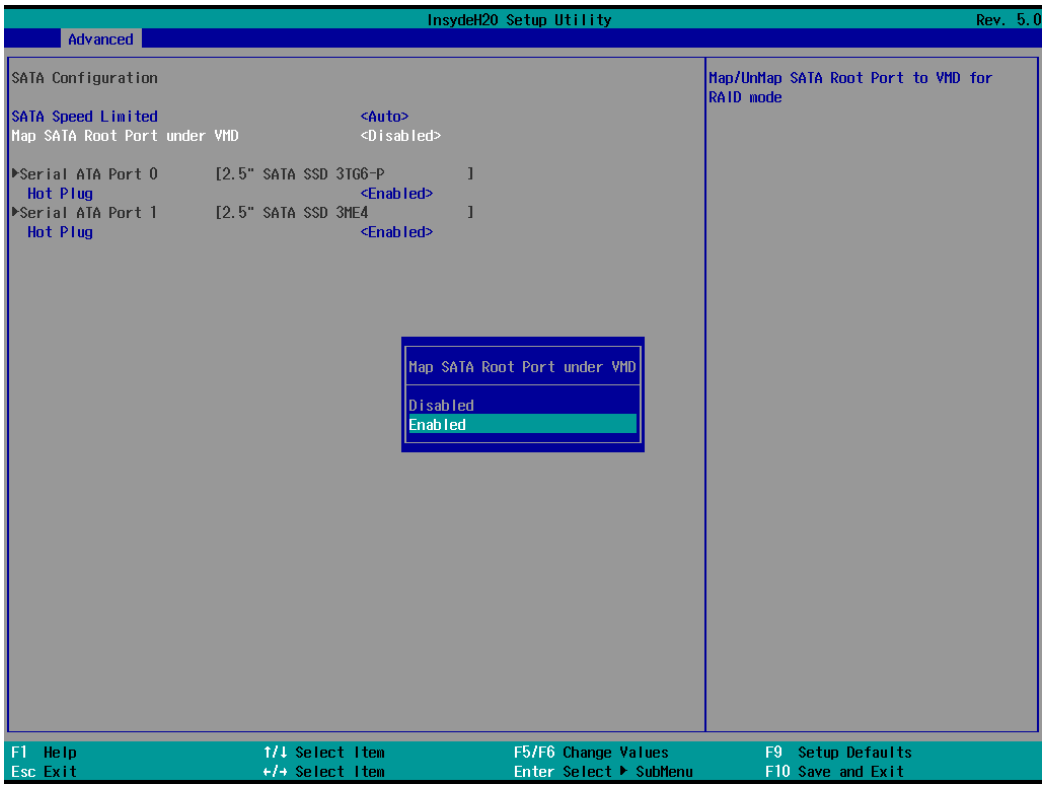


3. Select the SATA Configuration option.

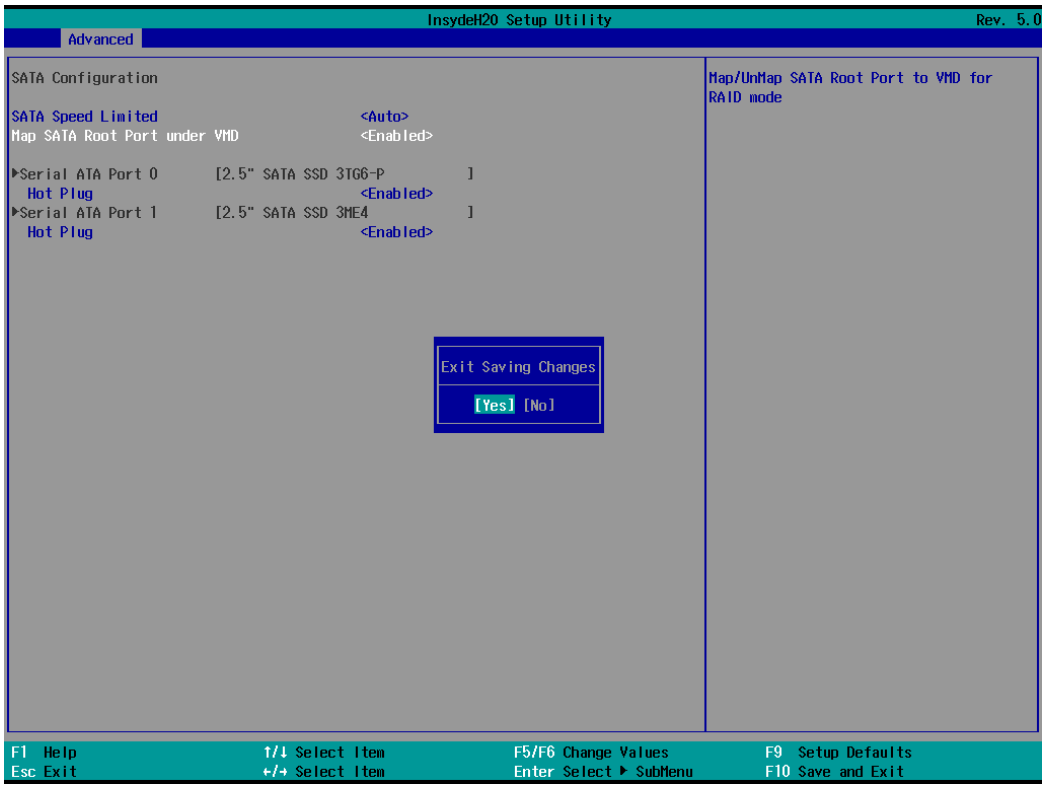



4. Select the Intel(R) Rapid Storage Technology option.

5. Select the Map SATA Root Port under VMD option and Enable.

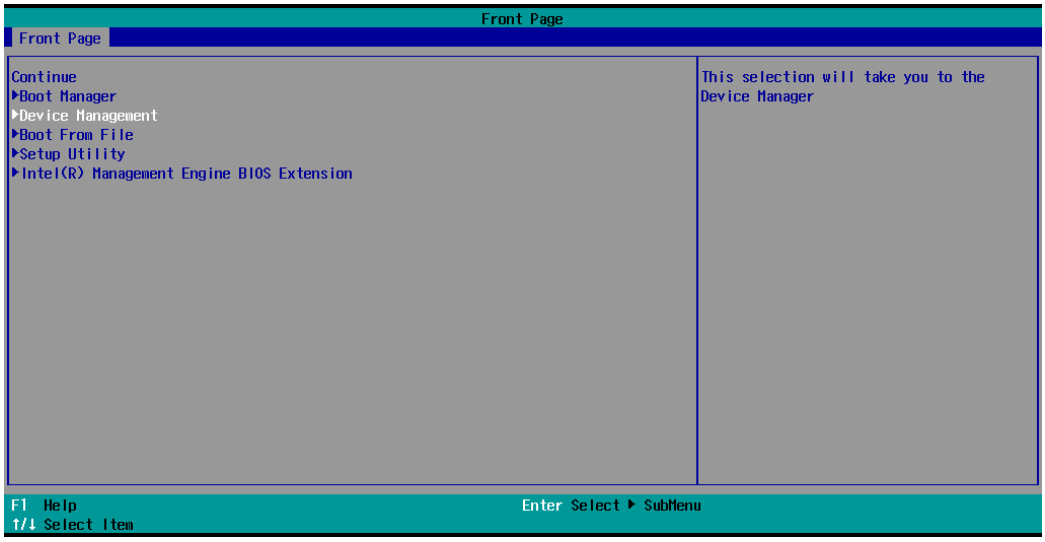



6. Press F10 to save the settings and Exit, and then select Yes to save the settings.



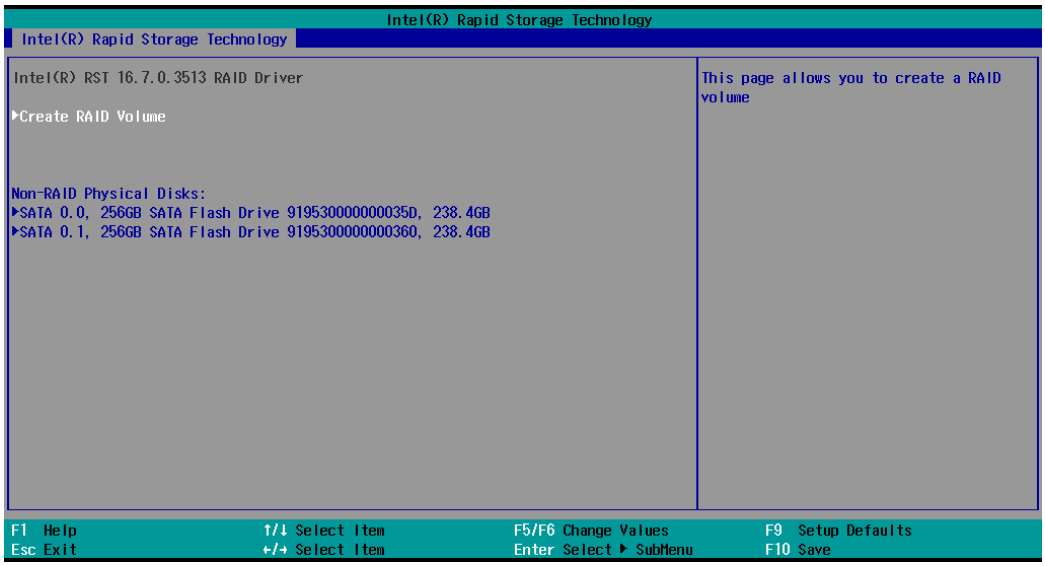
# Intel® RAID: Creating a RAID Disk in BIOS

1. Power on the computer and press F2 to enter the BIOS menu.
2. Select the Device Management option.

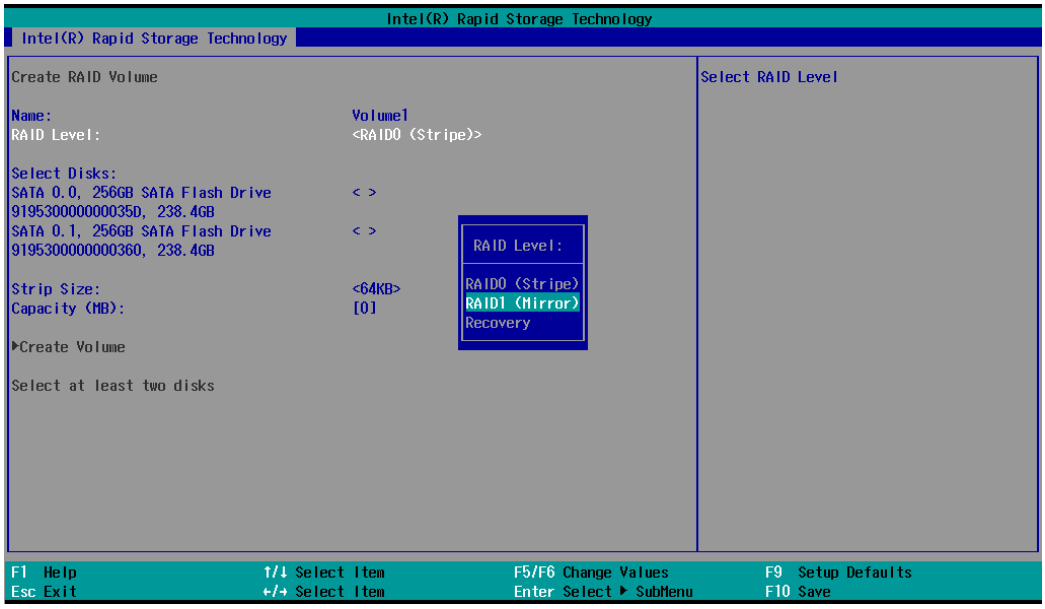


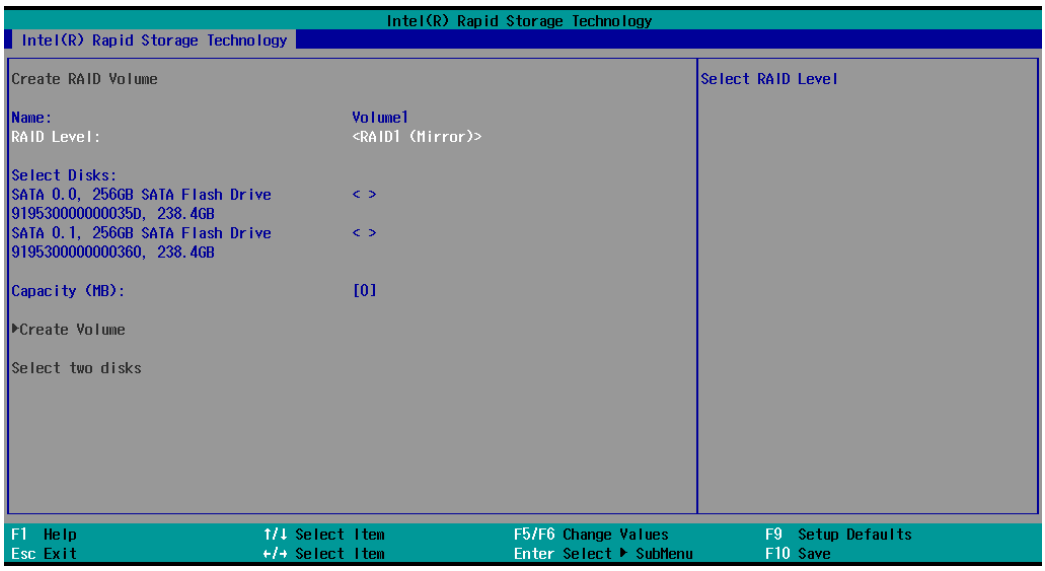

3. Select Intel® Rapid Storage Technology.

4. Select Create RAID Volume.

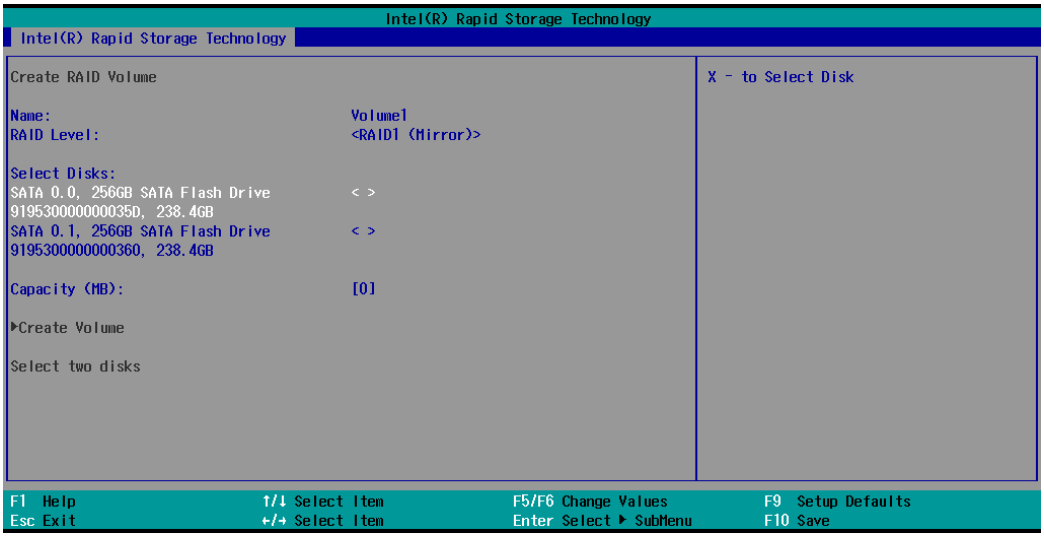



5. Select the RAID Level option and then press Enter to select the raid level; for example, RAID1(Mirror).

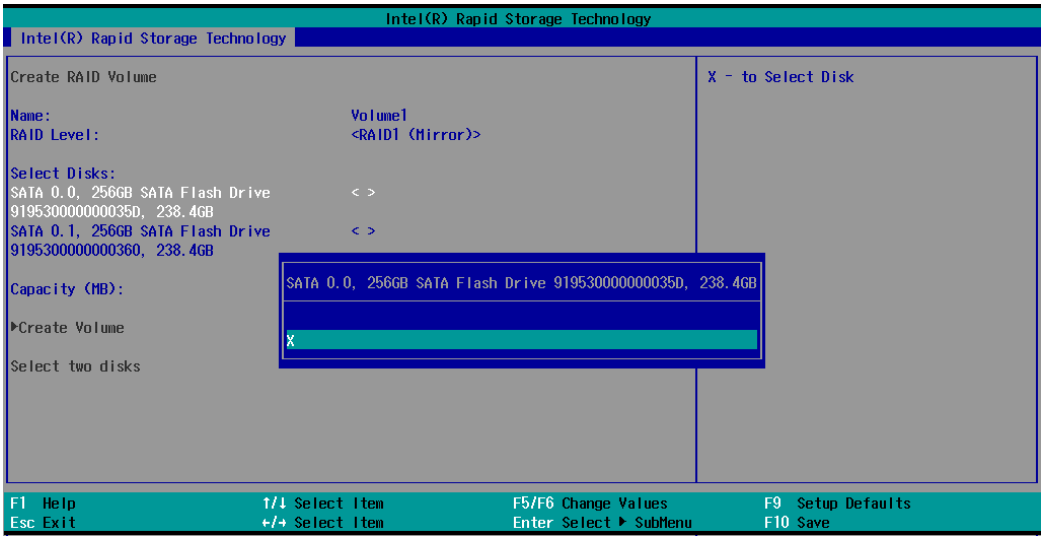


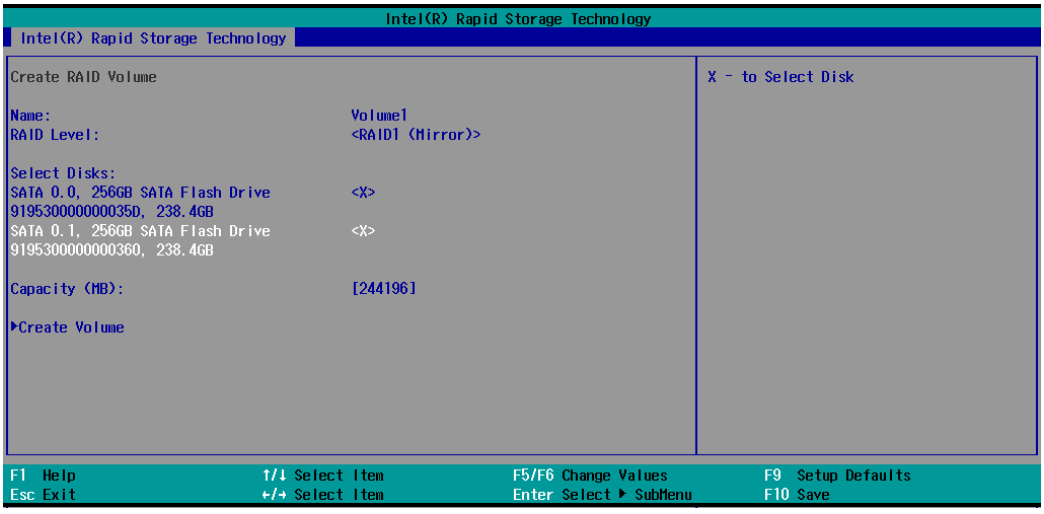
6. Select the target disk.



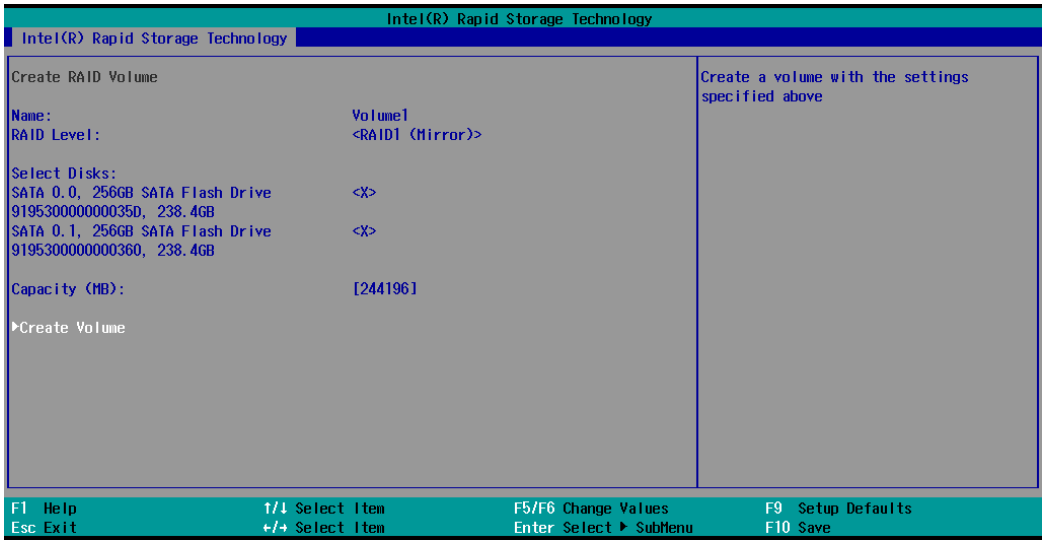

7. Enter X and then press Enter.



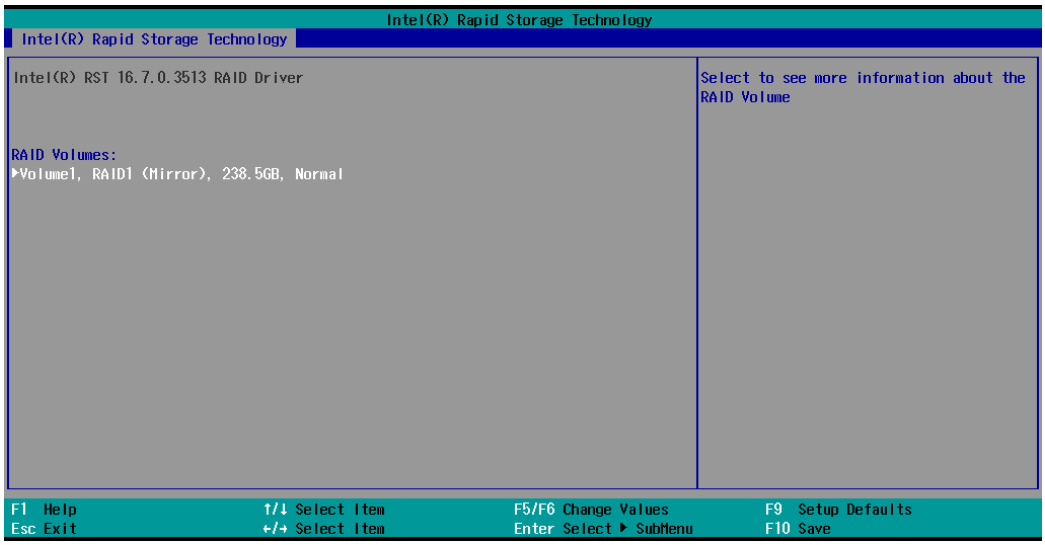

8. The disk is now marked with an X next to it to indicate the selection.



9. Select the Create Volume option.



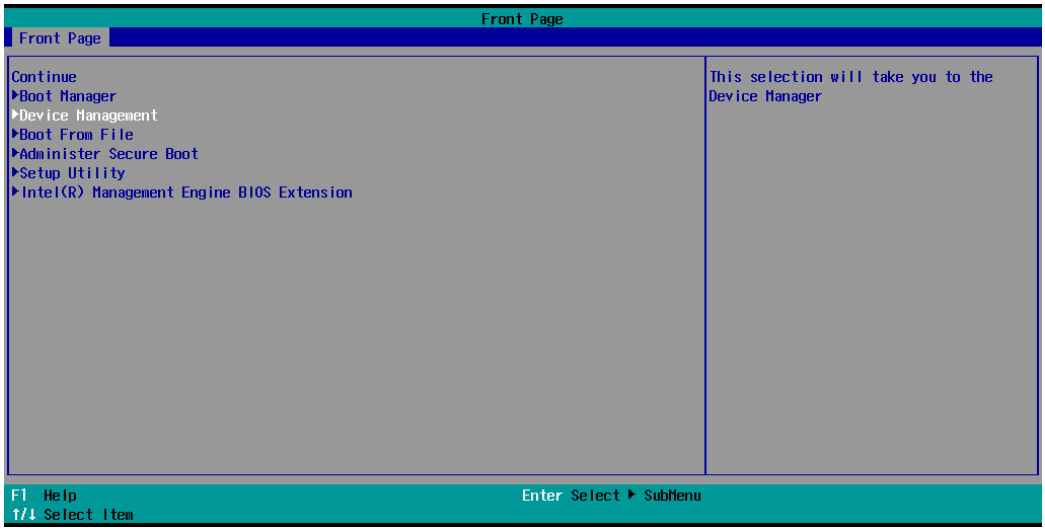

10. A RAID volume is created based on the settings specified.




11. Press F10 to save the settings.

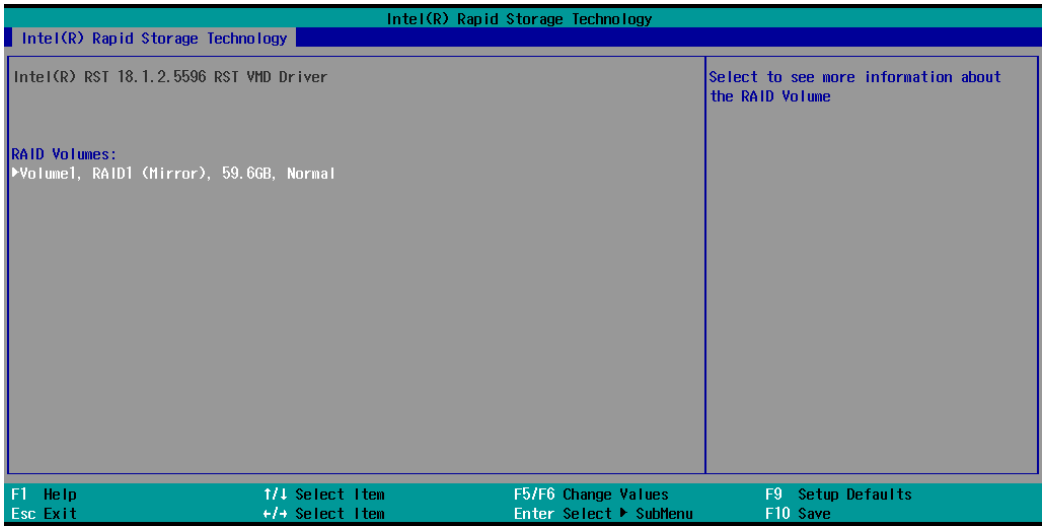
# Intel® RAID: Removing a RAID Volume From the BIOS

1. Power on the computer and press F2 to enter the BIOS menu.
2. Select Device Management.

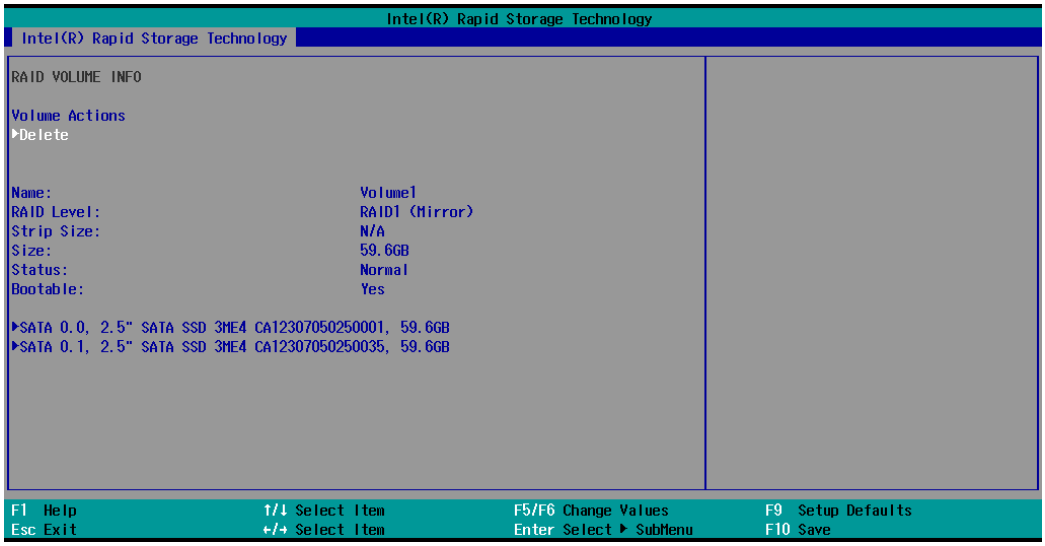



3. Select the Intel® Rapid Storage Technology option.

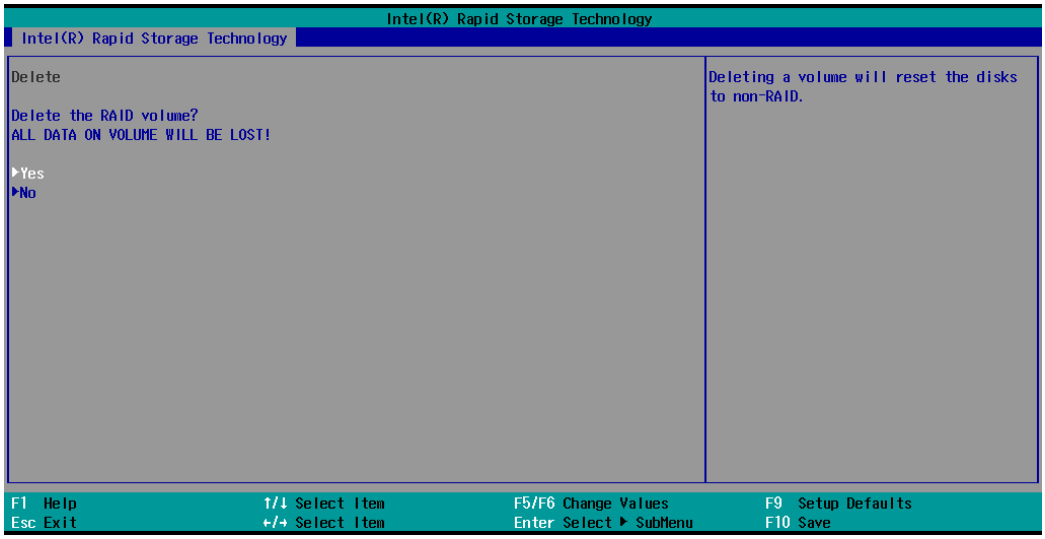
4. Select the RAID volume that you want to remove.




5. Select Delete and then press Enter.




6. Select Yes to confirm and then press Enter.




7. Press F10 to save the settings.

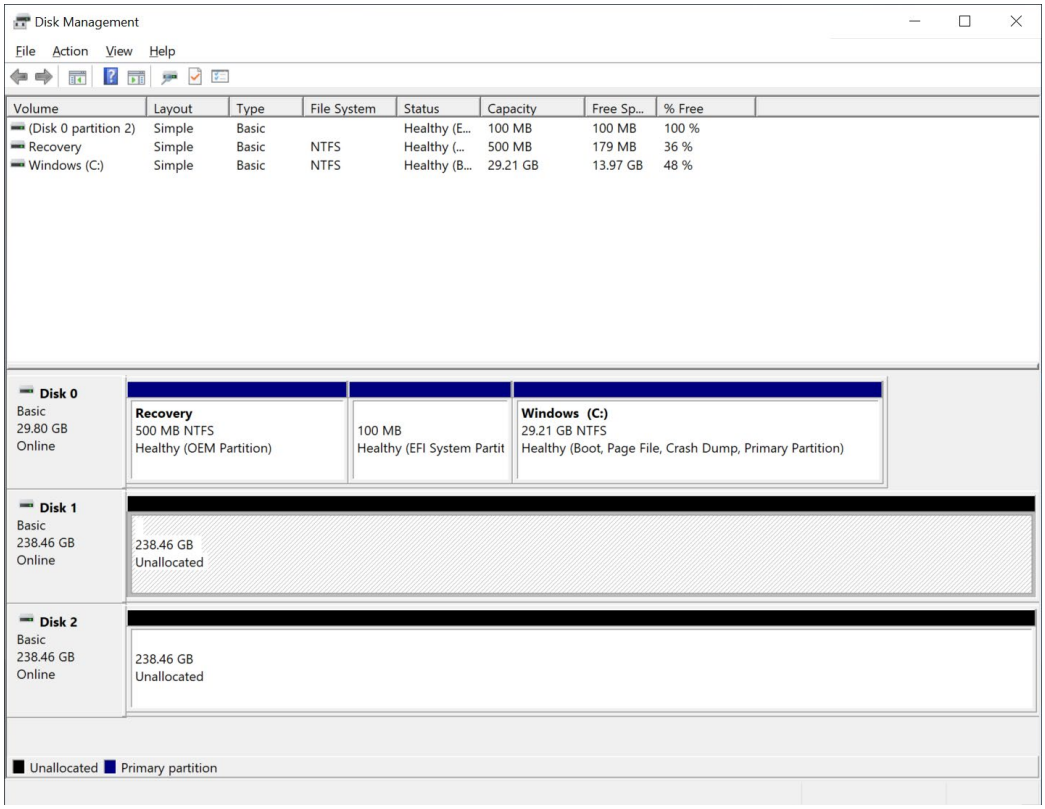
**NOTE**

Using hard disks of the same brand, same model and same capacity to create RAID for best performance.

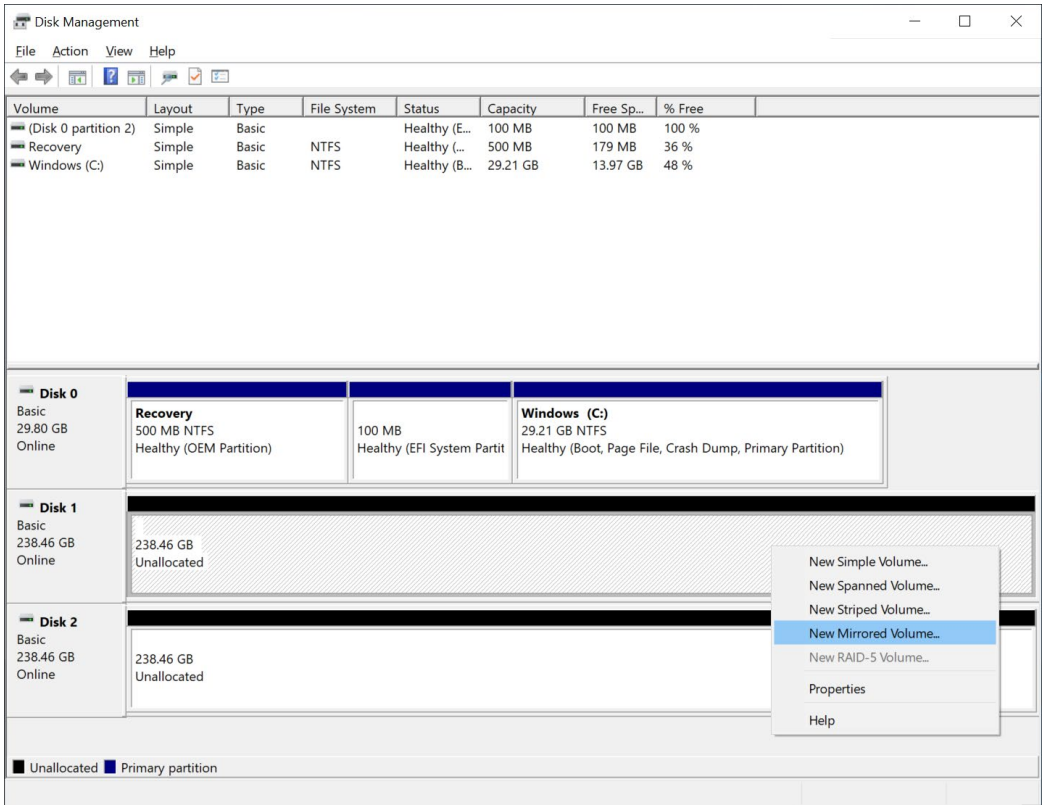
# SW RAID: Creating the RAID 0 or RAID 1 From Disk Management

1. Run the Disk Management.

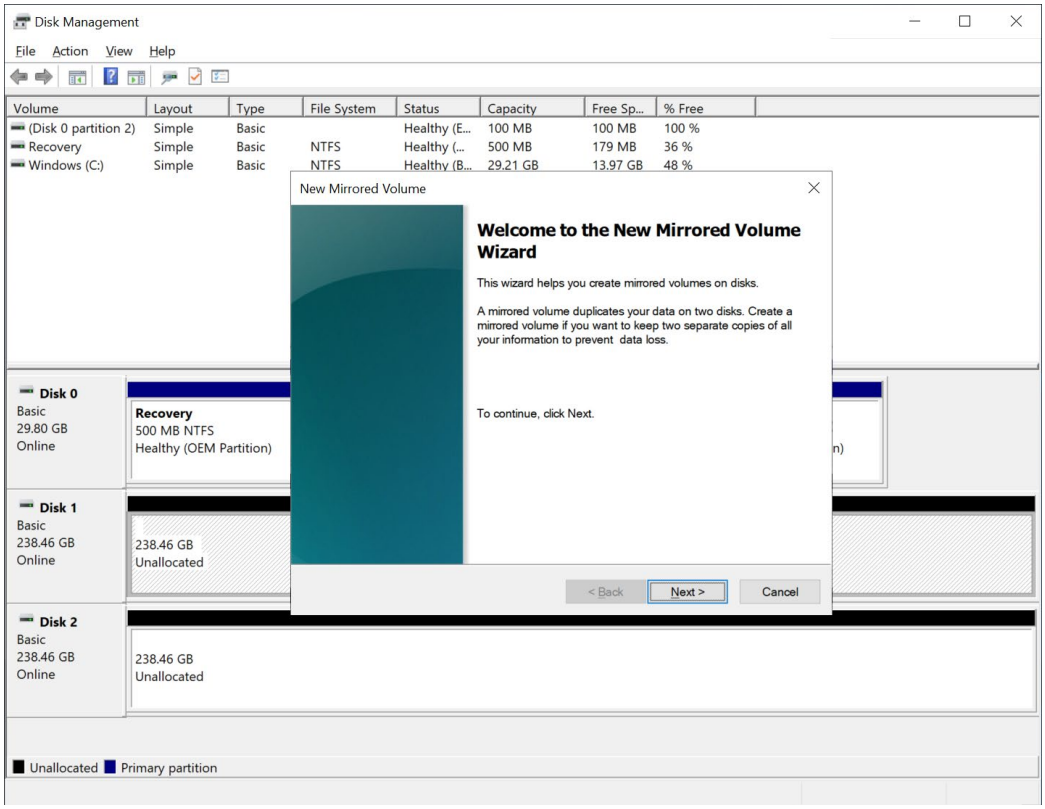
   All connected disks should have the disk status Unallocated. If the disk status is not Unallocated, you can right-click on the disk and select Delete Volume.



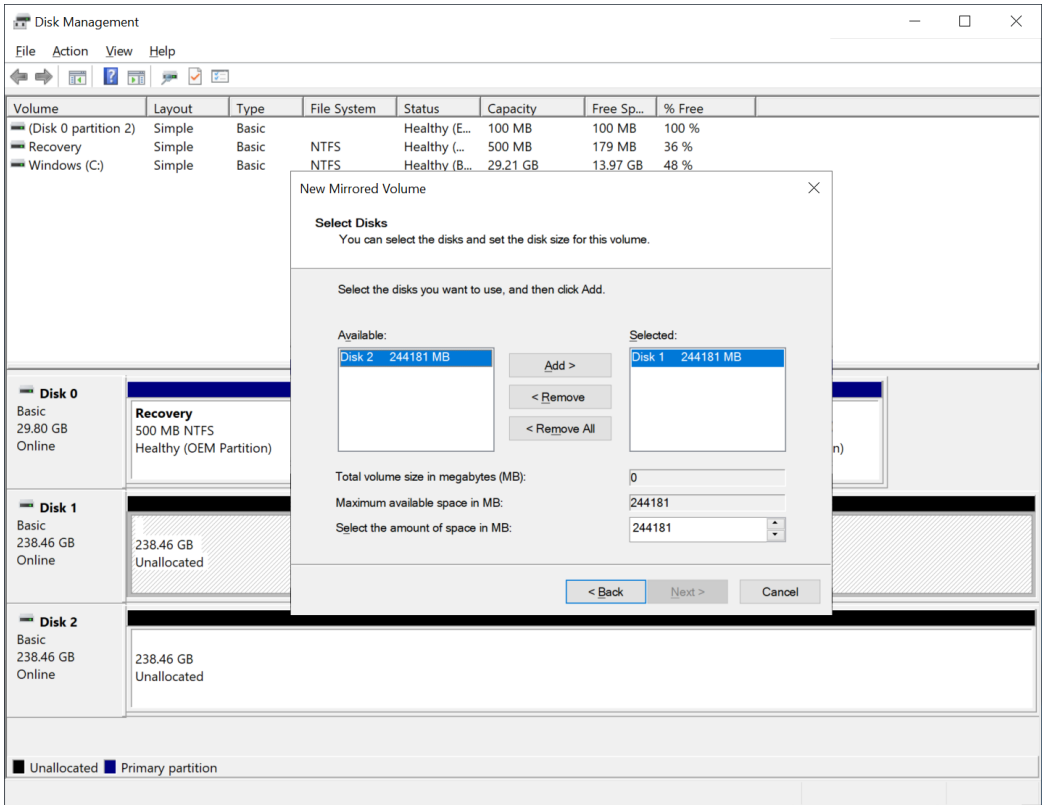
2. Right-click the target disk. Select the target volume type. For example: RAID1(Mirror).



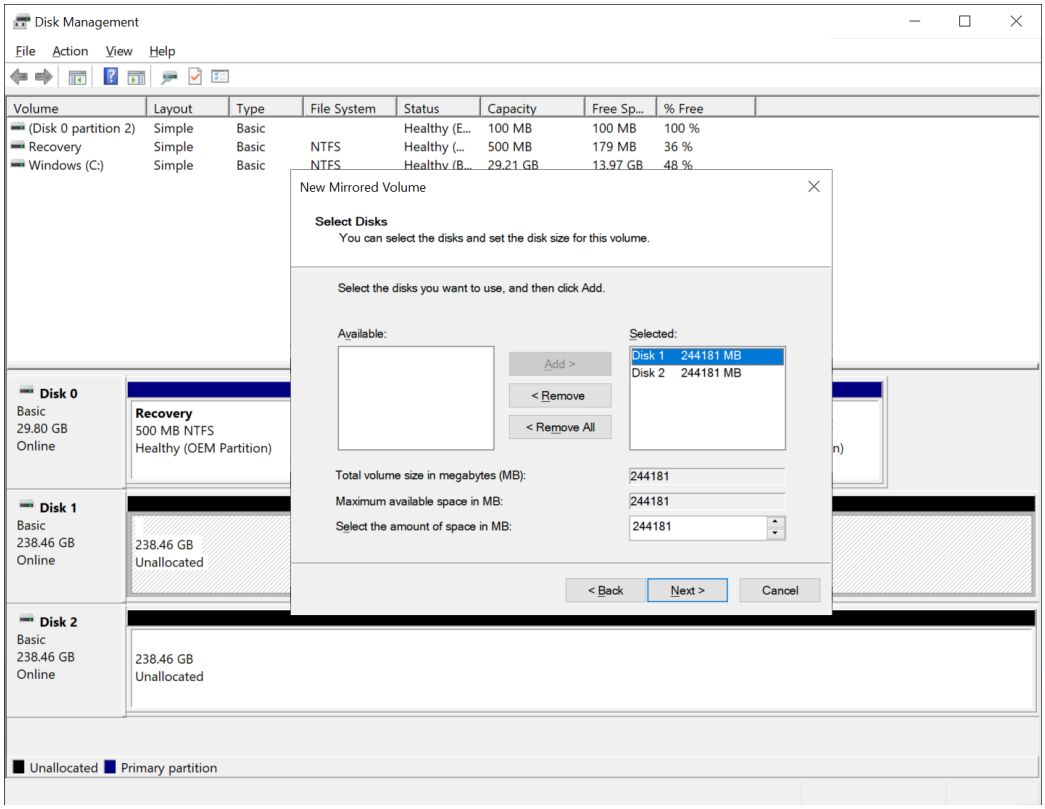

3. To continue, click Next.



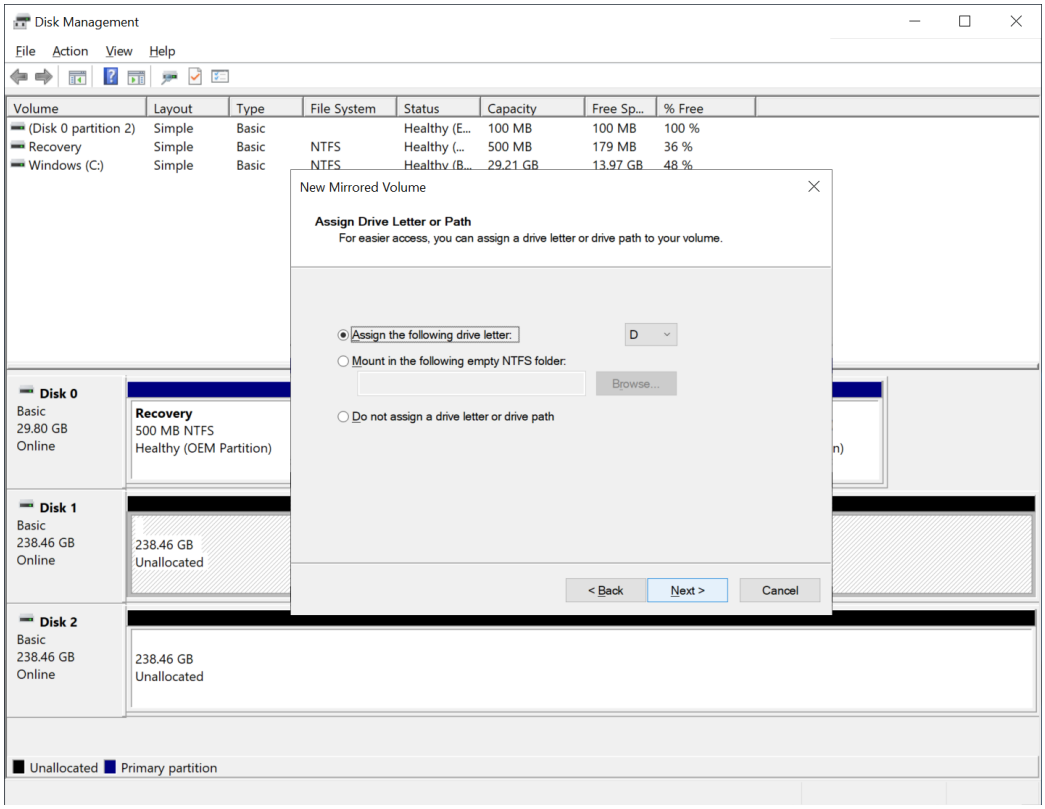
4. Select the disks you want to use, and then click Add.



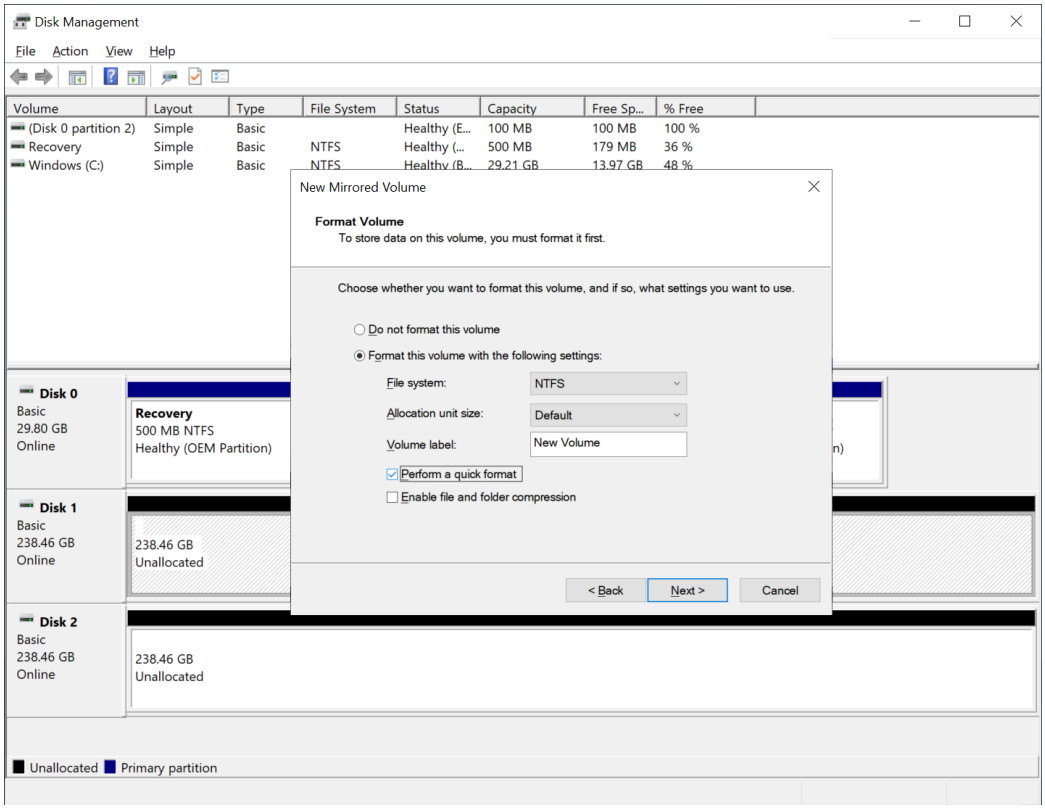

5. Click Next.



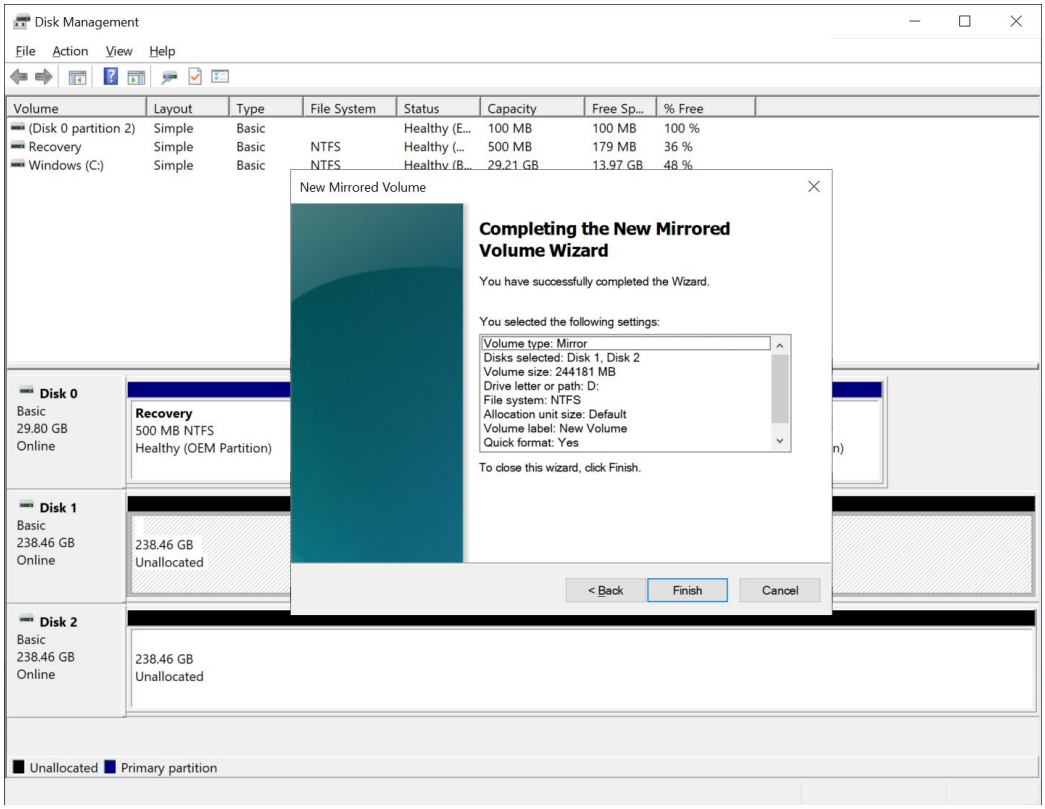
6. Assign the drive letter, click Next.



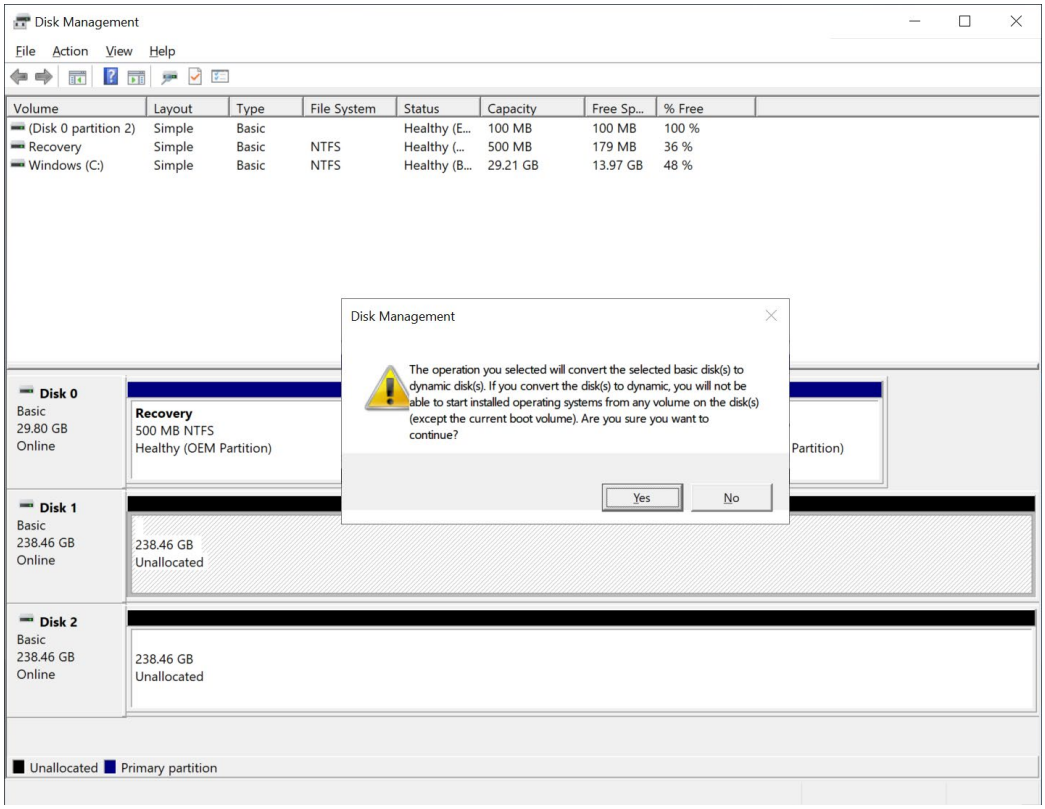

7. Format the volume using Quick Format, click Next.



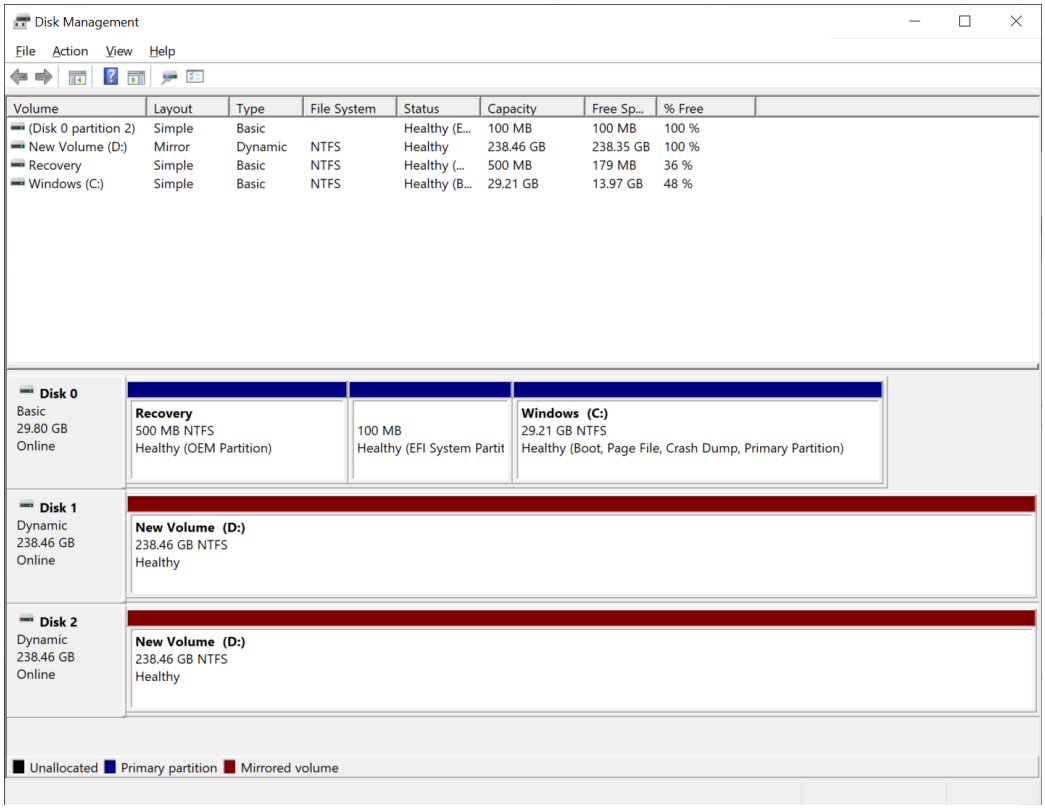
8. Checking the RAID1(Mirror) information. Click Finish to create the RAID1 volume.




9. System will show the warning message about SW RAID volume, click Yes to continue.

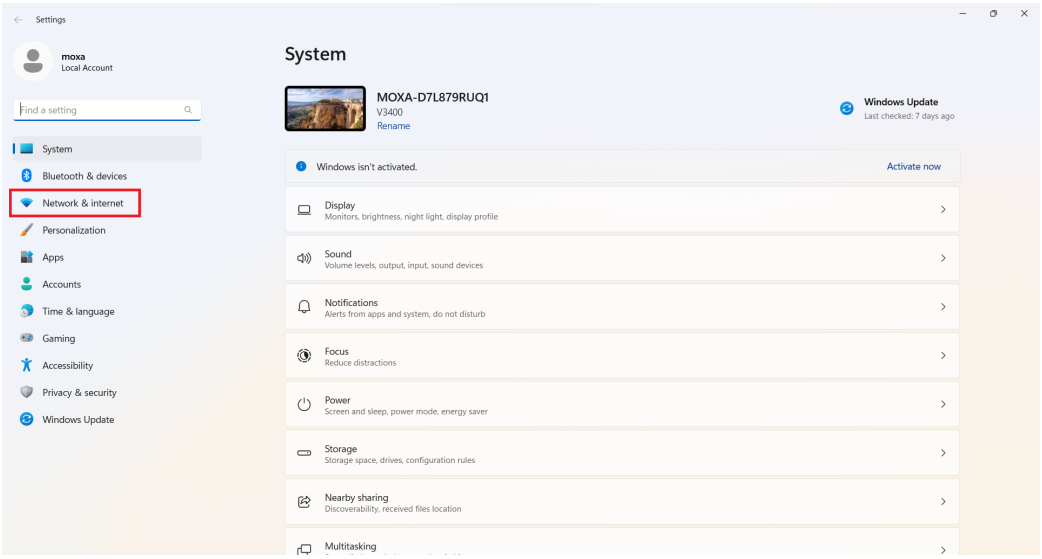


10. Checking the RAID1(Mirror) information from disk management.

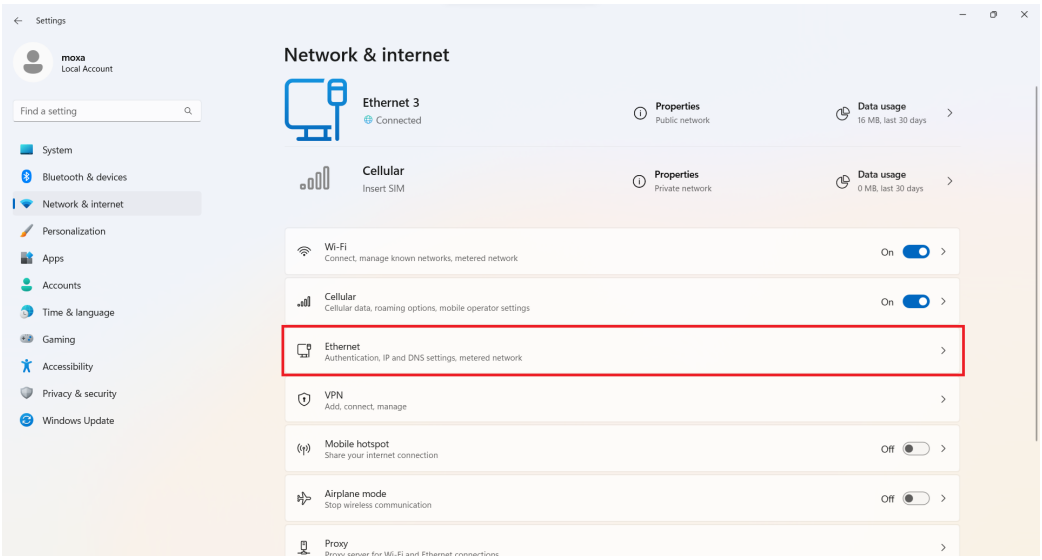


# Ethernet Setting

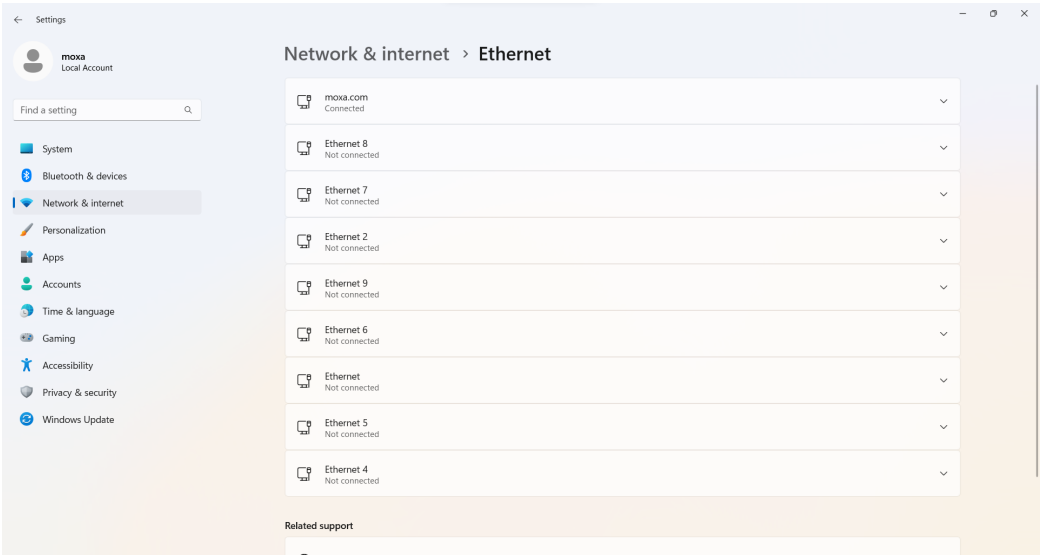
1. In Windows Settings, select **Network&Internet**.

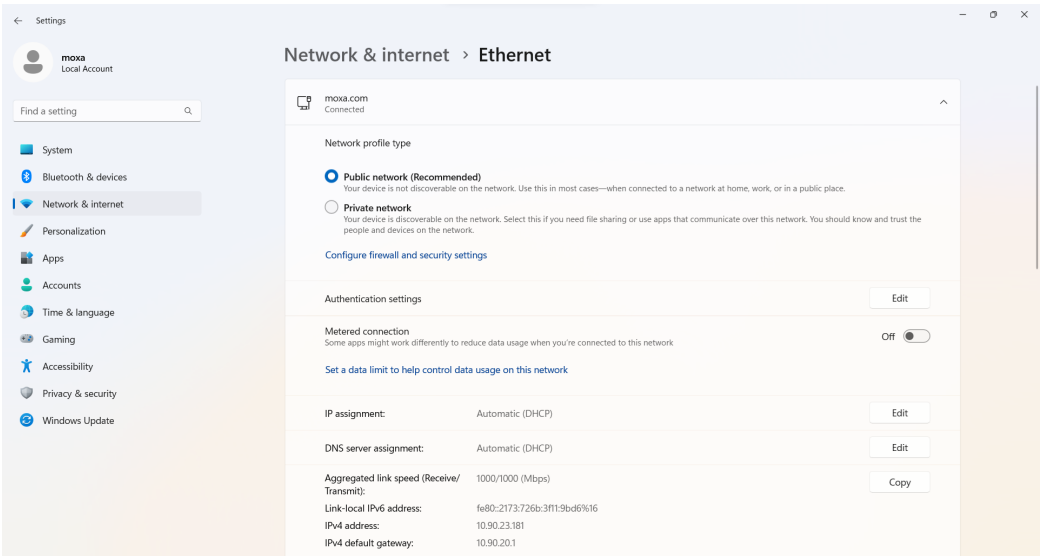



2. Select **Ethernet**.



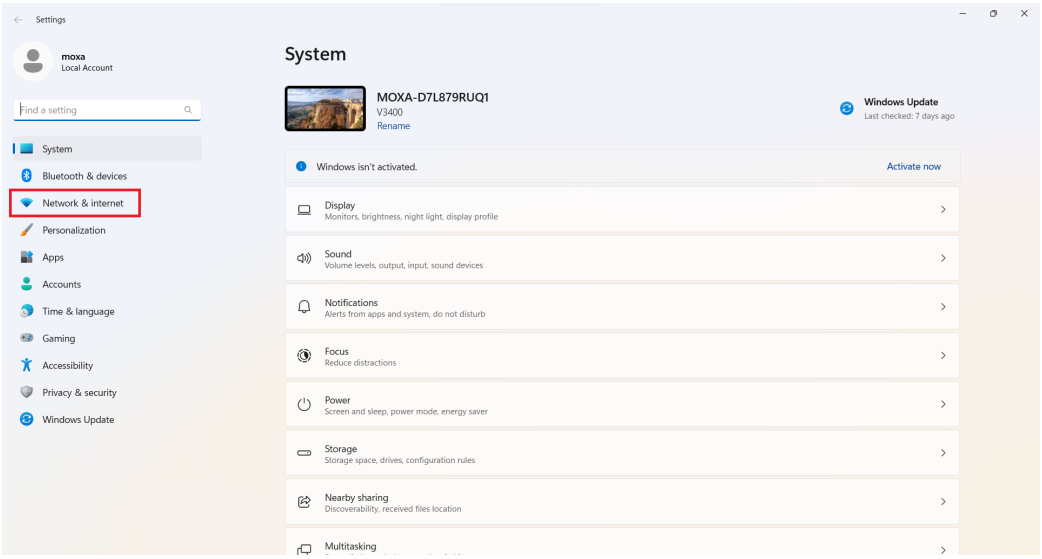
3. Select the Ethernet you want to set.



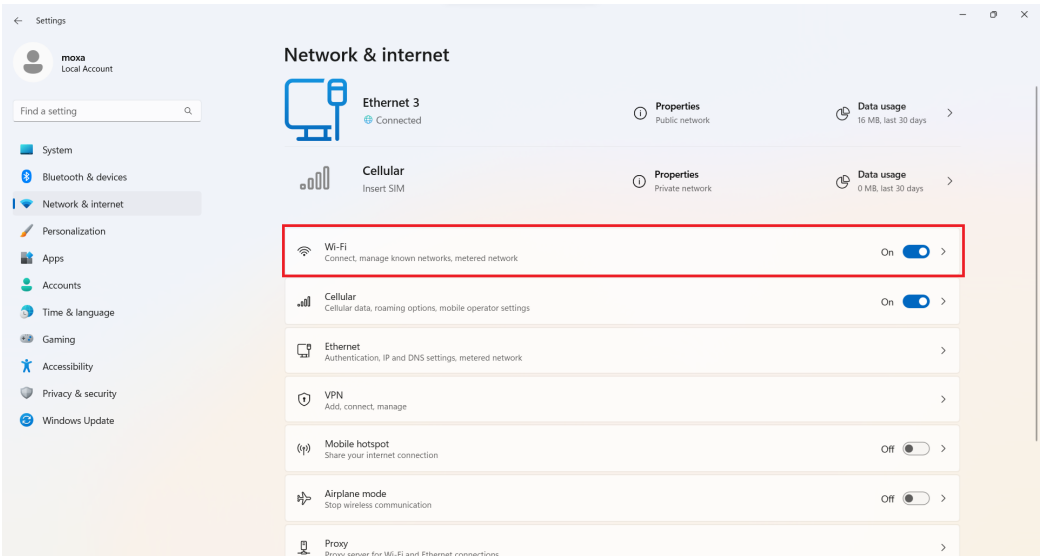



# WiFi Setting

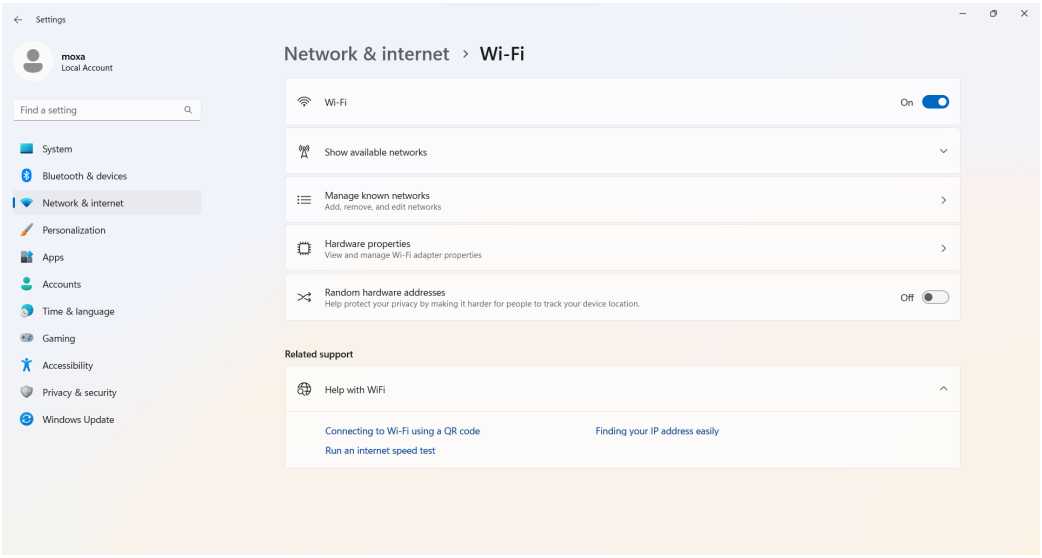
1. In Windows Settings, select **Network&Internet**.




2. Select **Wi-Fi**.



3. Edit Wi-Fi setting.



# 6. Intel® Active Management Technology

Intel® AMT is part of the Intel vPro technology offering. Platforms equipped with Intel® AMT can be managed remotely, regardless of its power state or whether it has a functioning OS or not. The Intel® Converged Security and Management Engine (Intel® CSME) powers the Intel® AMT system. As a component of the Intel vPro platform, Intel® AMT uses a number of elements in the Intel vPro platform architecture.

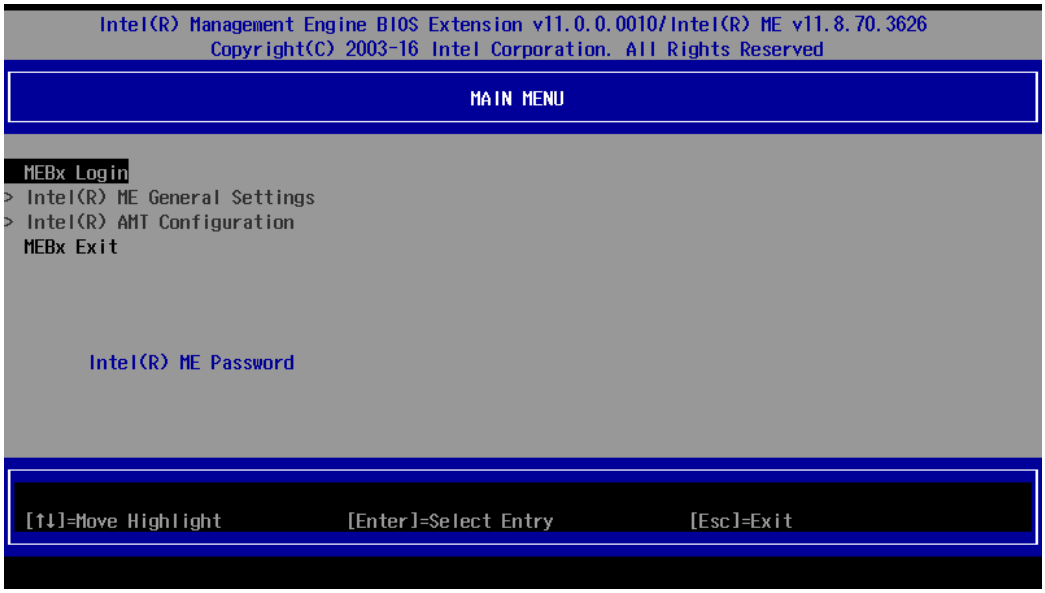
This chapter describes the setup process for the Intel® Active Management Technology. For more information about Intel® Active Management Technology: https://www.intel.com/content/www/us/en/developer/articles/guide/getting-started-with-activemanagement-technology.html?wapkw=AMT

**NOTE**

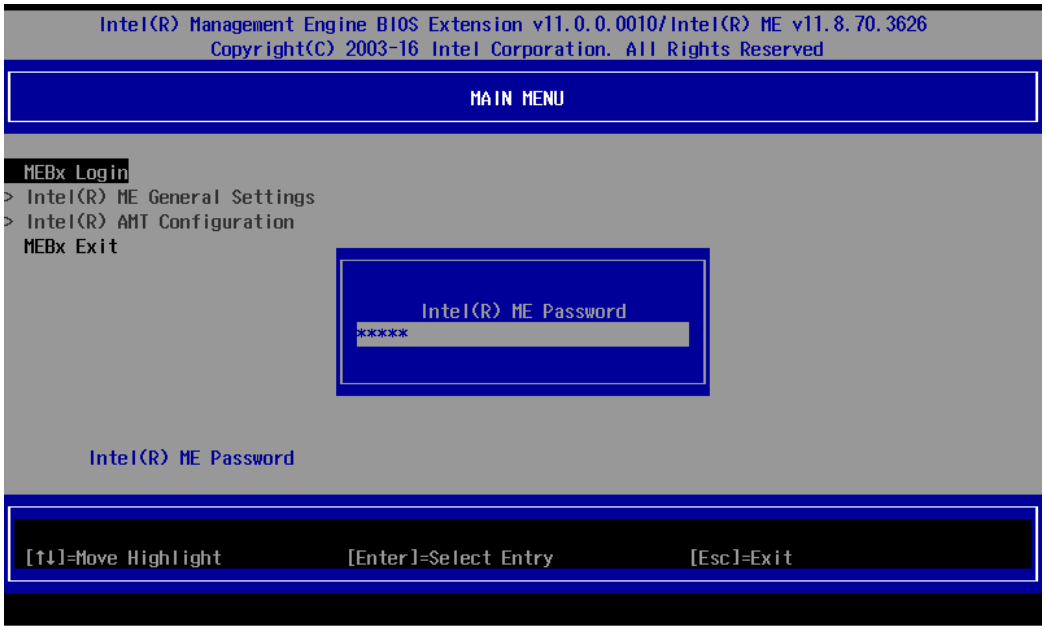
Intel® AMT is not supported in models with Intel® Celeron® and Intel® Core™ i3 processors.

## Turning on Intel® AMT on PC

1. Power on the computer and press F2 to enter the BIOS menu.
2. Select MEBx Login.



3. Enter the Intel® ME default password admin.




4. Enter a new password.

   The Intel® MEBX password must meet the following requirements for strong passwords:

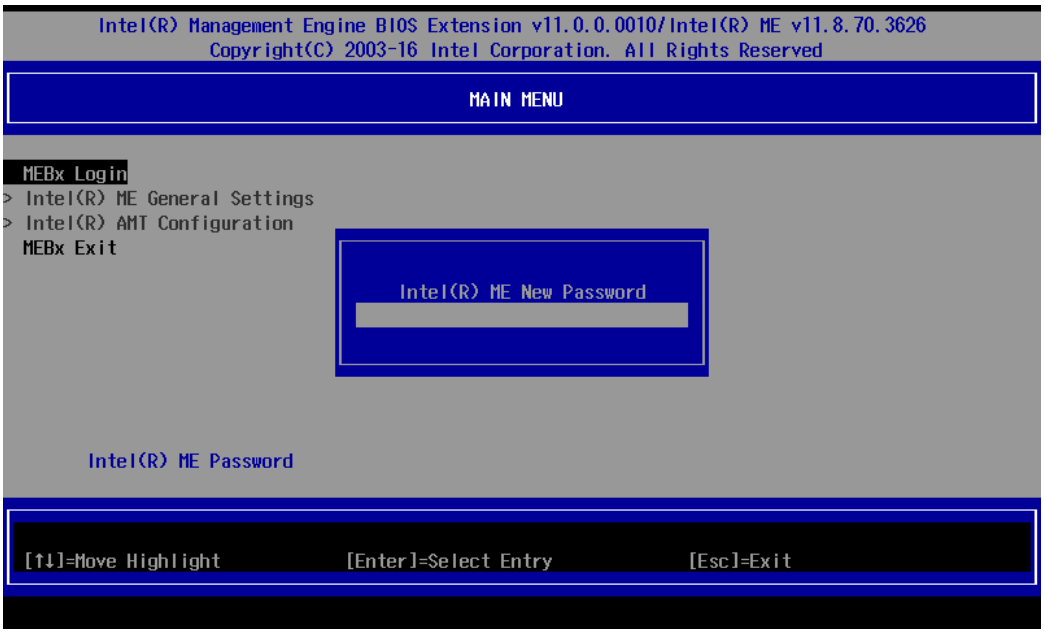
   a. Password Length: At least 8 characters, and no more than 32.

   b. Password Complexity: Password must include the following:

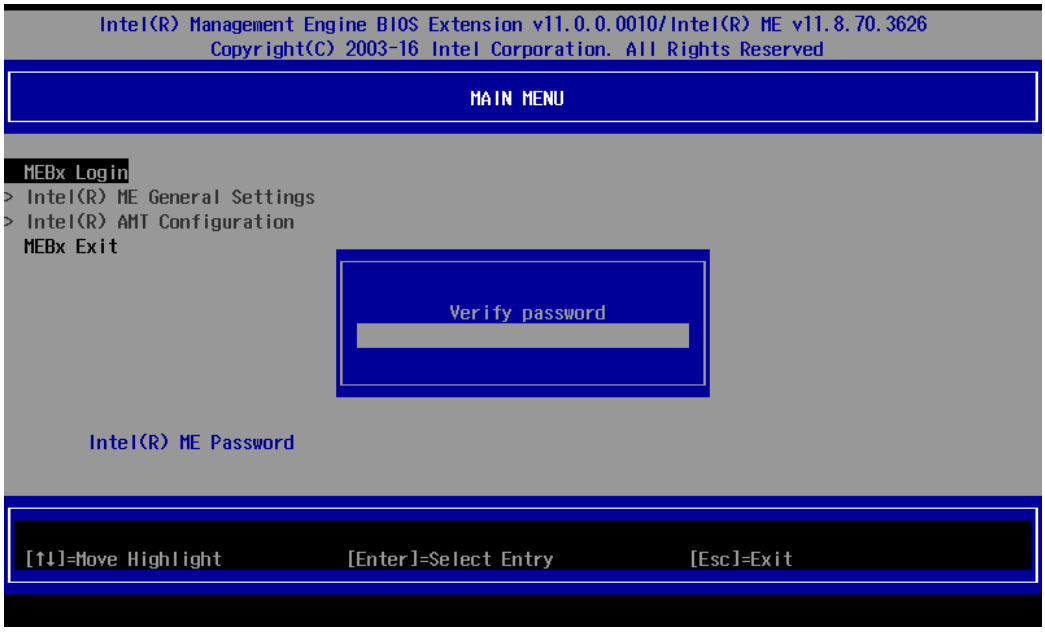
      i. At least one digit character ('0', '1', … '9').

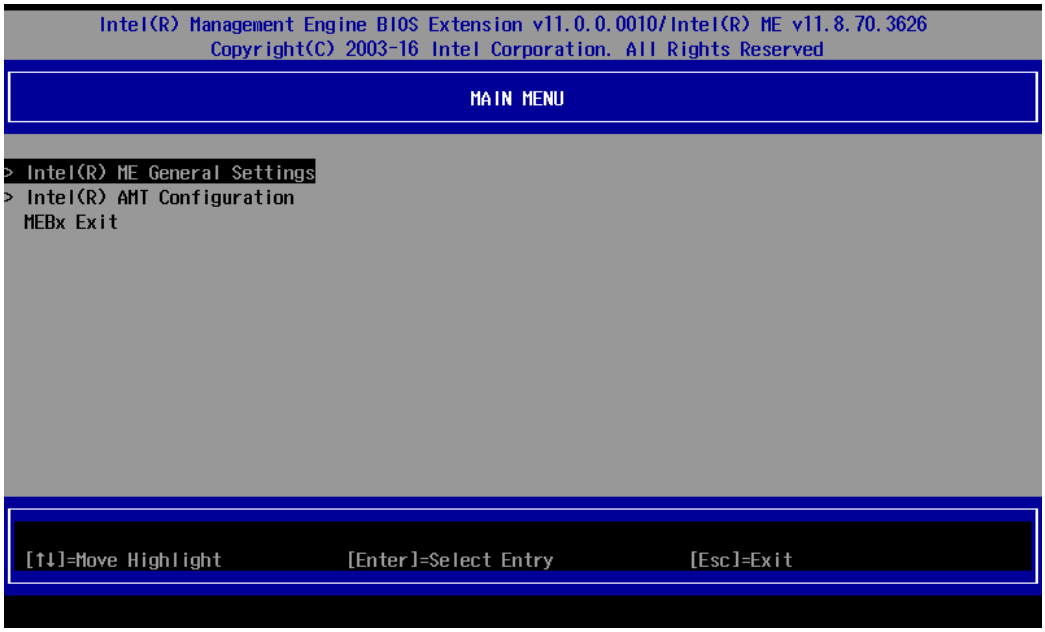
      ii. At least one 7-bit ASCII non alphanumeric character (e.g., '!', '$', ';'), but excluding ':', ',' and '"' characters.

      iii. At least one lower-case letter ('a', 'b'…'z') and at least one upper case letter ('A','B'…'Z').

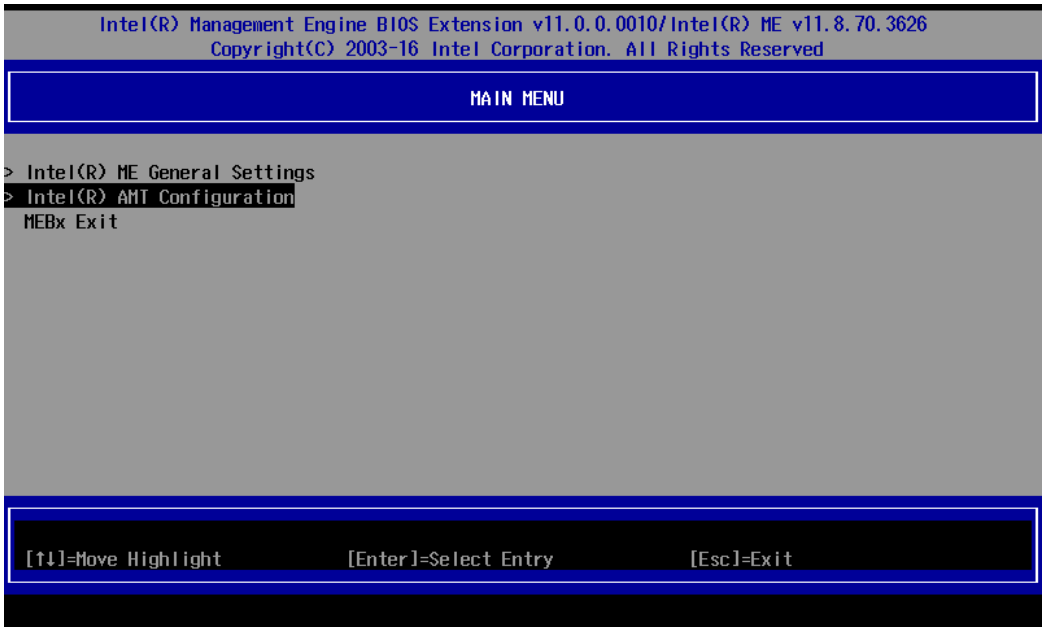


5. Verify password and login MEBx.

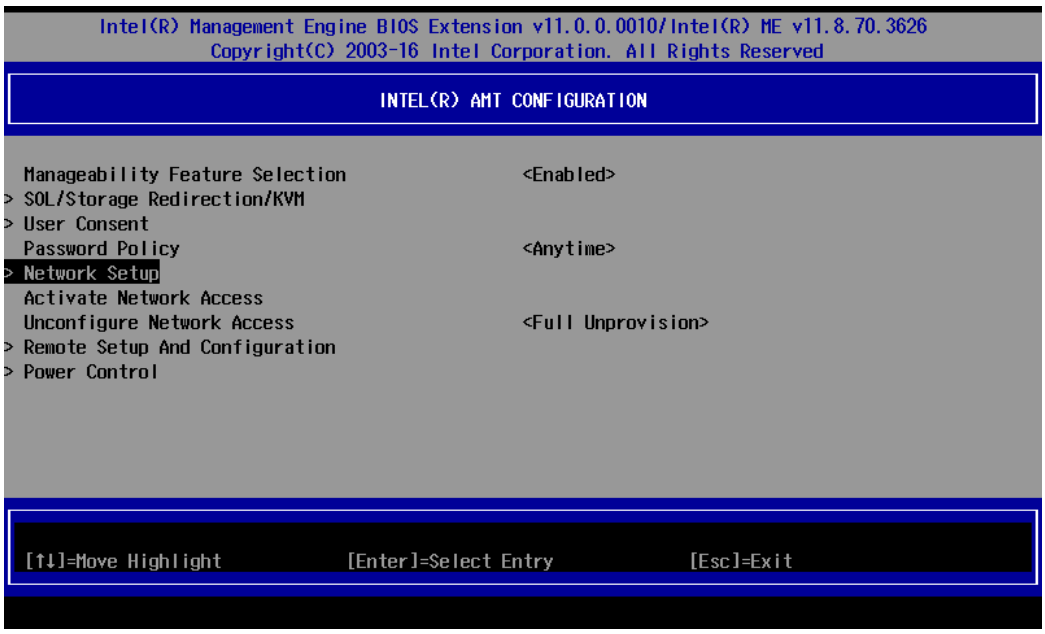


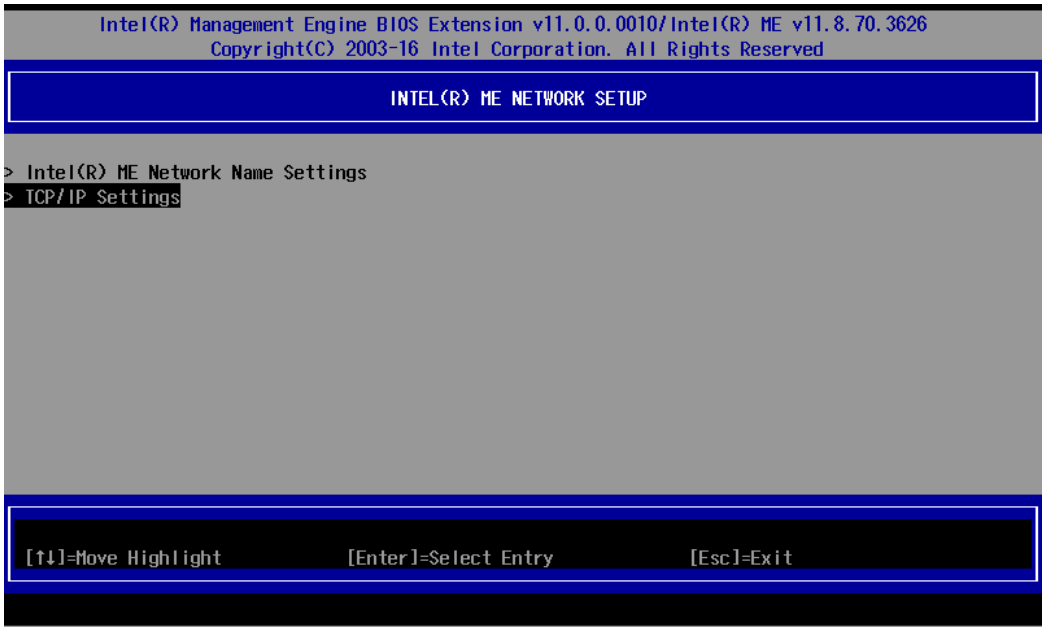
6. Select Intel(R) AMT Configuration.



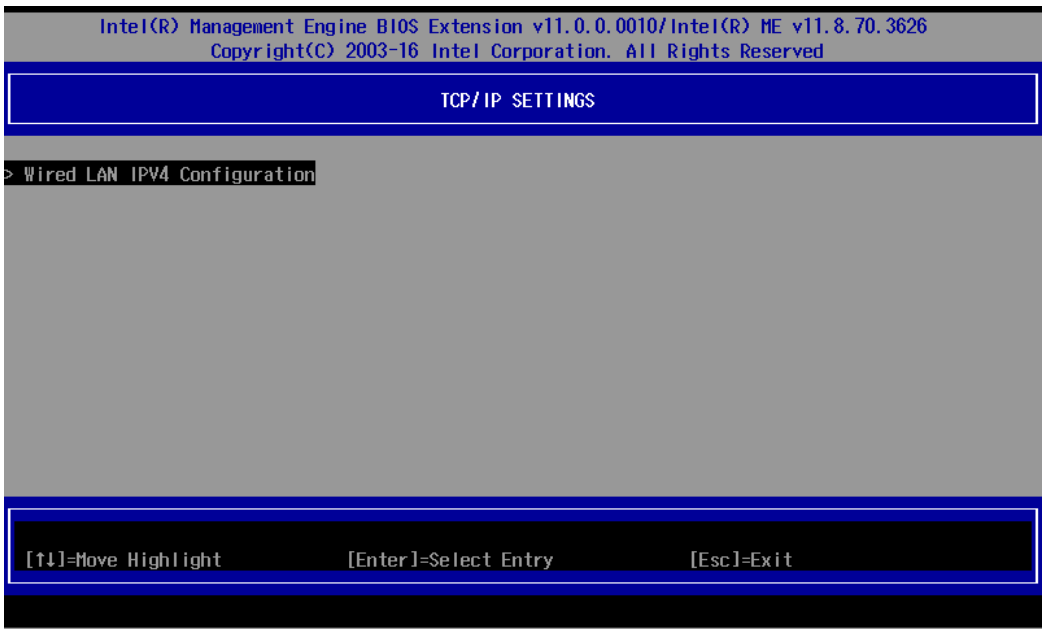

7. Select Network Setup.



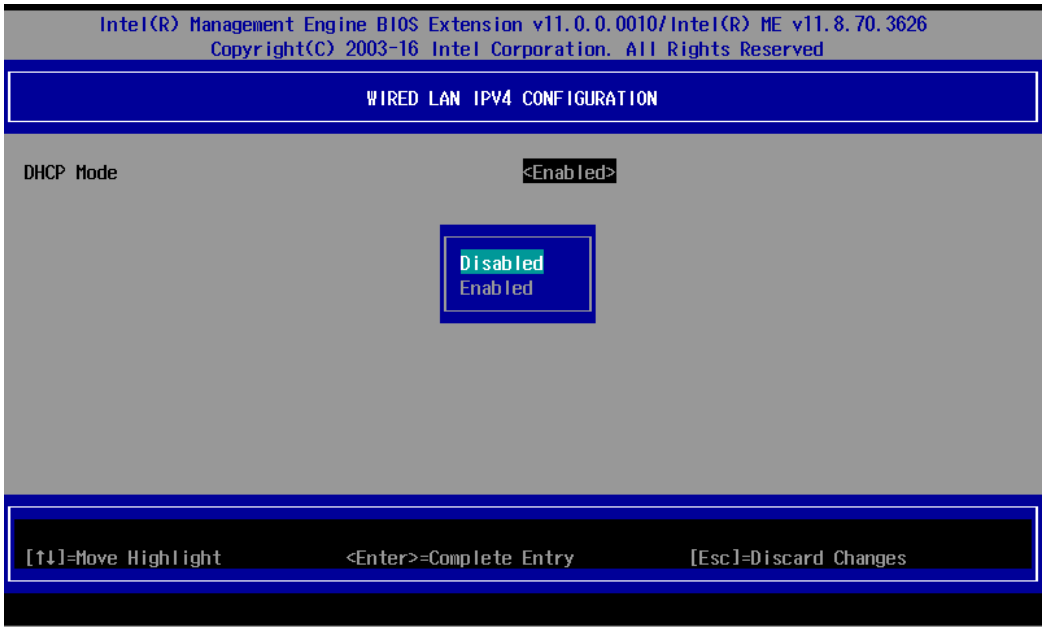
8. Select TCP/IP Settings.



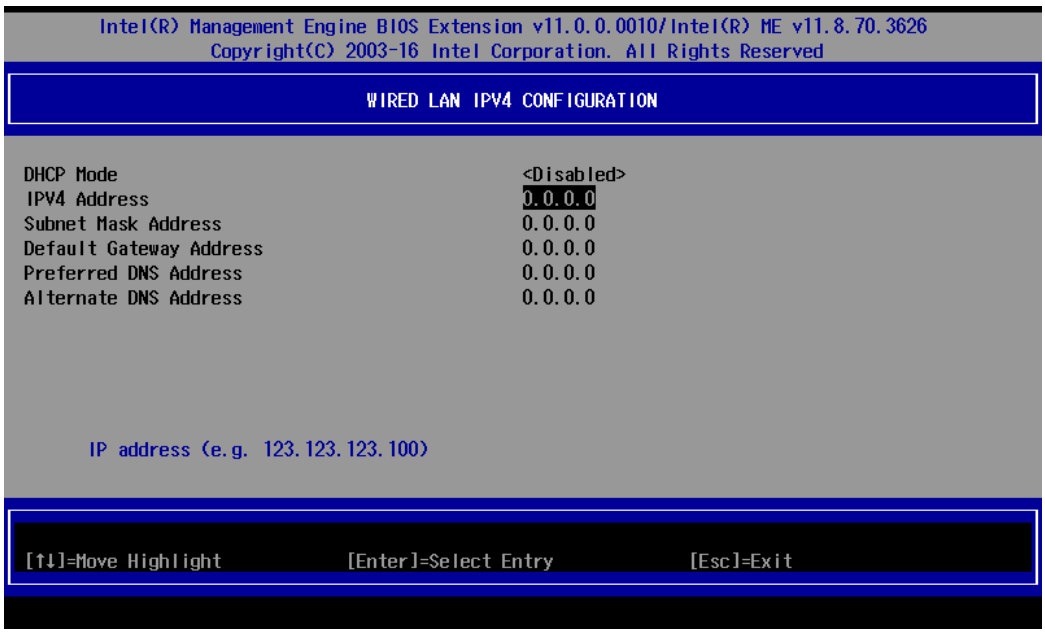

9. Select Wired LAN IPV4 Configuration.



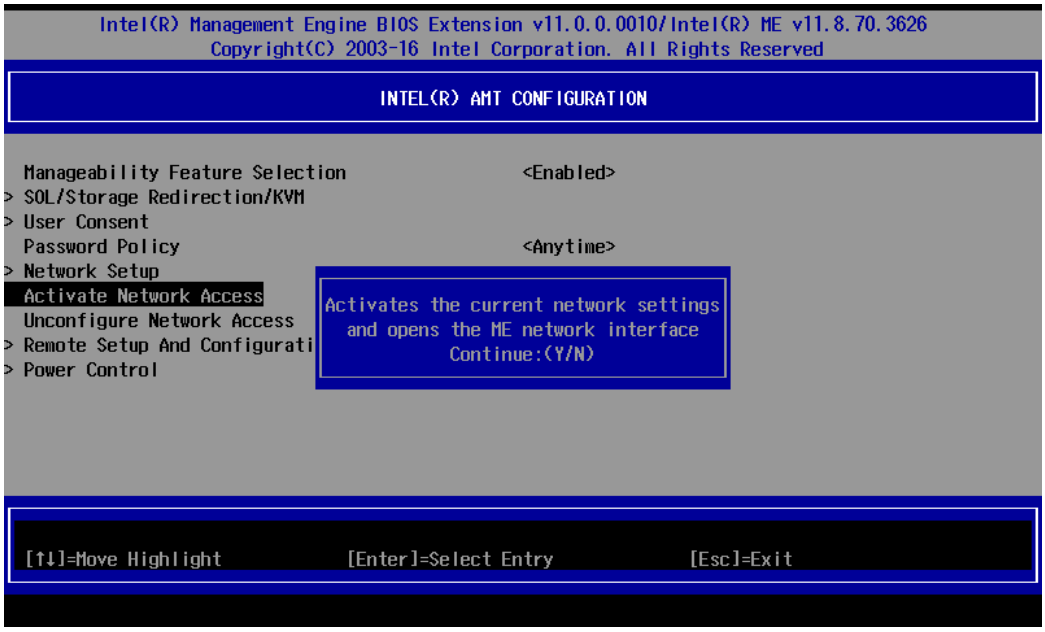
10. Select DHCP Mode and disable DHCP mode.



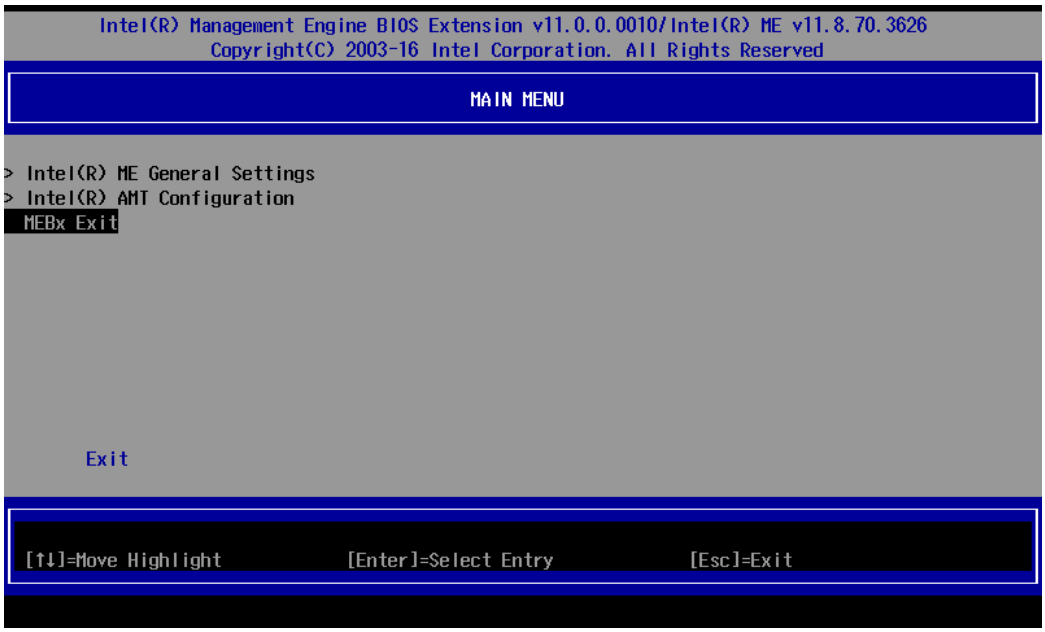

11. Type the network settings for Intel® Active Management Technology.



12. Back to the Intel(R) AMT Configuration page. Select Activate Network Access. Enter Y to continue.



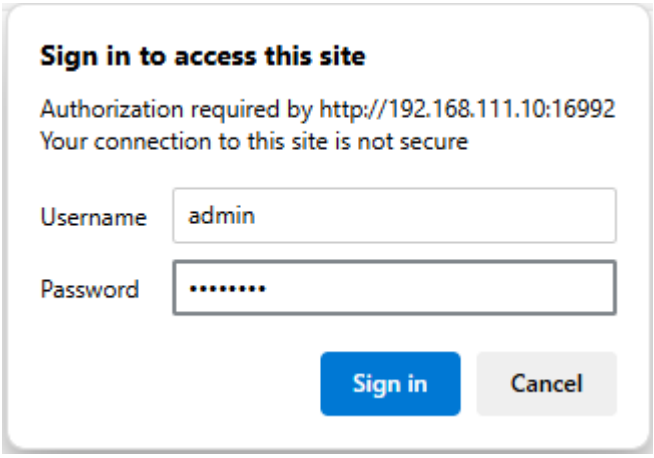

13. Back to the main menu. Select MEBx Exit to finish the Intel® Active Management Technology configuration.



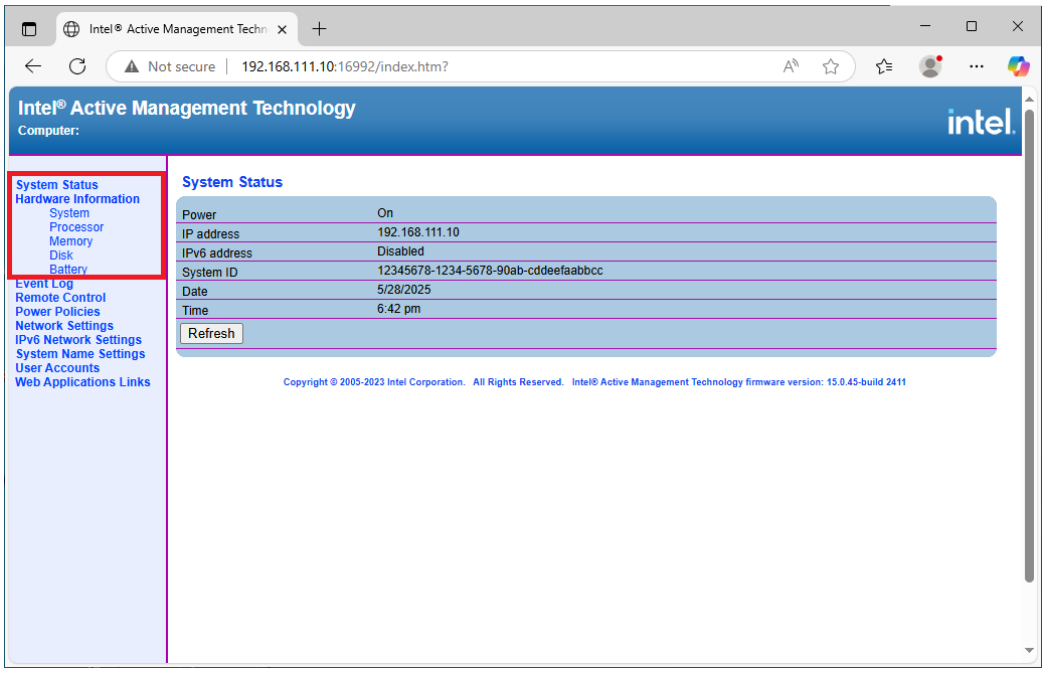
# Access the Intel® AMT From Website

1. Open the web browser and type the URL: Intel® AMT IP Address:16992 (ex: 192.168.111.10:16992).

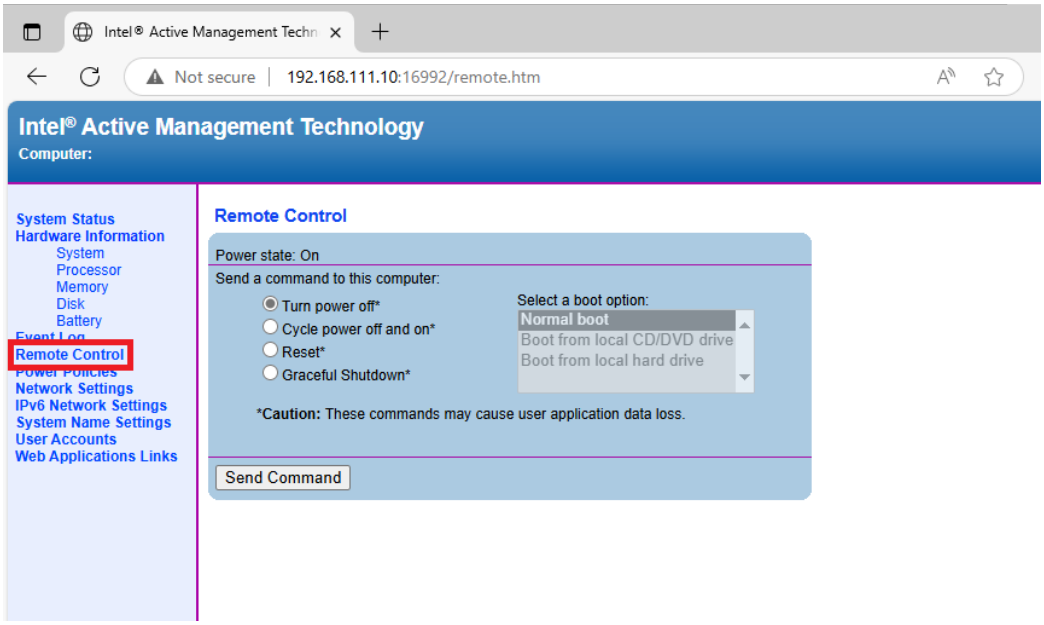
2. The browser would show the sign in message box. Type the Username and Password of Intel® AMT. The default username is admin.



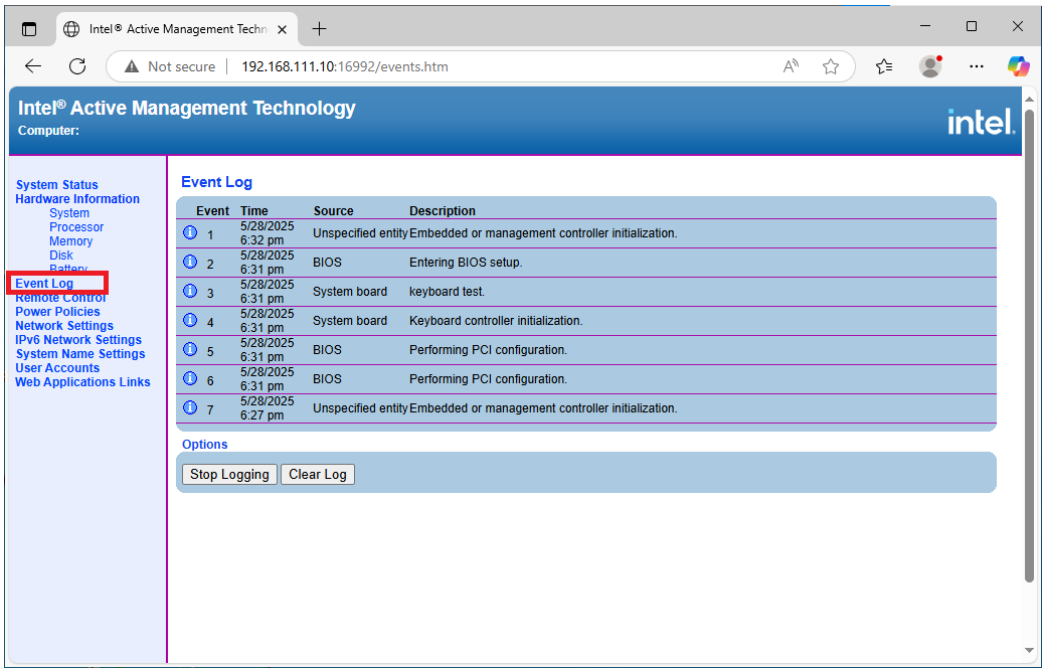

3. After signing in, you can check the system status and hardware information of your managed device.



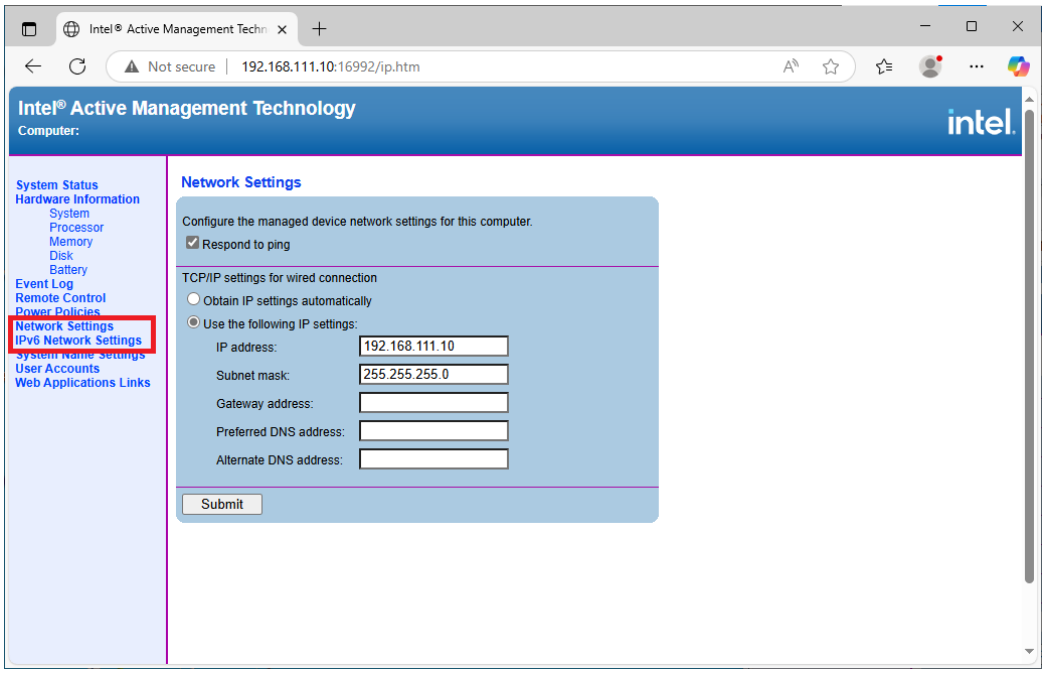
4. The Intel® AMT website provides the basic remote power control feature for the managed device. The advanced remote power control and the remote KVM feature please reference to next chapter.




5. The Event Manager deals with internal alerts that occur in both the host platform and the Intel® AMT device, regardless of the power state.

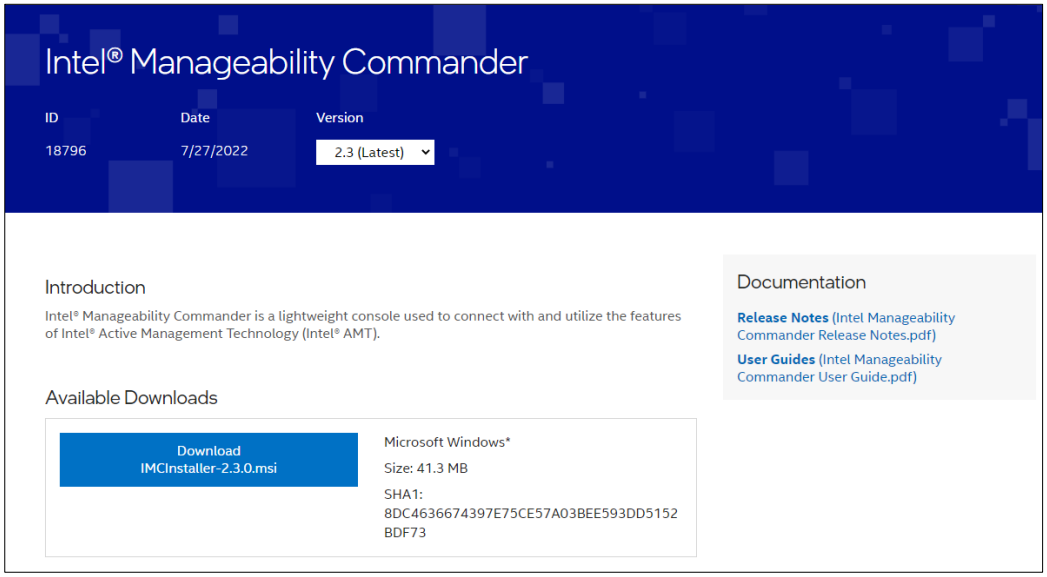


6. You can configure the managed device network settings from the website.

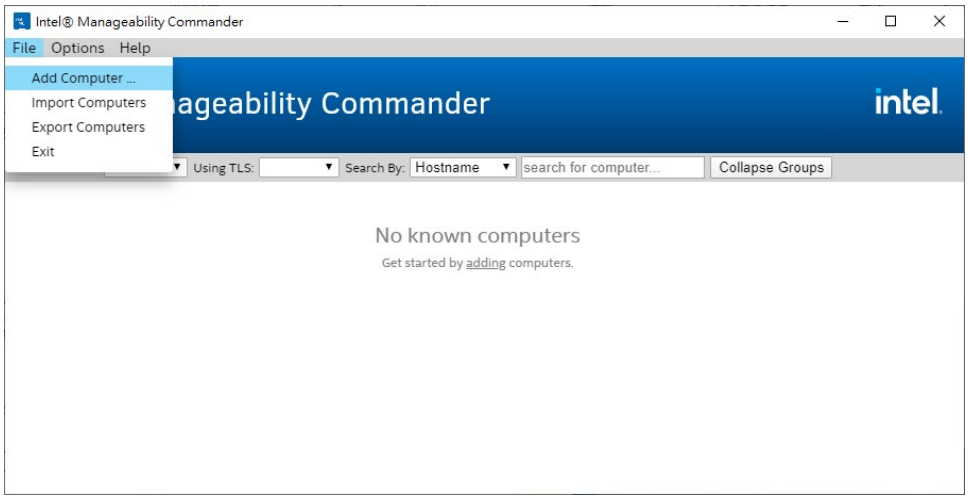



# Access the Intel® AMT by Intel® Manageability Commander

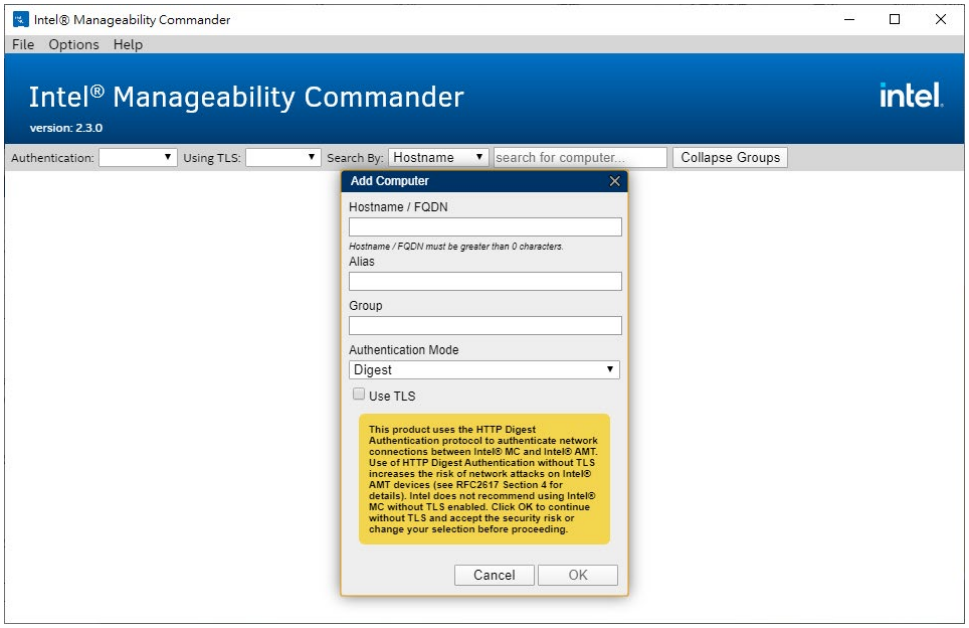
1. Download and install the latest version Intel® Manageability Commander from http://Intel.com.



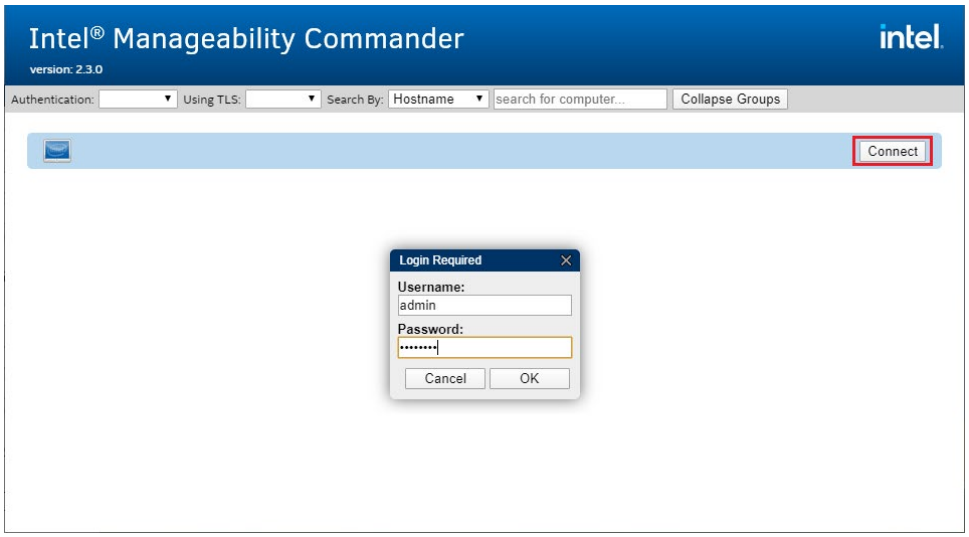
2. Execute the Intel® Manageability Commander as administrator. Select File > Add Computer....

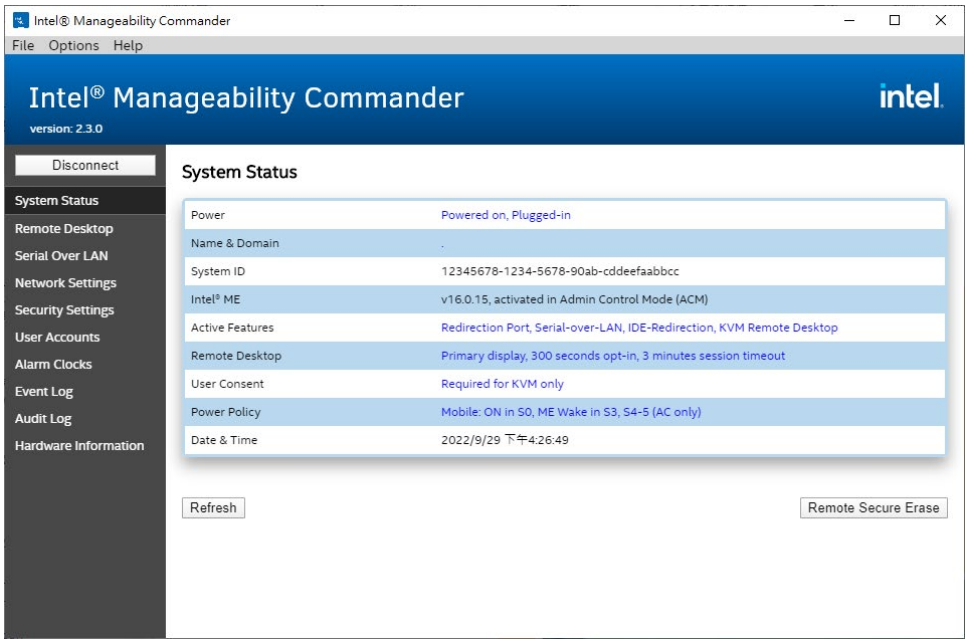



3. Enter the hostname, alias, and group. The hostname is the IP address of managed device Intel® AMT. The alias and group could keep empty value if not used. Click OK to apply settings.




4. Select Connect and enter the Username and Password of Intel® AMT. The default Username is admin.





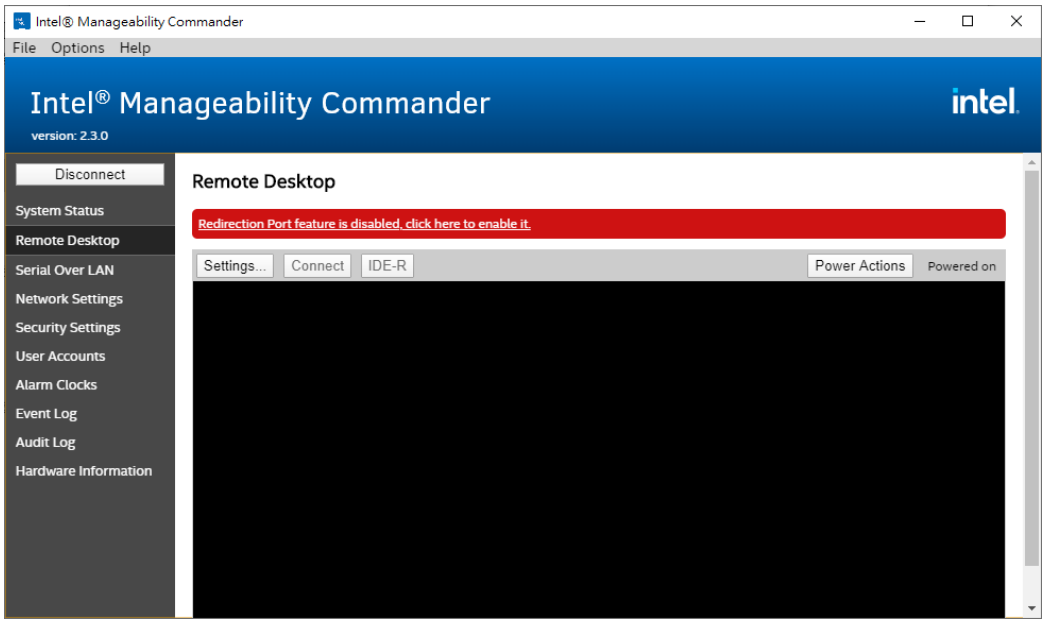

# Remote Desktop

Intel® Manageability Commander provides a remote KVM function.

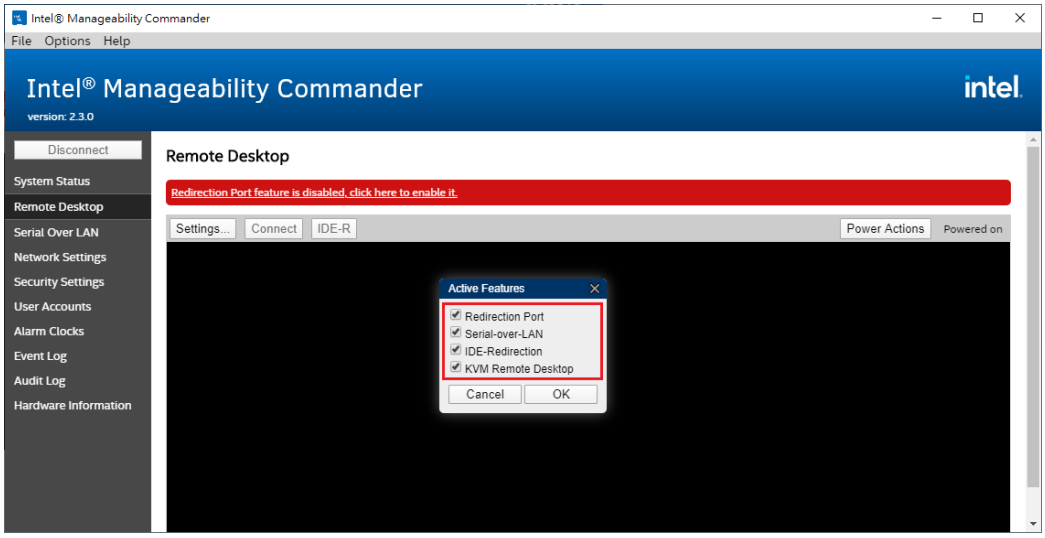
To connect to a managed device, do the following:

1. Select Remote Desktop.

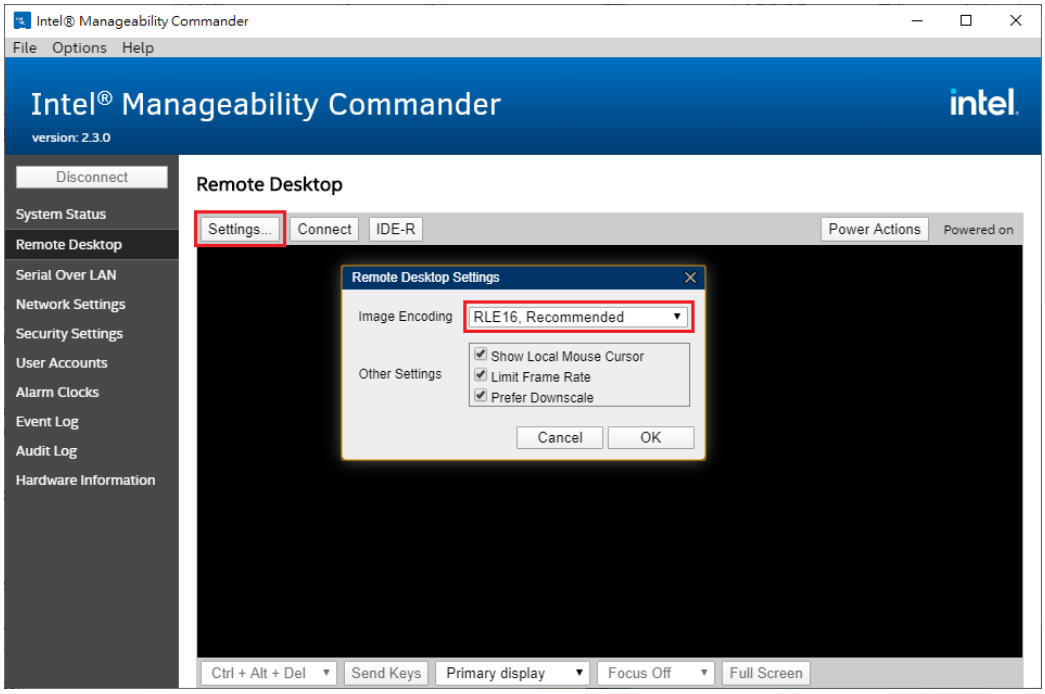
   Select Remote Desktop. The redirection port feature is disabled when the remote function is activated for the first time. Click Yes on the warning message to enable it.



2. Enable all features. Click OK to apply settings.

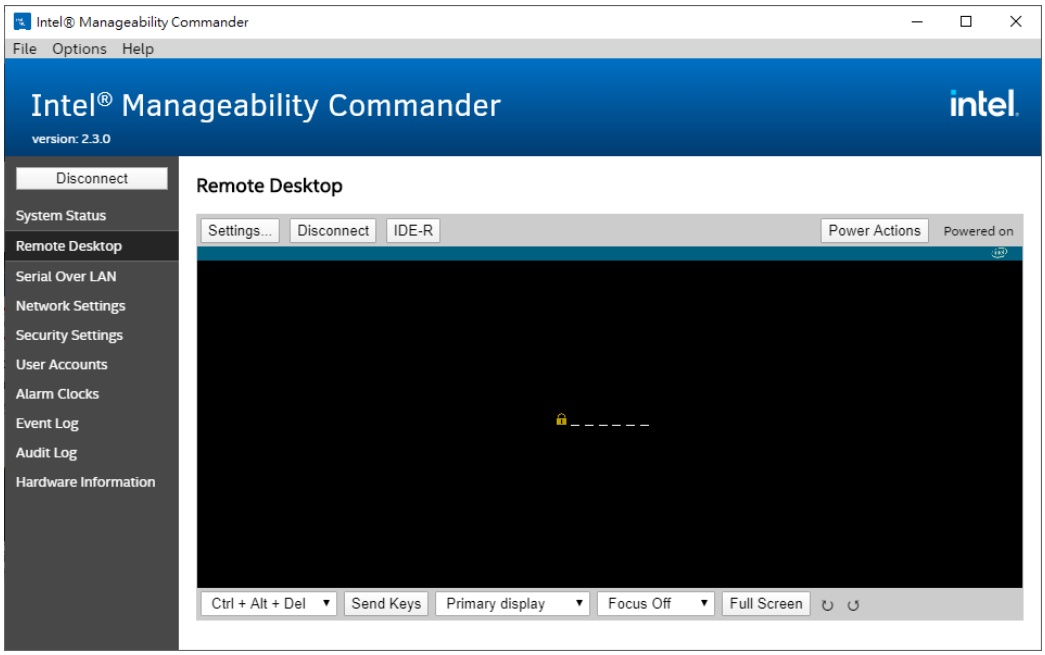



3. Click Settings..., Change the Image Encoding to RLE16, Recommended. Click OK to apply settings.

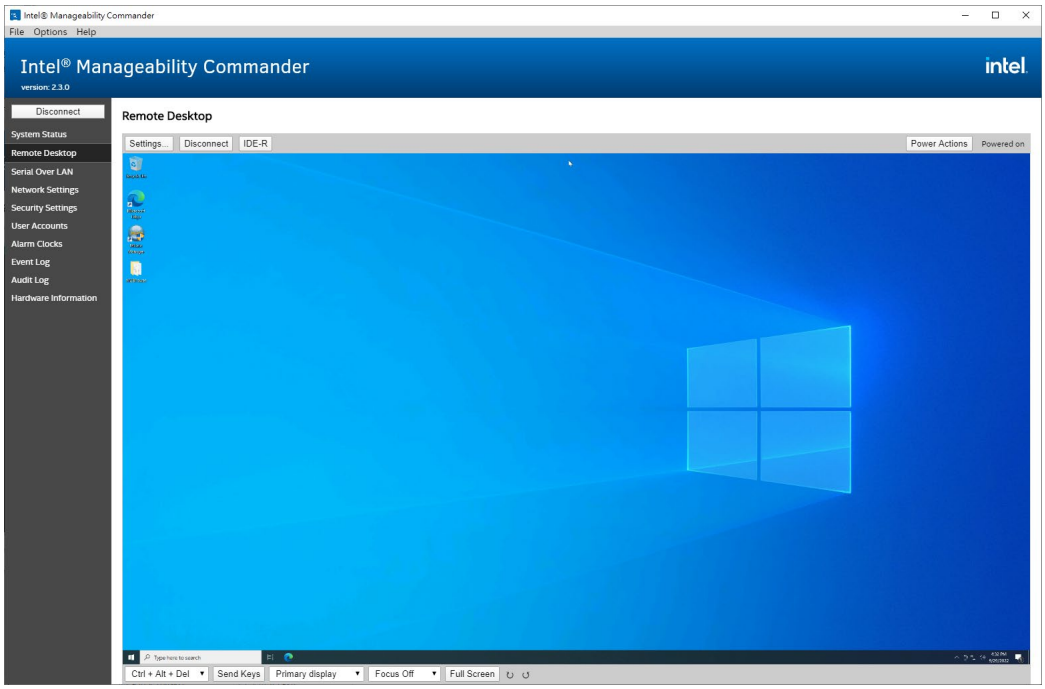


4. Click Connect and enter the User Consent code.

   The User Consent code is provided on the client-side as a sprite on the Intel® AMT device's display. This sprite is generated by Intel® GPU and is not available to the OS. This is a 6-digit code that the technician will use when making a connection that requires user consent, such as an Intel® KVM connection.

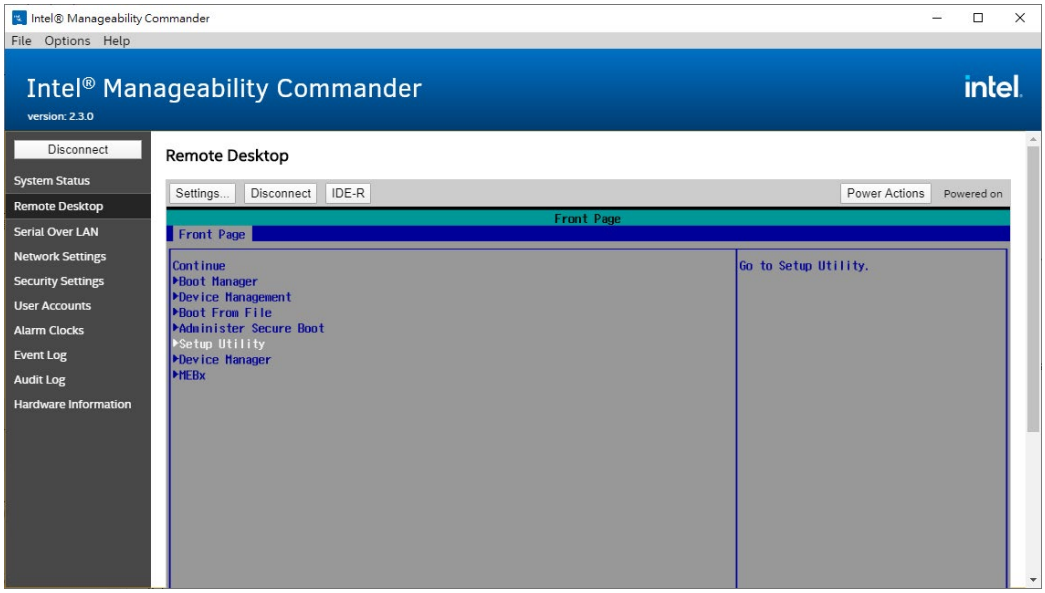


5. The Intel® AMT device authenticates the User Consent code following which you can run the remote desktop function.



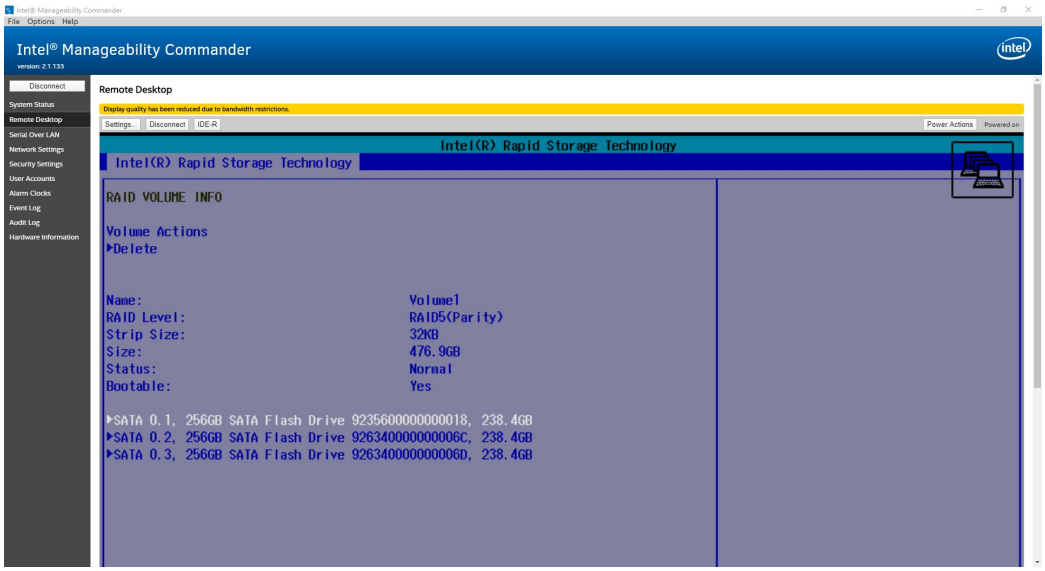

# Advanced Power Control

Intel® Manageability Commander provides the advanced power control features. Click Power Action to control managed device. For example, reset the device power and boot to the BIOS menu. With the advanced power control features, such as RAID Configuration, Boot Manager, and OS Recovery, you can configure the BIOS settings and manage the device more easily.



# Remote RAID Configuration

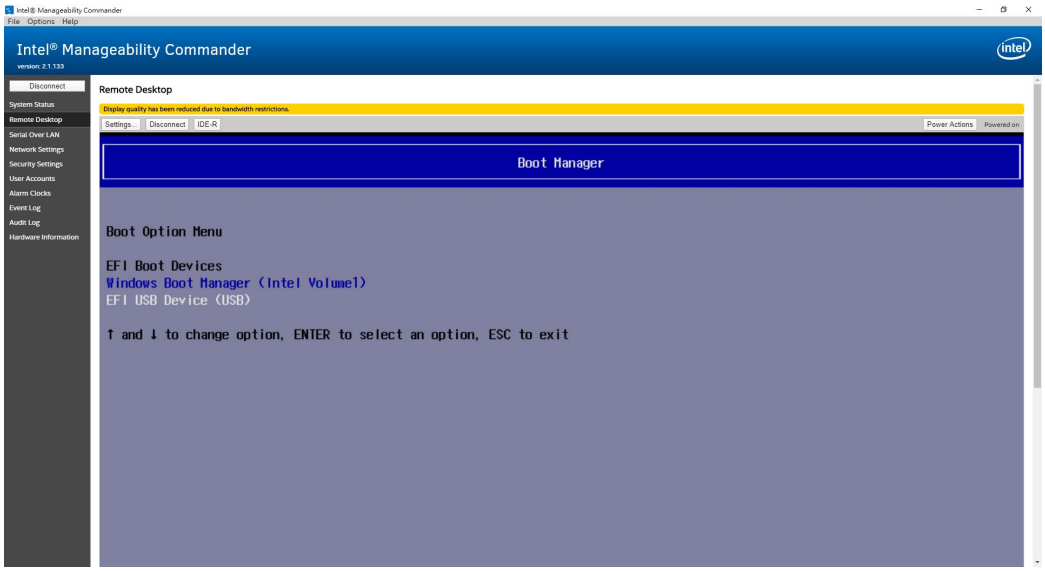
Reset to BIOS menu and select Setup Utility option. Reference Chapter 4 Configuring RAID to configure the RAID settings.




# Remote Boot Manager

Reset to BIOS menu and select Boot Manager.

You can select the bootable USB disk to back up the OS image or use a bootable disk, which includes recovery image, to recover the OS.

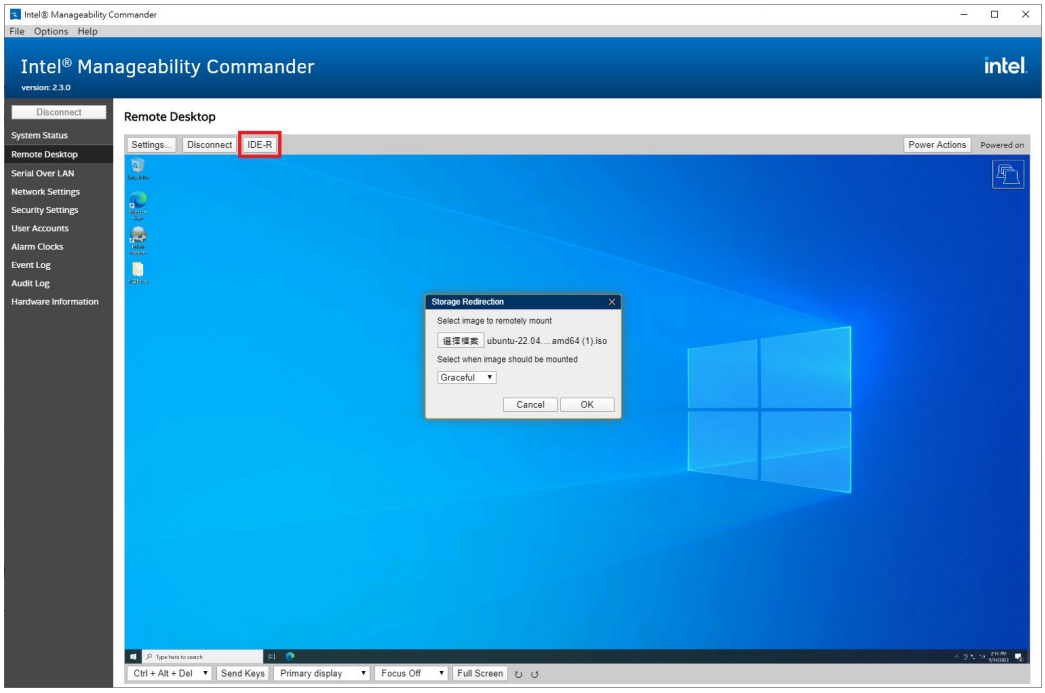


# Remote Mount/Recovery Image

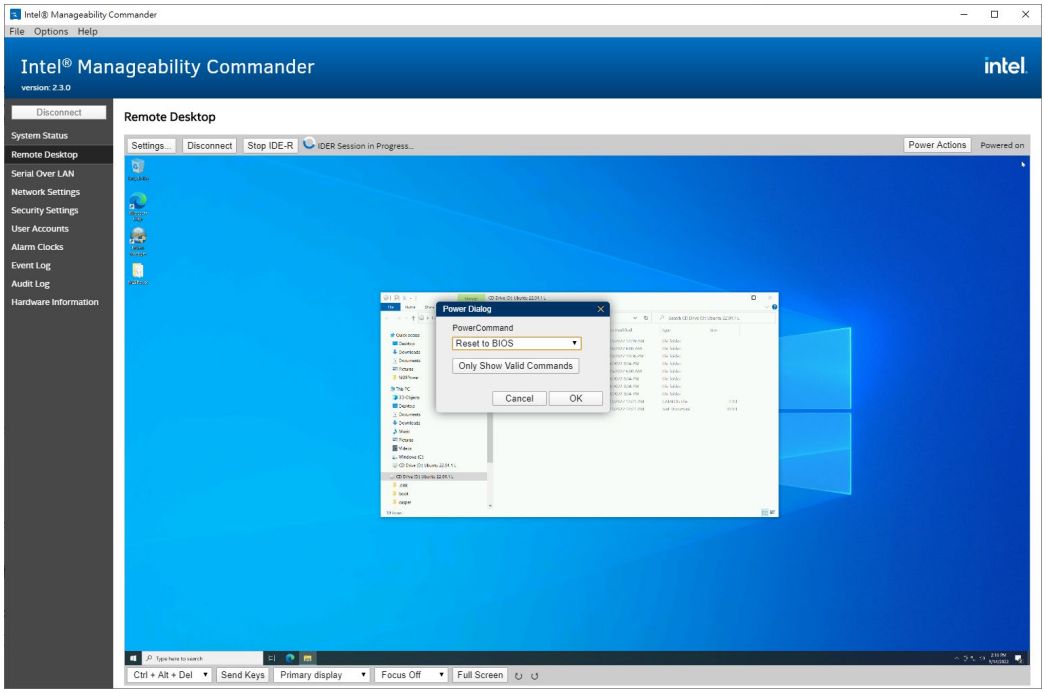
A recovery image can be remotely mounted on to a device.

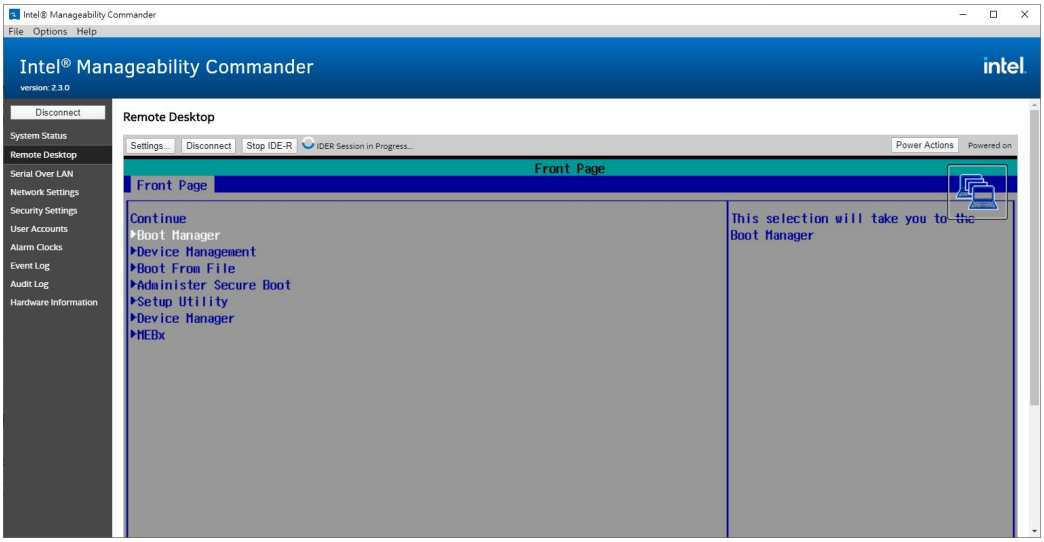
Do the following, to remotely install a recovery image on a device.

1. Click IDE-R option and select the target image.

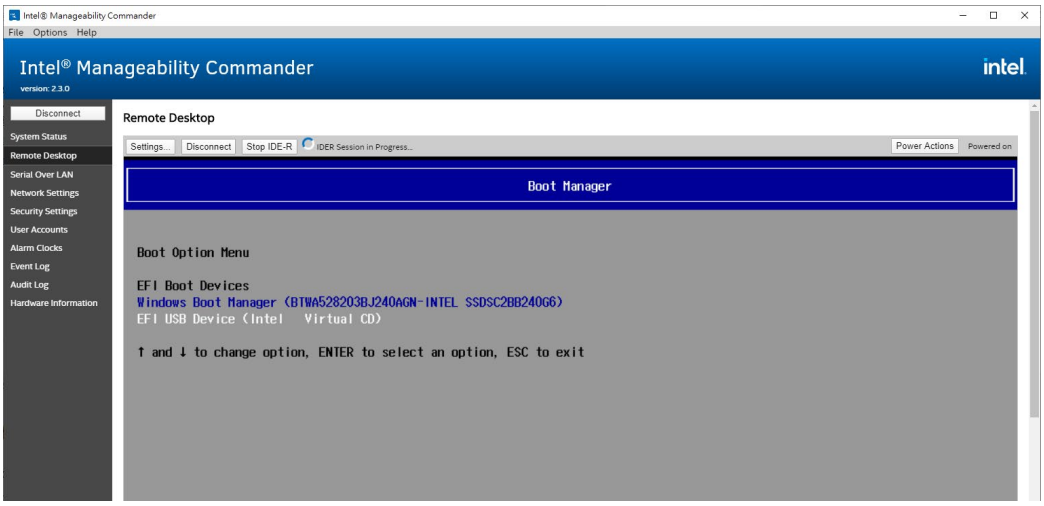



2. Reset to BIOS menu and select Boot Manager.

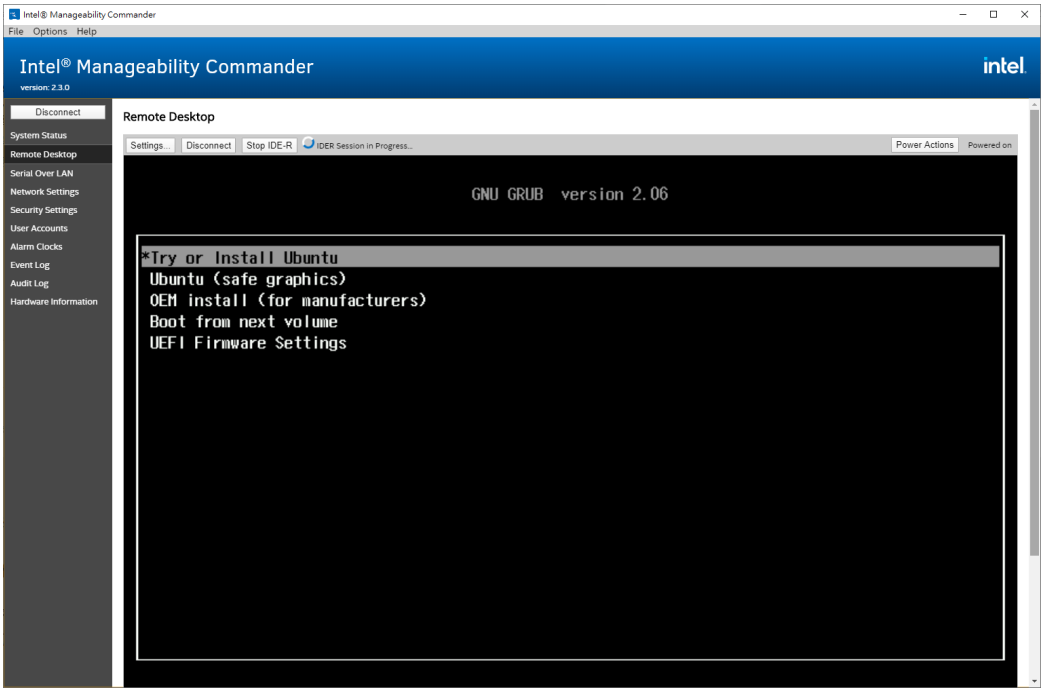




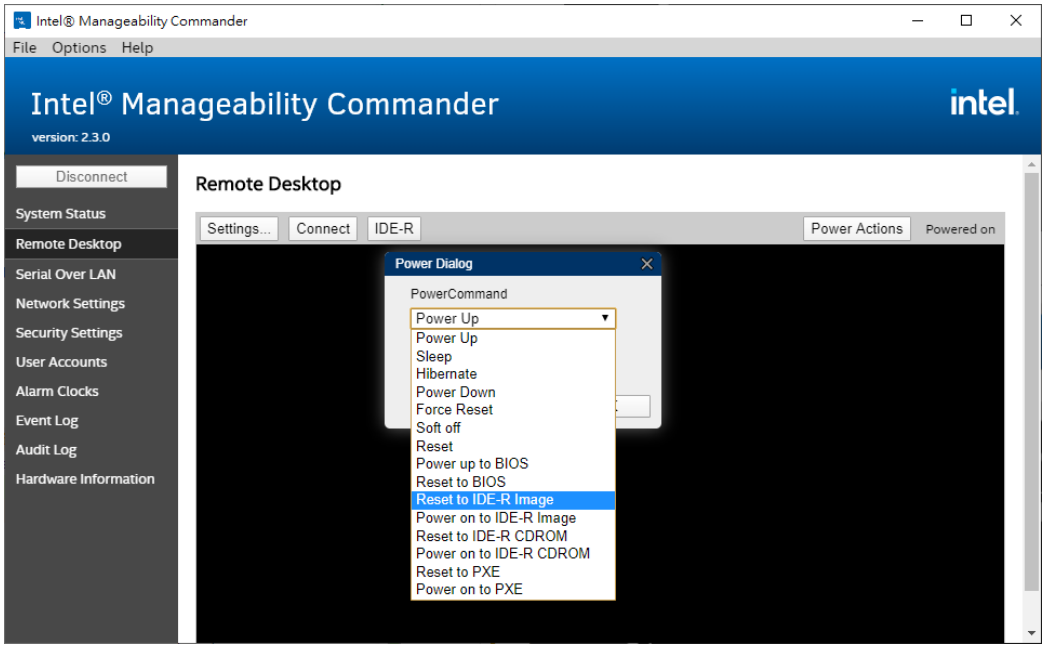

3. Select Intel Virtual CD to boot to mount image.




4. Following the steps to recovery image.

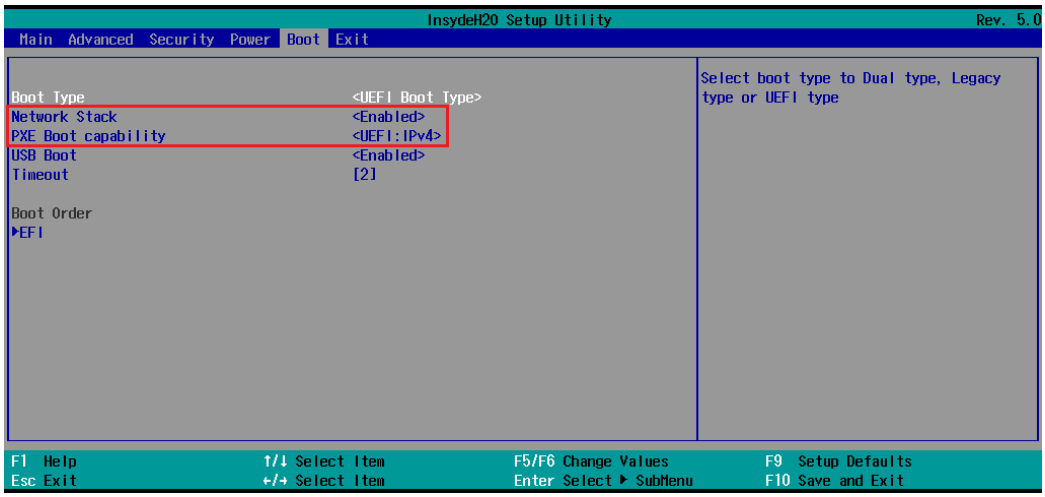


> **NOTE**
>
> Users can reset to IDE-R image directly using the power action.

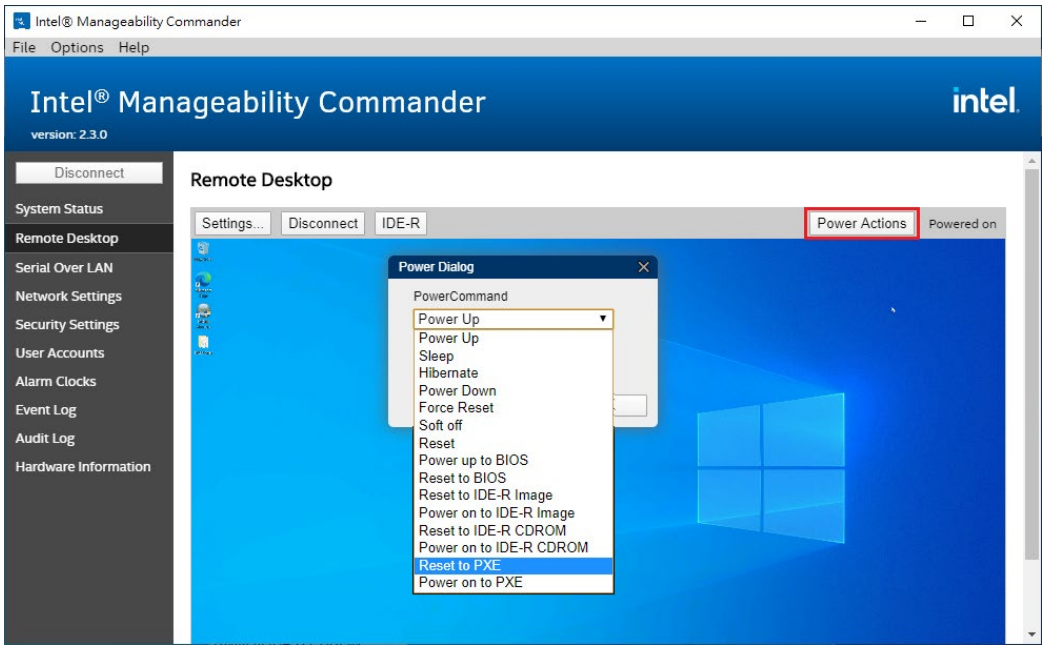



# Reset to PXE

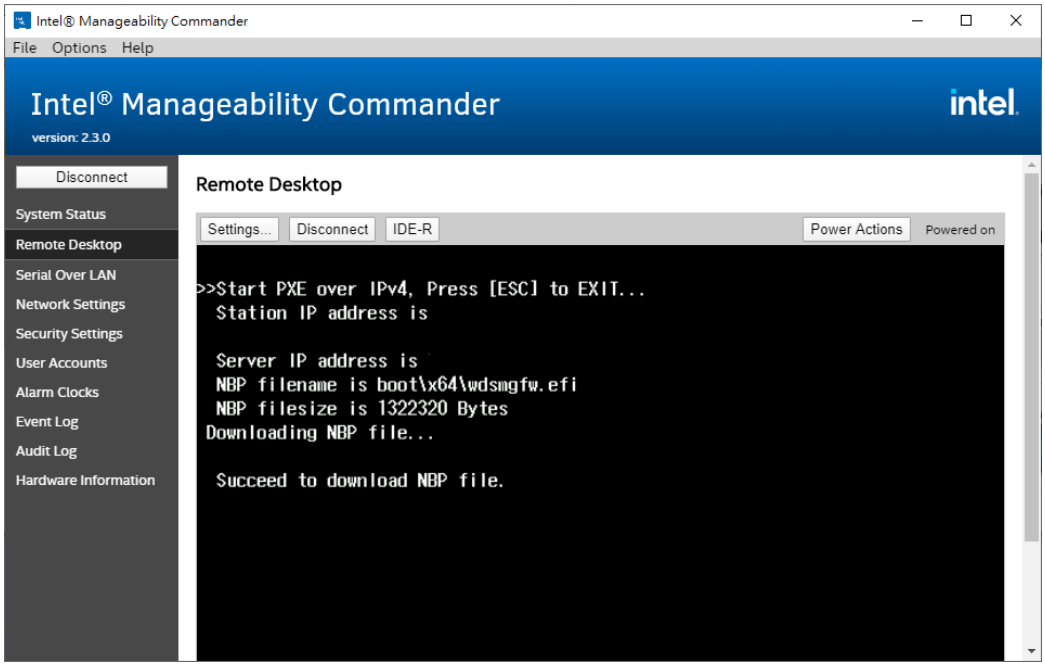
1. Enable the Network Stack and PXE Boot capability from BIOS menu of managed device.



2. Click Power Actions and select Reset to PXE.




3. The managed device will boot to PXE.



# 7. Unified Write Filter

Unified Write Filter (UWF) is an optional feature that helps to protect your drives by intercepting and redirecting any writes to the drive (app installations, settings changes, saved data) to a virtual overlay. The virtual overlay is a temporary location that is usually cleared during a reboot or when a guest user logs off.

UWF provides a clean experience for thin clients and workspaces that have frequent guests, like schools, library, or hotel computers. Guests can work, change settings, and install software. After the device reboots, the next guest receives a clean experience. It increases security and reliability for kiosks, IoT-embedded devices, and other devices where new apps are not expected to be frequently added.

This chapter describes how to use the Unified the Write Filter (UWF).

To use the UWF, you must first install the feature and enable it; the default is disable.

The first time you enable UWF on your device, UWF makes the following changes to your system to improve its performance:

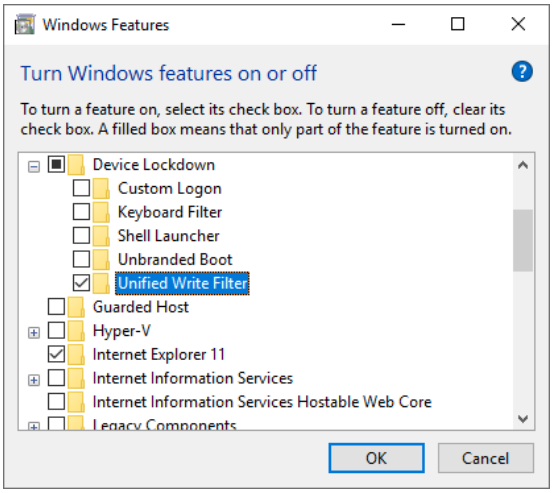
- **Paging files are disabled.**
- **System restore is disabled.**
- **SuperFetch is disabled.**
- **File indexing service is turned off.**
- **Fast boot is disabled.**
- **Defragmentation service is turned off.**
- **BCD setting bootstatuspolicy is set to ignoreallfailures.**

After UWF is enabled, you can select a drive that you want to protect and start using UWF. UWF can help you manage PCs and devices remotely using WMI.

# Turning on UWF on a Running PC

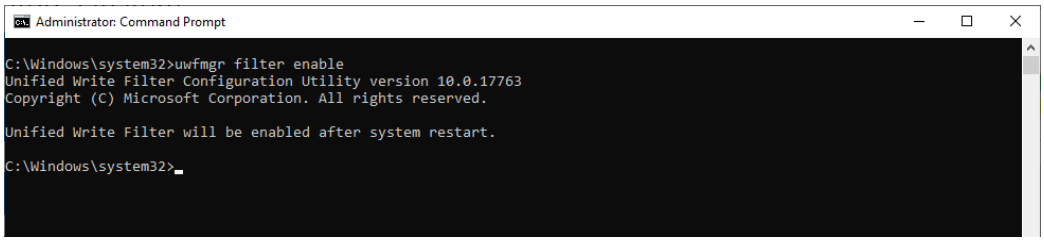
- Install UWF.
  a. In the Windows **Start** window, type **Turn Windows features on or off**.
  b. Open the **Windows Features** window and expand the **Device Lockdown** node.
  c. Select **Unified Write Filter** and click **OK**.
  d. Windows searches for the required files and displays a progress bar.

     Once the files are found, Windows applies the changes. When the changes are complete, a message to this effect is displayed.
  e. Click **Close**.



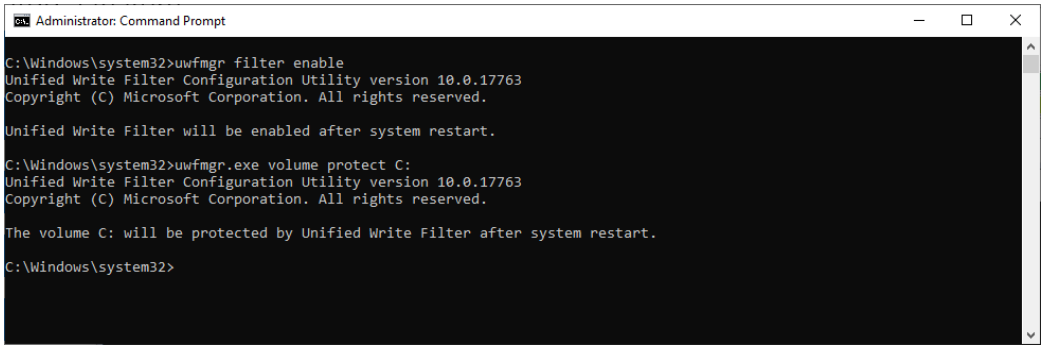
- Enable the following filter as an Administrator:

  **cmd uwfmgr filter enable**



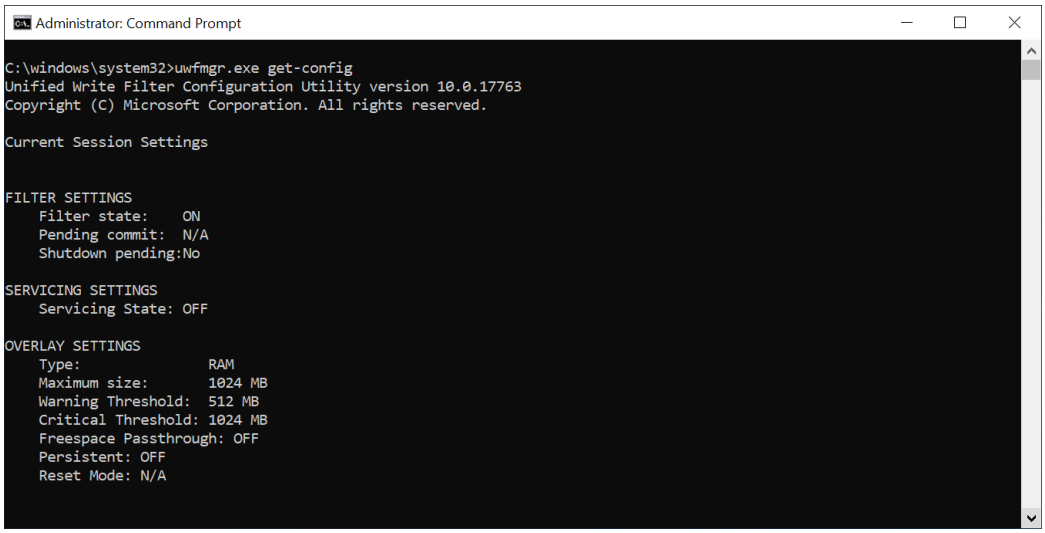

- Enable write protection for a drive:

  **cmd uwfmgr.exe volume protect C:**




- Restart your computer.
- Confirm that UWF is running:

  **cmd uwfmgr.exe get-config**



# Installing UWF Using WMI

If you have already installed Windows on your computer and you do not want to use a provisioning package, you can configure UWF by using Windows Management Instrumentation (WMI) providers.

To turn on UWF using WMI, use the **UWF_Filter** function, specifically the **UWF_Filter.Enable** method in one of the following ways:

- Use the WMI providers directly in a PowerShell script
- Use the WMI providers directly in an application
- Use the command line tool, uwfmgr.exe

## NOTE

You must restart your computer after you turn on or turn off UWF for the changes to take effect.

You can also change the settings after you turn on UWF. For example, you can move the page file location to an unprotected volume and re-enable paging files.

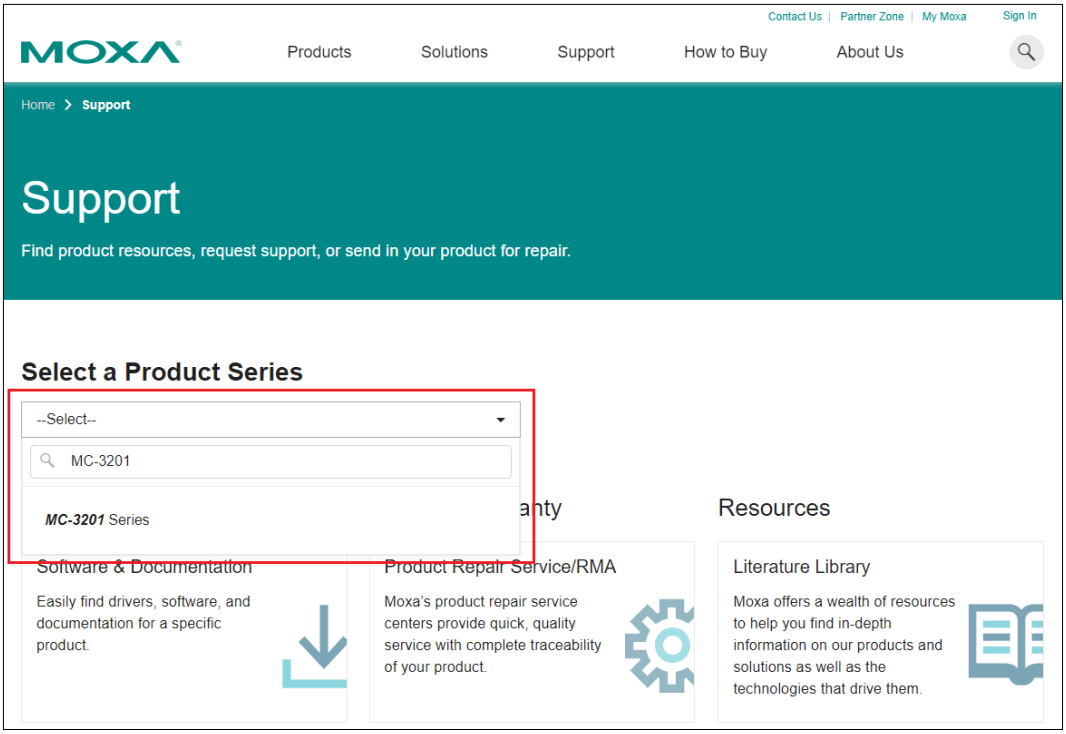
## IMPORTANT!

If you add UWF to your image by using SMI settings in the unattend.xml file, turning on UWF only sets the bootstatuspolicy BCD setting and turns off the defragmentation service. You must manually turn off the other features and services if you want to increase the performance of UWF.

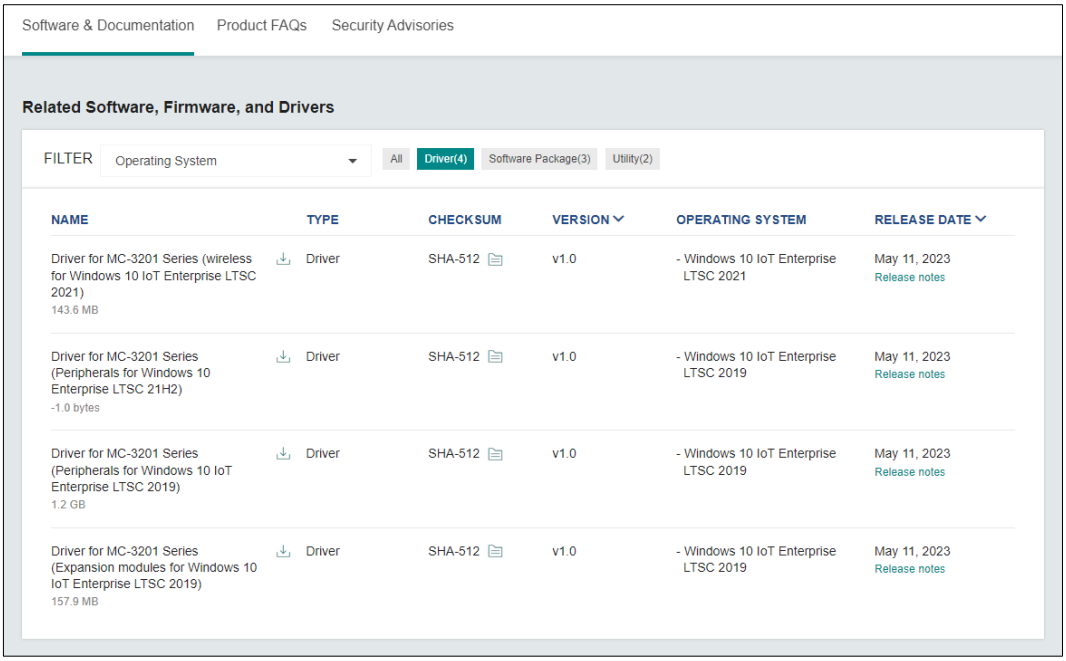
After the device is restarted, UWF maintains configuration settings for the current session in a registry. UWF automatically excludes these registry entries from its filter. Static configuration changes do not take effect until after a device restarts; the changes are saved in registry entries for use in the next session. Dynamic configuration changes occur immediately and persist after a device restarts.

# 8. Driver

Moxa provide verified drivers for each device on official website. Please access the Moxa support page(https://www.moxa.com/en/support) and search for the device from the searching window (For Example: MC-3201).




Form the **Software & Documentation** page filtered by **Driver** and download the driver package. The driver packages are categorized by OS version, with separate sections for **Peripheral** and **Expansion modules**.



# 9. Utility

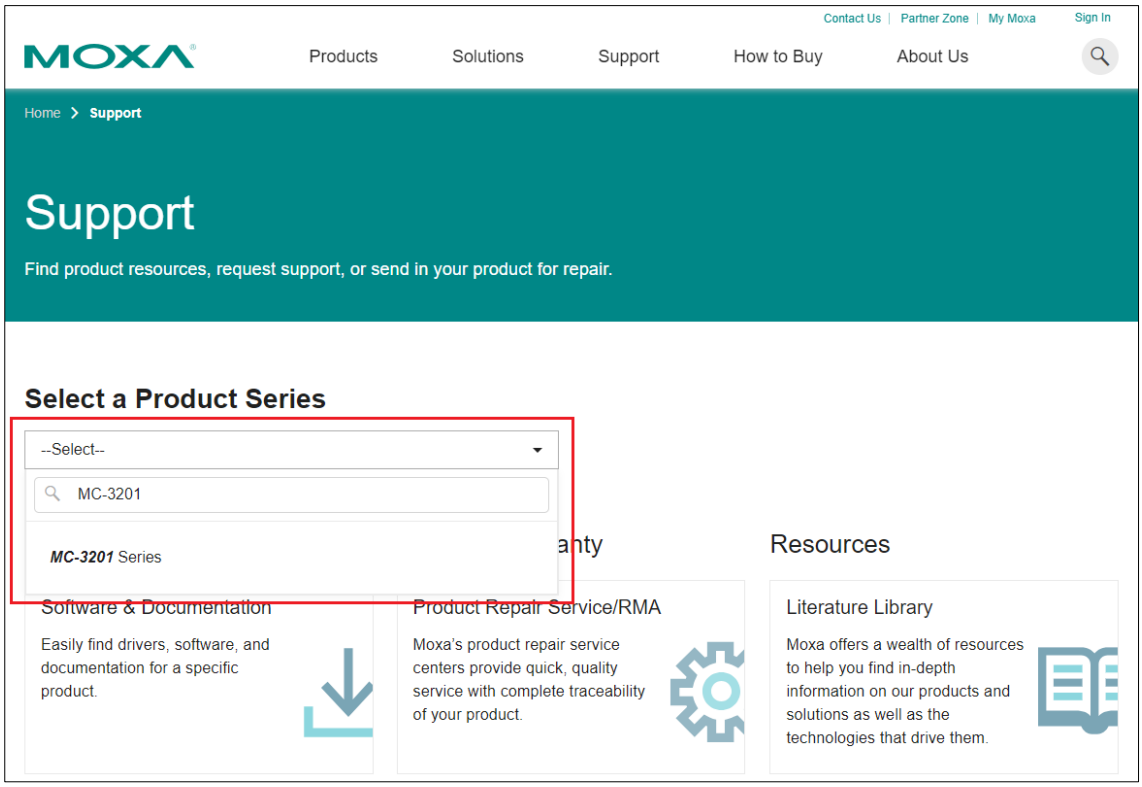
This chapter describes the usage of the following:

- Moxa IO Controller
- Serial Interface
- Moxa Sort Net Name
- Moxa Disk Hot Swap Application
- Moxa LANBypass Utility
- Moxa MCU Firmware Upgrade
- Moxa Tca9535 Initial Service
- Moxa Cellular Assistant
- Moxa WiFi AP Service
- Moxa Battery Detect

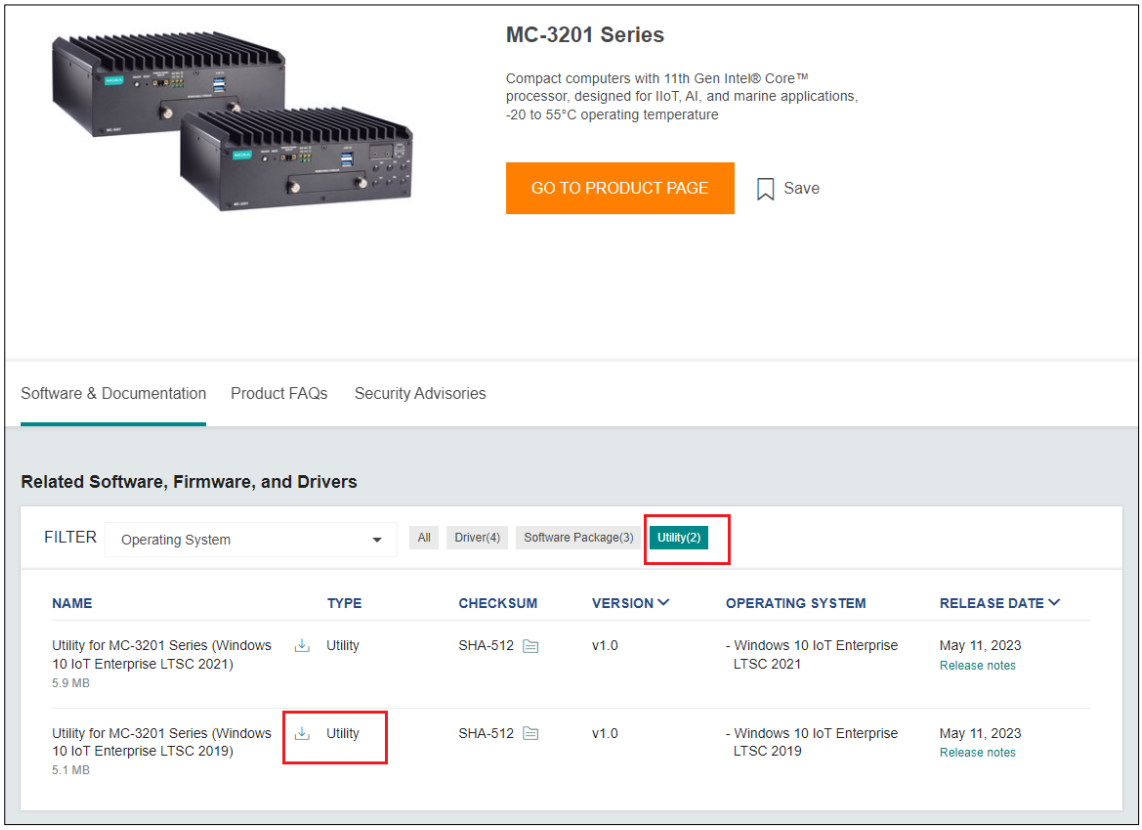
## Where to Find the Utility

The utilities are preinstalled on your device if the Windows 11 OS is provided by Moxa. If you install Windows 11 on your own, go to the Moxa support page (https://www.moxa.com/en/support) to download the utility.

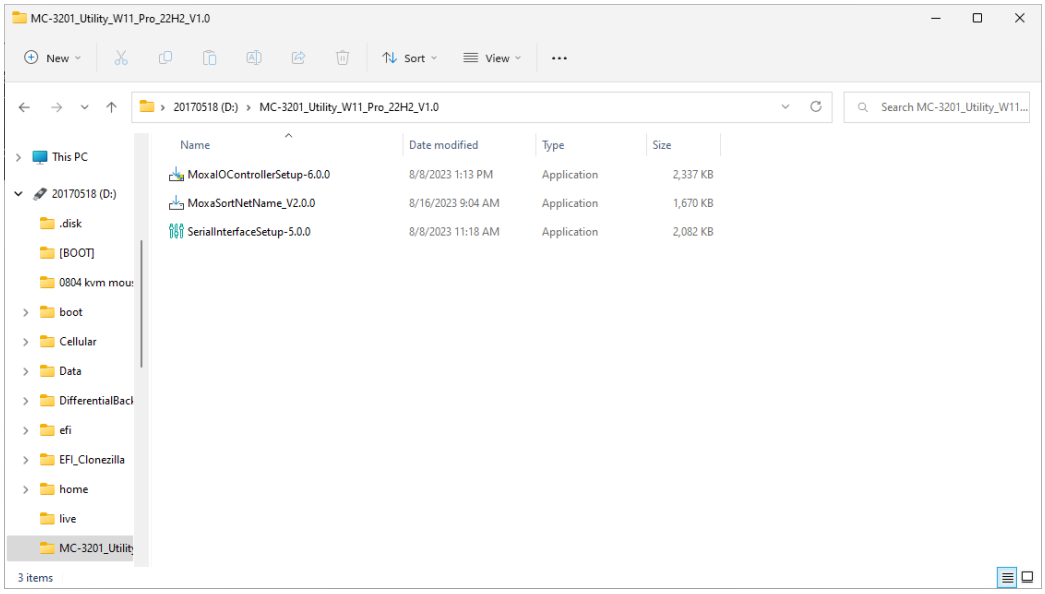
1. Search for the device from the searching window (example: MC-3201).



2. Go to the **Software & Documentation** page, filter by **Utility** and download the required file. The installation file for the device is in a *.zip file.




3. After extracting the files, you will obtain the following setup files.



# Dependent Packages

- After the installation of Windows 11 LTSC 24H2 and drivers are completed, you will need to install **dependent packages** to ensure the smooth operation of the utility. Please attach the following link to download and install the packages.
- Microsoft Visual C++ Redistributable:

  https://learn.microsoft.com/en-us/cpp/windows/latest-supported-vc-redist?view=msvc-170
- Microsoft .NET Framework 4.8:

  https://support.microsoft.com/en-us/topic/microsoft-net-framework-4-8-offline-installer-for-windows-9d23f658-3b97-68ab-d013-aa3c3e7495e0

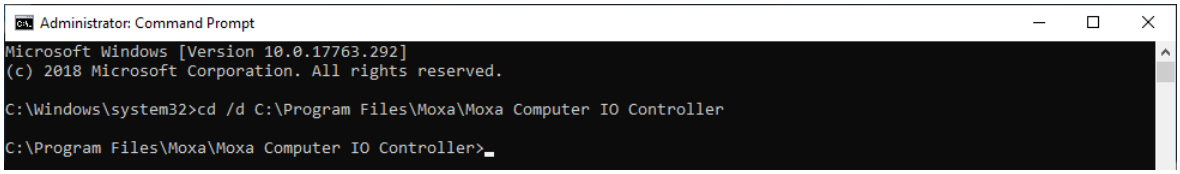
# Moxa IO Controller

Moxa IO Controller Utility is developed to control the peripherals' IO as well as expansion modules interface of the device.

This section describes how to use the Moxa IO Controller utility and the following content will be included.

- Setting the DIO Status
- Setting the UART Mode
- Setting the LED Status
- Setting the PCIE Slot Power Status
- Setting the PCIE Reset Pin Status
- Setting the M.2 B Key Socket 5G Module Power Status

Use the pre-installed utility or install the MoxaIOControllerSetup utility from the Moxa support page. To use the Moxa IO Controller utility, first install the utility and enable the utility to configure the DIO, UART, Relay and LED mode. After the installation process is complete, run the Windows command prompt as an Administrator and change the path to C:\Program Files\Moxa\Moxa IO Controller.

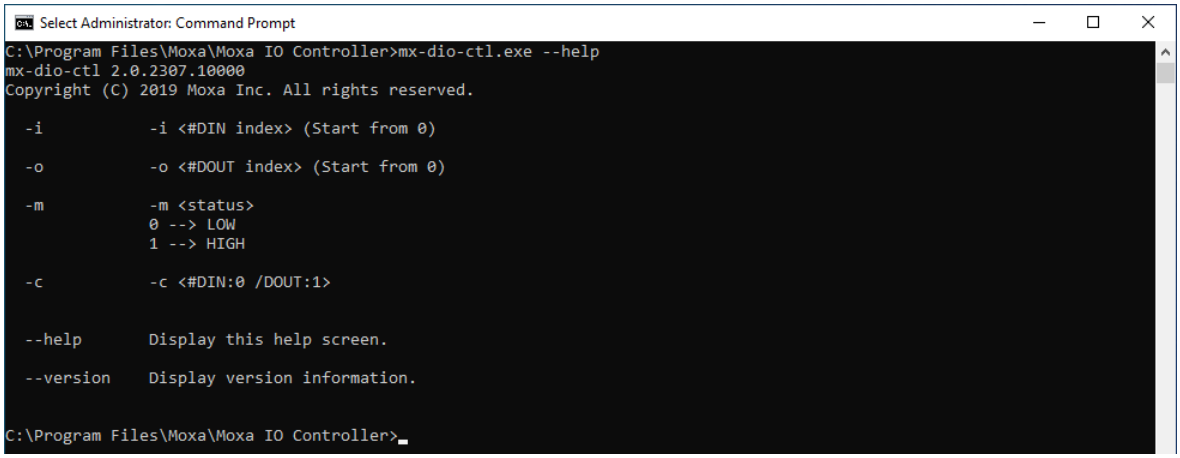


# Setting the DIO Status

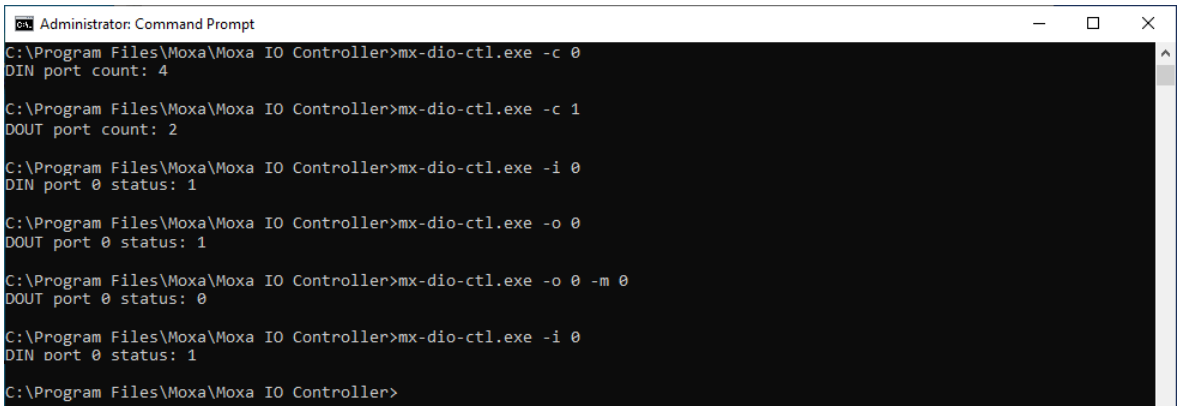
Run the mx-dio-ctl --help command to see the instructions on using this utility and follow them to get or set the DIO status.

## IMPORTANT!

The DIN and DOUT indices start at 0. Even though the console output starts at 1, the indices still start at 0.




### Example:

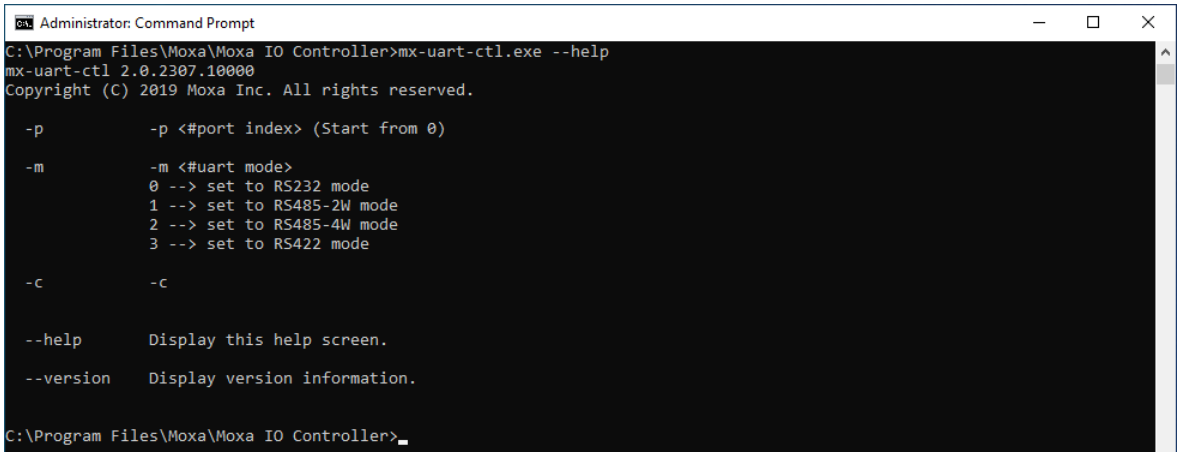


# Setting the UART Mode

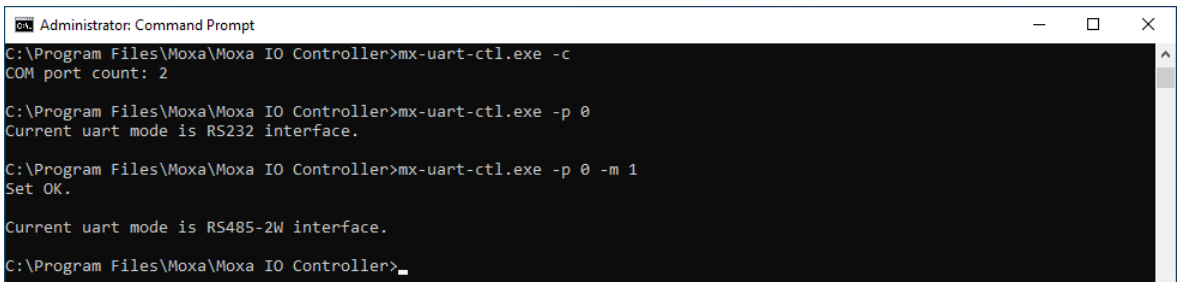
Run the mx-uart-ctl --help command to see instructions on using this utility and follow the onscreen instructions to get or set the UART mode.

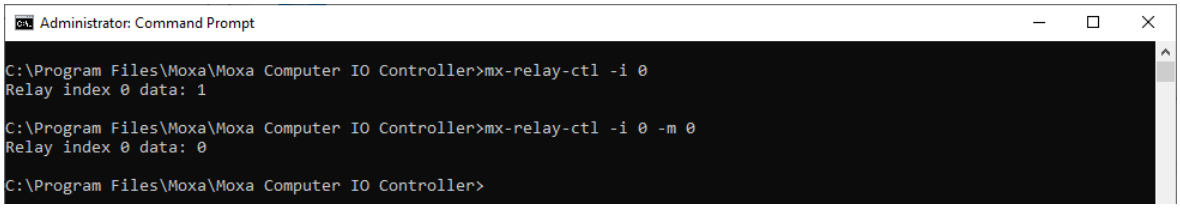
## IMPORTANT!

The UART index starts from 0. Even though the console output starts at 1, the index still starts at 0.




### Example:



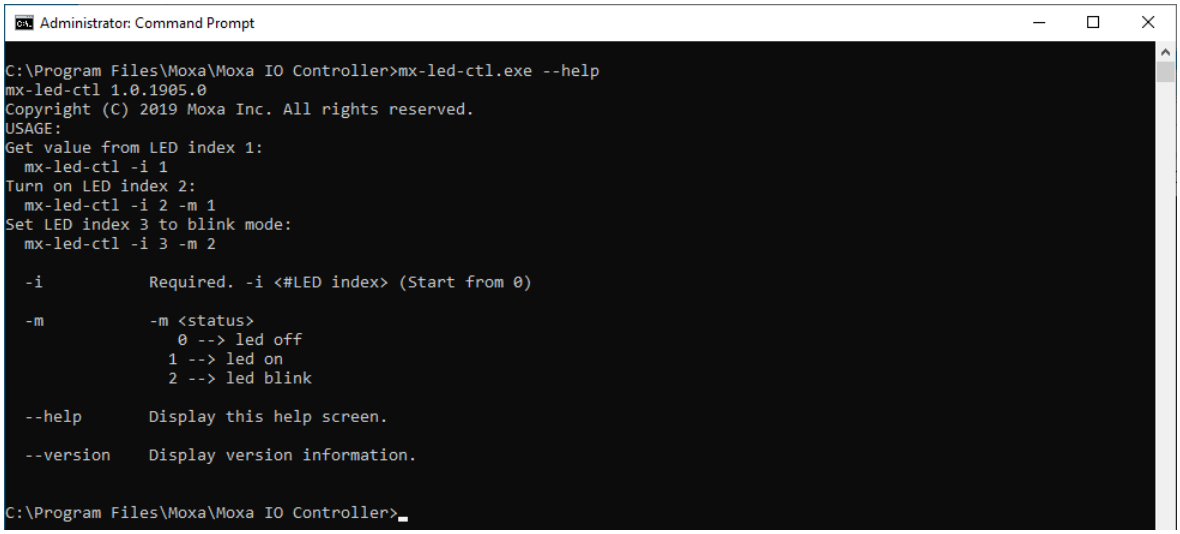



# Setting the LED Status

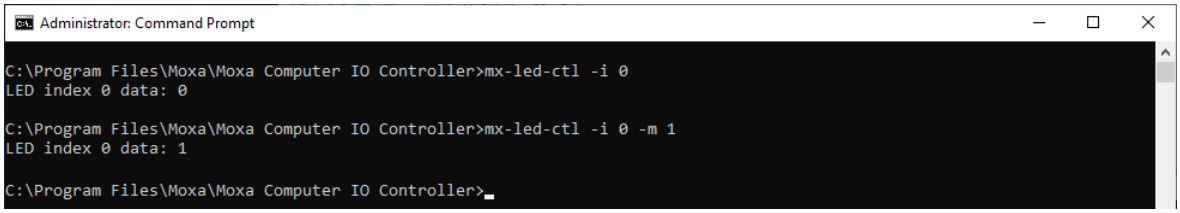
Run the mx-led-ctl --help command to see instructions on using this utility and follow the onscreen instructions to get or set the LED status.

## IMPORTANT!

The LED index starts from 0. Even though the console output starts at 1, the index still starts at 0.




**Example:**

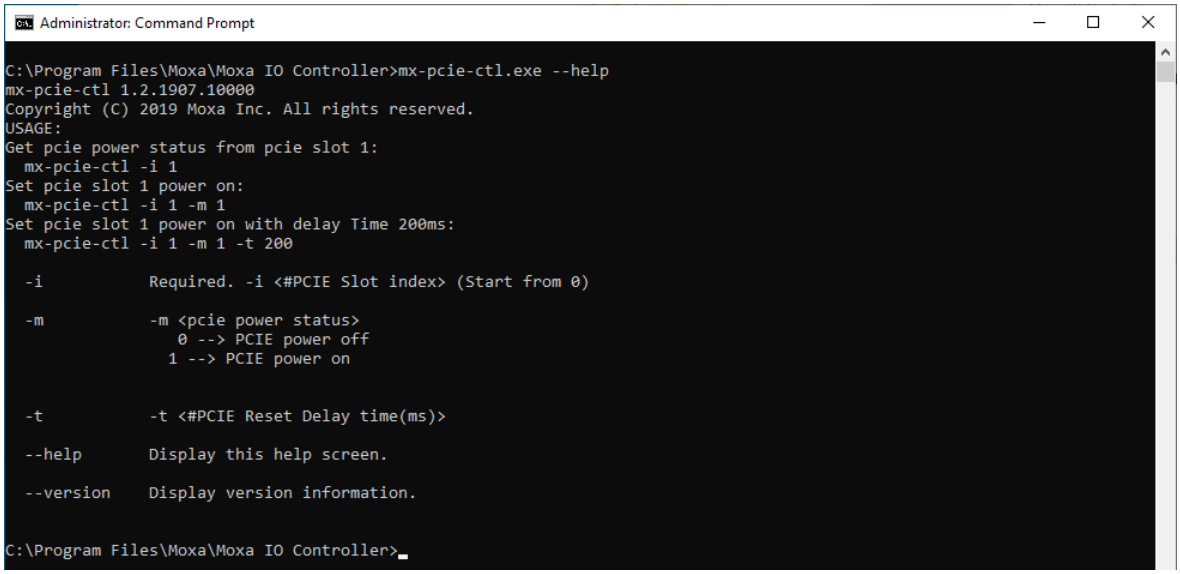


# Setting the PCIE Slot Power Status

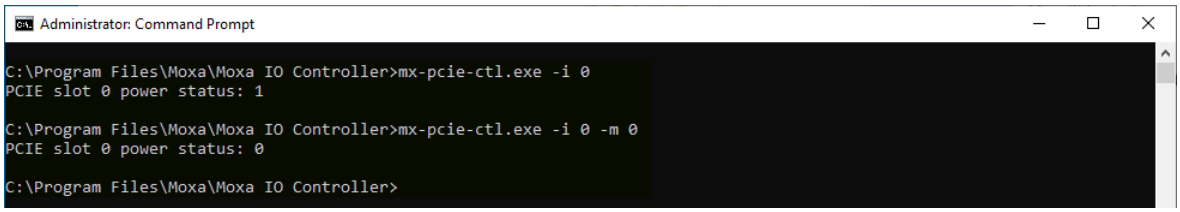
Run the mx-pcie-ctl --help command to see instructions on using this utility and follow the onscreen instructions to get or set the status of the PCIE slot power.

## IMPORTANT!

The PCIE Slot index starts from 0. Even though the console output starts at 1, the index still starts at 0.




**Example:**

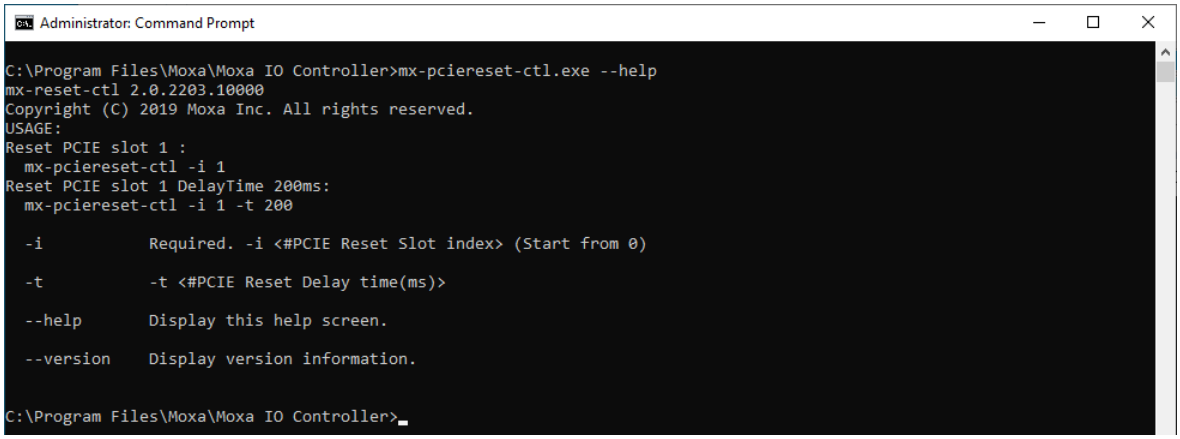


# Setting the PCIE Reset Pin Status

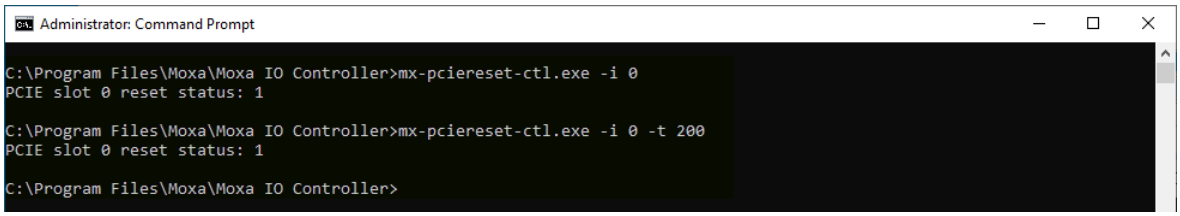
Run the mx-pciereset-ctl --help command to see instructions on using this utility and follow the onscreen instructions to get or set the PCIE reset pin status and delay time .

**IMPORTANT!**

The PCIE reset pin index starts from 0. Even though the console output starts at 1, the index still starts at 0.

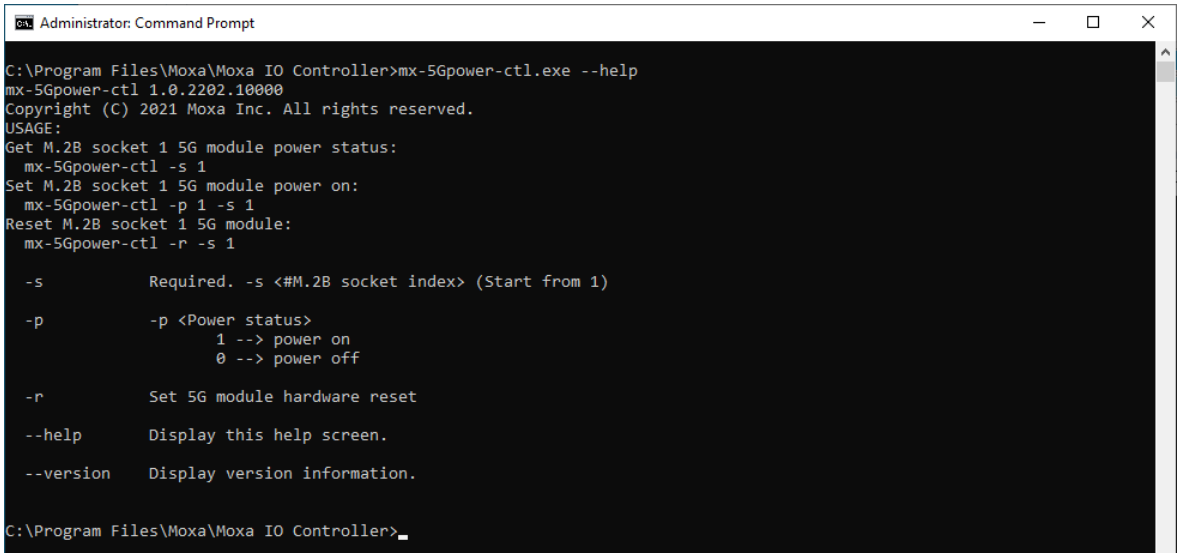



**Example:**

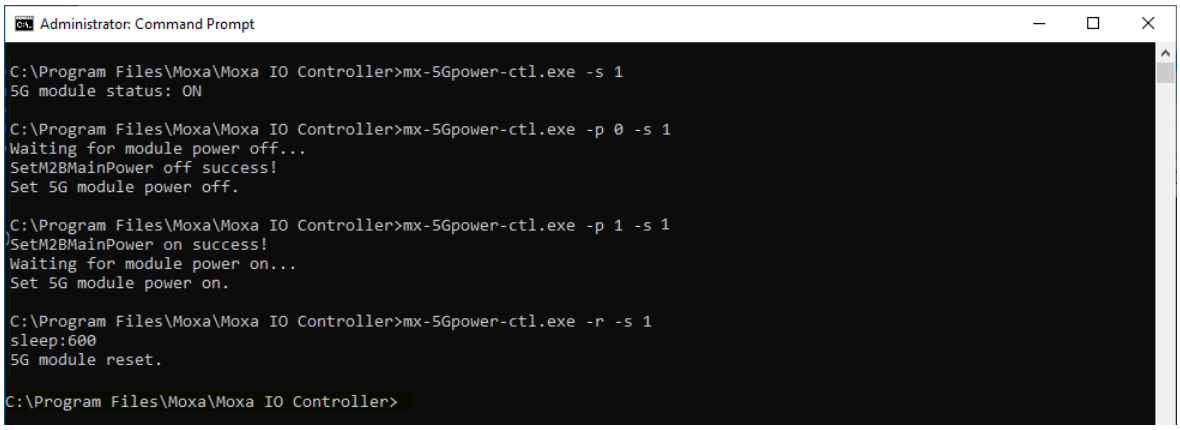



# Setting the M.2 B Key Socket 5G Module Power Status

Run the mx-5Gpower-ctl --help command to see instructions on using this utility and follow the onscreen instructions to get or set the status of the M.2 B Key socket power.



**Example:**

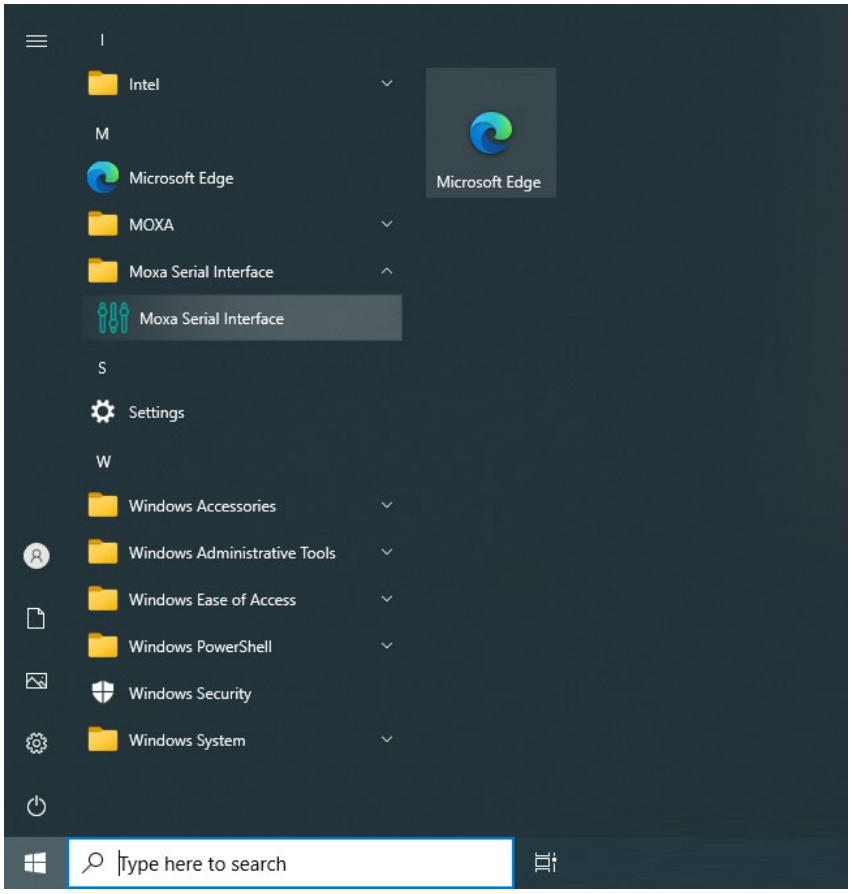



# Moxa Serial Interface Utility

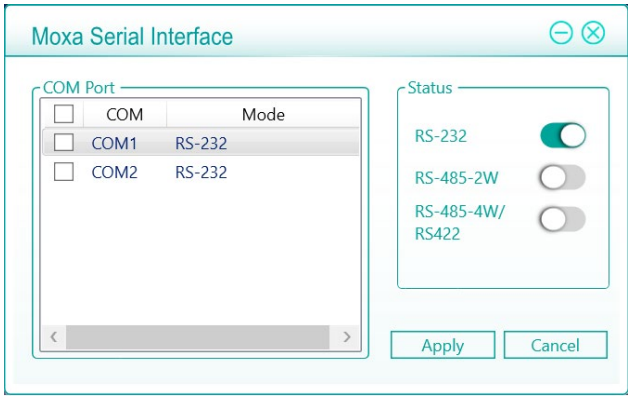
In this section, we describe how to use the Moxa Serial Interface utility to set the UART mode in your computer's serial interface.

## Setting the Serial Port Mode

1. Install the Moxa Serial Interface utility.
2. From the Windows Start menu, run the Moxa Serial Interface utility.



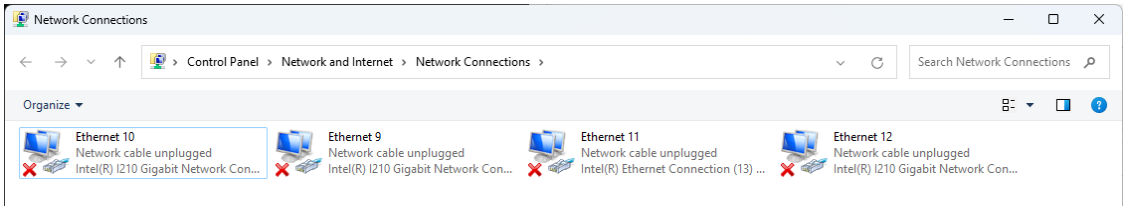
3. Select the target COM port and UART mode and click Apply to save the settings.



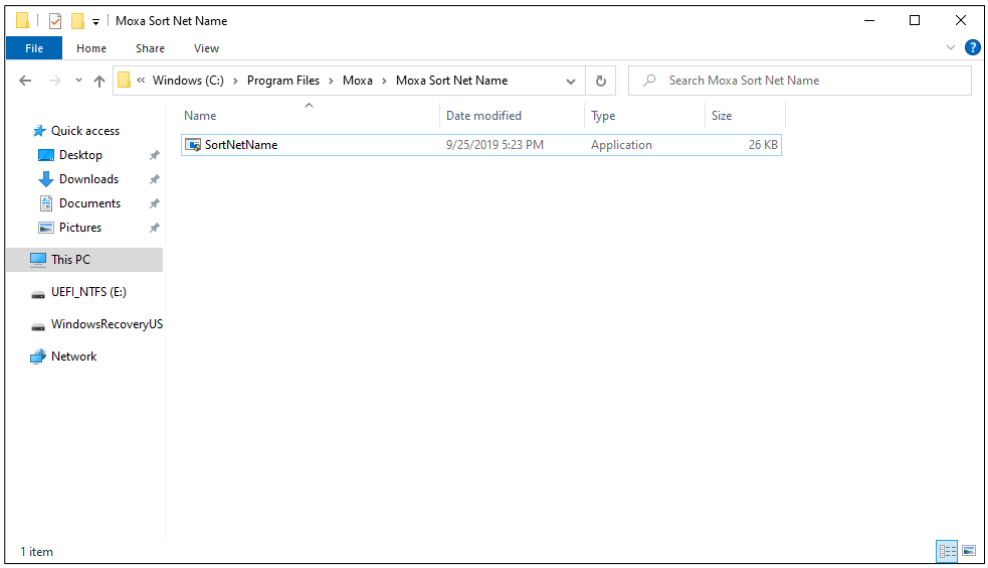

# Moxa Sort Net Name

In this section, we describe how to use the Moxa Sort Net Name utility to rename Ethernet adapter for mapping physical LAN port order on chassis.

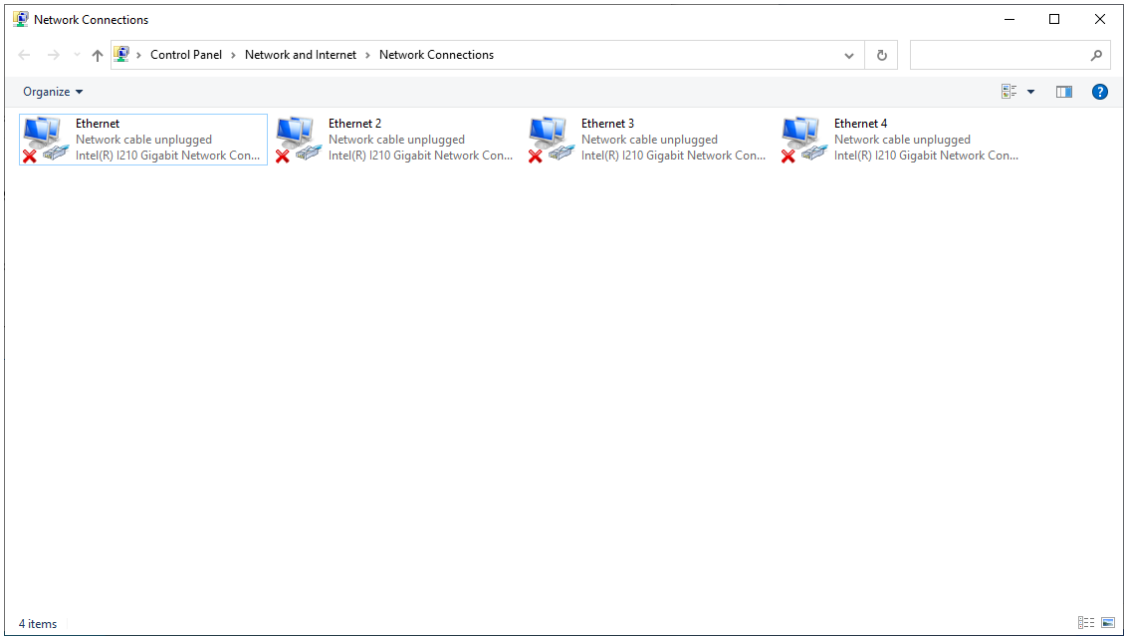
1. Use the pre-installed utility or install the MoxaSortNetName utility from the Moxa support page.
2. The initial order of network names may be random.




3. After the installation process is complete, run the SortNetName.exe from C:\Program Files\Moxa\Moxa Sort Net Name as an Administrator.



4. If you want to rename the Ethernet adapter, wait for the installation process to complete. The order of the Ethernet adapter will correspond to the order of the label (e.g., LAN 2 of the computer is mapped to Ethernet 2 in Windows).




# Moxa Disk Hot Swap Application

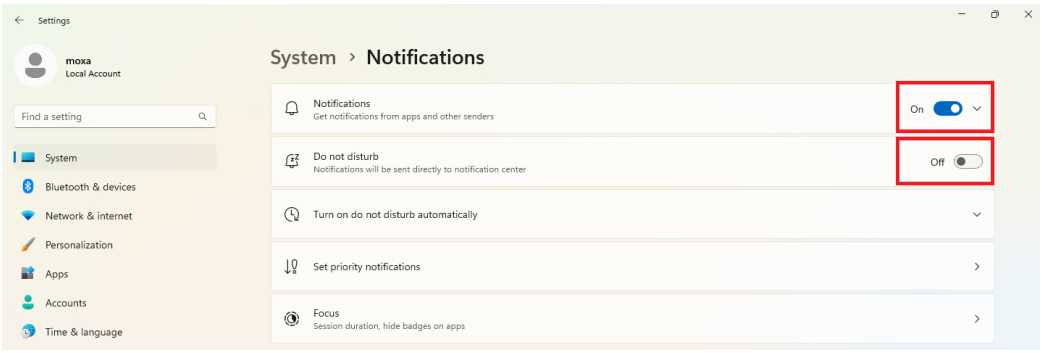
This utility allows users to configure hot-plug functionality for storage devices, enabling safe insertion and removal of disks while the system is powered on and operational.

The following topics are covered in this chapter:

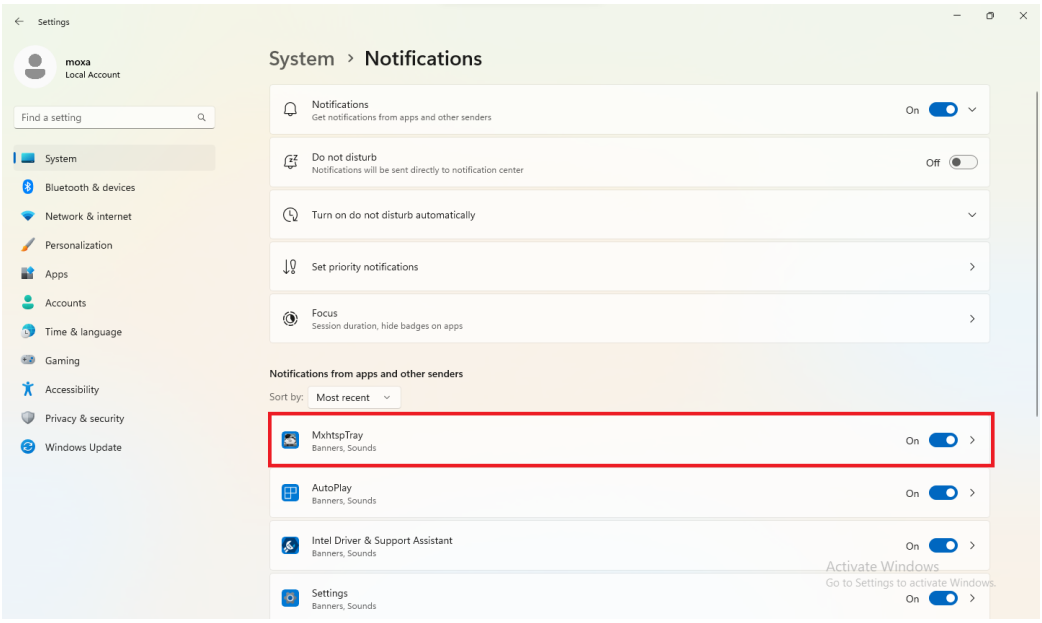
1. Setting Up the Notification
2. Setting Up the Disk Removal Policy
3. Setting Up the Drive Actions
   - Default Actions
4. Unmounting and Mounting Disks
   - Unmounting a Disk Drive
   - Mounting a Disk Drive
5. Hot Swap Function Notifications

# Setting Up the Notification

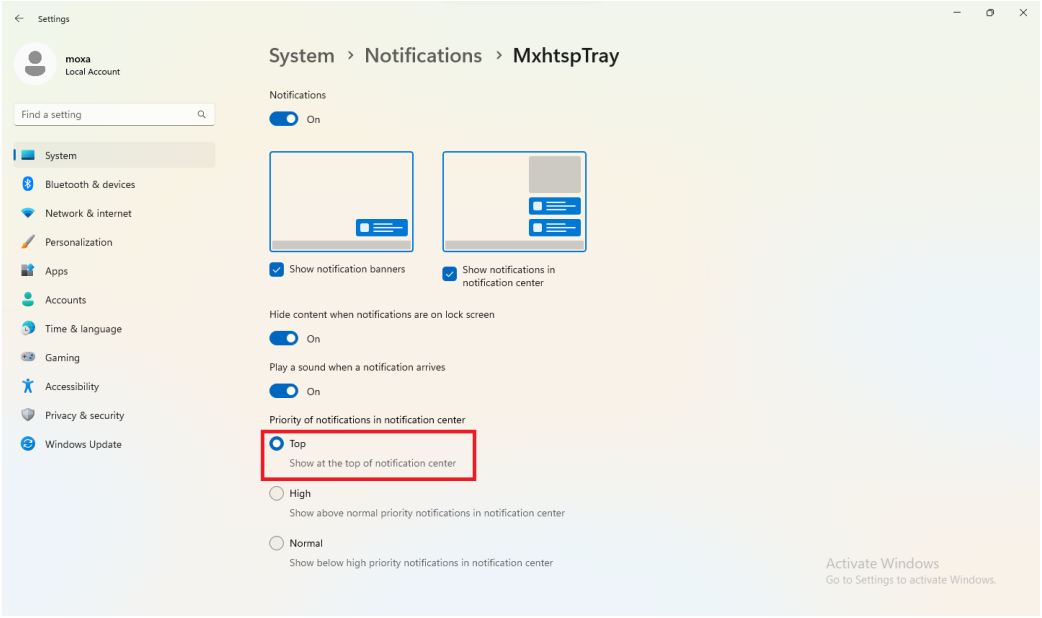
1. Enter System Settings from Windows Settings and click **Notification**.
2. Confirm to **enable** Notifications setting and **disable** Do not disturb setting.




3. Select **MxhtspTray** to enter the setting page.

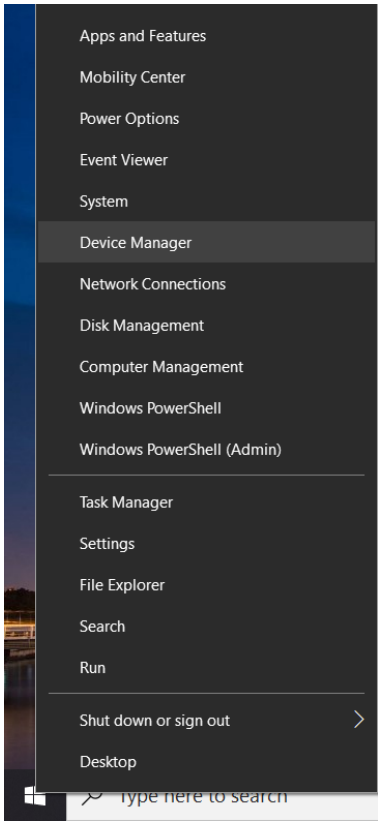



4. Set priority of notifications in notification center to **Top**.

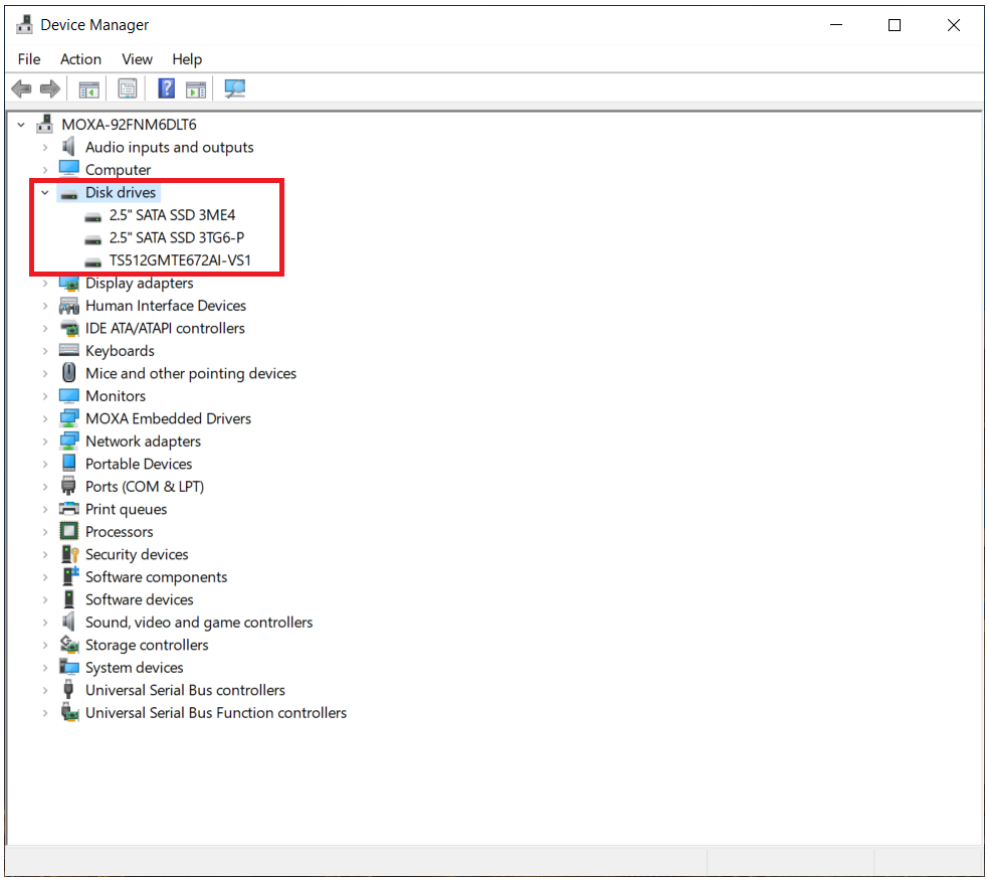


# Setting Up the Disk Removal Policy

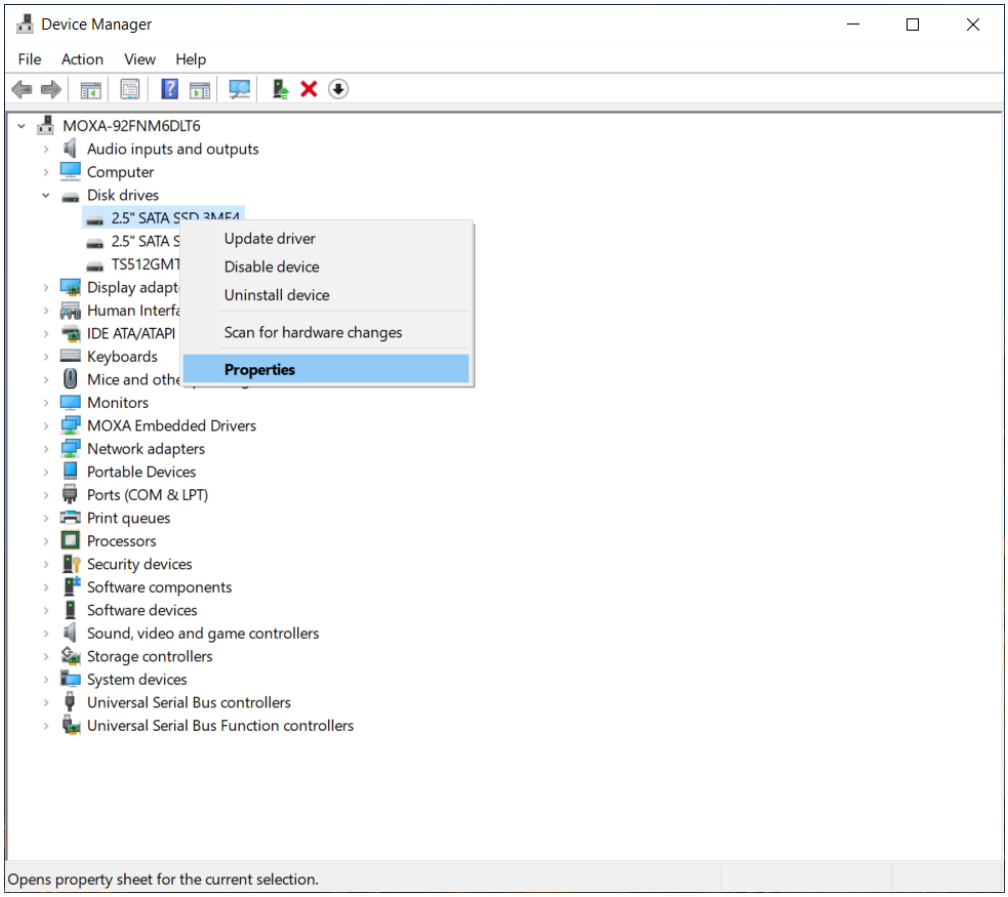
1. Right-click on the Windows Start icon and select Device Manager.



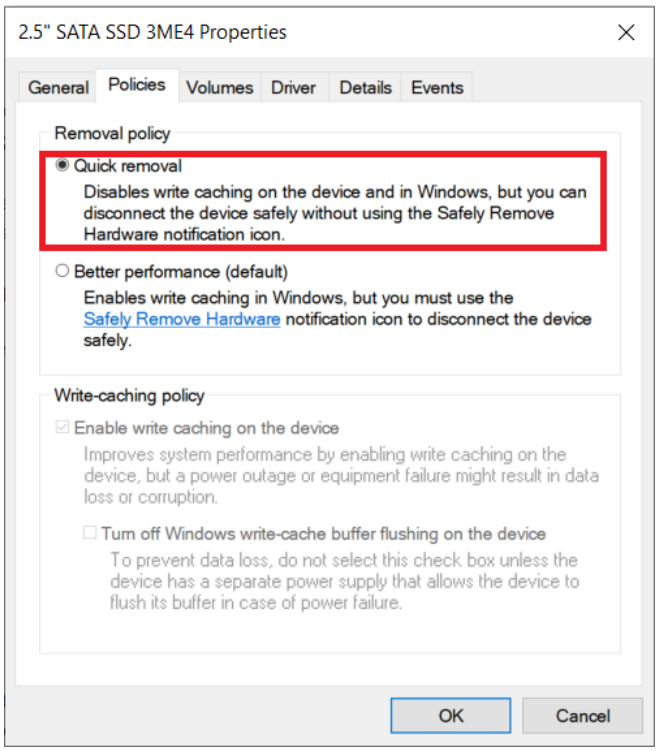

2. Expand the Disk drives node.



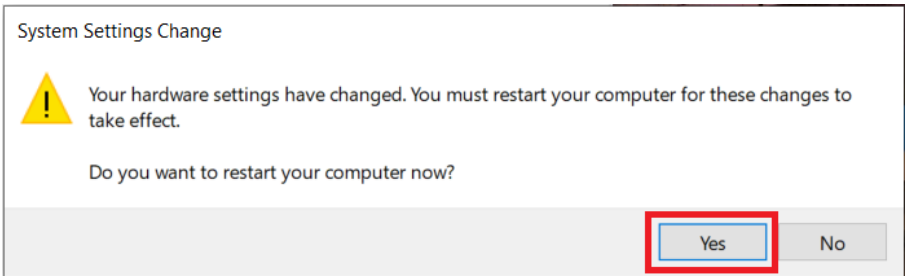
3. Right-click on the target drive and select Properties.




4. In the Policies tab, select Quick removal and click OK.

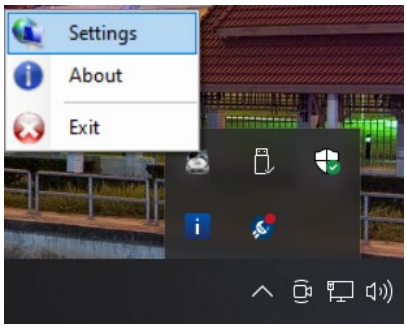


5. Click Yes to restart the computer and apply the new settings.

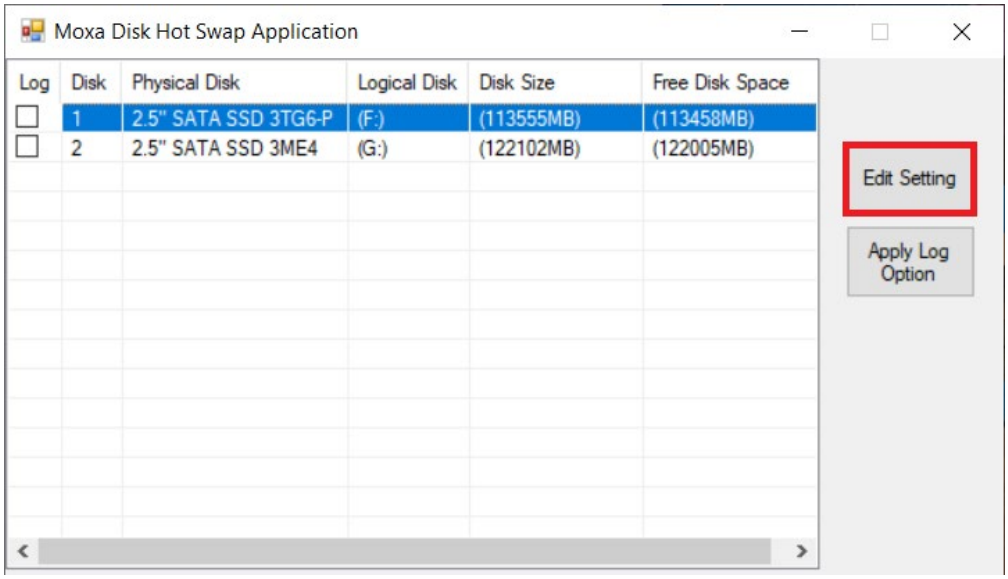



# Setting Up the Drive Actions

1. Right-click on the Disk Hot Swap application icon on the Windows taskbar and select Settings.

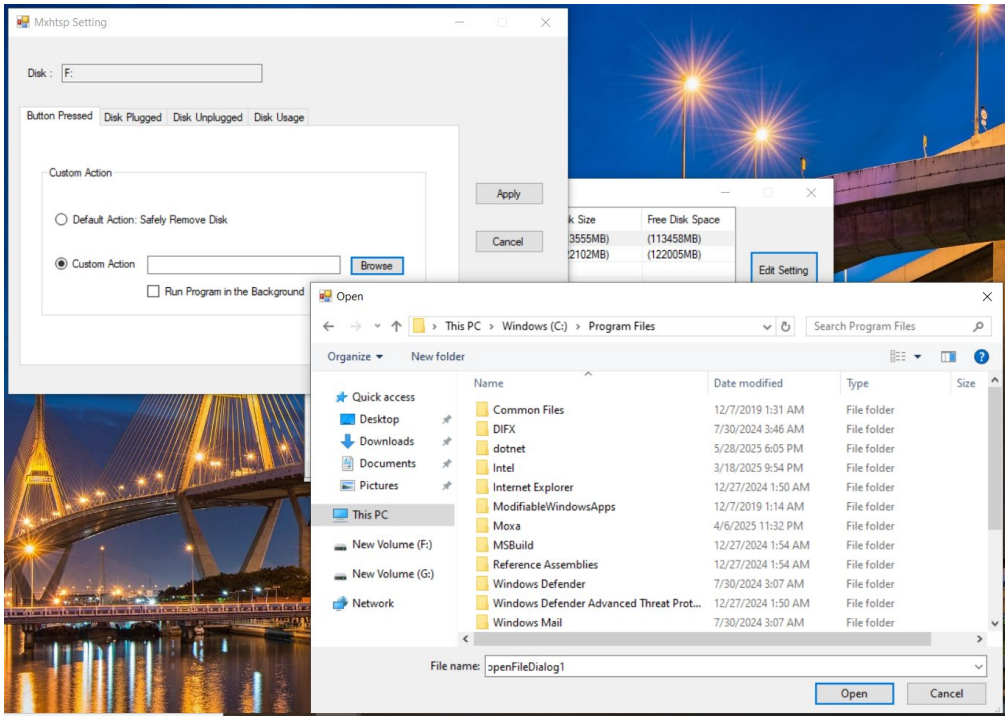



2. Select the target drive and click Edit Setting.




3. Open an event tab (Button Pressed, Disk Plugged, Disk Unplugged, or Disk Usage) and select Custom Action.

4. Associate the application that you want to run as the custom action when the event is triggered. The application will run in the background. The new settings will take effect the next time you run the Disk Hot Swap program.




**IMPORTANT!**

The action for the Button Pressed event is triggered only when the button is pressed for more than 3 seconds.

# Default Actions

If you do not configure a Custom Action, the following default actions are triggered by the events.

1. Button Pressed: Disk unmount procedure is triggered when the button is pressed for over 3 seconds.
2. Disk Plugged: The disk drive list in the Windows Device Manager is refreshed after a few seconds.
3. Disk Unplugged: The disk drive list in the Windows Device Manager is refreshed after a few seconds.
4. Disk Usage: A notification on the disk usage is displayed.

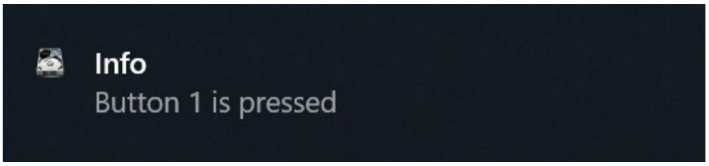
   This event is triggered if on Check Disk Usage, the disk usage is equal to or higher than the disk usage threshold.

# Unmounting and Mounting Disks

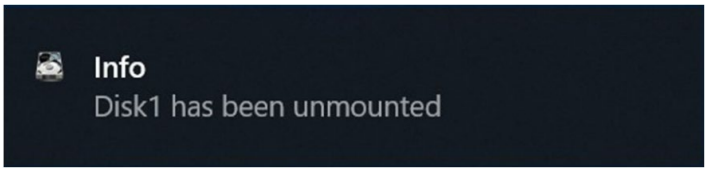
## Unmounting a Disk Drive

1. Press the button of the target disk for more than three seconds.

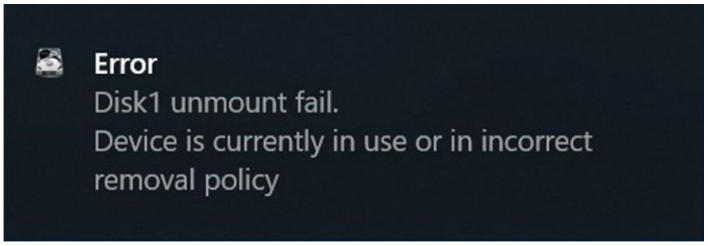
   The disk unmount procedure will start after the Button is pressed notification is displayed.



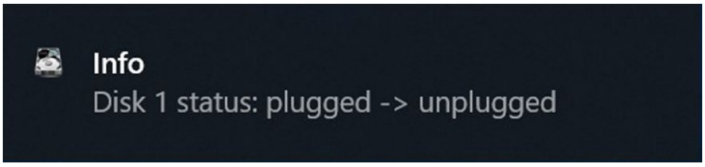
2. The Disk has been unmounted notification is displayed after the disk has been successfully unmounted.




If the disk unmount process fails, you will see the following message. Ensure that the target disk removal policy is set to Quick removal and the target disk is not busy before triggering the unmount process again.




3. After the disk has been unmounted, unplug the disk from your computer after you see the following notification.

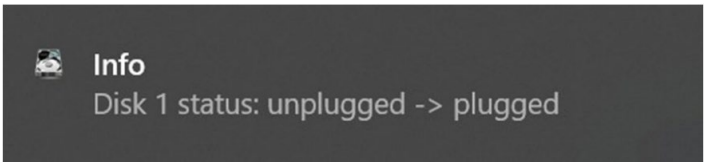



This completes the disk unmounting process.

## Mounting a Disk Drive

Insert the disk into the computer.

The following notification is displayed.



## Hot Swap Function Notifications

| Notification Type | Notification Content | Description |
|---|---|---|
| Info | Button is pressed | Displayed when a button is pressed for more than 3 seconds. |
| Info | Disk has been unmounted | Displayed after a disk is successfully unmounted. |
| Info | Disk status: unplugged -> plugged | Displayed after a disk is plugged into the computer. |
| Info | Disk status: plugged -> unplugged | Displayed after a disk is unplugged from the computer. |
| Info | Partition is over usage, execute user program | Displayed when the disk usage is over a predefined threshold. A user-specified program is run after this event is triggered. |
| Info | Partition is over usage, use system default action | Displayed when the disk usage is over a predefined threshold. A user-specified program is run after this event is triggered. |
| Error | Disk unmount fail. Device is currently in use or in incorrect removal policy | Displayed when disk unmount process fails; disk is in busy state or the removal policy is incorrect. |
| Error | Invalid handle | Displayed when an internal error occurs. |
| Application | Starting Moxa Disk Hot Swap Application | Displayed when the Moxa Disk Hot Swap application starts up. |
| Application | Stopping Moxa Disk Hot Swap Application | Displayed when the Moxa Disk Hot Swap application stops running. |

# Moxa LANBypass Utility

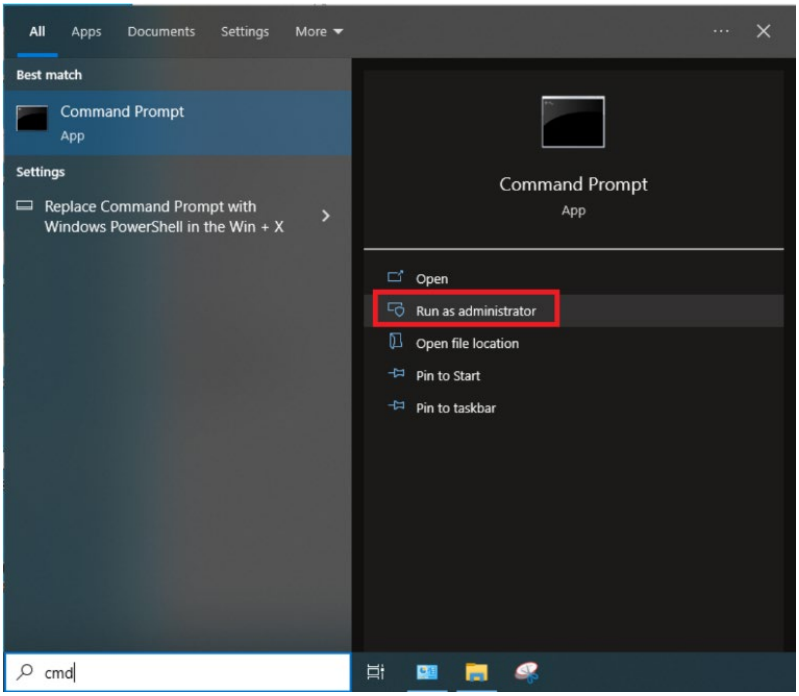
This tool allows users to independently update the firmware of the MCU (Microcontroller Unit).

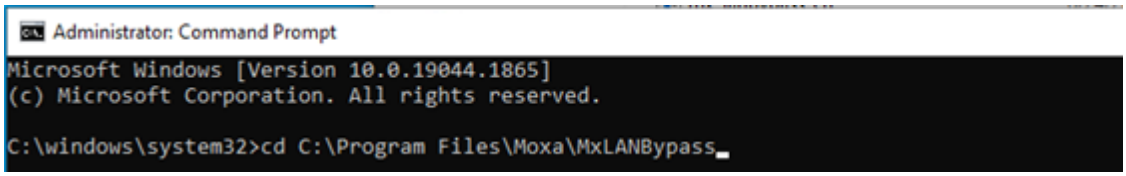
**NOTE**

This tool is only applicable to models equipped with 8 LAN ports.

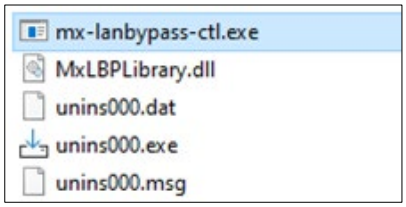
To run the utility, do the following:

1. Open the command prompt as Administrator and type the following to access the folder: **C:\ProgramFiles\Moxa\MxLANBypass**.

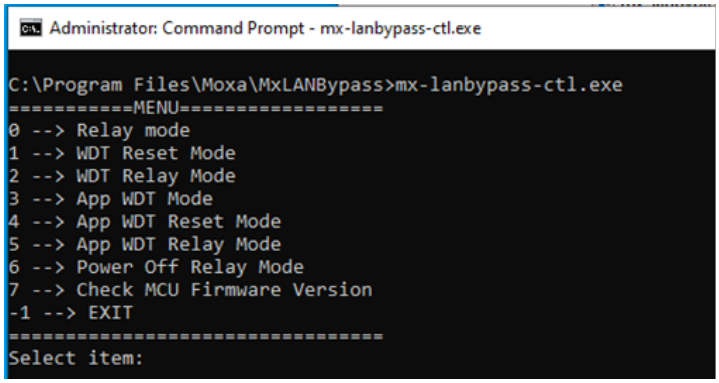





2. Run the mx-lanbypass-ctl.exe from the **C:\Program Files\Moxa\MxLANBypass** folder.




3. Type in the number corresponding to the function that you want to run.
   Refer to the LAN Bypass Modes section for a detailed description of the functions.

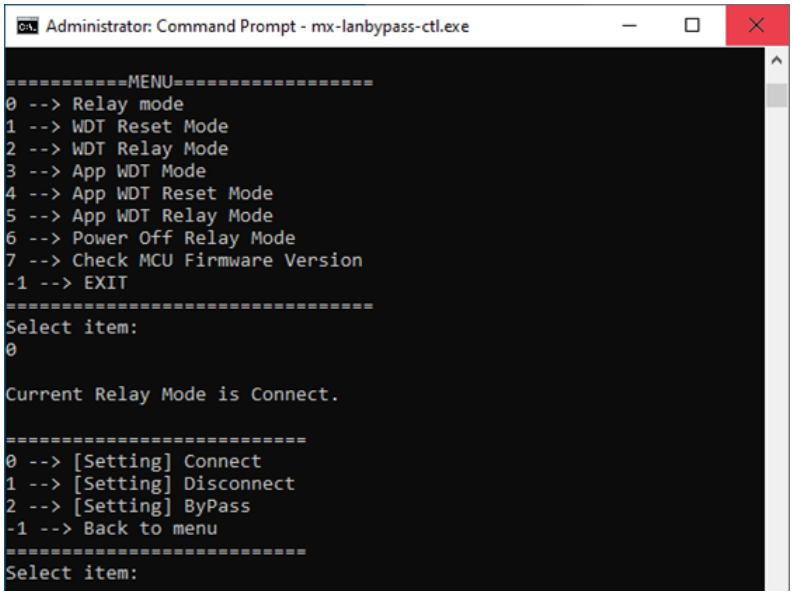



# Relay Mode

Use this mode to set the connection type between the target LANs.

After typing **0** in the prompt for **Relay mode**, select one of the following options:

| Index | Mode | Description |
|---|---|---|
| 0 | Connect | Target LANs connected. |
| 1 | Disconnect | Target LANs disconnected. |
| 2 | Bypass | Target LANs are physically connected without passing through the network device. |

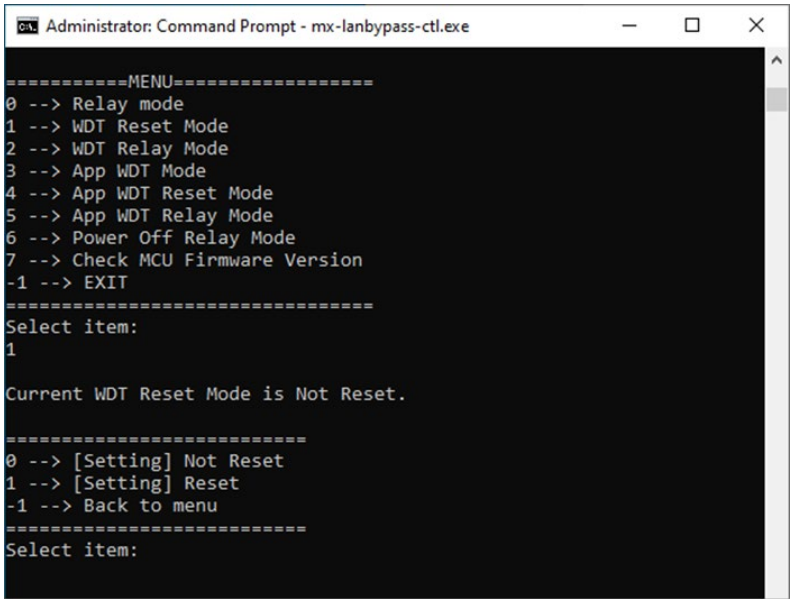


# WDT Reset Mode

Use this mode to indicate if the system needs to be reset after the watchdog is triggered.

After typing 1 for WDT Reset Mode, select one of the following options:

| Index | Mode | Description |
|---|---|---|
| 0 | Not Reset | The system will not reset when the watchdog is triggered. |
| 1 | Reset | The system will reset when the watchdog is triggered. |

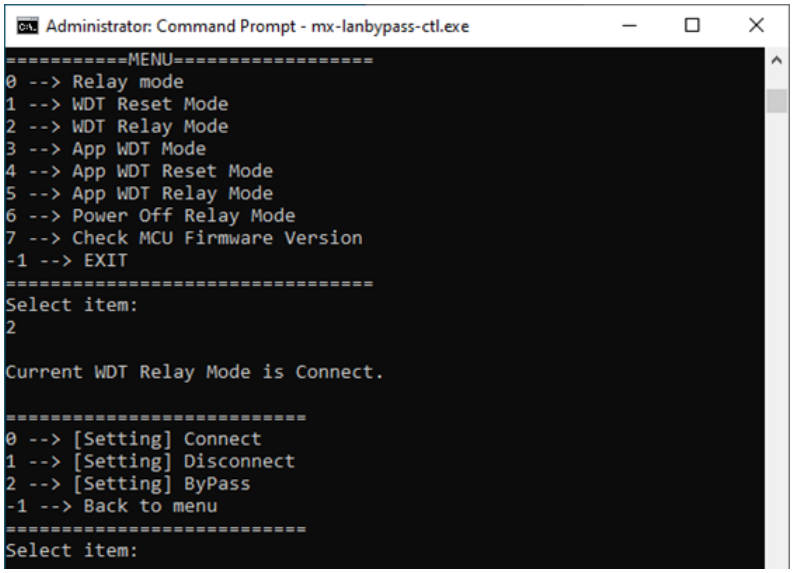



# WDT Relay Mode

Use this mode to indicate if the system needs to be reset after the watchdog is triggered.

After typing 2 for WDT Relay Mode, select one of the following options:

| Index | Mode | Description |
|---|---|---|
| 0 | Connect | Target LANs connected. |
| 1 | Disconnect | Target LANs disconnected. |
| 2 | Bypass | Target LANs are physically connected without passing through the network device. |



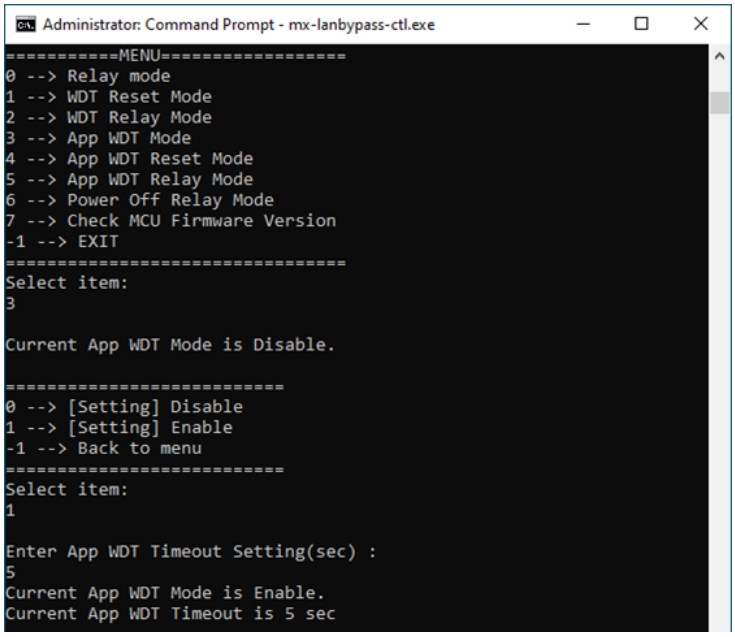
# App WDT Mode

Use this mode to enable or disable the watchdog application. Activating the watchdog function is key to creating a trigger to activate LAN bypass when your application encounters issues or is unresponsive.

After typing 3 for App WDT Mode, select one of the following options:

| Index | Mode | Description |
|---|---|---|
| 0 | Disable | Disable the app watchdog function. |
| 1 | Enable | Enable the app watchdog function. |

When you select Enable, you must enter an App WDT timeout (5 to 15 sec) to set the interval for sending a watchdog "kick" the MCU.

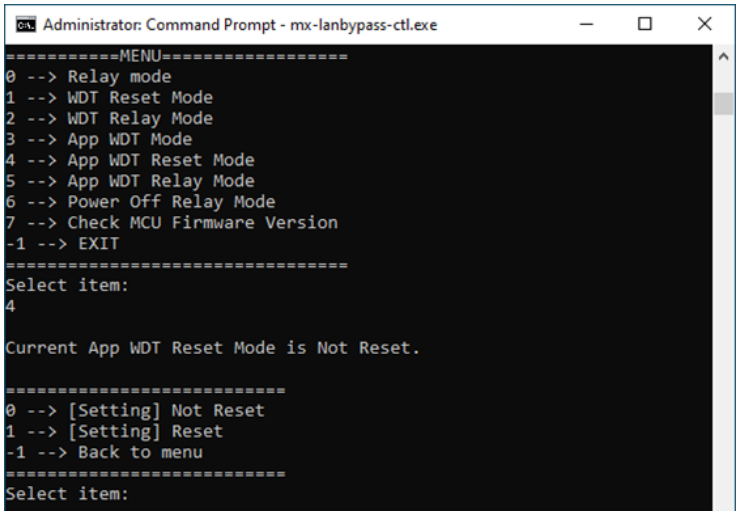



# App WDT Reset Mode

Use this mode to indicate if the system needs to be reset after the app watchdog is triggered.

After typing 4 for App WDT Reset Mode, select one of the following options:

| Index | Mode | Description |
|---|---|---|
| 0 | Not Reset | The system will not reset when the watchdog is triggered. |
| 1 | Reset | The system will reset when the watchdog is triggered. |

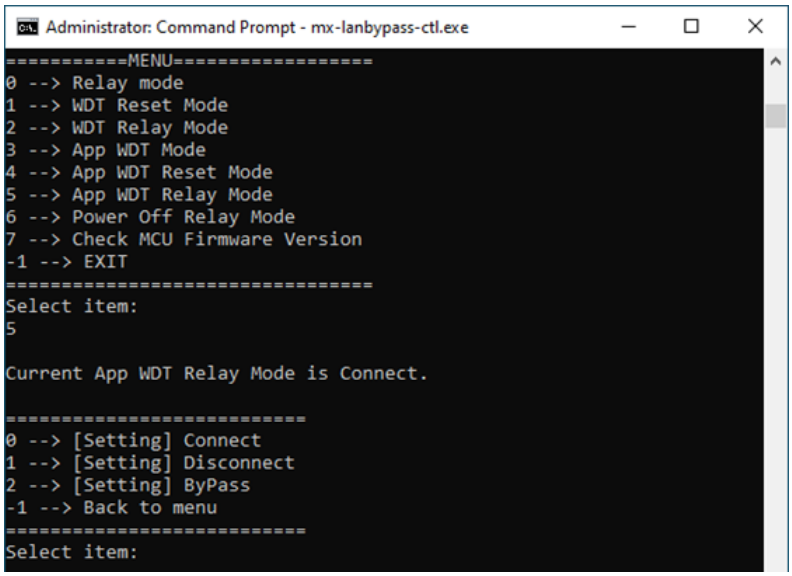


# App WDT Relay Mode

Use this mode to indicate the relay mode to switch to after the app watchdog is triggered.

After typing 5 for App WDT Relay Mode, select one of the following options:

| Index | Mode | Description |
|---|---|---|
| 0 | Connect | Target LANs connected. |
| 1 | Disconnect | Target LANs disconnected. |
| 2 | Bypass | Target LANs are physically connected without passing through the network device. |

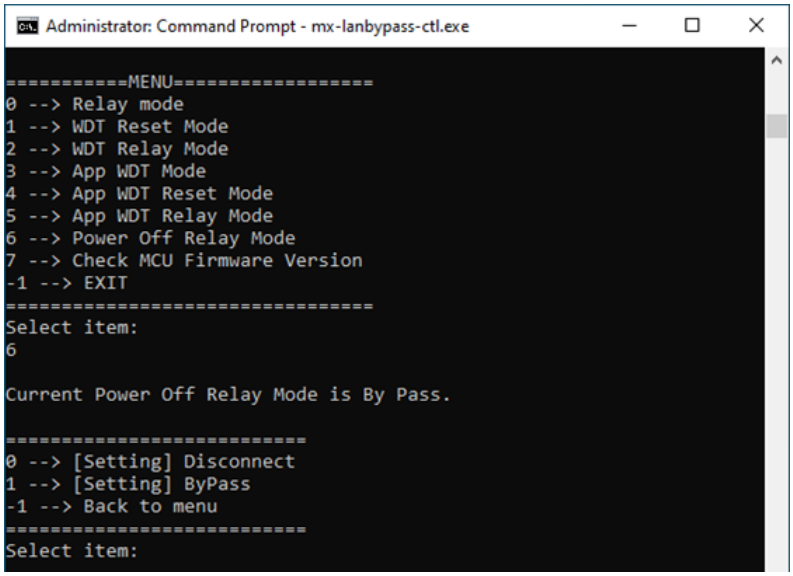



# Power Off Relay Mode

Use this mode to indicate the relay mode to switch to after the system is powered off.

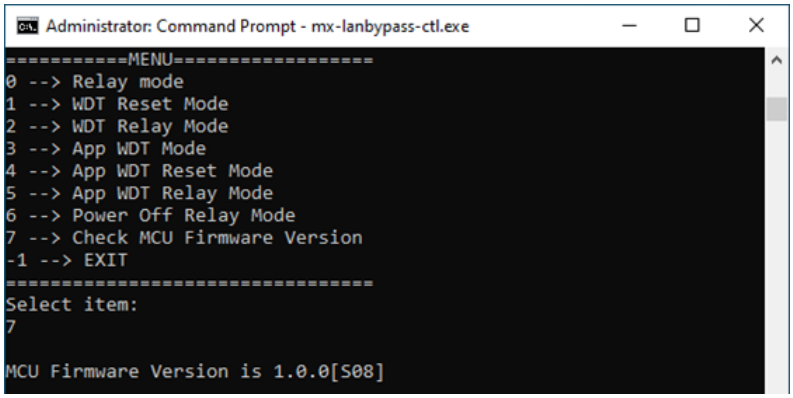
After typing 6 for Power Off Relay Mode, select one of the following options:

| Index | Mode | Description |
|---|---|---|
| 0 | Disconnect | Target LANs disconnected. |
| 1 | Bypass | Target LANs are physically connected without passing through the network device. |



## Checking the Firmware Version of the MCU

Use this option to check the firmware version on the MCU. Type 7 in the prompt to select Check MCU Firmware Version. The firmware version information is displayed at the prompt.




# Moxa MCU Firmware Upgrade

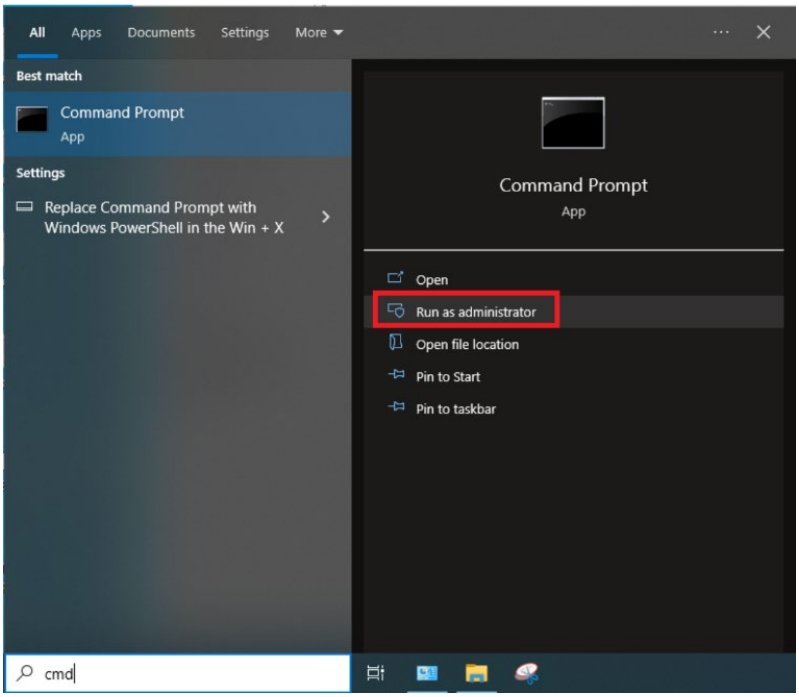
This tool allows users to independently update the firmware of the MCU (Microcontroller Unit).

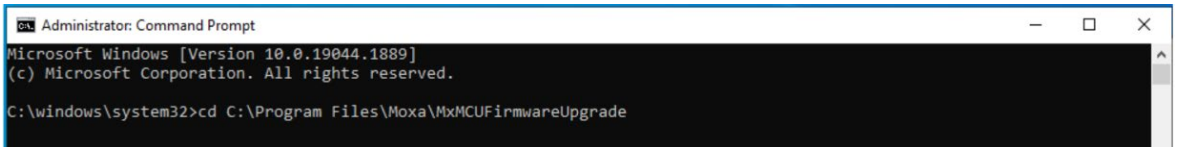
**NOTE**

This tool is only applicable to models equipped with 8 LAN ports.

To run the utility, do the following:

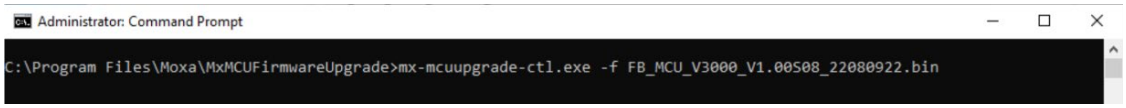
Open the Command Prompt as Administrator and enter C:\ProgramFiles\Moxa\MxMCUFirmwareUpgrade\.



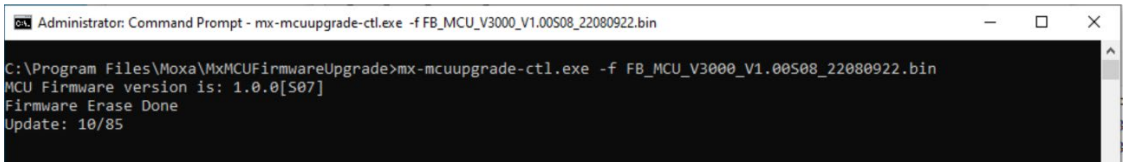



# Upgrade the MCU Firmware

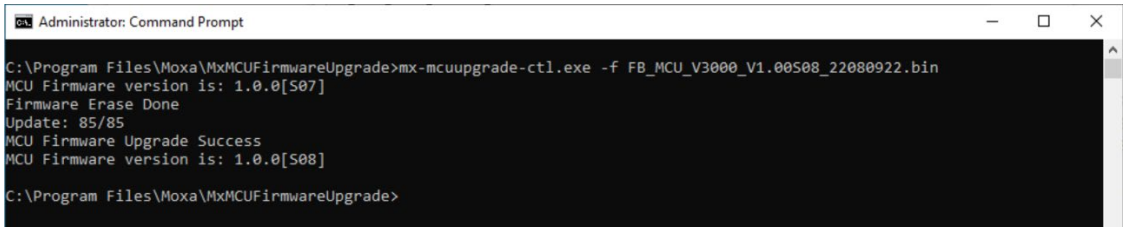
1. Download the latest MCU firmware upgrade file from Moxa's support website at https://www.moxa.com/en/support to the V3400 computer.
2. Copy the update files in C:\Program Files\Moxa\MxMCUFirmwareUpgrade\.
3. Run the Command Prompt as Administrator to upgrade MCU firmware in Command Prompt.




4. Waiting for update.



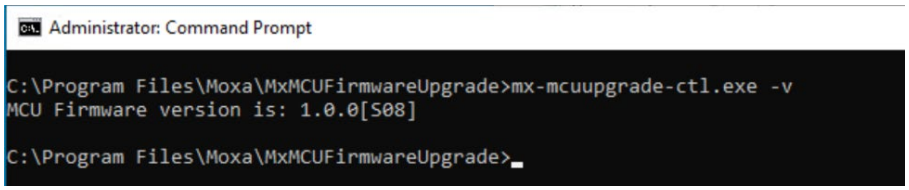

5. Update completed.




# Check MCU Firmware Version

Run the following command to check current MCU firmware version.

```
mx-mcuupgrade-ctl.exe -v
```



# Moxa Tca9535 Initial Service

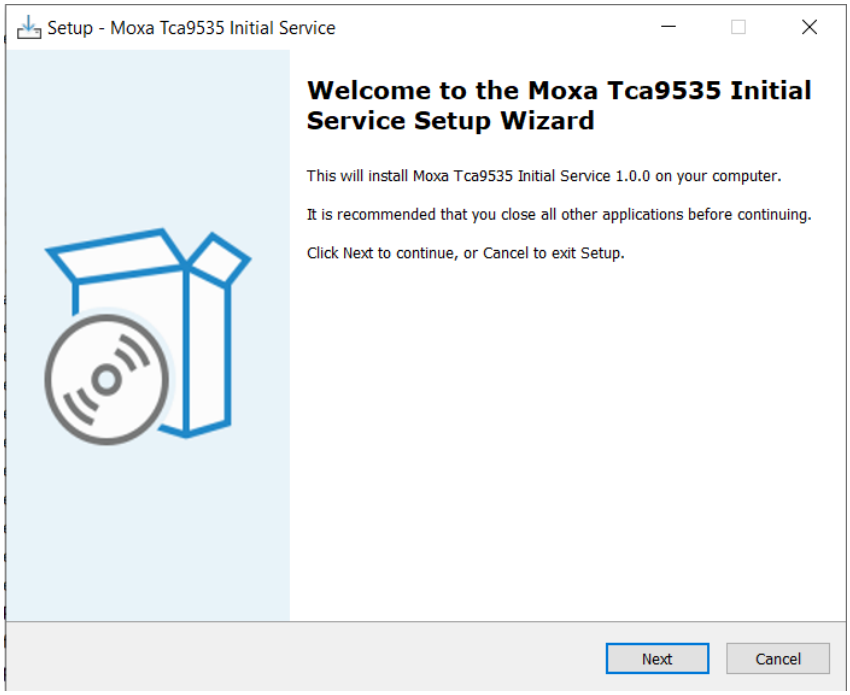
This service is used to initialize the TCA9535 and will automatically run in the background after each system startup.

## NOTE

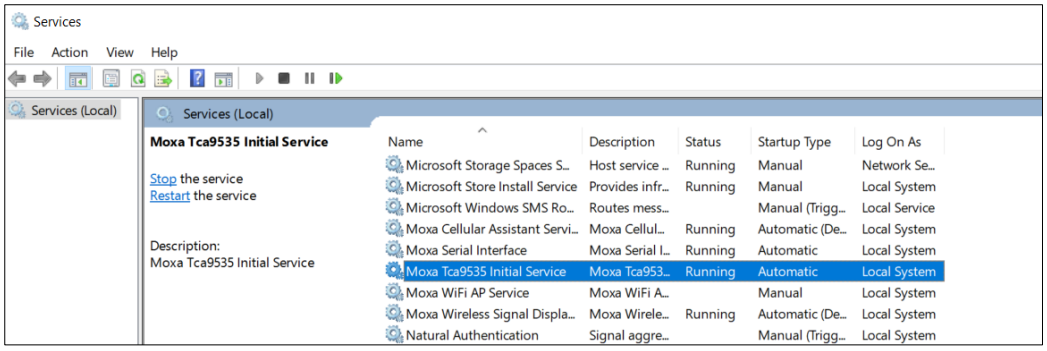
Installation is only required if the Moxa Cellular Assistant utility is needed. Please install the Moxa Tca9535Initial Service before installing the Moxa Cellular Assistant utility.

To run the utility, do the following:

1. Run **Tca9535InitialServiceSetup.exe** to Install Moxa Tca9535 Initial Service. The service will start running automatically after installation.




2. To verify whether the service is running successfully, go to the Windows Services utility and check the status of the 'Tca9535InitialService'.



# Moxa Cellular Assistant

Windows Cellular Assistant provides command line tool and GUI utility to configure the connection of Cellular module. And there is a Service running in the background and automatically initializing and setting the Cellular module according to configuration. This chapter describes how to set up and config Cellular module connection configuration. The following topics are covered in this chapter:

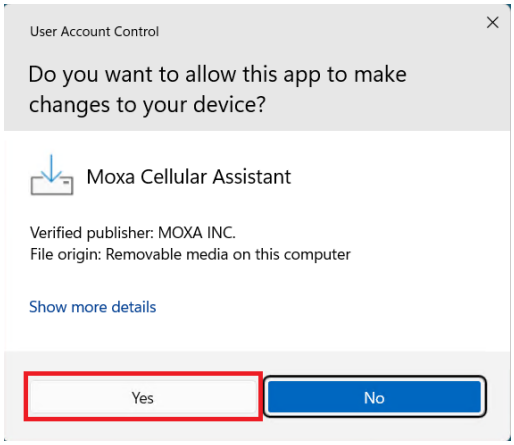
- Install Moxa Cellular Assistant
- Command Line Tool
  - Unlock SIM PIN
- GUI utility
  - Setup Cellular Connection
  - Configure Connection Check-alive
  - Disconnect Log

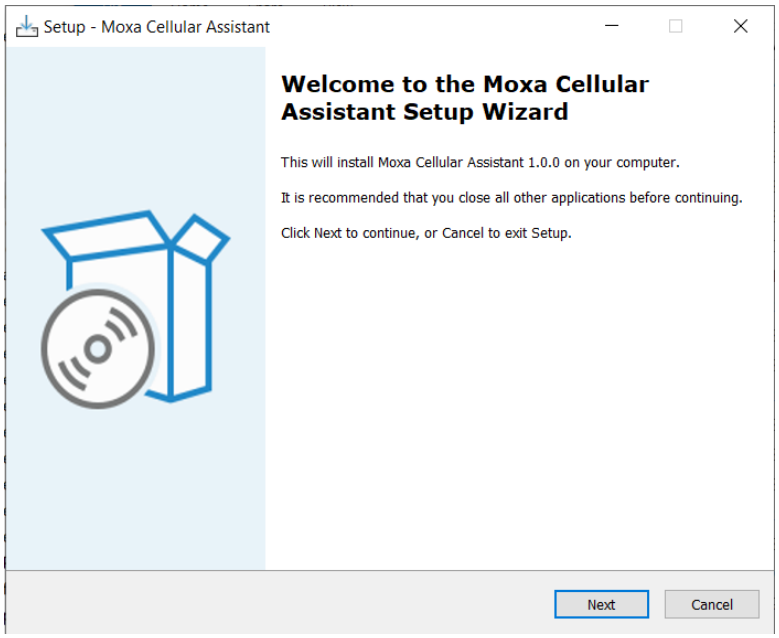
**NOTE**

Please install the Moxa Tca9535Initial Service before installing the Moxa Cellular Assistant utility.

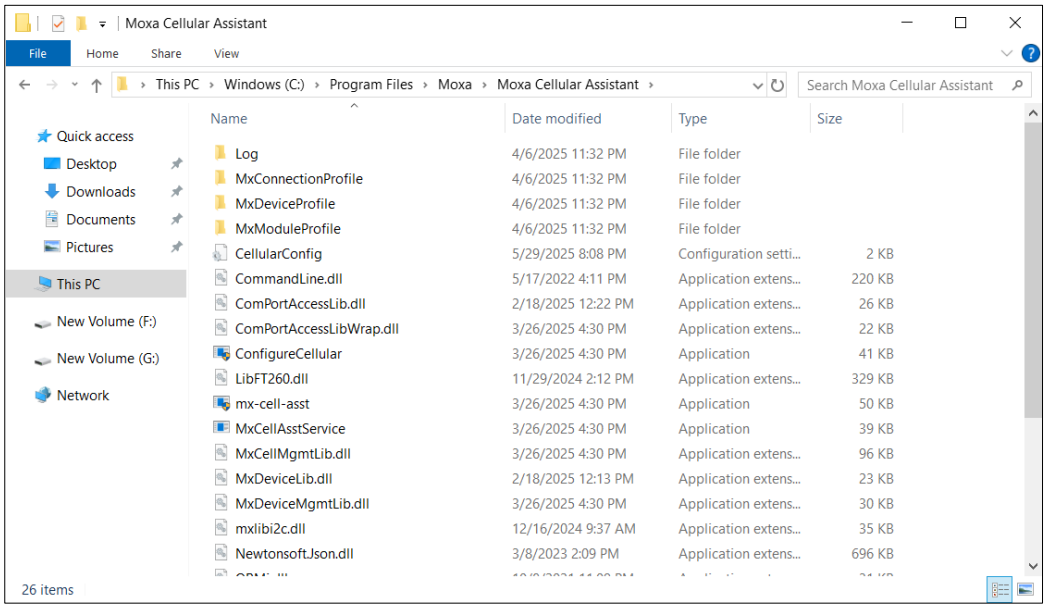
## Install Moxa Cellular Assistant

1. Run **CellularAssistantSetup.exe** to install Moxa Cellular Assistant Service.



2. After installation finished, **Moxa Cellular Assistant Service** will automatically start and initialize the Cellular module on the device.
3. You can find the command line tool **mx-cell-asst.exe** and **ConfigureCellular.exe** at the **C:\Program Files\Moxa\Moxa Cellular Assistant**. These two utilities will be used later for Cellular configuration.

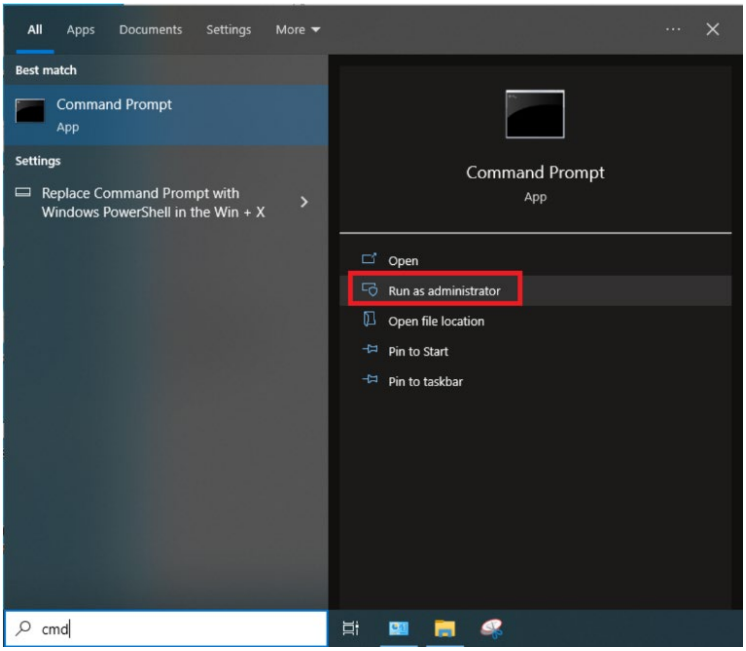


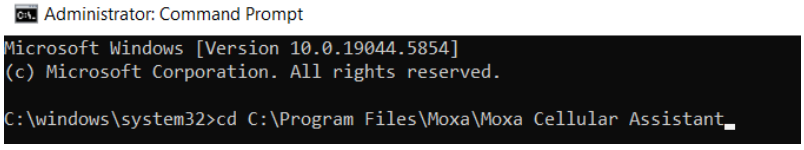

4. It may take 2-3 minutes for Service to initialize the Cellular module and the system to prepare the device.

# Command Line Tool

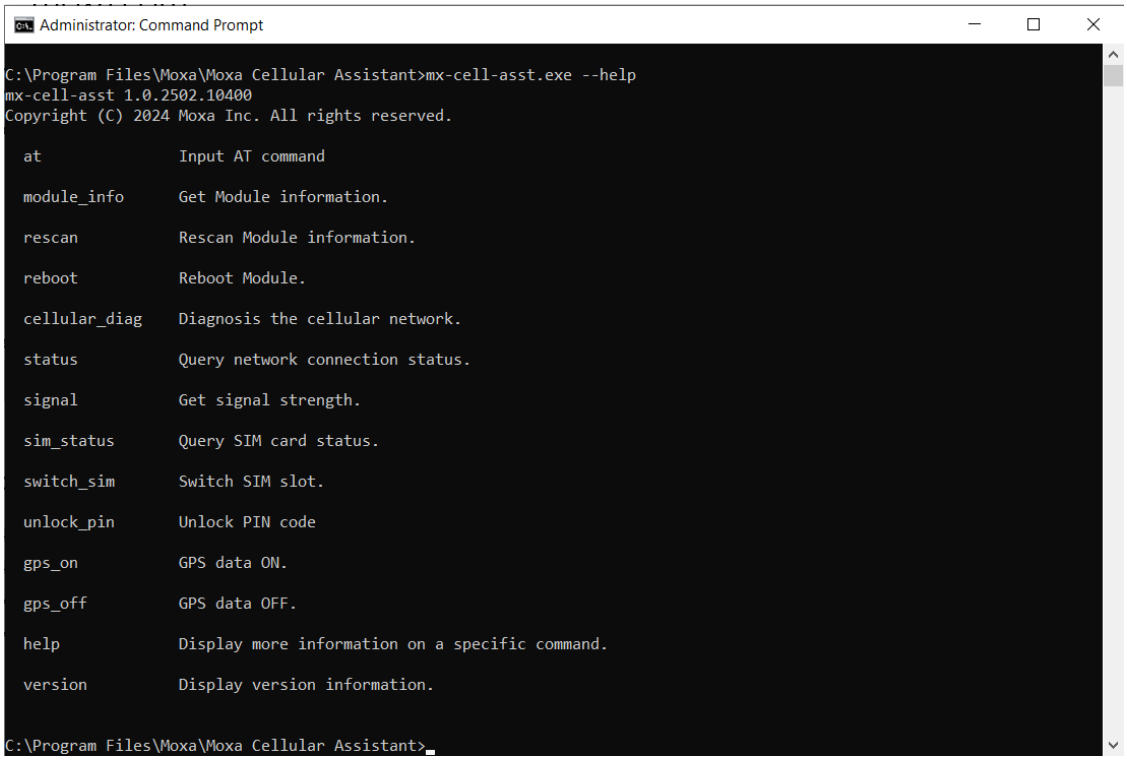
To run the utility, do the following:

- Open the command prompt as Administrator and type the following to access the folder: **C:\Program Files\Moxa\Moxa Cellular Assistant**.



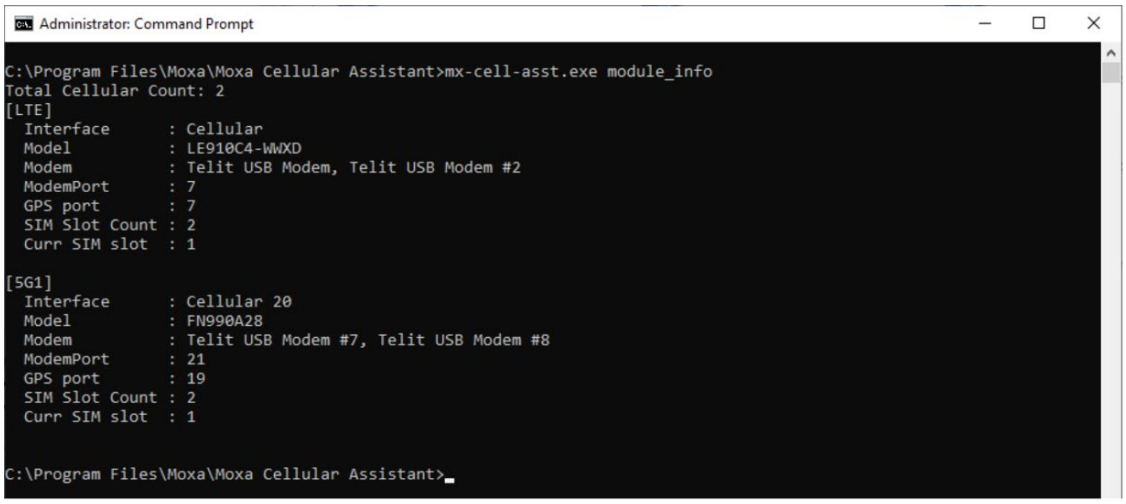
- Run the **"mx-cell-asst.exe --help"** command to see the instructions on using this utility and follow them to configure the cellular module status.




- **module_info**

  module_info command provides information of the Cellular module. The name in **"[ ]"** corresponds to the front text print. This information will be used in the following settings.

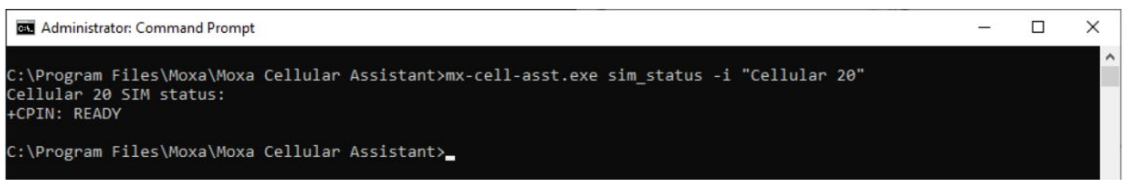
```
mx-cell-asst.exe module_info
```




- **sim_status -i [interface]**

  Using **sim_status** command with parameter **"Interface Name"** to get the SIM card status

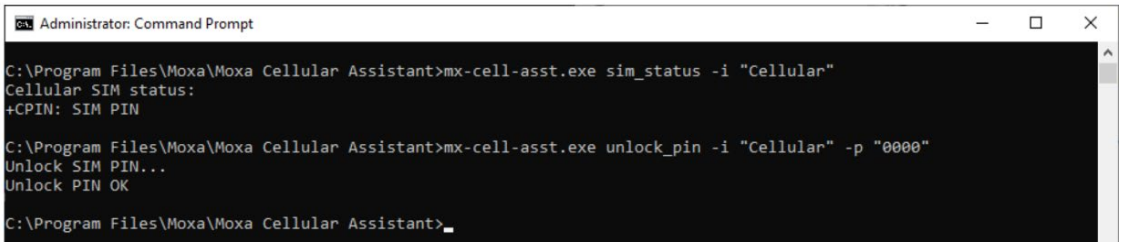
```
mx-cell-asst.exe sim_status -i "Cellular 20"
```



- **unlock_pin -i [interface] -p [pin_code]**

  Using **unlock_pin** command with parameter **"Interface Name"** and **"PIN Code"** to unlock SIM Pin.

```
mx-cell-asst.exe unlock_pin -i "Cellular" -p "0000"
```

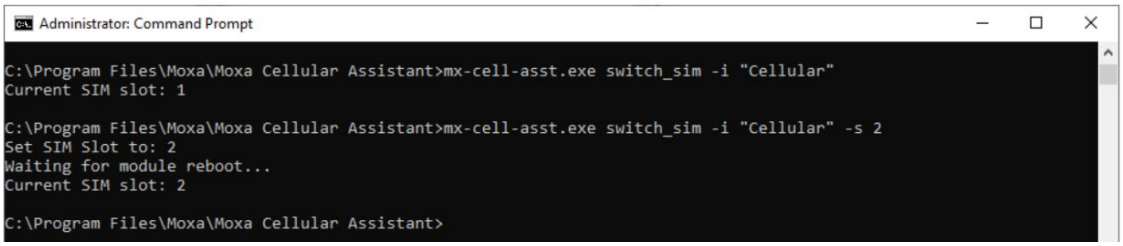



- **switch_sim -i [interface] -s [slot]**

  Using **switch_sim** command with parameter "Interface Name" to check current sim slot, using parameter "Interface Name" and "slot" to switch sim slot selection.

```
mx-cell-asst.exe switch_sim -i "Cellular"
```

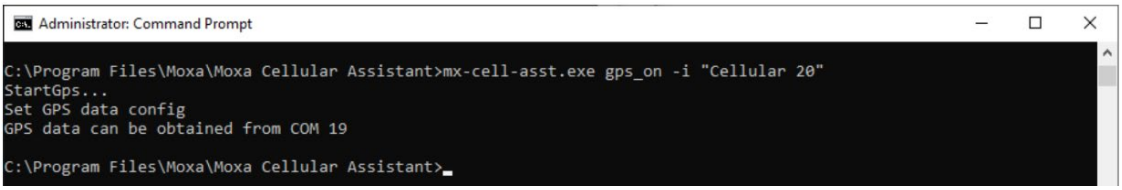
```
mx-cell-asst.exe switch_sim -i "Cellular" -s 2
```

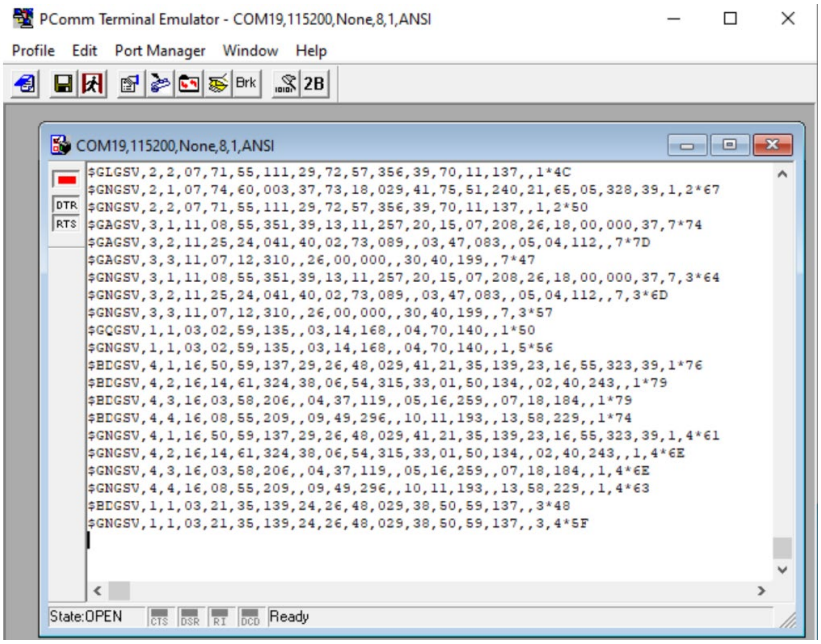



- **gps_on**

  Using **gps_on** command with parameter **"Interface Name"** to enable GPS data output and get GPS information from the specified com port.

```
mx-cell-asst.exe gps_on -i "Cellular"
```

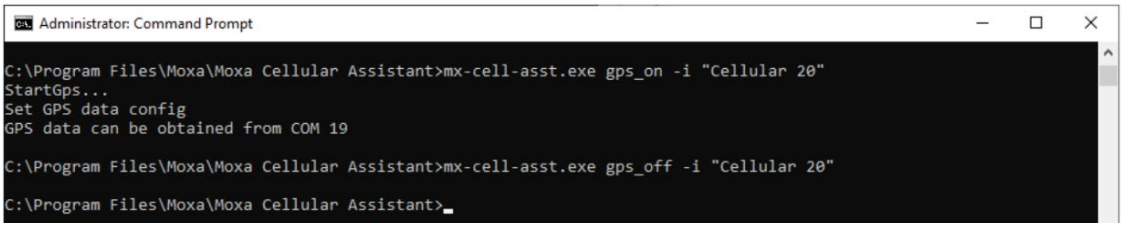


- **gps_off**

  Using **gps_off** command with parameter **"Interface Name"** to disable GPS data output.

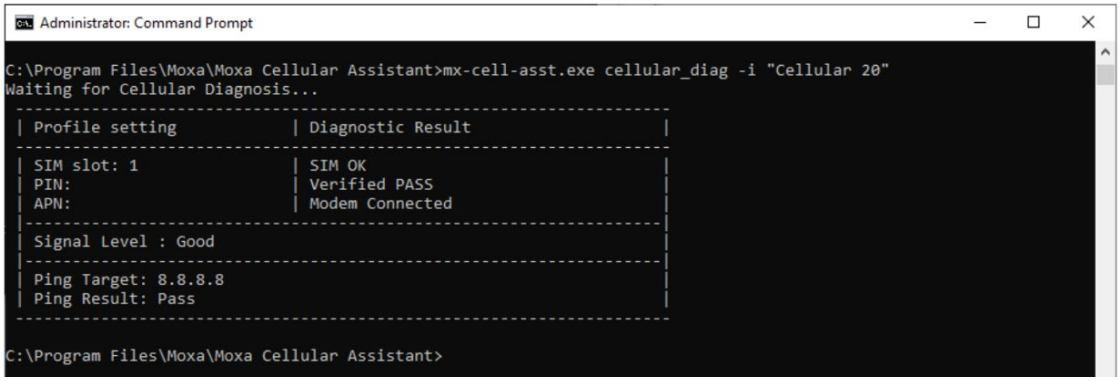
```
mx-cell-asst.exe gps_off -i "Cellular"
```




- **cellular_diag**

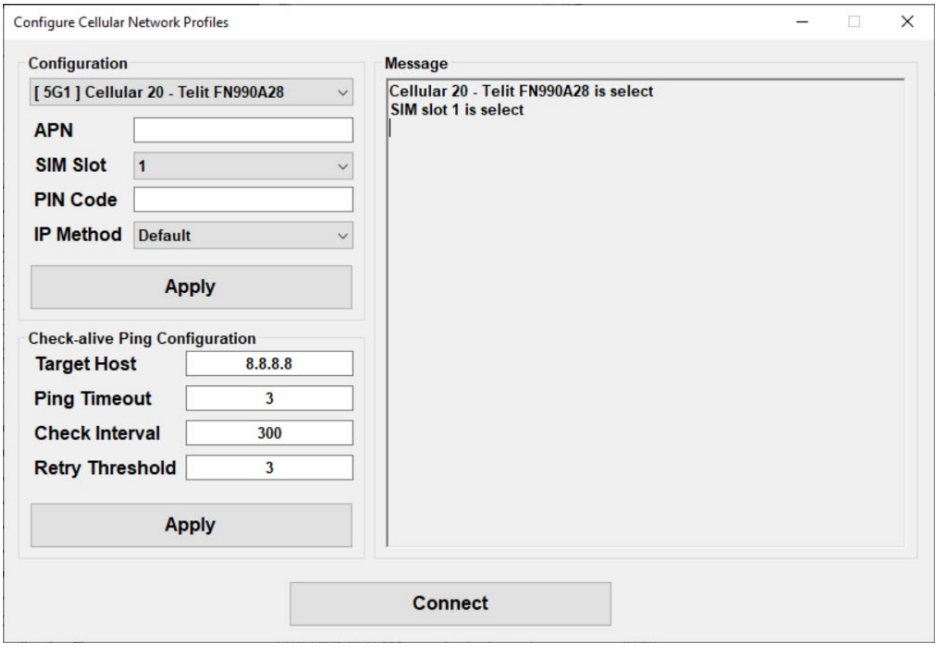
  Using **cellular_diag** command with parameter **"Interface Name"** to perform diagnosis on the cellular network.

```
mx-cell-asst.exe cellular_diag -i "Cellular"
```

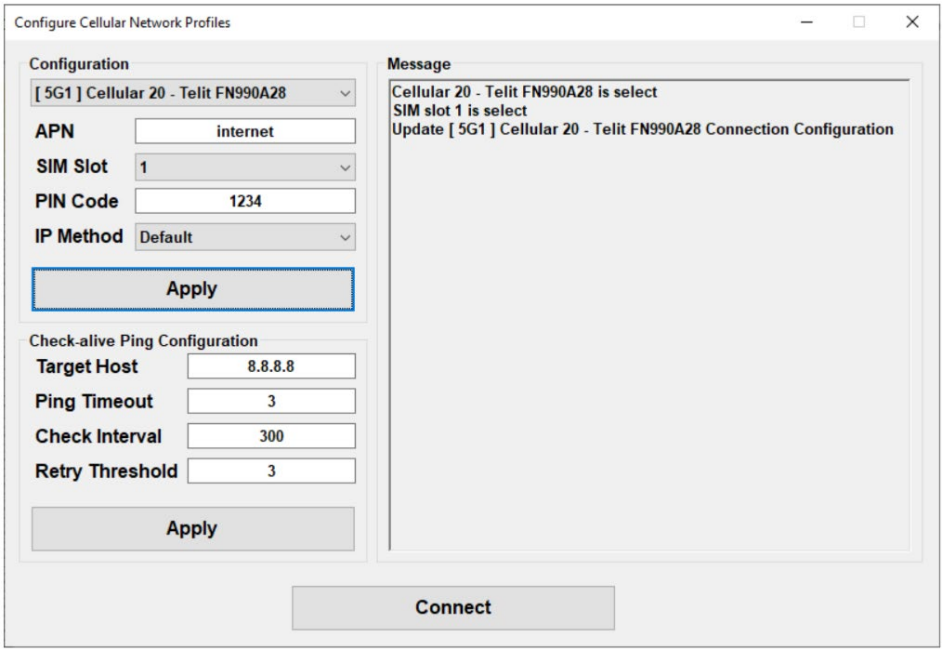



# GUI Utility

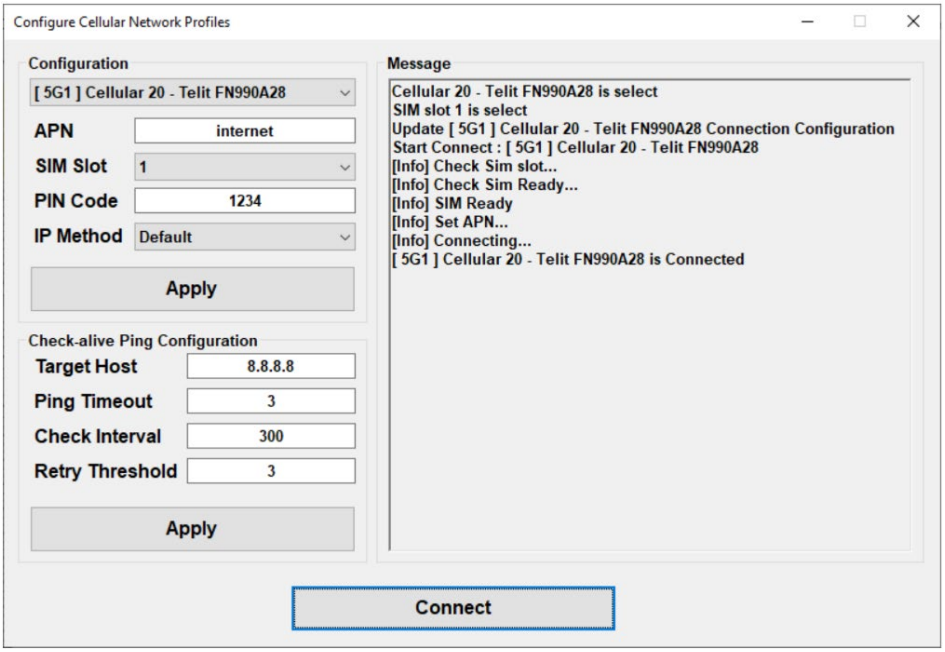
The UI program consists of two sections: **Cellular Configuration** and **Check-Alive Settings**. It also features an action message window.



- **Setup Cellular Connection Config**
  - Select the Cellular Interface, set APN, SIM Slot, PIN code, IP Method and click **Apply** to save connection configuration.




  - After every Cellular Interface's config finished, click **Connect** to connect all Cellular Interface to Internet.



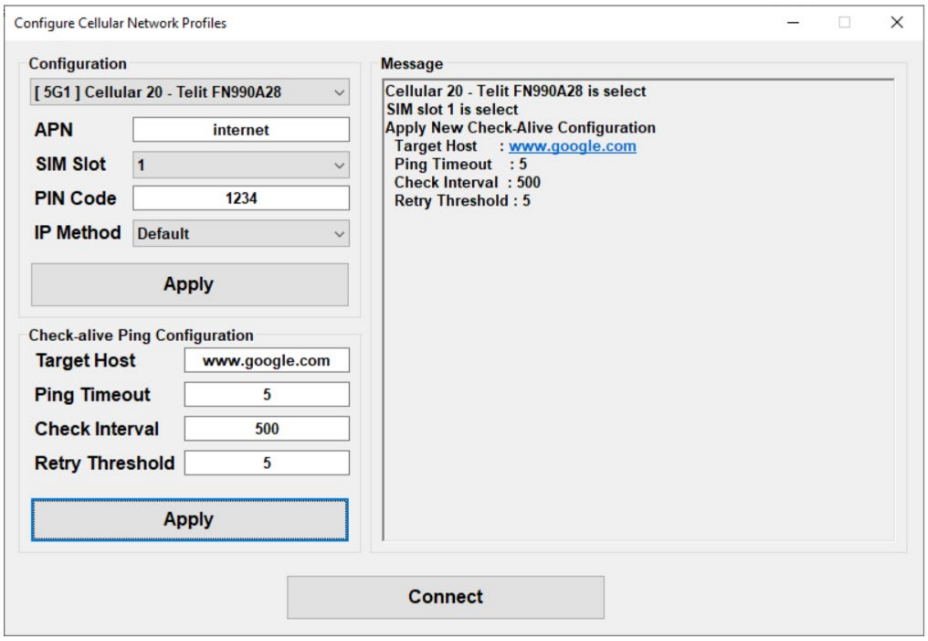
- **Configure Connection Check-alive**
  - Select a method to determine whether the connection is successful. The system uses the ping command to verify whether the connection between the device and the server is functioning properly.
    - Target Host:

      The default value is 8.8.8.8. It also support URL.
    - Ping Timeout:

      The default is 3 seconds, indicating the timeout of the ping command.
    - Check Interval:

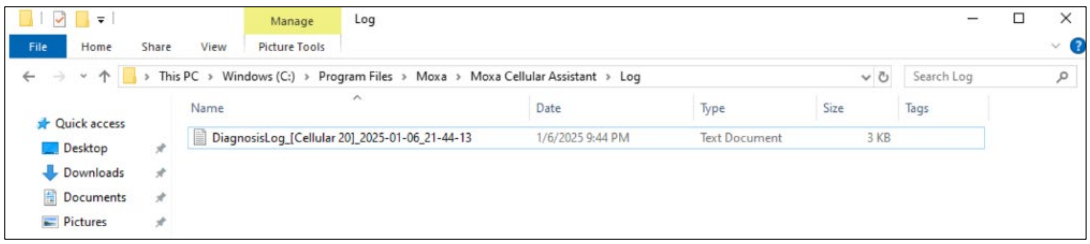
      The default is 300 seconds, which means the ping method will be checked for connectivity every 300 seconds.
    - Retry Threshold:

      The default is 3, which means each check-alive will ping 3 times to check for connectivity.
  - After set the check-alive config, click **Apply** to update check-alive configuration and the service will start check-alive using new configuration.




- **Disconnect Log**

  When the connection between the machine and the server is lost, service will automatically record module information for the problem can be analyzes later.



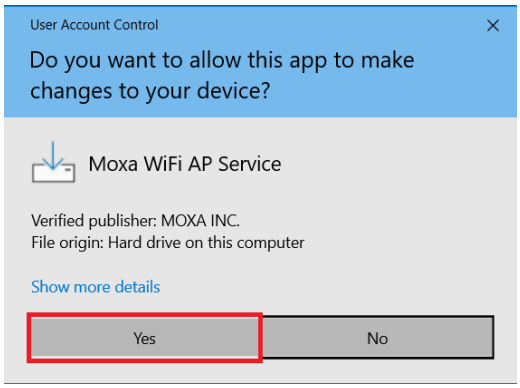
# Moxa WiFi AP Service

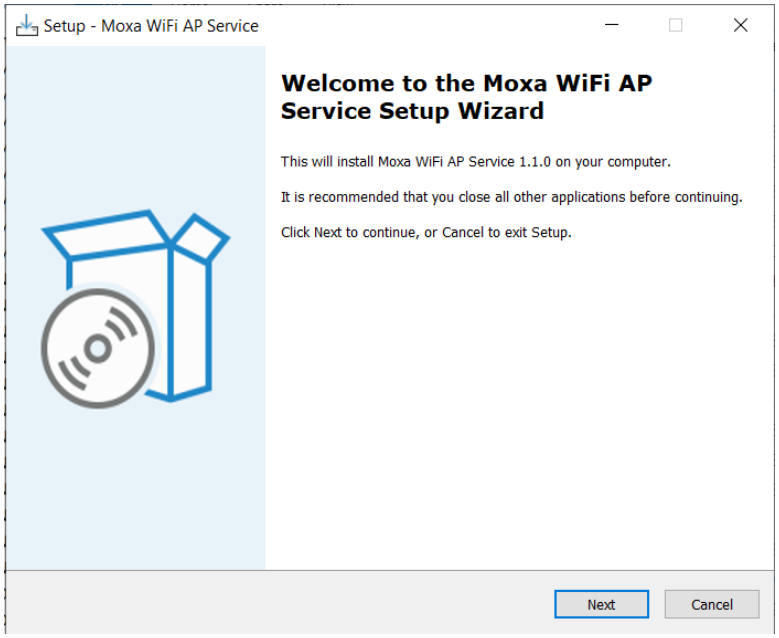
This chapter describes how to set up and config **WiFi AP Service** which using **WiFi Direct** technology to achieve WiFi AP functionality on Windows. The utility will set up a service to control the WiFi AP and a command line tool to set the config of the service and WiFi AP. The following topics are covered in this chapter:

- Install WiFi AP Service
- Configuring the WiFi AP Service

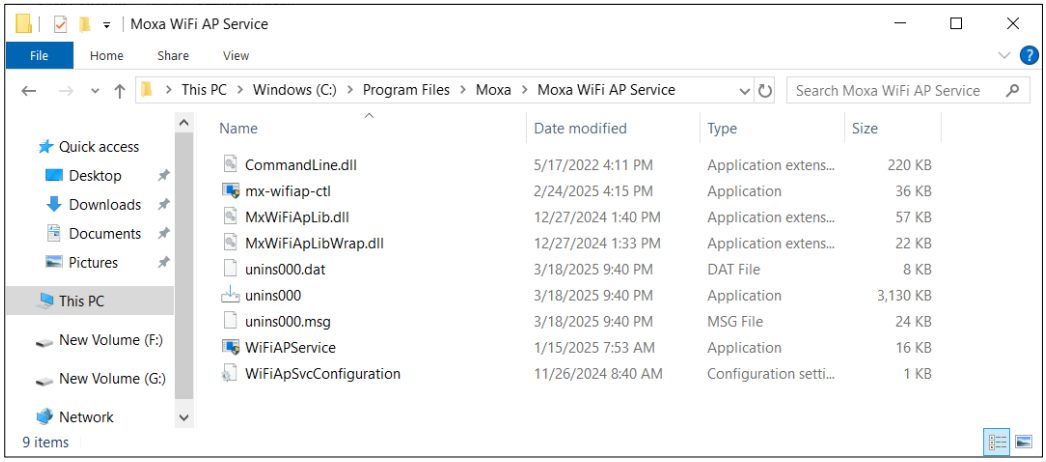
## Install WiFi AP Service

- Run "WiFiAPServiceSetup.exe" to install WiFi AP Service.



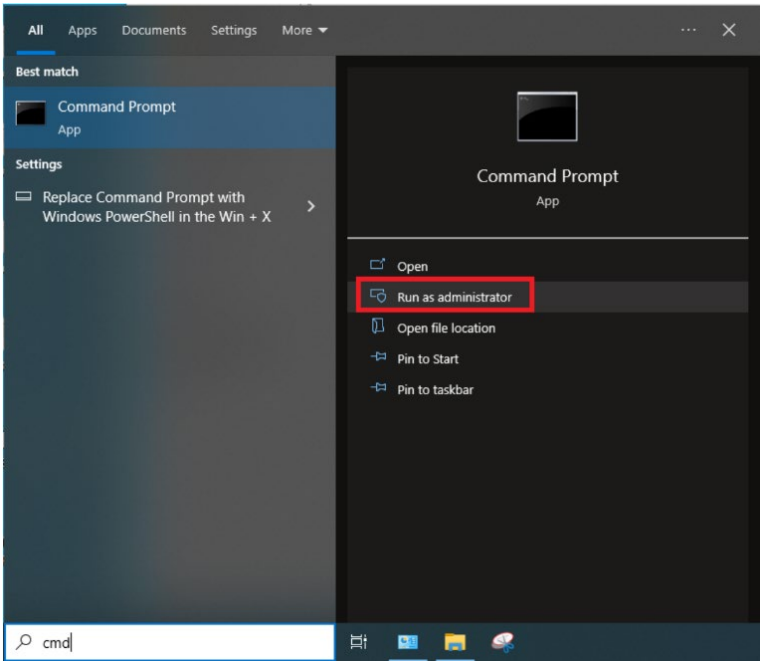



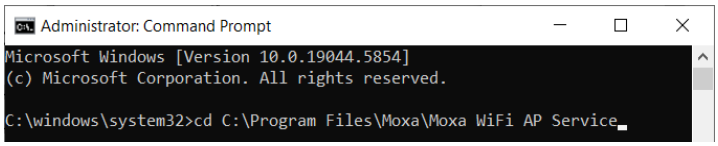
- After installation finished, you can find the command line tool **mx-wifiap-ctl.exe** at the **C:\Program Files\Moxa\Moxa WiFi AP Service**.



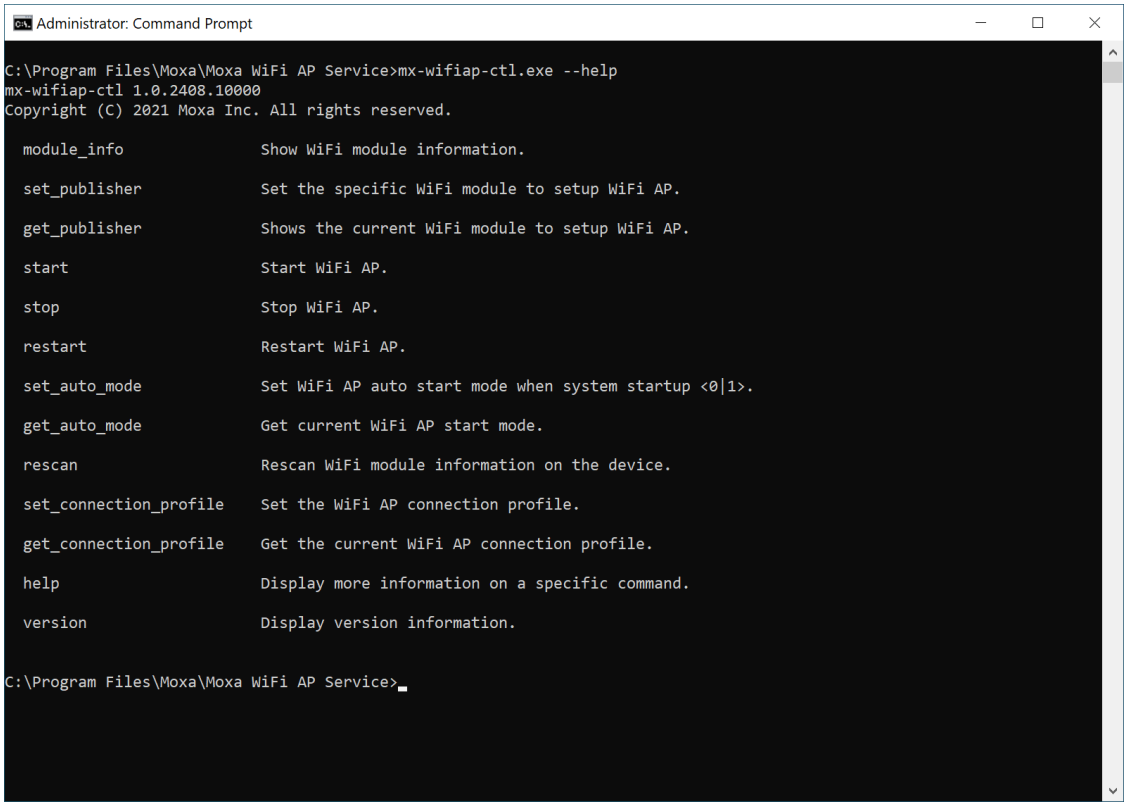

# Configuring the WiFi AP Service

- Open the command prompt as Administrator and type the following to access the folder: **C:\Program Files\Moxa\Moxa WiFi AP Service**.



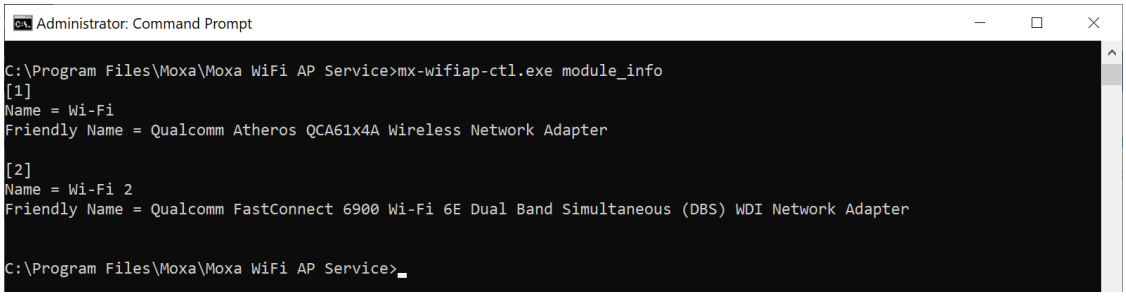



- Run the **mx-wifiap-ctl.exe --help** command to see the instructions on using this utility and follow them to configure the WiFi AP status.



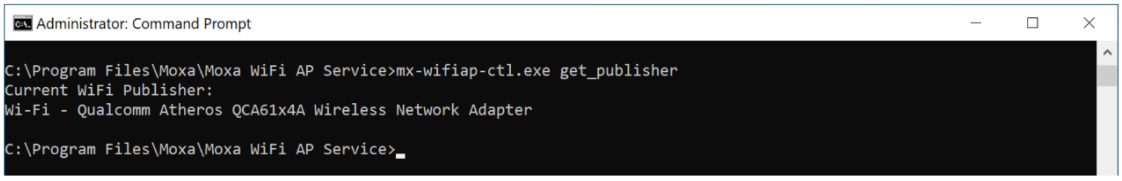

- **module_info**

  Provides information of the WiFi module and the module information will save to the config file.



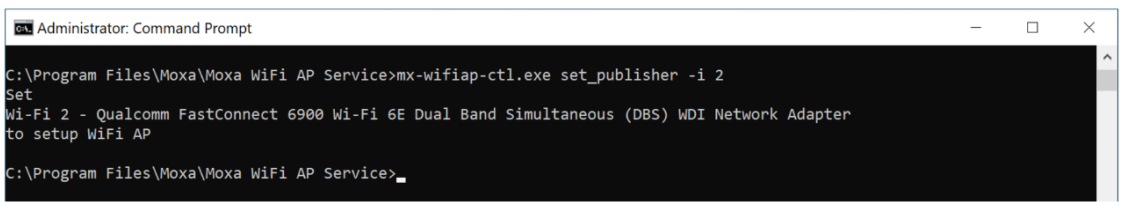

- **get_publisher**

  Get the current WiFi module to set up WiFi AP.



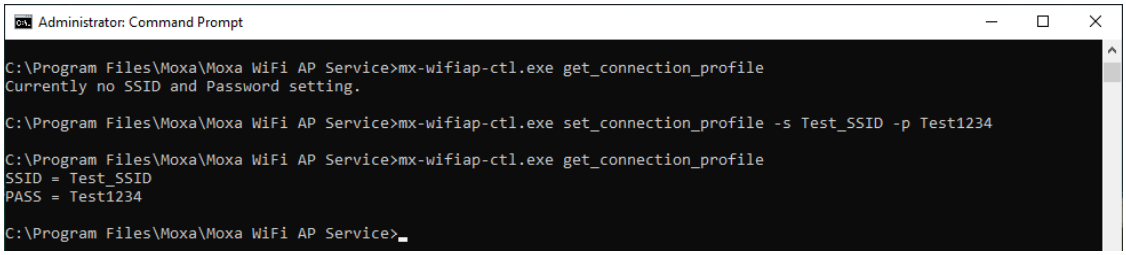

- **set_publisher -i [index]**

  Use the **WiFi index** captured by **module_info** command to set the WiFi module used to create WiFi AP.



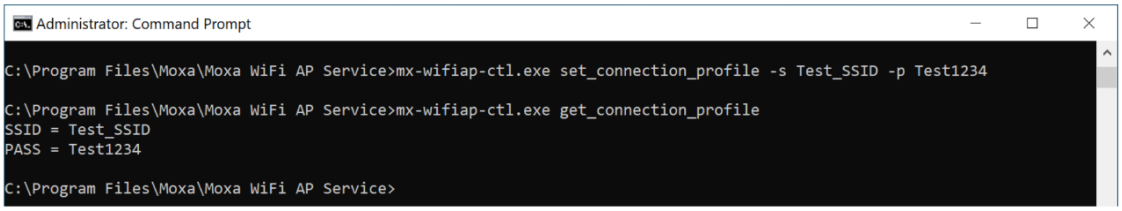
- **get_connection_profile**

  Get the current connection profile's **SSID** and **Passphrase** setting. There is no default SSID and Passphrase.

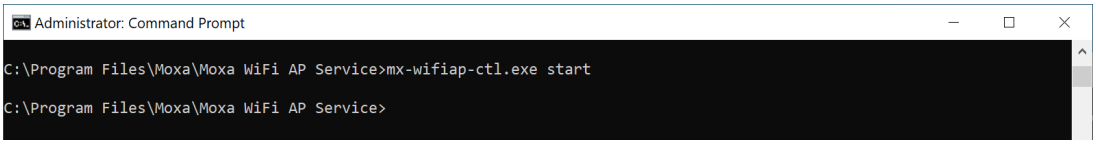



- **set_connection_profile -s SSID -p PASS**

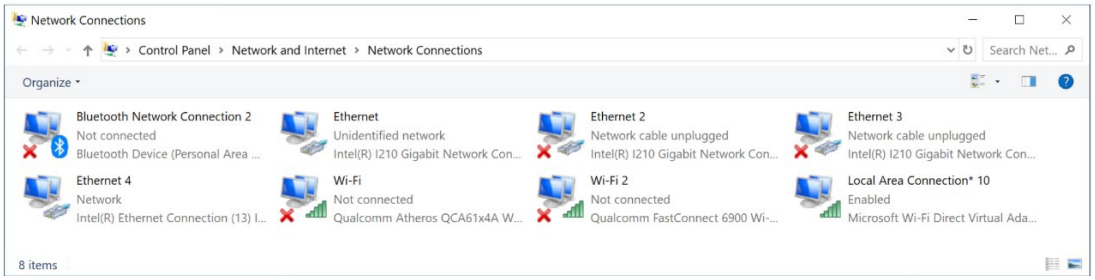
  Set the connection profile's **SSID** and **Passphrase**.



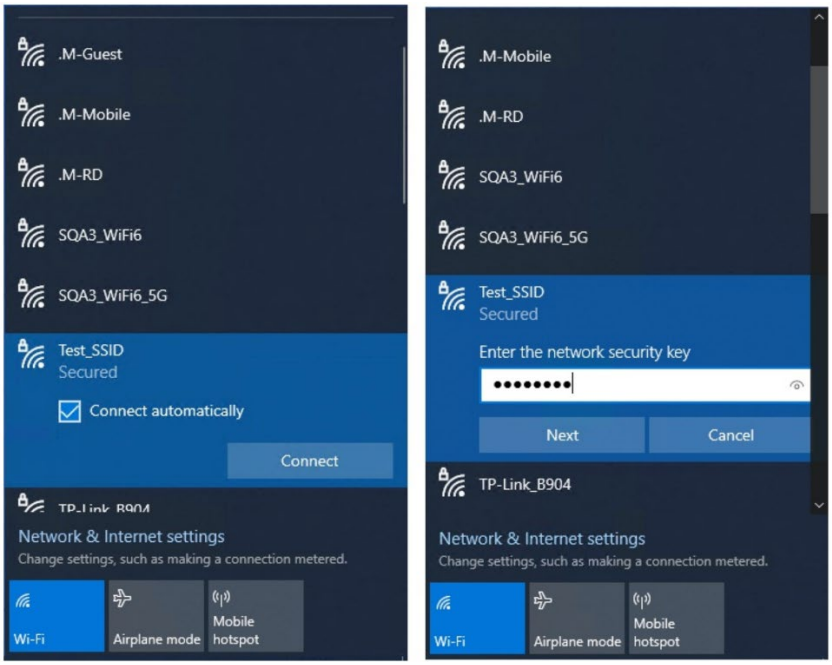

- **start**
  - Start the WiFi AP.




  - At the Network Connections page can be found that there is a **"Microsoft Wi-Fi Direct Virtual Adapter"** used to establish WiFi Direct connection.

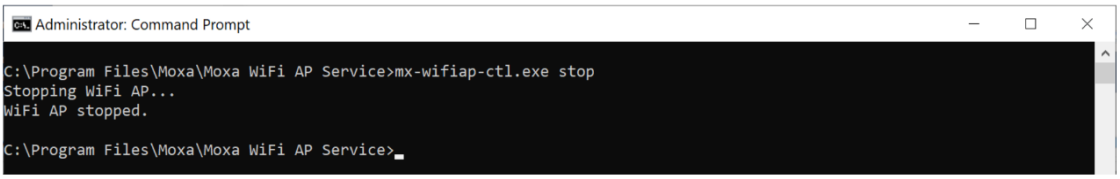


➢ You can find this WiFi access point on another computer and connect using the previously set password.



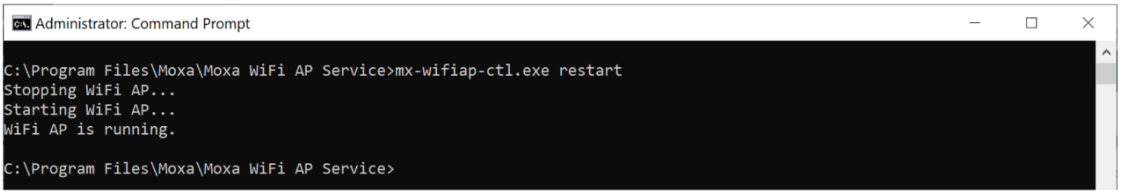

- **stop**

  Stop the WiFi AP and all the connection will be disconnected.



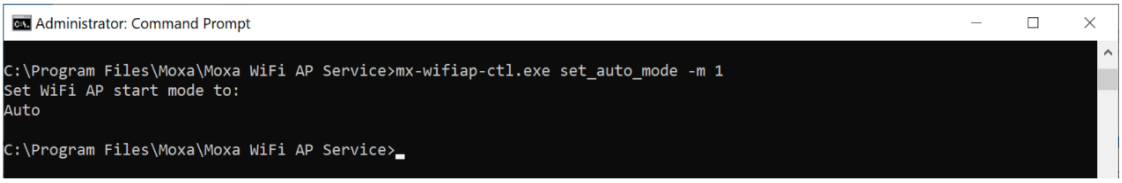

- **restart**

  Stop WiFi AP and then restart the WiFi AP.



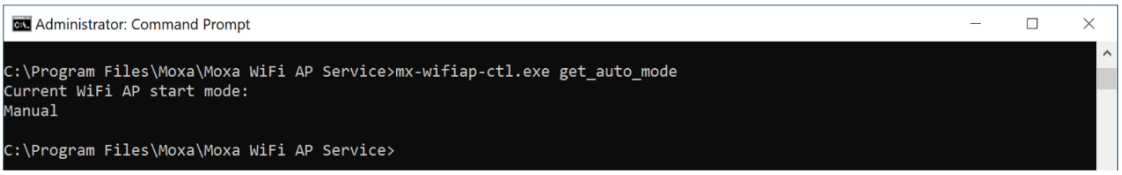

- **set_auto_mode -m[mode]**

  Set the WiFi AP start automatically when the system is turned on. (1:auto, 0:manual)



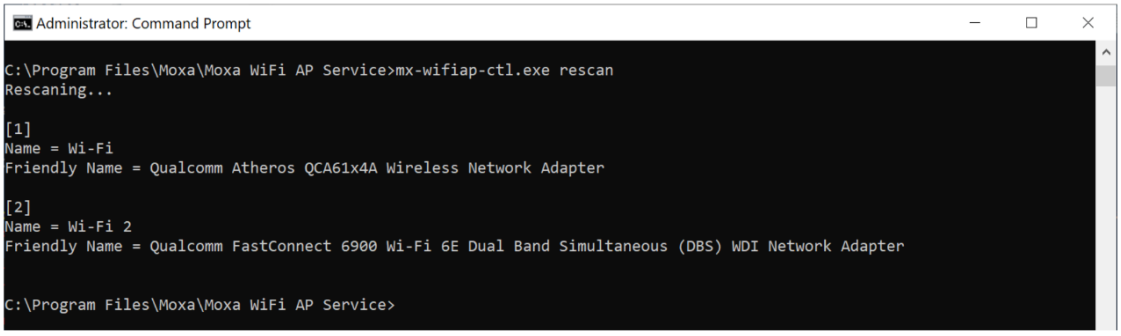

- **get_auto_mode**

  Shows the current WiFi AP start mode.



- **rescan**

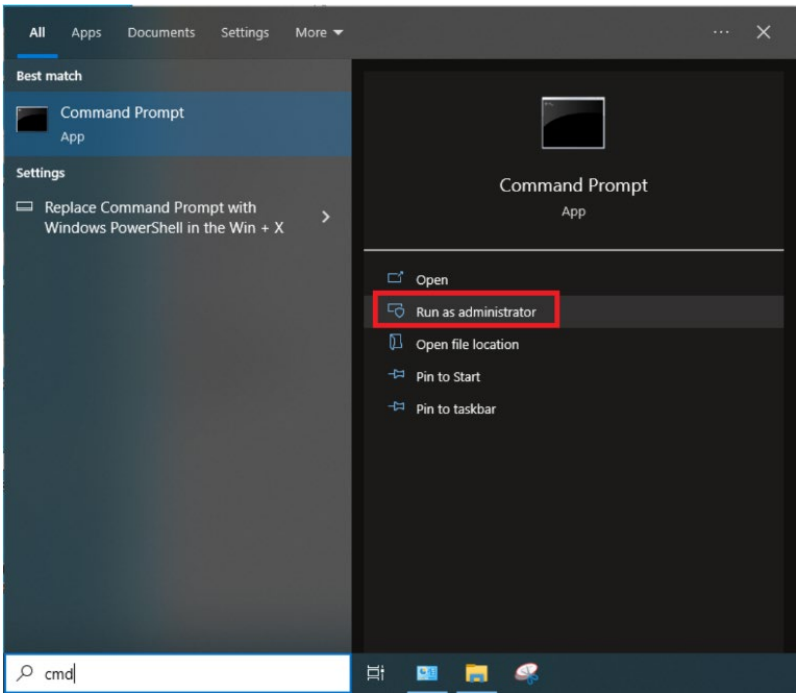
  Rescan the WiFi module on the system, it's for the case when the WiFi module or WiFi adapter is changed.

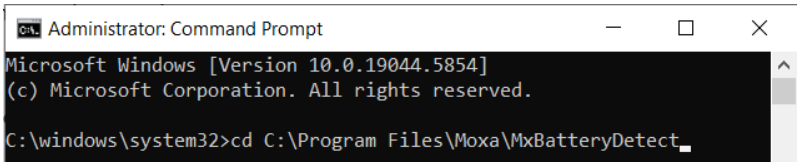



# Moxa Battery Detect

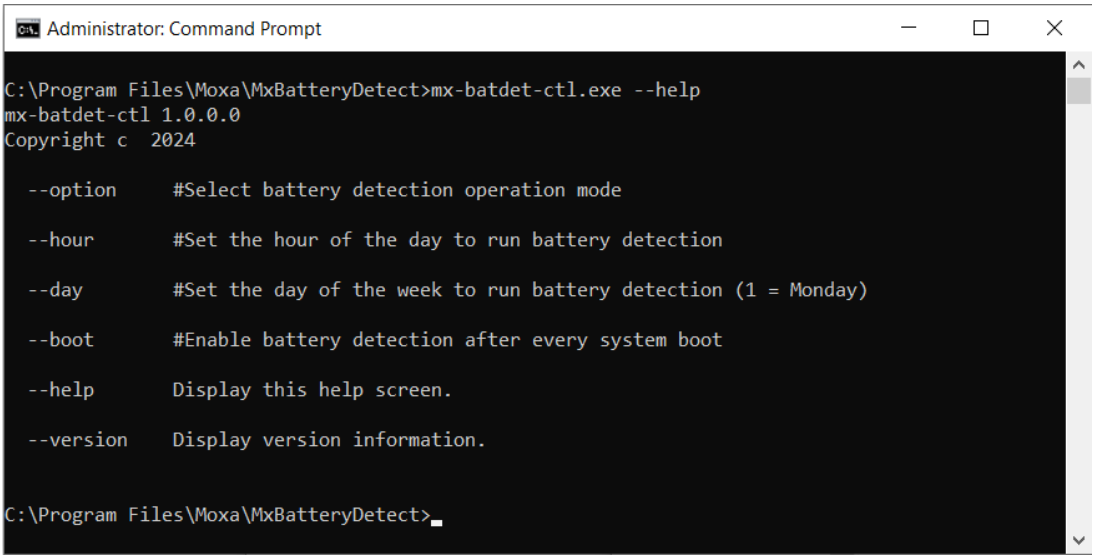
This tool is used to check whether the RTC battery voltage has dropped below 2.4 volts. Users can configure the detection interval, and a notification will alert them to replace the battery when the voltage is too low.

- Open the command prompt as Administrator and type the following to access the folder: **C:\Program Files\Moxa\MxBatteryDetect**.



- Run the **mx-batdet-ctl.exe --help** command to see the instructions on using this utility and follow them to configure the battery detect setting.




- Setting parameters
  - Options:

| Option | Description |
|---|---|
| 1 | Start RTC battery detection |
| 2 | Stop RTC battery detection |
| 3 | Confirm detection settings |
| 4 | Detect RTC battery status once |

  - hour

    Set the hour of the day to perform RTC battery detection (Range: 0-23).
  - day

    Set the day of the week for RTC battery detection (Range: 1-7, where 1 = Monday).
  - boot

    Enable RTC battery detection on every system boot.

    1 → Enable
- Example:
  - Set to detect once every morning at 8 o'clock.

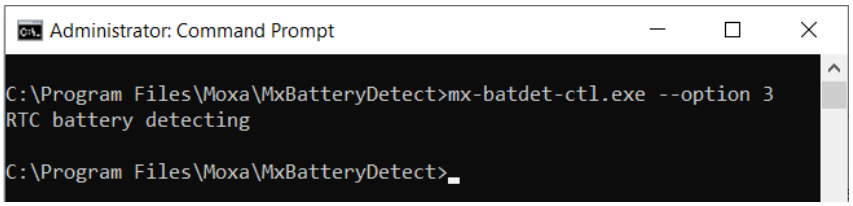
    ```
    mx-batdet-ctl.exe --option 1 --hour 8
    ```
  - Set to detect once every Monday at 8 am.

    ```
    mx-batdet-ctl.exe --option 1 --hour 8 --day 1
    ```
  - Set battery detection to run after every boot.

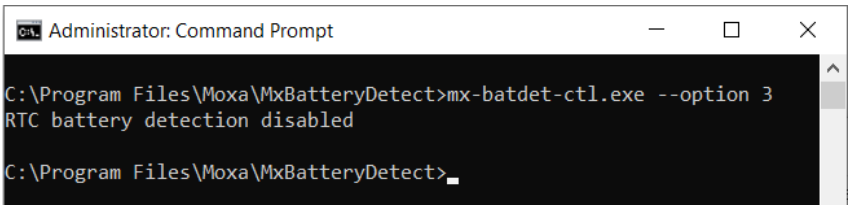
    ```
    mx-batdet-ctl --option 1 --boot 1
    ```
  - Stop detecting RTC battery.

    ```
    mx-batdet-ctl --option 2
    ```
  - Confirm detection setting.

    ```
    mx-batdet-ctl --option 3
    ```
    - If battery detection is set.



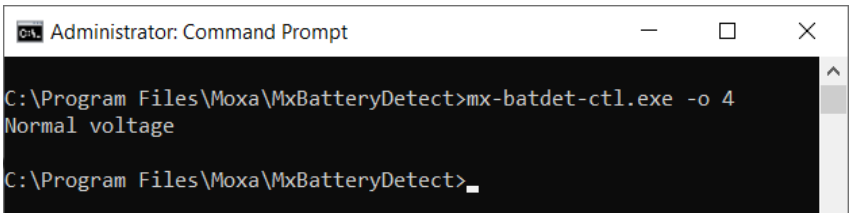
❒ If battery detection is not set.



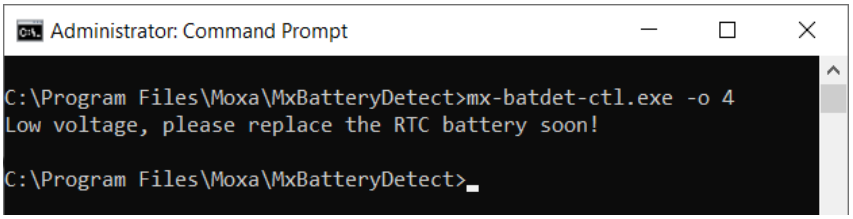

➢ Detect RTC battery status once.

```
mx-batdet-ctl -o 4
```

❒ If the RTC battery voltage is normal (greater than 2.4 volts).

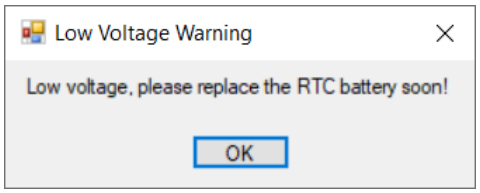



❒ If the RTC battery voltage is too low or if the RTC battery is not installed.




- Detection:

  After configuring the scheduled detection, if the RTC battery voltage is too low or the battery is not installed, a warning message will pop up. The user must click **OK** to close the message window.

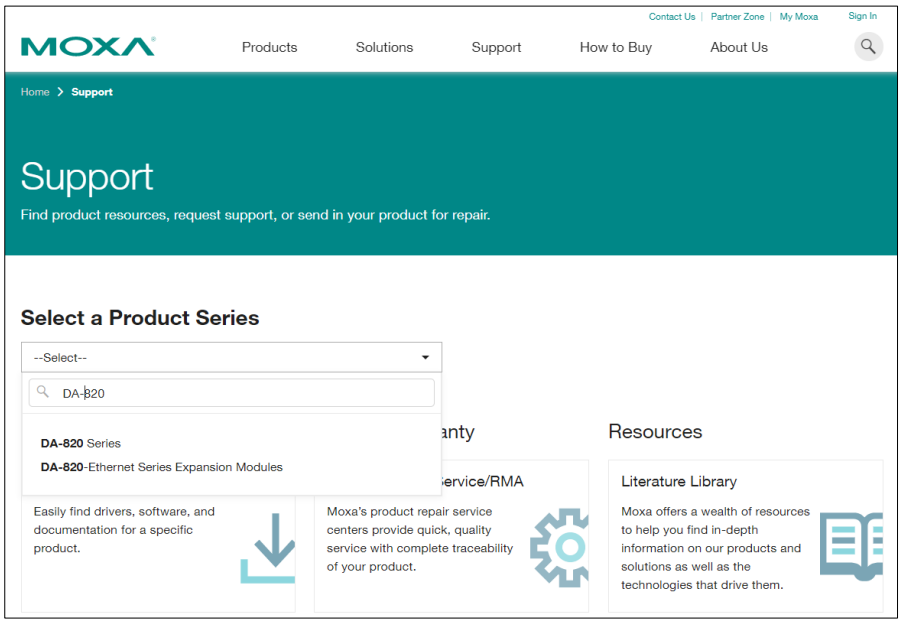


# 10. IO Control API

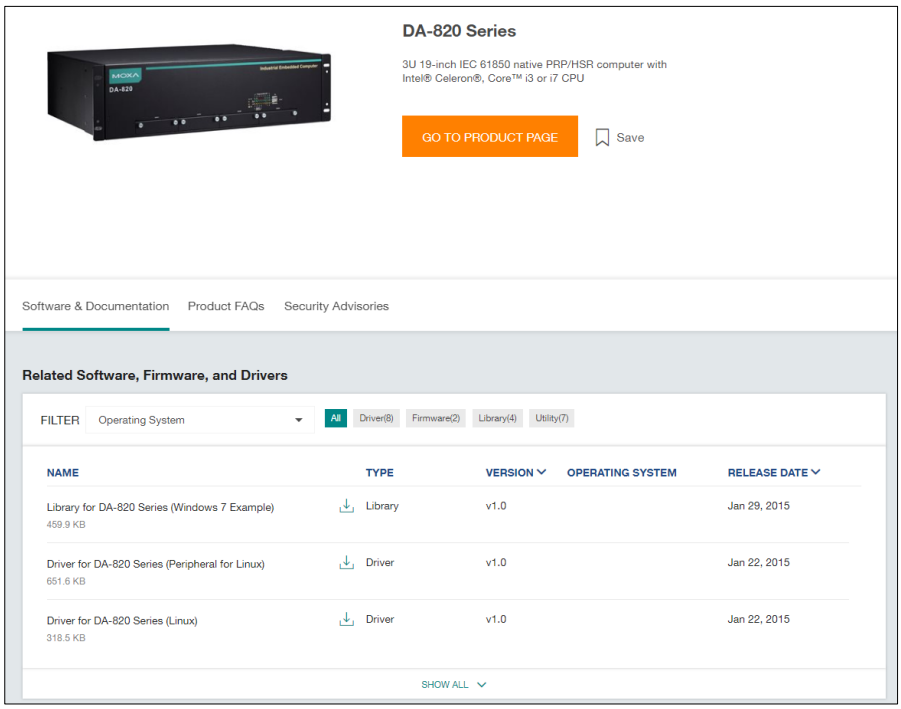
This chapter describes how to use the IO Control API.

# Downloading the API

1. Access the Moxa support page: https://www.moxa.com/en/support.
2. Select the product series (example: DA-820).




3. Download the related files.



# mxdgio

The **mxdgio** library operates on the digital I/Os and consists of the following:

- **GetDinCount**
- **GetOutCount**
- **GetDinStatus**
- **GetDoutStatus**
- **SetDoutStatus**

## GetDinCount

**Syntax**

```
int GetDinCount();
```

**Description**

Get the numbers of a digital input port.

**Parameters**

N/A.

**Return Value**

The numbers of the digital input port.

**Error codes**

The following error codes can be retrieved using the **DIO_STATUS** function.

| Name | Value | Meaning |
|---|---|---|
| LIB_INITIALIZE_FAIL | -1 | The mxdgio library initialization failed. Cannot open json profile. |

**Requirements**

| Name | Items |
|---|---|
| Header | mxdgio.h |
| Library | mxdgio.lib |
| DLL | mxdgio.dll |
| Profile | MxdgioProfile[*ModelName*].json |

# GetDoutCount

**Syntax**

```
int GetDoutCount();
```

**Description**

Get the numbers of a digital output port.

**Parameters**

N/A.

**Return Value**

The numbers of the digital output port.

**Error codes**

The following error codes can be retrieved using the **DIO_STATUS** function.

| Name | Value | Meaning |
|---|---|---|
| LIB_INITIALIZE_FAIL | -1 | The mxdgio library initialization failed. Cannot open json profile. |

**Requirements**

| Name | Items |
|---|---|
| Header | mxdgio.h |
| Library | mxdgio.lib |
| DLL | mxdgio.dll |
| Profile | MxdgioProfile[*ModelName*].json |

# GetDinStatus

**Syntax**

```
int GetDinStatus(int port);
```

**Description**

Gets the status of a digital input port.

**Parameters**

*port*: The index of the digital input port; starts at 0.

**Return Value**

The status of the digital input port; 0 for low and 1 for high.

**Error codes**

The following error codes can be retrieved using the **DIO_STATUS** function.

| Name | Value | Meaning |
|---|---|---|
| LIB_INITIALIZE_FAIL | -1 | The mxdgio library initialization failed. Cannot open json profile. |
| PORT_OUTOF_INDEX | -2 | Target port index is out of range. |

**Requirements**

| Name | Items |
|---|---|
| Header | mxdgio.h |
| Library | mxdgio.lib |
| DLL | mxdgio.dll |
| Profile | MxdgioProfile[*ModelName*].json |

# GetDoutStatus

**Syntax**

```
int GetDoutStatus(int port);
```

**Description**

Gets the status of a digital output port.

**Parameters**

*port*: The index of the digital output port; starts at 0.

**Return Value**

The status of the digital output port; 0 for low and 1 for high.

**Error codes**

The following error codes can be retrieved using the **DIO_STATUS** function.

| Name | Value | Meaning |
|---|---|---|
| LIB_INITIALIZE_FAIL | -1 | The mxdgio library initialization failed. Cannot open json profile. |
| PORT_OUTOF_INDEX | -2 | Target port index is out of range. |

**Requirements**

| Name | Items |
|---|---|
| Header | mxdgio.h |
| Library | mxdgio.lib |
| DLL | mxdgio.dll |
| Profile | MxdgioProfile[*ModelName*].json |

# SetDoutStatus

**Syntax**

```
int SetDoutStatus(int port, int status);
```

**Description**

Sets the status of a digital output port.

**Parameters**

*port*: The index of the digital output port; starts at 0.

*status*: The status of the digital output port; 0 for low and 1 for high.

**Return Value**

Returns the value 0 if the digital output status is successfully set.

**Error codes**

The following error codes can be retrieved using the **DIO_STATUS** function.

| Name | Value | Meaning |
|---|---|---|
| LIB_INITIALIZE_FAIL | -1 | The mxdgio library initialization failed. Cannot open json profile. |
| PORT_OUTOF_INDEX | -2 | Target port index is out of range. |
| SET_STATUS_ERR | -3 | Status setting failed or is defined with a bad format. |

**Requirements**

| Name | Items |
|---|---|
| Header | mxdgio.h |
| Library | mxdgio.lib |
| DLL | mxdgio.dll |
| Profile | MxdgioProfile[*ModelName*].json |

# mxsp

The **mxsp** library operates on the serial port and consists of the following:

- **GetUartCount**
- **GetUartMode**
- **SetUartMode**

## GetUartCount

**Syntax**

```
int GetUartCount();
```

**Description**

Gets the numbers of the UART port.

**Parameters**

N/A

**Return Value**

The numbers of the UART port.

**Error codes**

The following error codes can be retrieved using the **UART_STATUS** function.

| Name | Value | Meaning |
|---|---|---|
| LIB_INITIALIZE_FAIL | -1 | The mxsp library initialization failed. Cannot open json profile. |

**Requirements**

| Name | Items |
|---|---|
| Header | mxsp.h |
| Library | mxsp.lib |
| DLL | mxsp.dll |
| Profile | MxspProfile[*ModelName*].json |

# GetUartMode

**Syntax**

```
int GetUartMode(int port);
```

**Description**

Gets the status of the UART port.

**Parameters**

port: The index of the UART port; starts at 0.

**Return Value**

The mode of a UART interface; 0 for RS-232, 1 for RS-485-2W, 2 for RS-485-4W, and 3 for RS-422.

**Error codes**

The following error codes can be retrieved using the **UART_STATUS** function.

| Name | Value | Meaning |
|---|---|---|
| LIB_INITIALIZE_FAIL | -1 | The mxsp library initialization failed. Cannot open json profile. |
| PORT_OUTOF_INDEX | -2 | Target port index is out of range. |

**Requirements**

| Name | Items |
|---|---|
| Header | mxsp.h |
| Library | mxsp.lib |
| DLL | mxsp.dll |
| Profile | MxspProfile[*ModelName*].json |

# SetUartMode

**Syntax**

```
int SetUartMode(int port, int mode);
```

**Description**

Sets the status of the UART port.

**Parameters**

*port*: The index of the UART port; starts at 0.

*mode*: The mode of a UART interface; 0 for RS-232, 1 for RS-485-2W, 2 for RS-485-4W, and 3 for RS-422.

**Return Value**

Returns 0 if the UART mode is successfully set.

**Error codes**

The following error codes can be retrieved using the **UART_STATUS** function.

| Name | Value | Meaning |
|---|---|---|
| LIB_INITIALIZE_FAIL | -1 | The mxsp library initialization failed. Cannot open json profile. |
| PORT_OUTOF_INDEX | -2 | Target port index is out of range. |
| SET_STATUS_ERR | -3 | Status setting failed or is defined with a bad format. |
| NOT_SUPPORT_MODE | -4 | Target mode is not supported for this port. |

**Requirements**

| Name | Items |
|---|---|
| Header | mxsp.h |
| Library | mxsp.lib |
| DLL | mxsp.dll |
| Profile | MxspProfile[*ModelName*].json |

# mxled

The **mxled** library operates on the relay output and consists of the following:

- **GetLedData**
- **SetLedData**

## GetLedData

**Syntax**

```
int GetLedData(int port);
```

**Description**

Gets the status of the LED port.

**Parameters**

*port*: The index of the LED port; starts at 0.

**Return Value**

The status of an LED port; 0 for OFF, 1 for ON.

**Error codes**

The following error codes can be retrieved by the **LED_STATUS** function.

| Name | Value | Meaning |
|---|---|---|
| LIB_INITIALIZE_FAIL | -1 | The mxled library initialization failed. Cannot open json profile. |
| PORT_OUTOF_INDEX | -2 | Target port index is out of range. |

**Requirements**

| Name | Items |
|---|---|
| Header | mxled.h |
| Library | mxled.lib |
| DLL | mxled.dll |
| Profile | MxledProfile[*ModelName*].json |

## SetLedData

**Syntax**

```
int SetLedData(int port, int status);
```

**Description**

Sets the status of the LED port.

**Parameters**

*port*: The index of the LED port; starts at 0.

*status*: The status of the LED; 0 for OFF, 1 for ON, and 2 for blinking.

**Return Value**

Returns 0 if the LED status is set successfully.

**Error codes**

The following error codes can be retrieved by the **LED_STATUS** function.

| Name | Value | Meaning |
|---|---|---|
| LIB_INITIALIZE_FAIL | -1 | The mxled library initialization failed. Cannot open json profile. |
| PORT_OUTOF_INDEX | -2 | Target port index is out of range. |
| SET_STATUS_ERR | -3 | Status setting failed or is defined with a bad format. |

**Requirements**

| Name | Items |
|---|---|
| Header | mxled.h |
| Library | mxled.lib |
| DLL | mxled.dll |
| Profile | MxledProfile[*ModelName*].json |

# mxwdg

The **mxwdg** library operates on the watchdog and consists of the following:

- **mxwdg_open**
- **mxwdg_refresh**
- **mxwdg_close**

## mxwdg_open

**Syntax**

```
PVOID mxwdg_open(unsigned long time);
```

**Description**

Initializes the watchdog timer.

**Parameters**

*time*: The interval at which the watchdog timer is refreshed; the unit is second.

**Return Value**

Returns the pointer to the watchdog handle; returns -1 on failure to initialize the watchdog timer.

**Requirements**

| Name | Items |
|---|---|
| Header | mxwdg.h |
| Library | mxwdg.lib |
| DLL | mxwdg.dll |

# mxwdg_refresh

**Syntax**

```
int mxwdg_refresh(PVOID fd);
```

**Description**

Refreshes the watchdog timer.

**Parameters**

*fd*: The handle of the watchdog timer.

**Return Value**

Returns 0 on success; otherwise, the function has failed.

**Requirements**

| Name | Items |
|---|---|
| Header | mxwdg.h |
| Library | mxwdg.lib |
| DLL | mxwdg.dll |

# mxwdg_close

**Syntax**

```
void mxwdg_close(PVOID fd);
```

**Description**

Disables the watchdog timer.

**Parameters**

*fd*: The handle of the watchdog timer.

**Return Value**

This function does not return a value.

**Requirements**

| Name | Items |
|---|---|
| Header | mxwdg.h |
| Library | mxwdg.lib |
| DLL | mxwdg.dll |

# mxpcie

The **mxpcie** library operates on the power of PCIE slot and consists of the following:

- **GetPCIESlotStatus**
- **SetPCIESlotStatus**
- **SetPCIESlotStatusWithReset**

## GetPCIESlotStatus

**Syntax**

```
int GetPCIESlotStatus(int port);
```

**Description**

Gets the PCIE slot power status.

**Parameters**

*port*: The index of the PCIE slot; starts at 0.

**Return Value**

The status of a PCIE slot power; 0 for OFF, 1 for ON.

**Error codes**

The following error codes can be retrieved using the **PCIE_STATUS** function.

| Name | Value | Meaning |
|---|---|---|
| LIB_INITIALIZE_FAIL | -1 | The mxpcie library initialization failed. Cannot open json profile. |
| PORT_OUTOF_INDEX | -2 | Target port index is out of range. |

**Requirements**

| Name | Items |
|---|---|
| Header | mxpcie.h |
| Library | mxpcie.lib |
| DLL | mxpcie.dll |
| Profile | MxpcieProfile[*ModelName*].json |

# SetPCIESlotStatus

**Syntax**

```
int SetPCIESlotStatus(int port, int status);
```

**Description**

Sets the PCIE slot power status.

**Parameters**

*port*: The index of the PCIE slot; starts at 0.

*status*: The status of the PCIE slot power; 0 for OFF, 1 for ON.

**Return Value**

Returns 0 if the PCIE slot power is successfully set.

**Error codes**

The following error codes can be retrieved using the **PCIE_STATUS** function.

| Name | Value | Meaning |
|---|---|---|
| LIB_INITIALIZE_FAIL | -1 | The mxpcie library initialization failed. Cannot open json profile. |
| PORT_OUTOF_INDEX | -2 | Target port index is out of range. |
| SET_STATUS_ERR | -3 | Status setting failed or is defined with a bad format. |

**Requirements**

| Name | Items |
|---|---|
| Header | mxpcie.h |
| Library | mxpcie.lib |
| DLL | mxpcie.dll |
| Profile | MxpcieProfile[*ModelName*].json |

# SetPCIESlotStatusWithReset

**Syntax**

```
int SetPCIESlotStatusWithReset(int port, int status, int time);
```

**Description**

Sets the PCIE slot power status and PCIE slot reset pin turn ON and OFF.

**Parameters**

*port*: The index of the PCIE slot; starts at 0.

*status*: The status of the PCIE slot power and PICE reset pin; 0 for OFF, 1 for ON.

*time*: The delay time between PCIE slot reset pin turn ON and OFF.

**Return Value**

Returns 0 if the PCIE slot power and PCIE reset pin are successfully set.

**Error codes**

The following error codes can be retrieved using the **PCIE_STATUS** function.

| Name | Value | Meaning |
|---|---|---|
| LIB_INITIALIZE_FAIL | -1 | The mxpcie library initialization failed. Cannot open json profile. |
| PORT_OUTOF_INDEX | -2 | Target port index is out of range. |
| SET_STATUS_ERR | -3 | Status setting failed or is defined with a bad format. |

**Requirements**

| Name | Items |
|---|---|
| Header | mxpcie.h |
| Library | mxpcie.lib |
| DLL | mxpcie.dll |
| Profile | MxpcieProfile[*ModelName*].json |

# mxpciereset

The **mxpciereset** library operates on the PCIE reset pin status and consists of the following:

- **GetRESETSlotStatus**
- **SetRESETSlotStatus**

## GetRESETSlotStatus

**Syntax**

```
int GetRESETSlotStatus (int port);
```

**Description**

Gets the PCIE slot reset pin status.

**Parameters**

*port*: The index of the PCIE slot; starts at 0.

**Return Value**

The status of a PCIE slot reset pin; 0 for OFF, 1 for ON.

**Error codes**

The following error codes can be retrieved using the **RESET_STATUS** function.

| Name | Value | Meaning |
|---|---|---|
| LIB_INITIALIZE_FAIL | -1 | The mxpciereset library initialization failed. Cannot open json profile. |
| PORT_OUTOF_INDEX | -2 | Target port index is out of range. |

**Requirements**

| Name | Items |
|---|---|
| Header | mxpciereset.h |
| Library | mxpciereset.lib |
| DLL | mxpciereset.dll |
| Profile | MxpcieresetProfile[*ModelName*].json |

# SetRESETSlotStatus

**Syntax**

```
int SetRESETSlotStatus (int port, int time);
```

**Description**

Sets the PCIE slot reset pin ON/OFF cycle and delay time.

**Parameters**

*port*: The index of the PCIE slot; starts at 0.

*time*: The delay time between PCIE slot reset pin turn ON and OFF.

**Return Value**

Returns 0 if the PCIE slot reset pin is successfully set.

**Error codes**

The following error codes can be retrieved using the **RESET_STATUS** function.

| Name | Value | Meaning |
|---|---|---|
| LIB_INITIALIZE_FAIL | -1 | The mxpciereset library initialization failed. Cannot open json profile. |
| PORT_OUTOF_INDEX | -2 | Target port index is out of range. |
| SET_STATUS_ERR | -3 | Status setting failed or is defined with a bad format. |

**Requirements**

| Name | Items |
|---|---|
| Header | mxpciereset.h |
| Library | mxpciereset.lib |
| DLL | mxpciereset.dll |
| Profile | MxpcieresetProfile[*ModelName*].json |

# mxm2bpower

The **mxm2bpower** library operates on the power of M.2 B Key slot and consists of the following:

- **GetPowerStatus**
- **SetPowerStatus**

## GetPowerStatus

**Syntax**

```
int GetPowerStatus (int port);
```

**Description**

Gets the M.2 B Key slot power status.

**Parameters**

*port*: The index of the M.2 B Key slot; starts at 0.

**Return Value**

The status of a M.2 B Key slot power; 0 for OFF, 1 for ON.

**Error codes**

The following error codes can be retrieved by the **M2BPOWER_STATUS** function.

| Name | Value | Meaning |
|---|---|---|
| LIB_INITIALIZE_FAIL | -1 | The mxm2bpower library initialization failed. Cannot open json profile. |
| PORT_OUTOF_INDEX | -2 | Target port index is out of range. |

**Requirements**

| Name | Items |
|---|---|
| Header | m2bpower.h |
| Library | m2bpower.lib |
| DLL | m2bpower.dll |
| Profile | Mxm2bpowerProfile[*ModelName*].json |

# SetPowerStatus

**Syntax**

```
int SetPowerStatus (int port, int status);
```

**Description**

Sets the M.2 B Key slot power status.

**Parameters**

*port*: The index of the M.2 B Key slot; starts at 0.

*status*: The status of the M.2 B Key slot power; 0 for OFF, 1 for ON.

**Return Value**

Returns 0 if the M.2 B Key slot power is successfully set.

**Error codes**

The following error codes can be retrieved by the **M2BPOWER_STATUS** function.

| Name | Value | Meaning |
|---|---|---|
| LIB_INITIALIZE_FAIL | -1 | The mxm2bpower library initialization failed. Cannot open json profile. |
| PORT_OUTOF_INDEX | -2 | Target port index is out of range. |

**Requirements**

| Name | Items |
|---|---|
| Header | m2bpower.h |
| Library | m2bpower.lib |
| DLL | m2bpower.dll |
| Profile | Mxm2bpowerProfile[*ModelName*].json |

# MxLBPLibrary

The **MxLBPLibrary** library operates LAN Bypass modes and consists of the following:

- RelayMode
  - GetRelayMode
  - SetRelayMode
- WDTResetMode
  - GetWDTResetMode
  - SetWDTResetMode
- WDTRelayMode
  - GetWDTRelayMode
  - SetWDTRelayMode
- AppWDTMode
  - GetAppWDTMode
  - SetAppWDTMode
- AppWDTKick
  - SendAppWDTKick
- AppWDTResetMode
  - GetAppWDTResetMode
  - SetAppWDTResetMode
- AppWDTRelayMode
  - GetAppWDTRelayMode
  - SetAppWDTRelayMode
- PowerOffRelayMode
  - GetPowerOffRelayMode
  - SetPowerOffRelayMode
- MCUFirmwareVersion
  - GetMCUFirmwareVersion

# Relay Mode

Relay Mode indicates using relay to control the connection mode of the target LANs. The following topics are covered in this chapter:

- GetRelayMode
- SetRelayMode

## GetRelayMode

**Syntax**

```
int GetRelayMode(out int relayMode);
```

**Description**

Gets relay mode from MCU.

**Parameters**

*relayMode* (call by reference variable)

- The mode of relay

| Index | Mode | Description |
|---|---|---|
| 0 | Connect | Target LANs connected. |
| 1 | Disconnect | Target LANs disconnected. |
| 2 | Bypass | Target LANs are physically connected directly without passing through the network device. |

**Return Value**

If the operation completes successfully, the return value is zero. If the operation fails, the return value is one.

**Requirements**

| Name | Items |
|---|---|
| DLL | MxLBPLibrary.dll |

## SetRelayMode

**Syntax**

```
int SetRelayMode(int relayMode);
```

**Description**

Sets relay mode.

**Parameters**

*relayMode* (call by reference variable)

- The mode of relay

| Index | Mode | Description |
|---|---|---|
| 0 | Connect | Target LANs connected. |
| 1 | Disconnect | Target LANs disconnected. |
| 2 | Bypass | Target LANs are physically connected directly without passing through the network device. |

**Return Value**

If the operation completes successfully, the return value is zero. If the operation fails, the return value is one.

**Requirements**

| Name | Items |
|---|---|
| DLL | MxLBPLibrary.dll |

# WDTResetMode

Watchdog Reset Mode indicates whether the system needs to be reset after the watchdog is triggered. The following topics are covered in this chapter:

- GetWDTResetMode
- SetWDTResetMode

## GetWDTResetMode

**Syntax**

```
int GetWDTResetMode(out int WDTResetMode);
```

**Description**

Gets watchdog reset mode from MCU.

**Parameters**

*WDTResetMode* (call by reference variable)

- The mode of watchdog reset

| Index | Mode | Description |
|---|---|---|
| 0 | Not Reset | The system will not reset when the watchdog is triggered. |
| 1 | Reset | The system will reset when the watchdog is triggered. |

**Return Value**

If the operation completes successfully, the return value is zero. If the operation fails, the return value is one.

**Requirements**

| Name | Items |
|---|---|
| DLL | MxLBPLibrary.dll |

## SetWDTResetMode

**Syntax**

```
int SetWDTResetMode(int WDTResetMode);
```

**Description**

Sets watchdog reset mode.

**Parameters**

*WDTResetMode* (call by reference variable)

- The mode of watchdog reset

| Index | Mode | Description |
|---|---|---|
| 0 | Not Reset | The system will not reset when the watchdog is triggered. |
| 1 | Reset | The system will reset when the watchdog is triggered. |

**Return Value**

If the operation completes successfully, the return value is zero. If the operation fails, the return value is one.

**Requirements**

| Name | Items |
|---|---|
| DLL | MxLBPLibrary.dll |

# WDTRelayMode

Watchdog Relay Mode indicates which relay mode to switch to after the watchdog is triggered. The following topics are covered in this chapter:

- GetWDTRelayMode
- SetWDTRelayMode

## GetWDTRelayMode

**Syntax**

```
int GetWDTRelayMode(out int WDTrelayMode);
```

**Description**

Gets watchdog relay mode from MCU.

**Parameters**

*WDTrelayMode* (call by reference variable)

- The mode of watchdog relay

| Index | Mode | Description |
|---|---|---|
| 0 | Connect | Target LANs connected. |
| 1 | Disconnect | Target LANs disconnected. |
| 2 | Bypass | Target LANs are physically connected directly without passing through the network device. |

**Return Value**

If the operation completes successfully, the return value is zero. If the operation fails, the return value is one.

**Requirements**

| Name | Items |
|---|---|
| DLL | MxLBPLibrary.dll |

## SetWDTRelayMode

**Syntax**

```
int SetWDTRelayMode(int WDTrelayMode);
```

**Description**

Sets watchdog relay mode.

**Parameters**

*WDTrelayMode* (call by reference variable)

- The mode of watchdog relay

| Index | Mode | Description |
|---|---|---|
| 0 | Connect | Target LANs connected. |
| 1 | Disconnect | Target LANs disconnected. |
| 2 | Bypass | Target LANs are physically connected directly without passing through the network device. |

**Return Value**

If the operation completes successfully, the return value is zero. If the operation fails, the return value is one.

**Requirements**

| Name | Items |
|---|---|
| DLL | MxLBPLibrary.dll |

# AppWDTMode

App Watchdog Mode determines whether to enable the APP Watchdog function. The following topics are covered in this chapter:

- GetAppWDTMode
- SetAppWDTMode

## GetAppWDTMode

**Syntax**

```
int GetAppWDTMode(out int appWDTMode, out int timeout);
```

**Description**

Gets app watchdog mode from MCU.

**Parameters**

- *appWDTMode* (call by reference variable)
  - The mode of app watchdog

| Index | Mode | Description |
|---|---|---|
| 0 | Disable | Disable the app watchdog function. |
| 1 | Enable | Enable the app watchdog function. |

- timeout
  - Timeout (sec), (Notice: timeout must be between 5 and 15.)

**Return Value**

If the operation completes successfully, the return value is zero. If the operation fails, the return value is one.

**Requirements**

| Name | Items |
|---|---|
| DLL | MxLBPLibrary.dll |

## SetAppWDTMode

**Syntax**

```
int SetAppWDTMode(int appWDTMode, int timeout);
```

**Description**

Sets app watchdog mode from MCU.

**Parameters**

- *appWDTMode* (call by reference variable)
  - The mode of app watchdog

| Index | Mode | Description |
|---|---|---|
| 0 | Disable | Disable the app watchdog function. |
| 1 | Enable | Enable the app watchdog function. |

- timeout
  - Timeout (sec), (Notice: timeout must be between 5 and 15.)

**Return Value**

If the operation completes successfully, the return value is zero. If the operation fails, the return value is one.

**Requirements**

| Name | Items |
|---|---|
| DLL | MxLBPLibrary.dll |

# AppWDTKick

App Watchdog Kick indicates which relay mode to switch to after the watchdog is triggered. The following topics are covered in this chapter:

- SendAppWDTKick

## SendAppWDTKick

**Syntax**

```
int SendAppWDTKick();
```

**Description**

Sends the kick of app watchdog.

**Return Value**

If the operation completes successfully, the return value is zero. If the operation fails, the return value is one.

**Requirements**

| Name | Items |
|---|---|
| DLL | MxLBPLibrary.dll |

# AppWDTResetMode

App Watchdog Reset Mode indicates whether the system needs to be reset after the watchdog is triggered. The following topics are covered in this chapter:

- GetAppWDTResetMode
- SetAppWDTResetMode

## GetAppWDTResetMode

**Syntax**

```
int GetAppWDTResetMode(out int appWDTResetMode);
```

**Description**

Gets app watchdog reset mode from MCU.

**Parameters**

*appWDTResetMode* (call by reference variable)

- The mode of app watchdog reset

| Index | Mode | Description |
|---|---|---|
| 0 | Not Reset | The system will not reset when the watchdog is triggered. |
| 1 | Reset | The system will reset when the watchdog is triggered. |

**Return Value**

If the operation completes successfully, the return value is zero. If the operation fails, the return value is one.

**Requirements**

| Name | Items |
|---|---|
| DLL | MxLBPLibrary.dll |

## SetAppWDTResetMode

**Syntax**

```
int SetAppWDTResetMode(int appWDTResetMode);
```

**Description**

Sets app watchdog reset mode.

**Parameters**

*appWDTResetMode* (call by reference variable)

- The mode of app watchdog reset

| Index | Mode | Description |
|---|---|---|
| 0 | Not Reset | The system will not reset when the watchdog is triggered. |
| 1 | Reset | The system will reset when the watchdog is triggered. |

**Return Value**

If the operation completes successfully, the return value is zero. If the operation fails, the return value is one.

**Requirements**

| Name | Items |
|---|---|
| DLL | MxLBPLibrary.dll |

# AppWDTRelayMode

App Watchdog Relay Mode indicates which relay mode to switch to after the watchdog is triggered. The following topics are covered in this chapter:

- GetAppWDTRelayMode
- SetAppWDTRelayMode

## GetAppWDTRelayMode

**Syntax**

```
int GetAppWDTRelayMode(out int appWDTrelayMode);
```

**Description**

Gets app watchdog relay mode from MCU.

**Parameters**

*appWDTrelayMode* (call by reference variable)

- The mode of watchdog relay

| Index | Mode | Description |
|---|---|---|
| 0 | Connect | Target LANs connected. |
| 1 | Disconnect | Target LANs disconnected. |
| 2 | Bypass | Target LANs are physically connected directly without passing through the network device. |

**Return Value**

If the operation completes successfully, the return value is zero. If the operation fails, the return value is one.

**Requirements**

| Name | Items |
|---|---|
| DLL | MxLBPLibrary.dll |

## SetAppWDTRelayMode

**Syntax**

```
int SetAppWDTRelayMode(int appWDTrelayMode);
```

**Description**

Sets app watchdog relay mode.

**Parameters**

*appWDTrelayMode* (call by reference variable)

- The mode of watchdog relay

| Index | Mode | Description |
|---|---|---|
| 0 | Connect | Target LANs connected. |
| 1 | Disconnect | Target LANs disconnected. |
| 2 | Bypass | Target LANs are physically connected directly without passing through the network device. |

**Return Value**

If the operation completes successfully, the return value is zero. If the operation fails, the return value is one.

**Requirements**

| Name | Items |
|---|---|
| DLL | MxLBPLibrary.dll |

# PowerOffRelayMode

Power Off Relay Mode indicates which relay mode to switch to after the system is power off. The following topics are covered in this chapter:

- GetPowerOffRelayMode
- SetPowerOffRelayMode

## GetPowerOffRelayMode

**Syntax**

```
int GetPowerOffRelayMode(out int powerOffRelayMode);
```

**Description**

Gets power off relay mode from MCU.

**Parameters**

*powerOffRelayMode* (call by reference variable)

- The mode of power off relay

| Index | Mode | Description |
|---|---|---|
| 1 | Disconnect | Target LANs disconnected. |
| 2 | Bypass | Target LANs are physically connected directly without passing through the network device. |

**Return Value**

If the operation completes successfully, the return value is zero. If the operation fails, the return value is one.

**Requirements**

| Name | Items |
|---|---|
| DLL | MxLBPLibrary.dll |

## SetPowerOffRelayMode

**Syntax**

```
int SetPowerOffRelayMode(int powerOffRelayMode);
```

**Description**

Sets power off relay mode from MCU.

**Parameters**

*powerOffRelayMode* (call by reference variable)

- The mode of power off relay

| Index | Mode | Description |
|---|---|---|
| 1 | Disconnect | Target LANs disconnected. |
| 2 | Bypass | Target LANs are physically connected directly without passing through the network device. |

**Return Value**

If the operation completes successfully, the return value is zero. If the operation fails, the return value is one.

**Requirements**

| Name | Items |
|---|---|
| DLL | MxLBPLibrary.dll |

# MCUFirmwareVersion

MCU Firmware Version indicates the firmware version of MCU. The following topics are covered in this chapter:

- GetMCUFirmwareVersion

## GetMCUFirmwareVersion

**Syntax**

```
int GetMCUFirmwareVersion(out string version);
```

**Description**

Gets MCU firmware version from MCU.

**Parameters**

*version* (call by reference variable)

- The string of MCU firmware version, e.g., 1.0.0[S00].

**Return Value**

If the operation completes successfully, the return value is zero. If the operation fails, the return value is one.

**Requirements**

| Name | Items |
|---|---|
| DLL | MxLBPLibrary.dll |

# 11. System Backup Restore

This chapter describes the usage of the following for system backup and restoration.

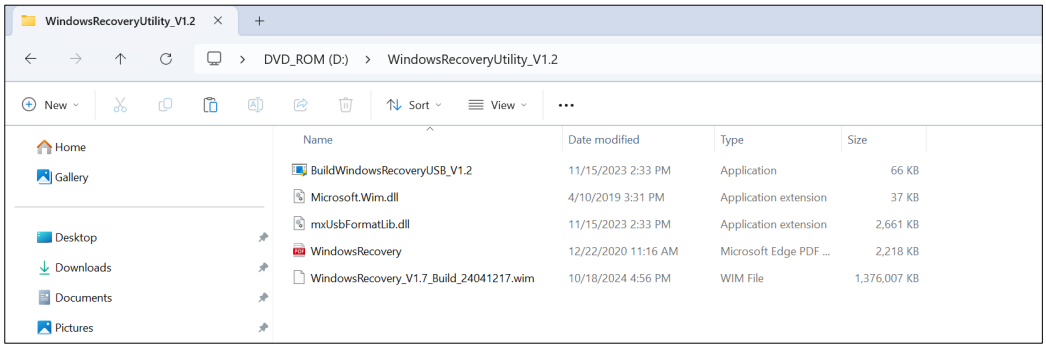
- **WindowsRecovery**

# WindowsRecovery

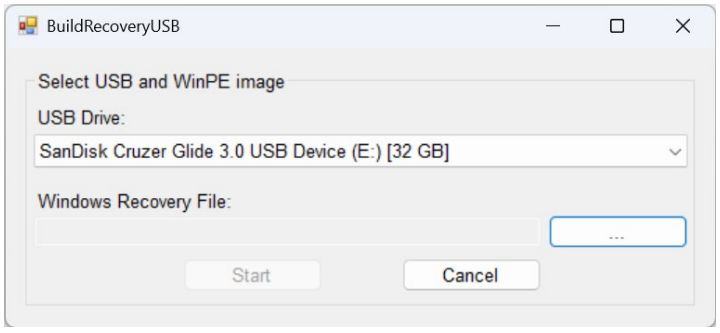
WindowsRecovery is an OS image backup and restore program for system deployment, backup, and recovery. You will first need to create a WindowsRecovery USB disk. This WindowsRecovery disk can only be used to boot a **UEFI BIOS** machine. This chapter describes the setup process of the Windows Recovery function.

## Preparing the USB Device

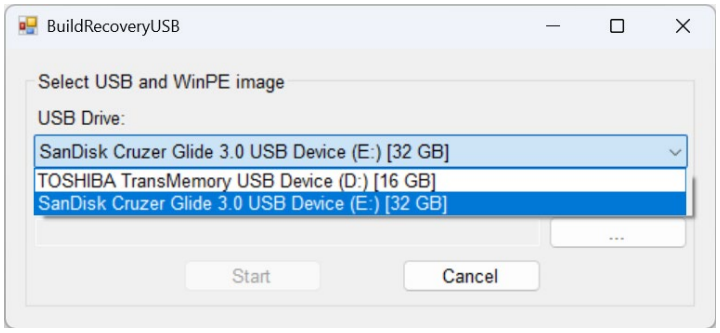
1. Contact a Moxa technical staff and get the required file.



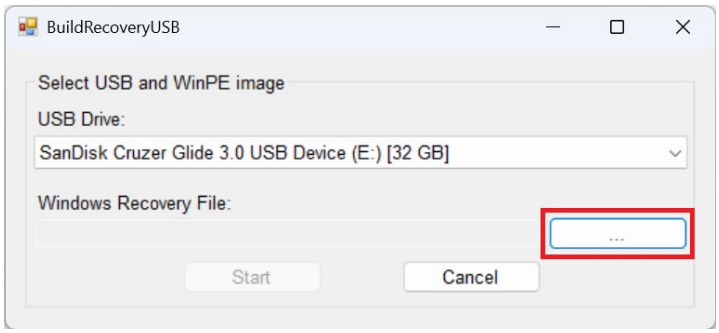

2. Run the **BuildWindowsRecoveryUSB_V1.2.0.exe**.

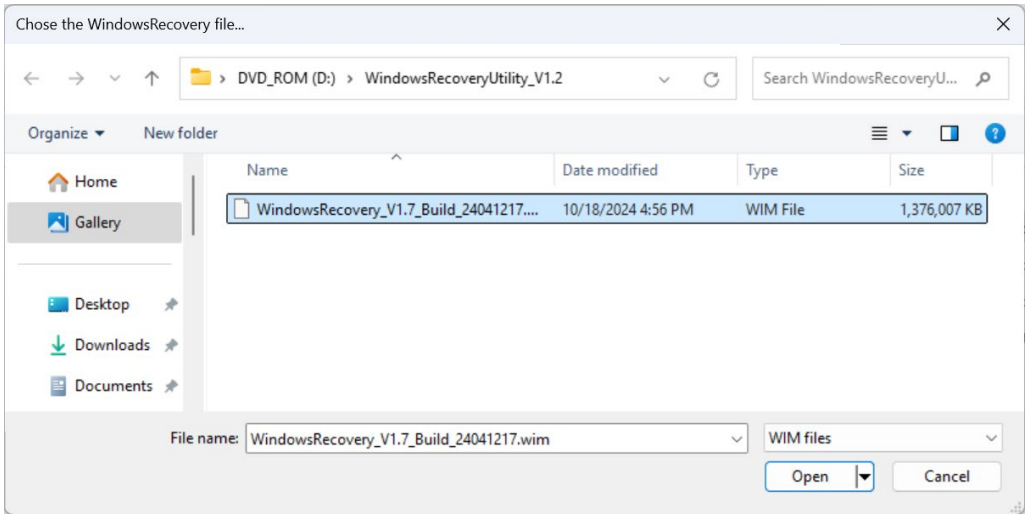



3. Select the USB drive to format.

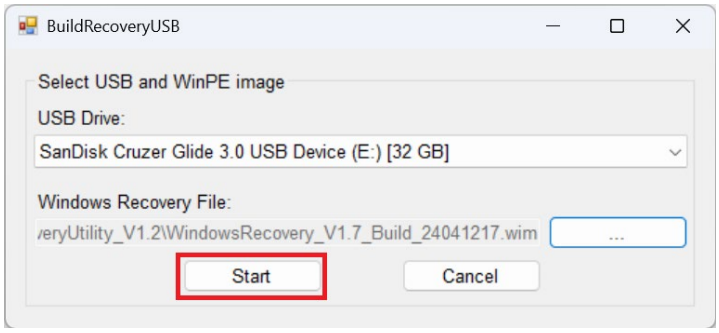


4. Click **...** to select **.wim** file from the folder.

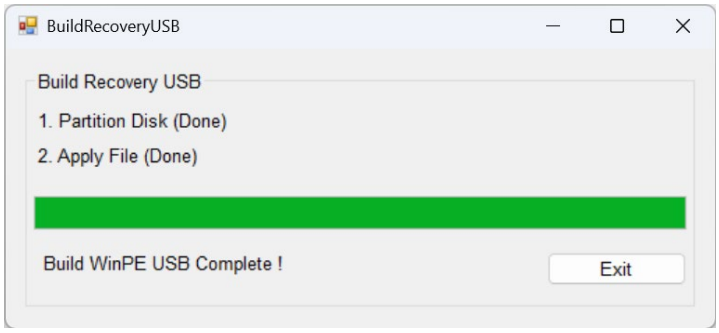


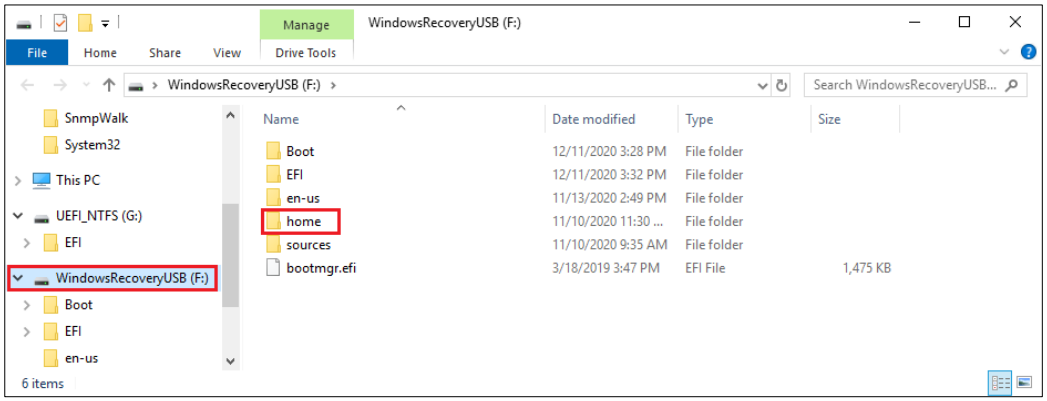

5. Click **Start** and make sure the selected USB can be formatted. Click **Yes** to start creating the recovery USB.

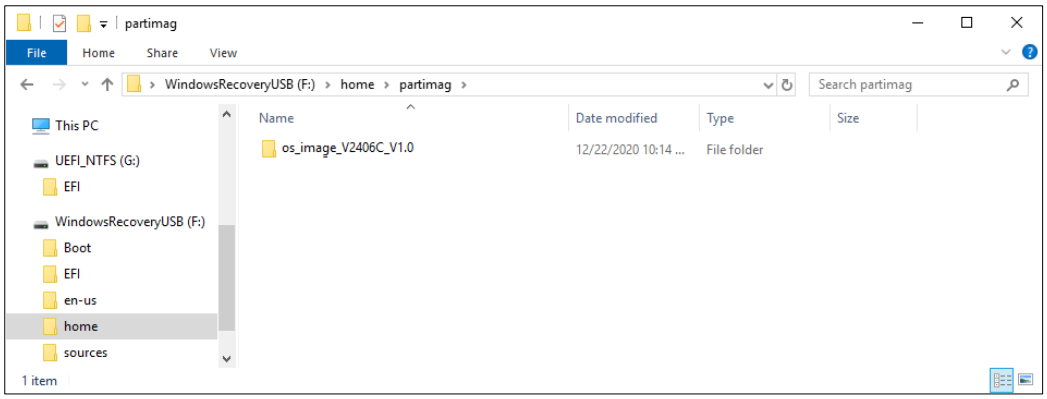



6. Wait for the process to finish. The program will format the USB device and create a UEFI bootable volume and a WinPE volume. You may see additional windows about folder information; do not close these. You can close the windows after the process finishes.



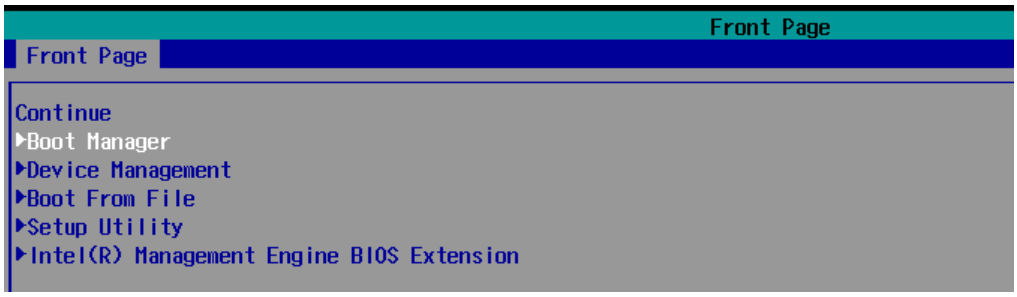
7. To create a recovery USB disk with the Windows 11 image, copy the **os_image_ModelName** directory to the **\home\partimag** folder in the USB drive.




# Booting From the USB Disk

1. Turn on the computer and press **F2** when you hear the beep sound to enter the BIOS setup menu, select **Boot Manager** and press **Enter** to continue.




2. Select the **EFI USB Device** on the computer and press **Enter** to continue to boot from the USB device.

# System Image Backup

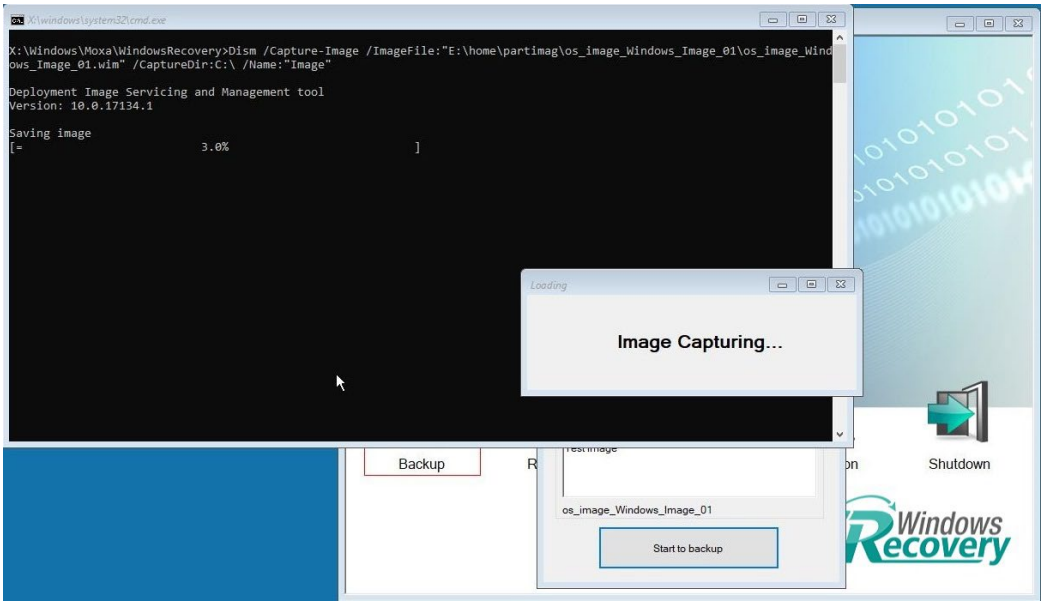
To back up the image from the USB disk, run **Windows Preinstallation Environment(WinPE)** and the **Windows Recovery utility** will display. Follow these steps.

1. Click **Backup**.

2. Select the **Source disk** to backup and **Destination USB** to store the OS image, also give an image name and description. Click **Start to backup**.

3. Click **Yes** to continue.

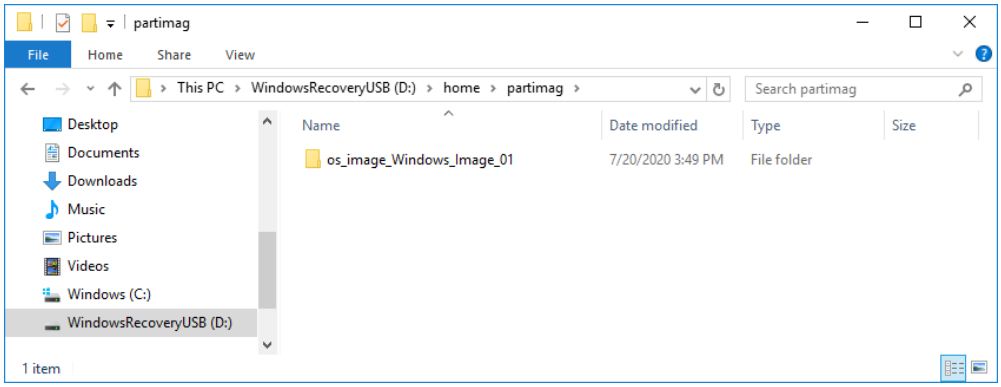
4. Wait for the backup process to complete.



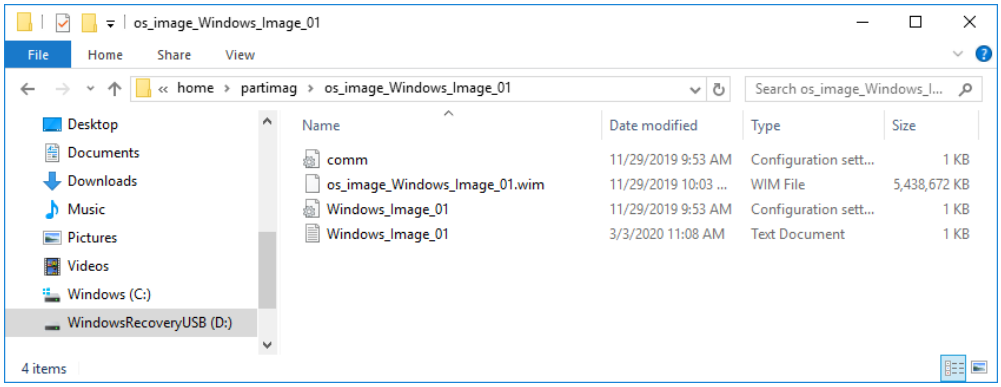
5. When the process is done, click **OK**.

6. Click **OK**, the computer will shut down.

7. The OS image will be saved in USB disk at **home\partimag**.




8. In the **os_image** folder you can view the backup information and the image files.




# Restoring the System From a Backup

To restore the image, run the **Windows Preinstallation Environment(WinPE)** and the **Windows Recovery utility** will display. Follow these steps.

1. Click **Recovery**.

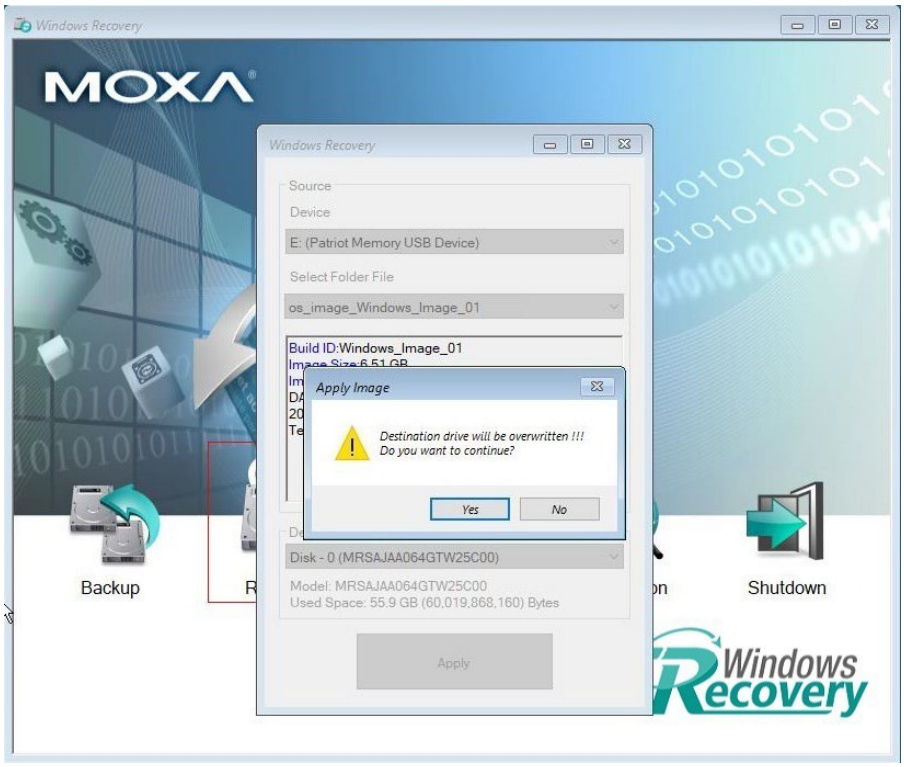
2. Select the **Source USB Device**, **Image Folder File** and check the image information, select the **Destination Drive** to restore. Click **Apply**.

## NOTE

If dual operating systems are required, it is recommended to restore the image to the destination drive with the PCIe interface (if available) first.

3. Click **Yes** to continue the process.

4. Click **Yes** to overwrite the destination drive.




5. Wait for the process to complete.

6. Click **OK**.

## NOTE

When you restart the computer, you will need to wait about 5 minutes for the computer to go through two cycles of the reboot process. The system configuration files will be initiated during the first boot- up process. Do not turn off or shut down the computer while the system is restarting.