# MPC-3000 Series Windows 10 IoT Enterprise LTSC 2021 (21H2) User Manual

**Version 2.0, August 2026**

**www.moxa.com/products**

# MPC-3000 Series Windows 10 IoT Enterprise LTSC 2021 (21H2) User Manual



## Copyright Notice



## Trademarks



## 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

This Windows 10 IoT Enterprise LTSC 2021 (21H2) 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 in this manual.

## 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.

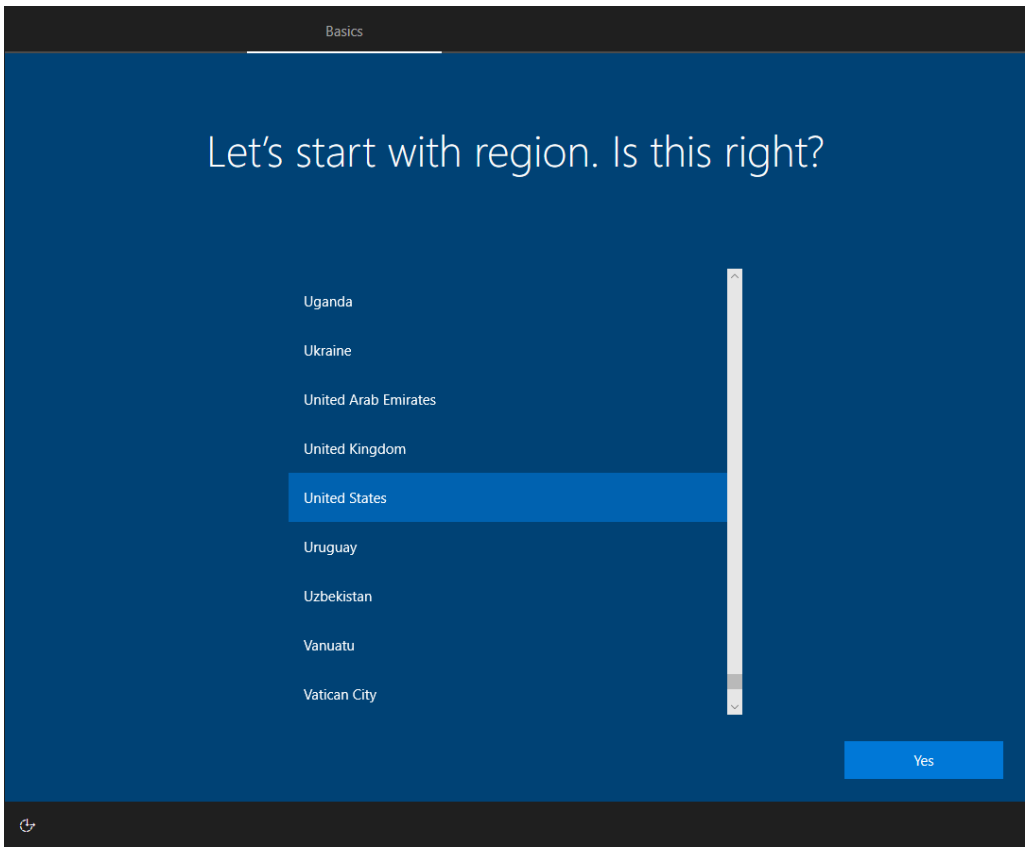
# 2. System Initialization

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.

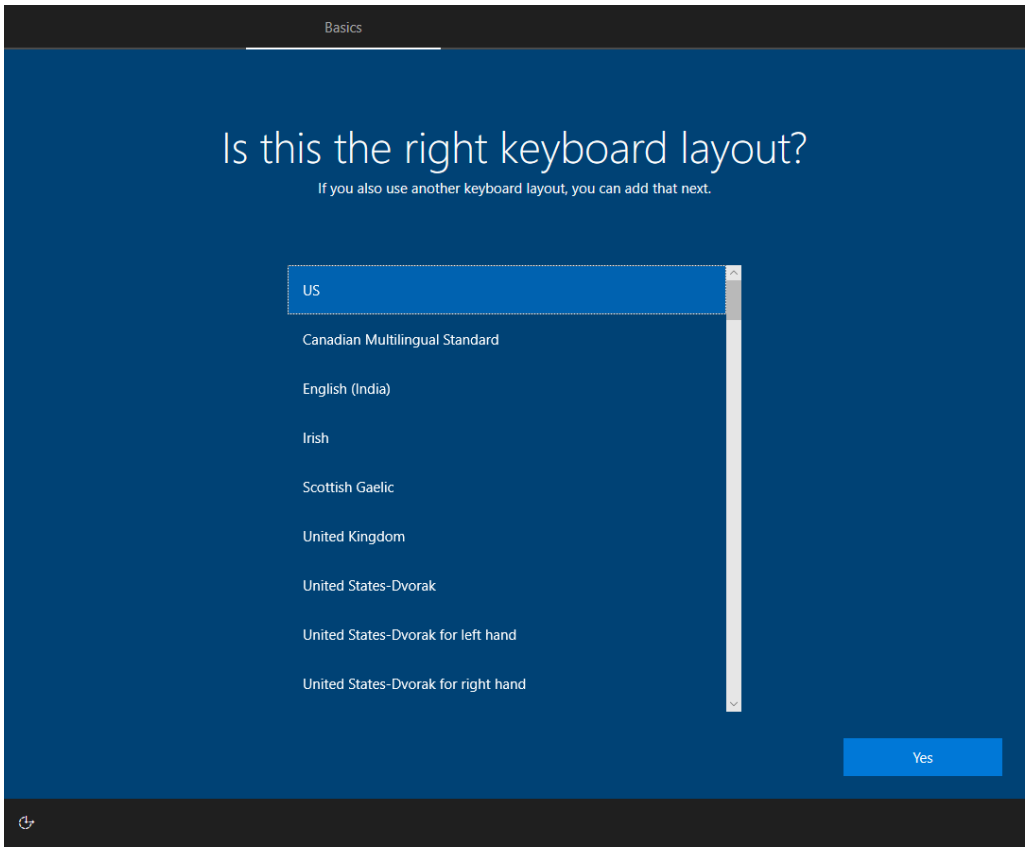
## Initializing User Settings

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

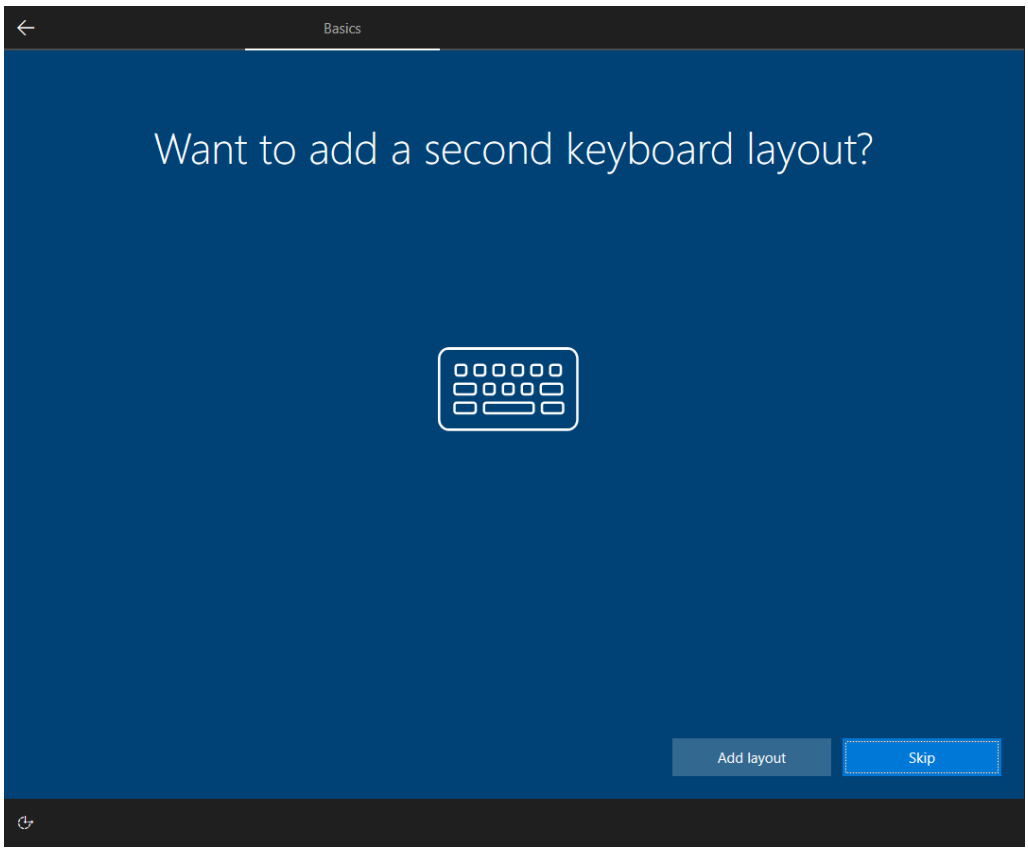
1. Select a region.



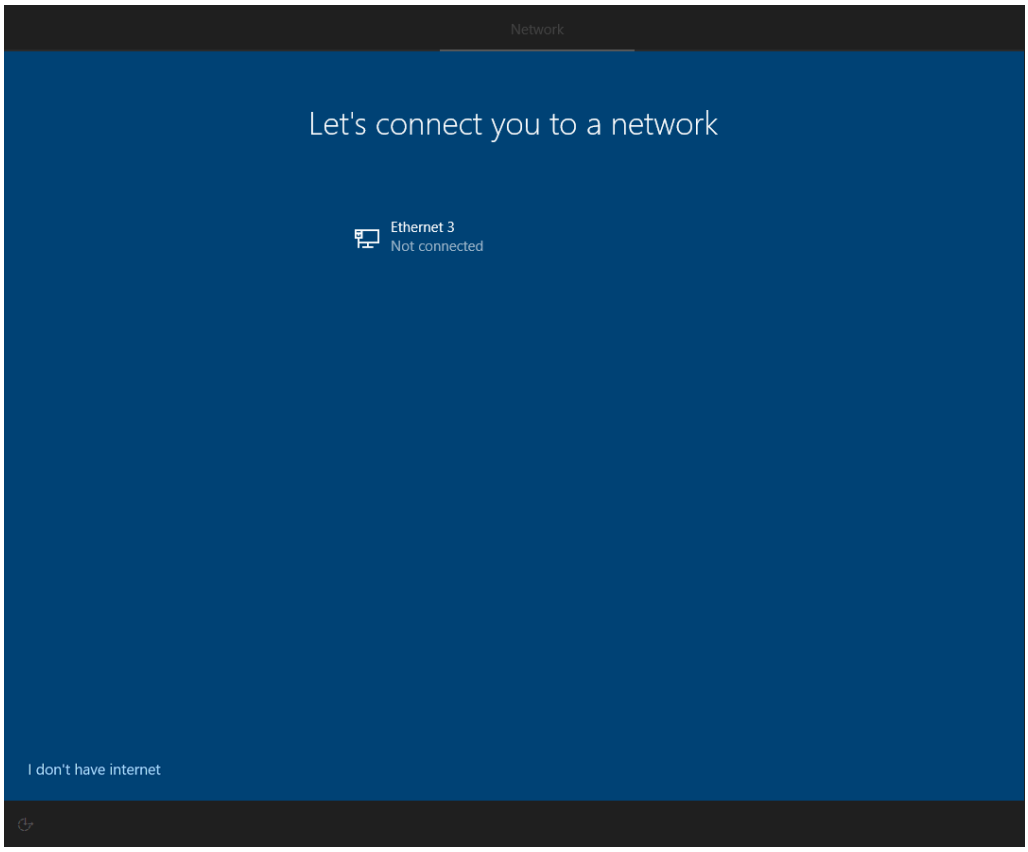
2. Select a keyboard.

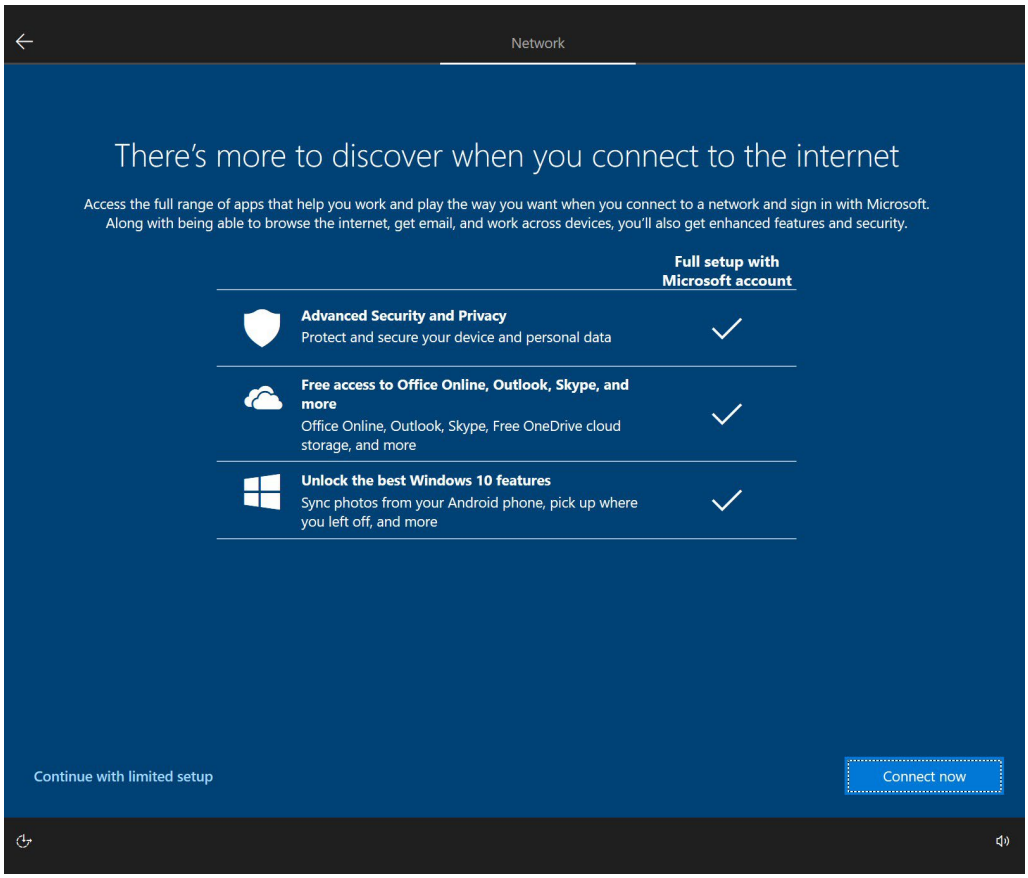



3. Select a second keyboard.

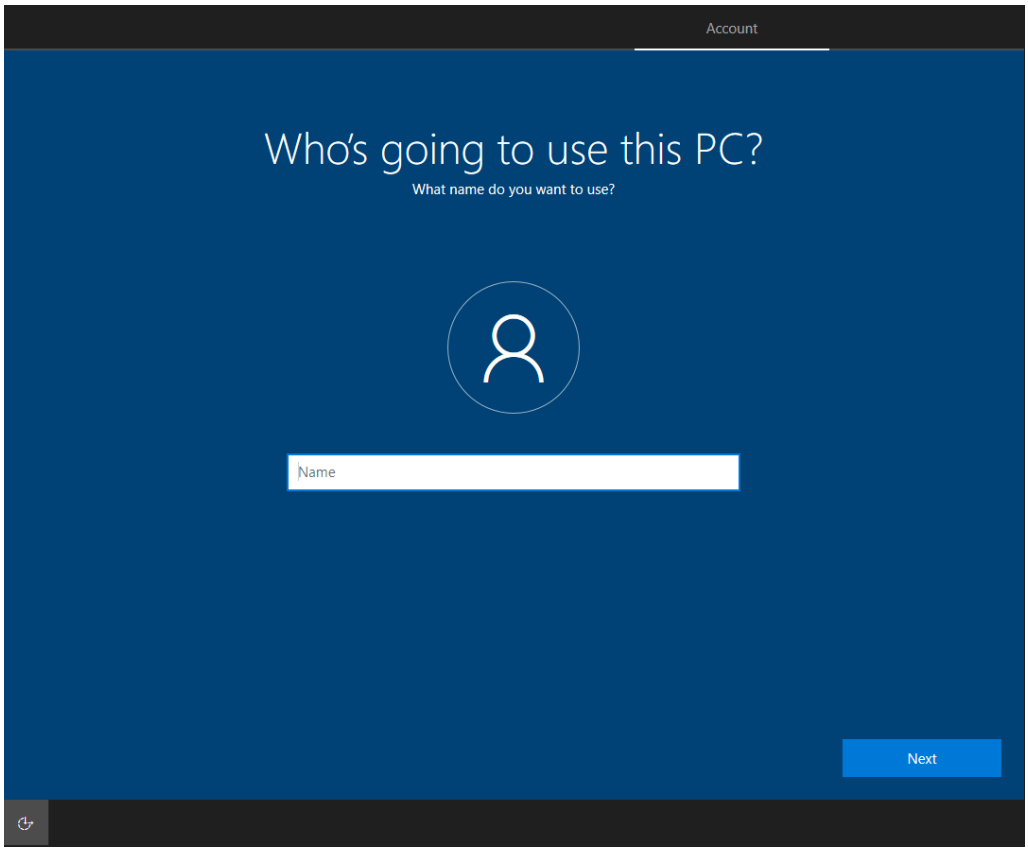


4. Connect to a network or continue with limited setup.

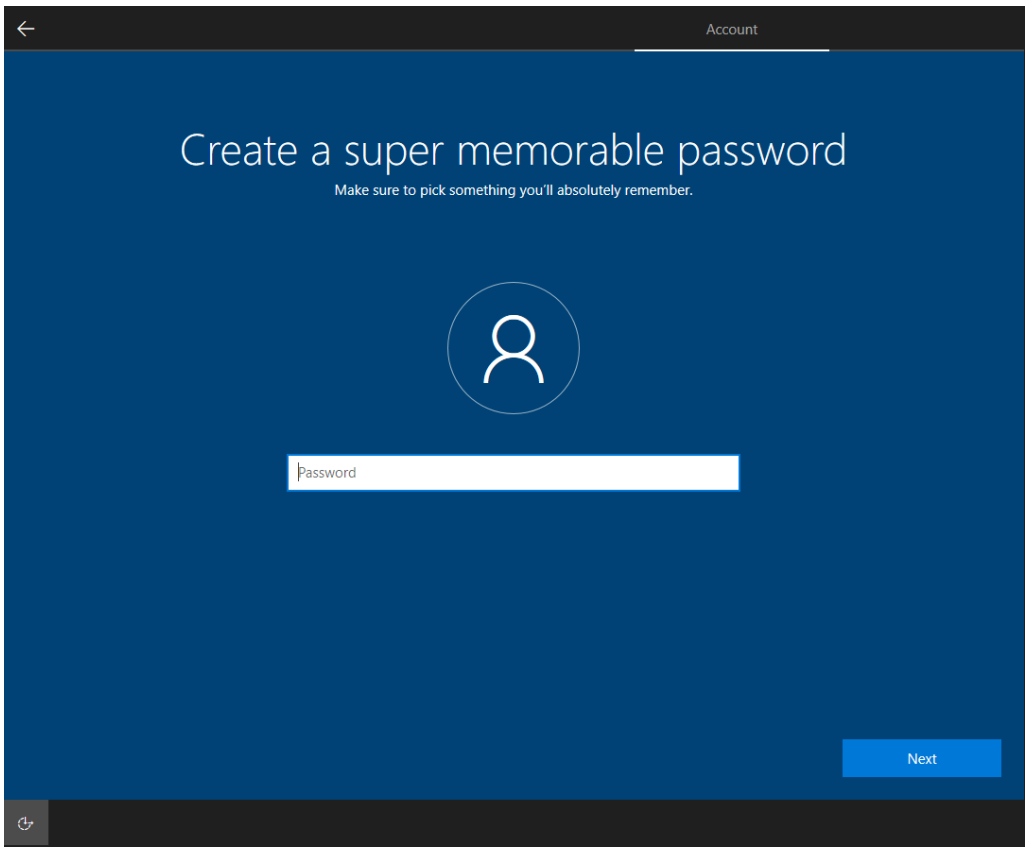


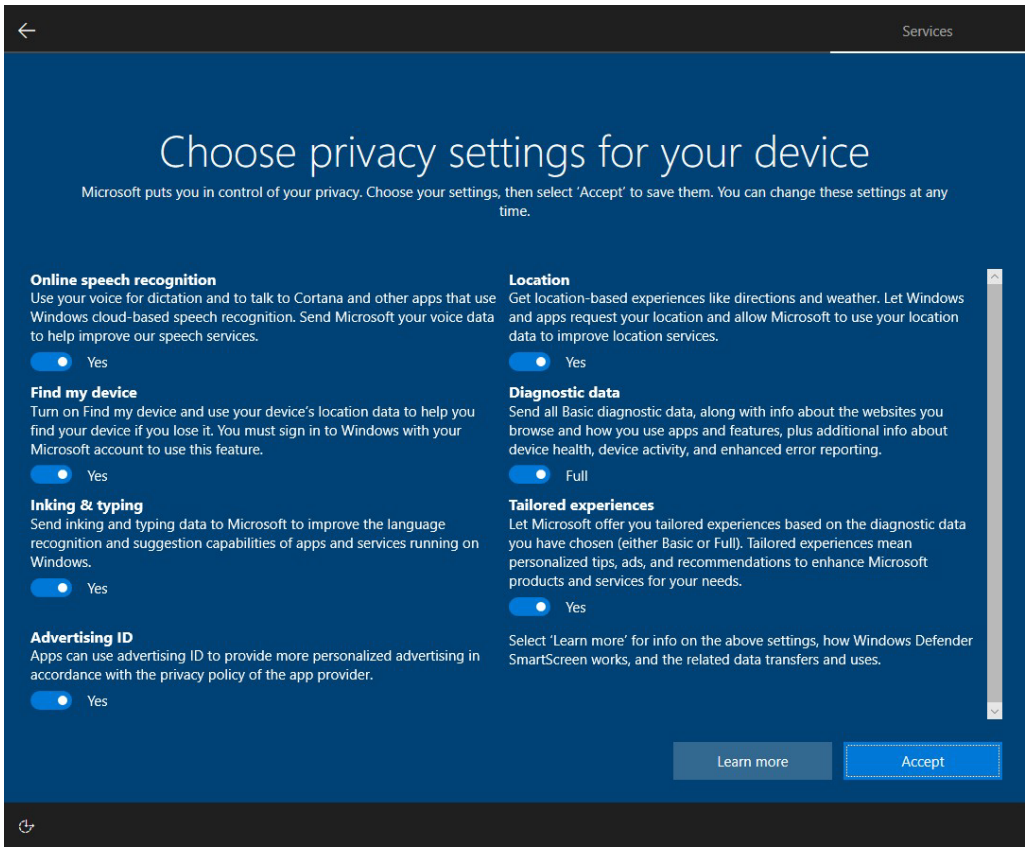
5. Sign in to or create a local account or a Microsoft account (MSA).




6. Set a password.

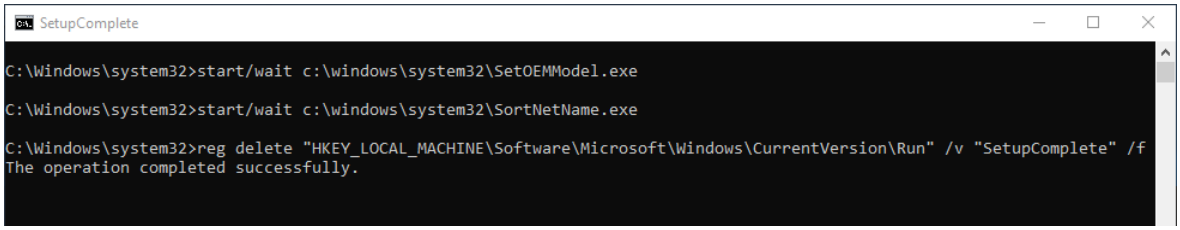


7. Choose your privacy settings.




# Initializing System

After the OOBE settings, you will be redirected to the device desktop of the device. 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.

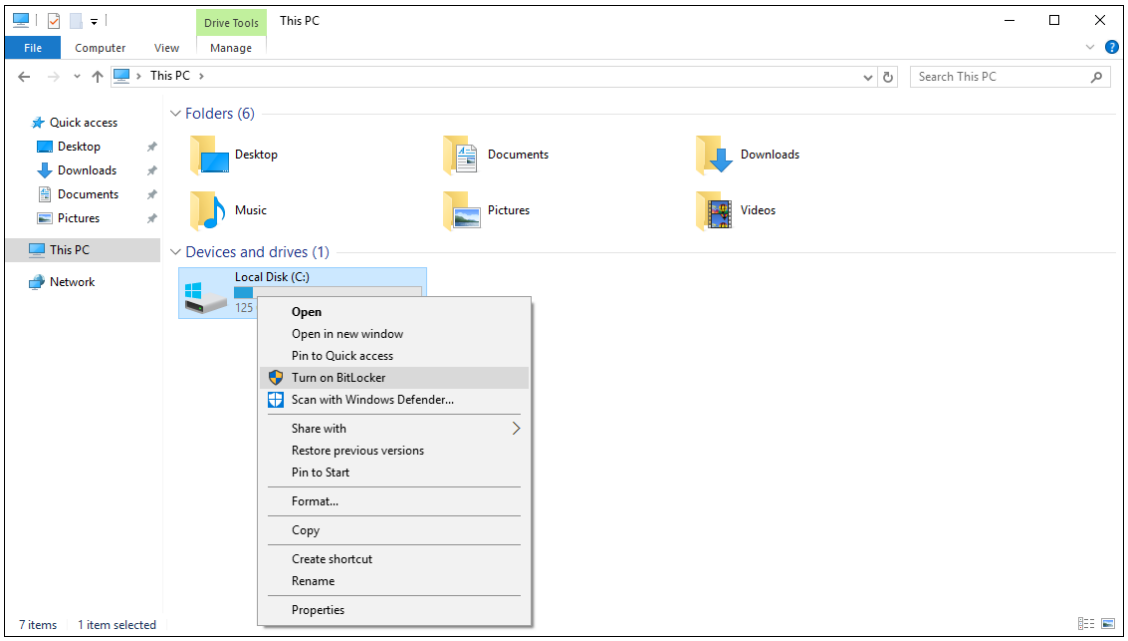
For more information about BitLocker, go to:

https://learn.microsoft.com/en-us/windows/security/operating-system-security/data-protection/bitlocker/

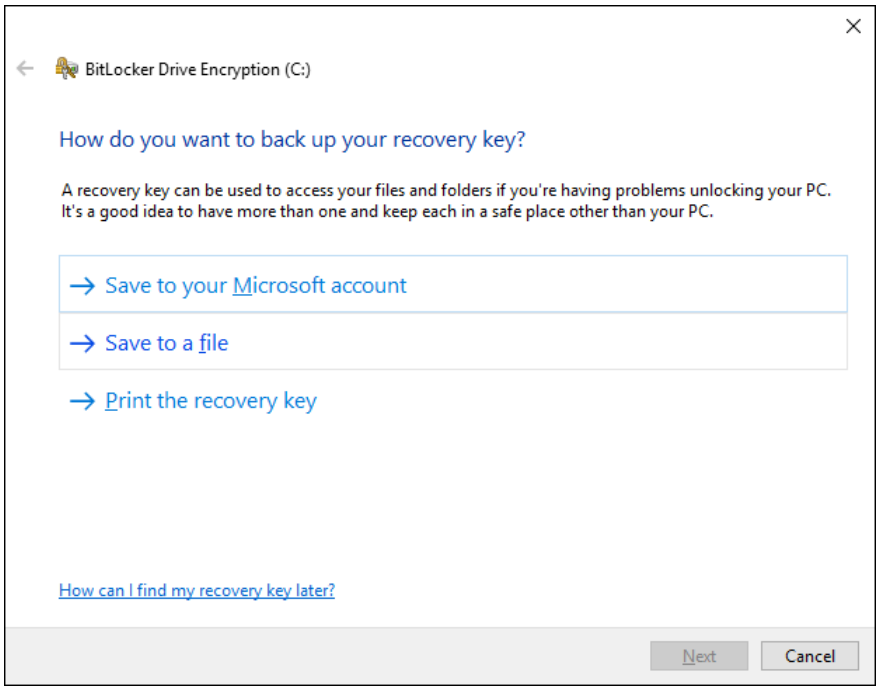
This chapter describes the BitLocker setup process.

# Enabling the BitLocker

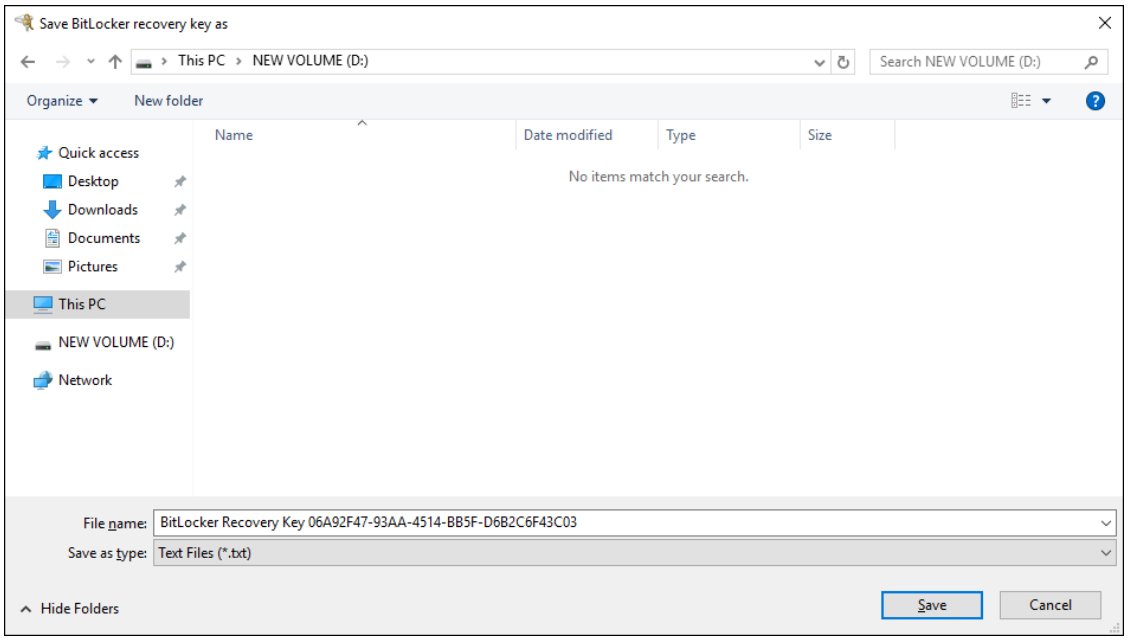
1. In the **Windows Devices and drives**, right-click on the drive and select **Turn on BitLocker**.



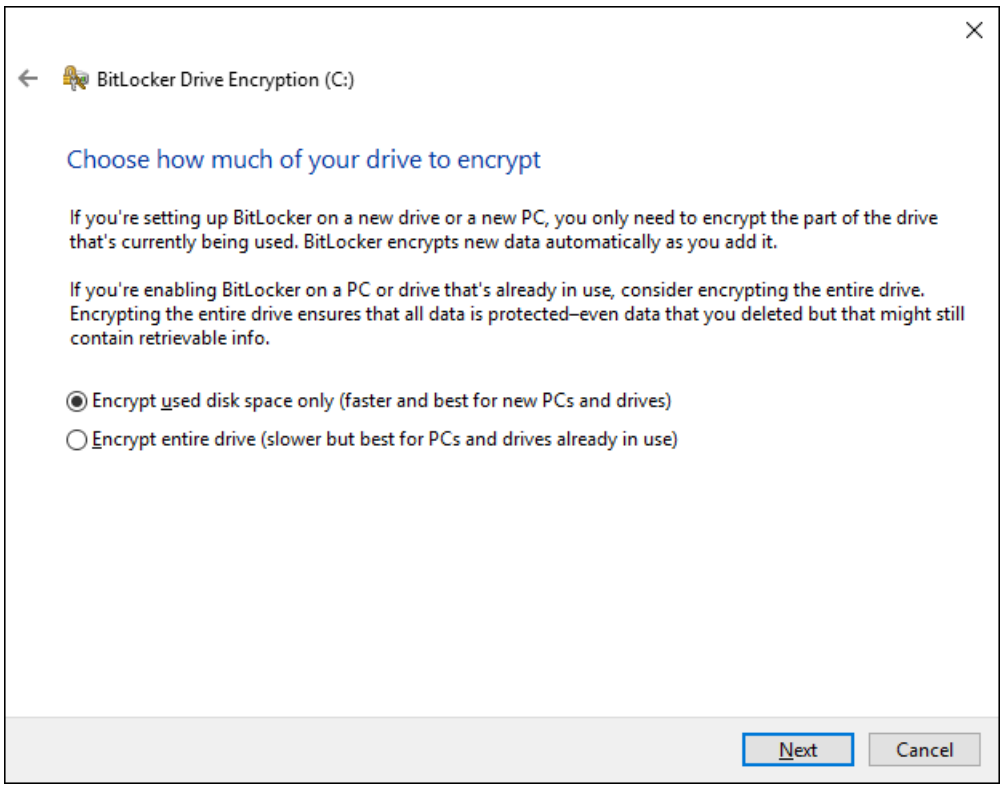
2. Select an option to back up the recovery key. For example, select **Save to a file**.

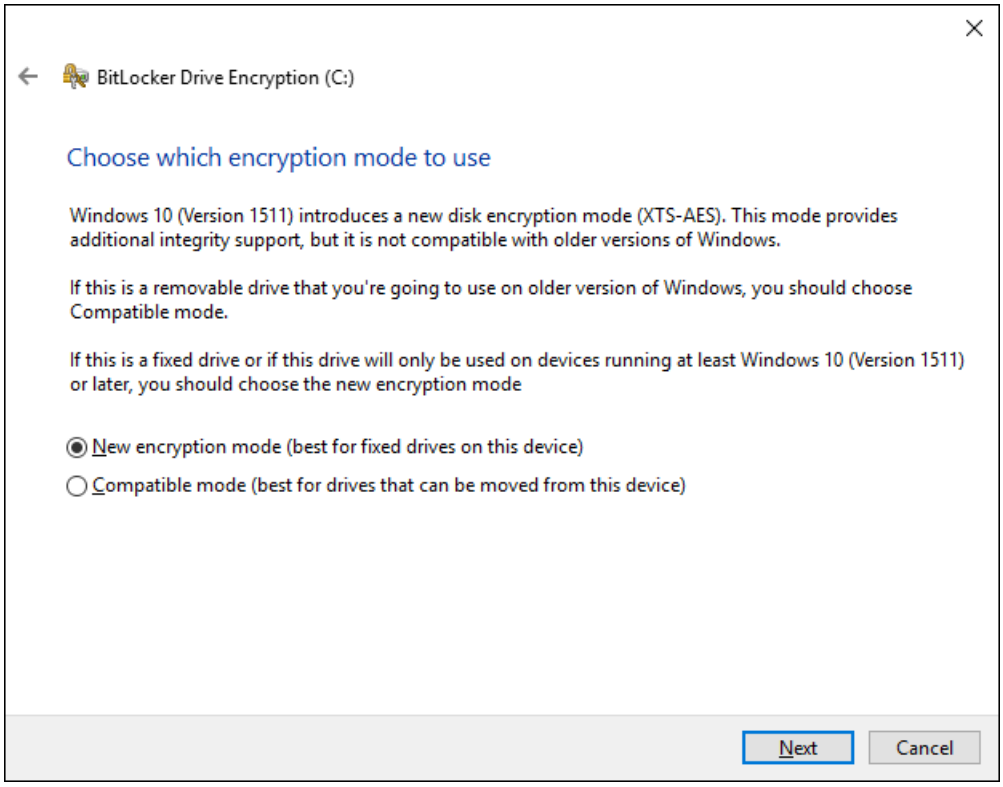



3. Select the path to store the file in.

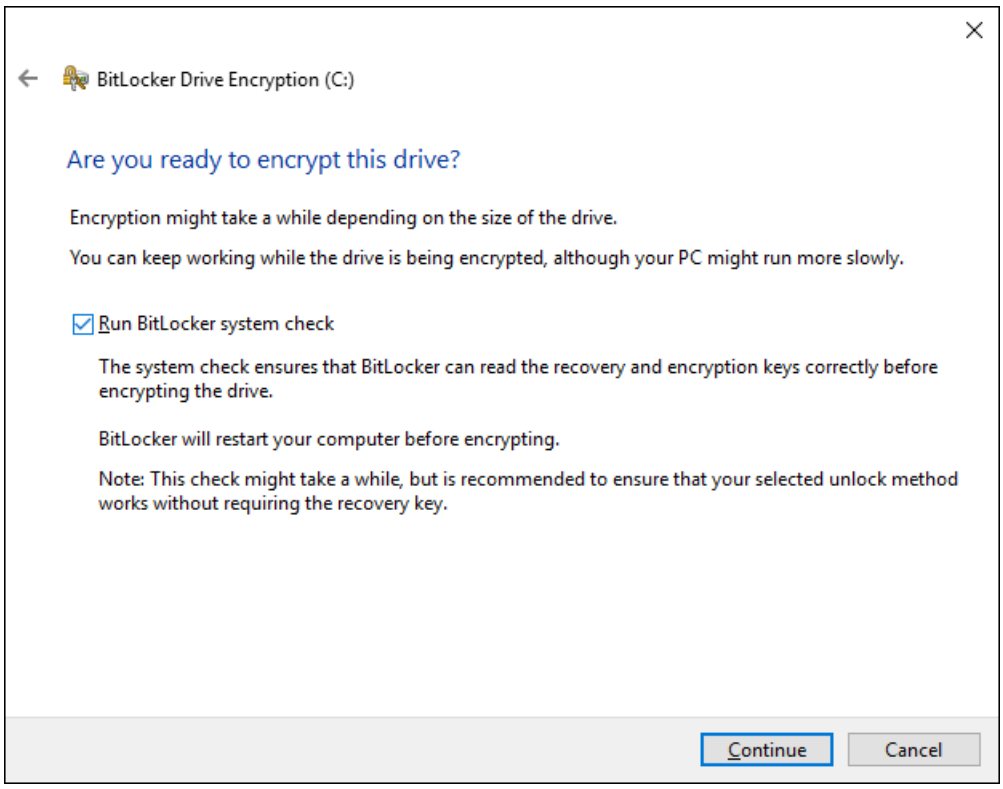


4. Follow the onscreen instructions to specify the drive encryption options.

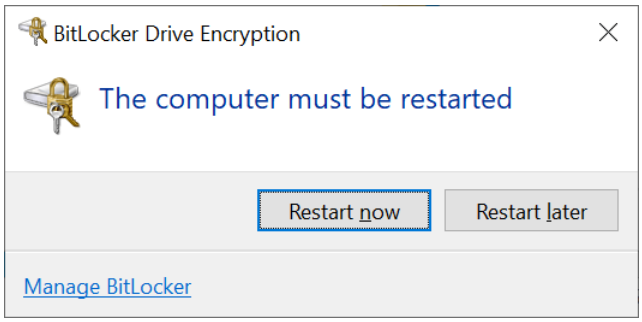


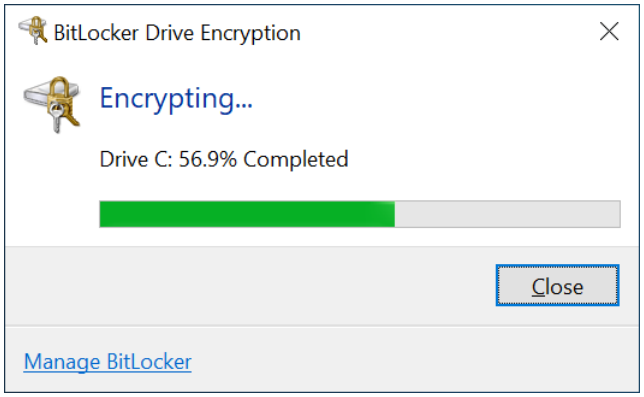
5. Click **Continue**.




6. Restart the computer.

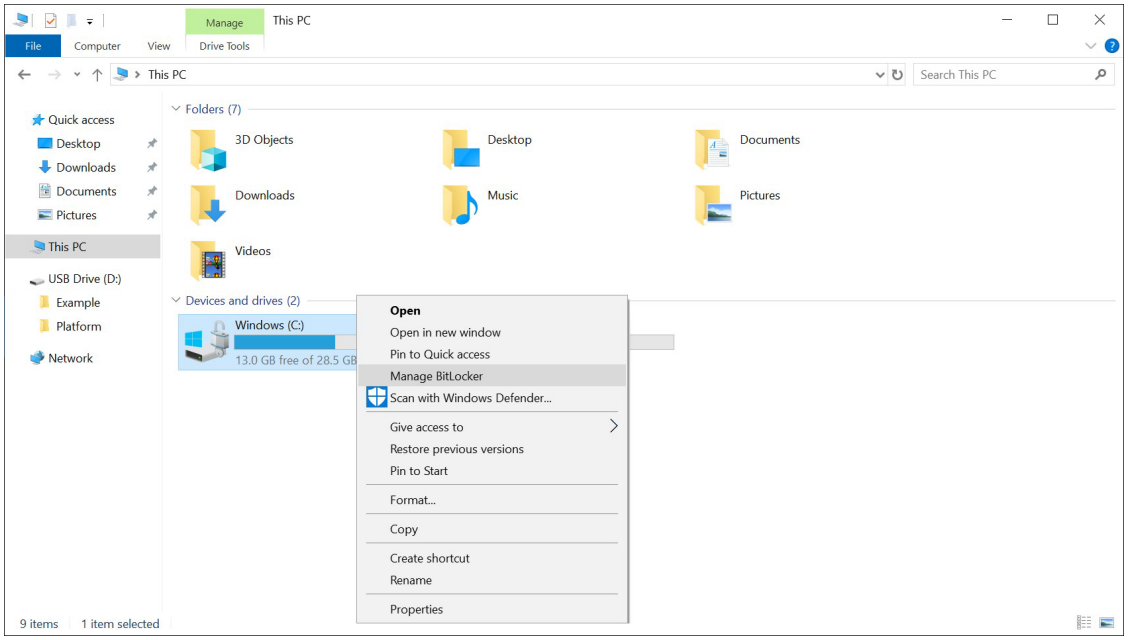



7. Wait for the encryption process to complete and then click **Close**.

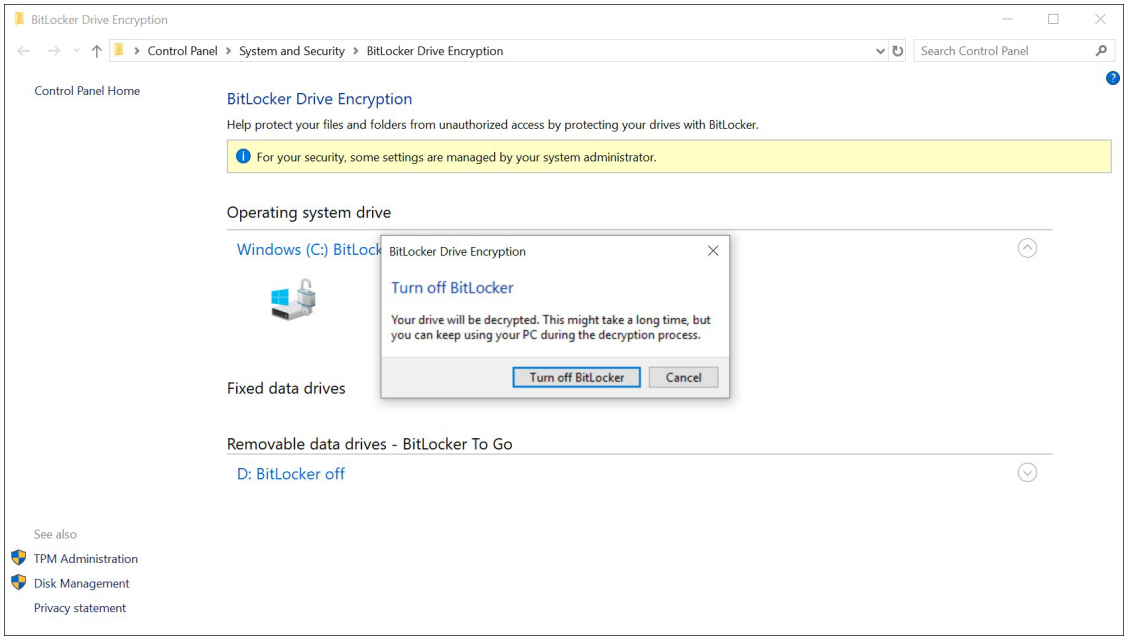


# Disabling the BitLocker

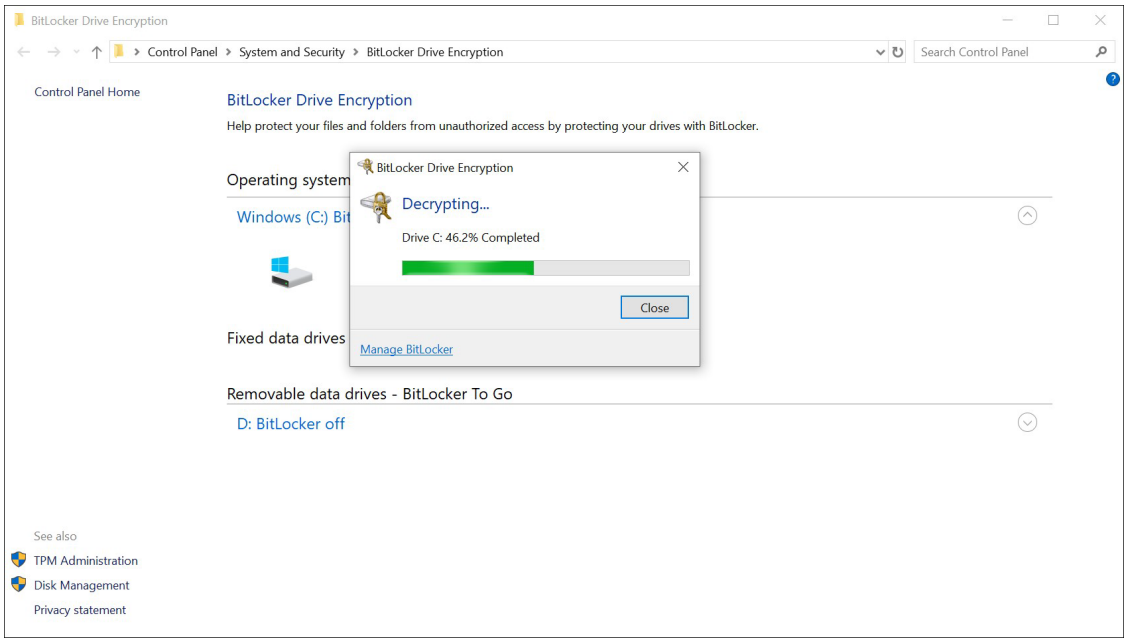
1. In the **Windows Devices and drives**, right-click on the drive and select **Manage BitLocker**.

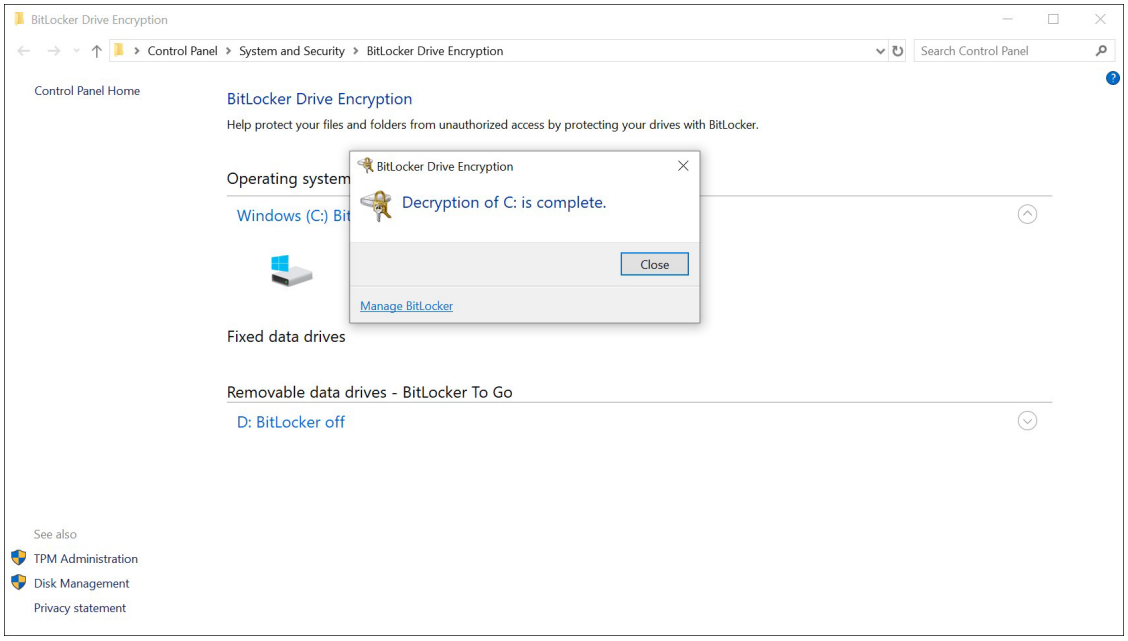



2. Click on **Turn off BitLocker**.

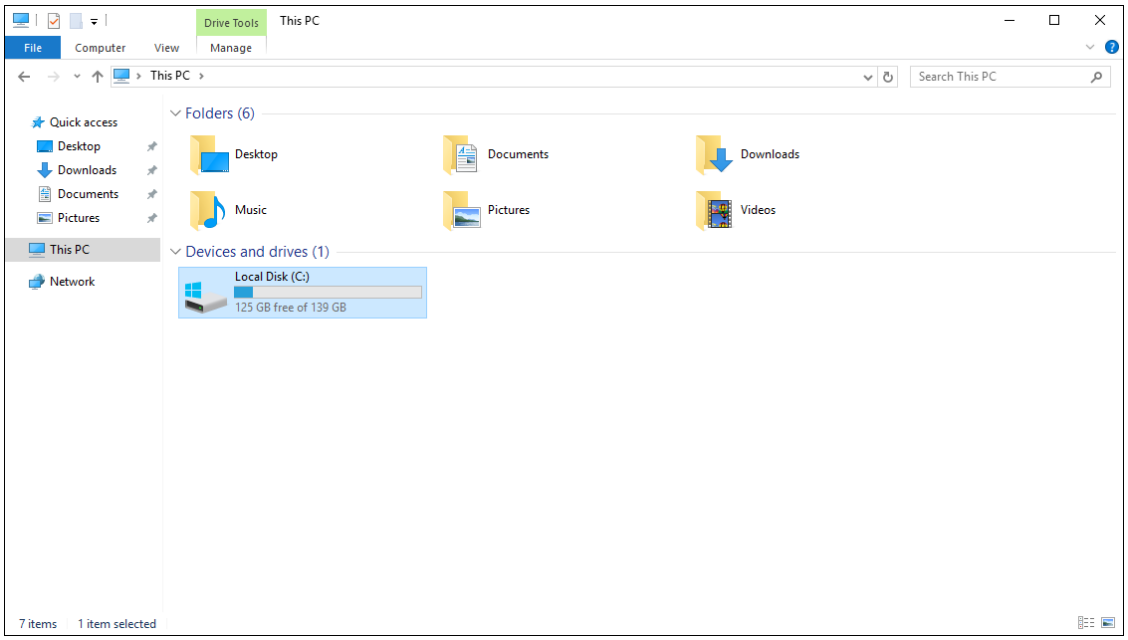


3. Wait for the decryption process to complete and click **Close** to exit the program.



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



# 4. Teaming

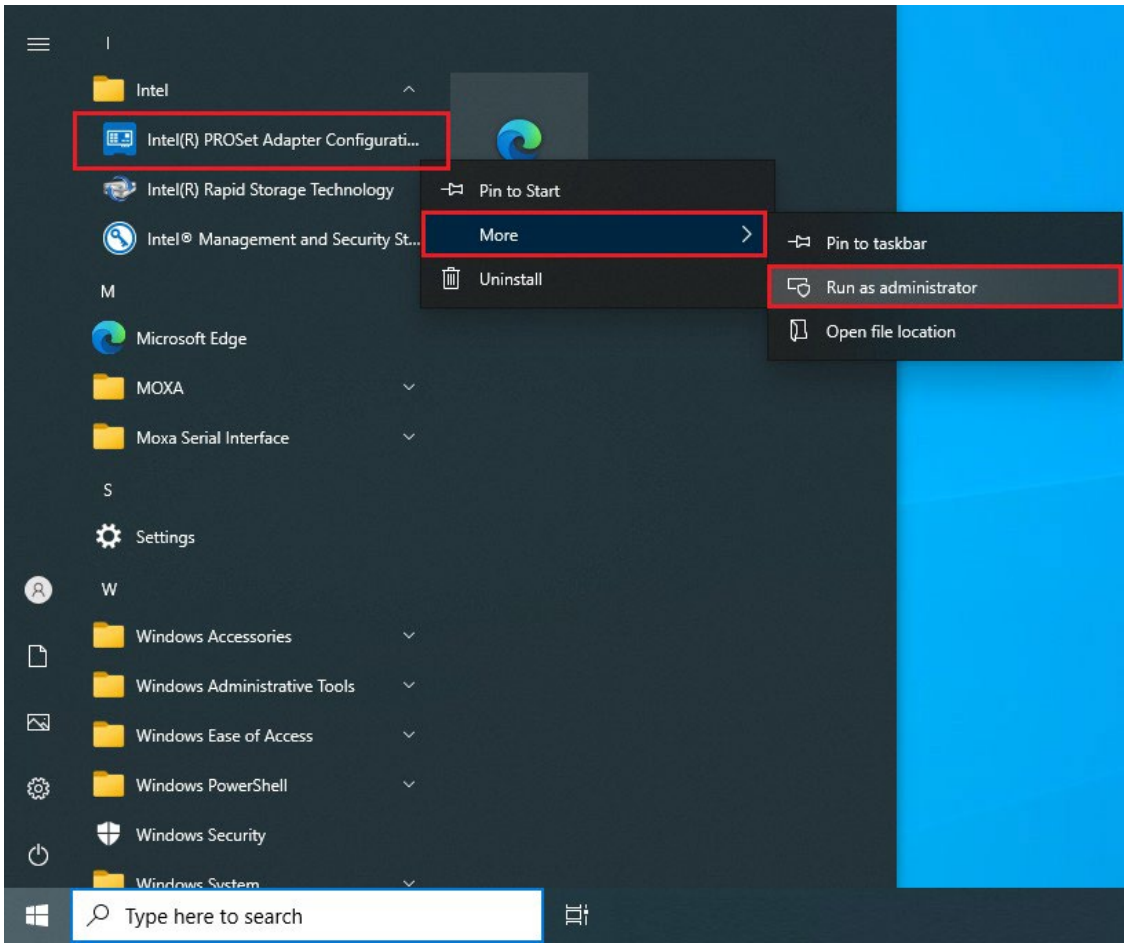
NIC Teaming, also known as load balancing and failover (LBFO), allows multiple network adapters on a computer to be placed into a team for bandwidth aggregation or traffic failover to prevent connectivity loss in the event of a network component failure.

# Intel® Net Team

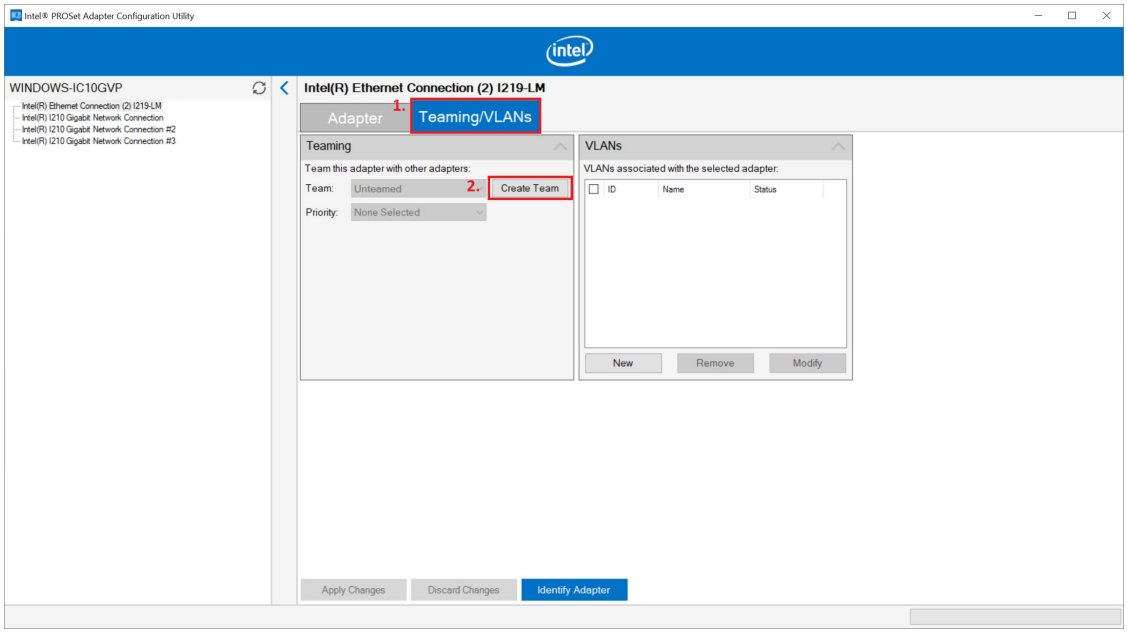
This chapter describes the setup process for the Intel® Teaming function.

## Creating an Intel® Net Team

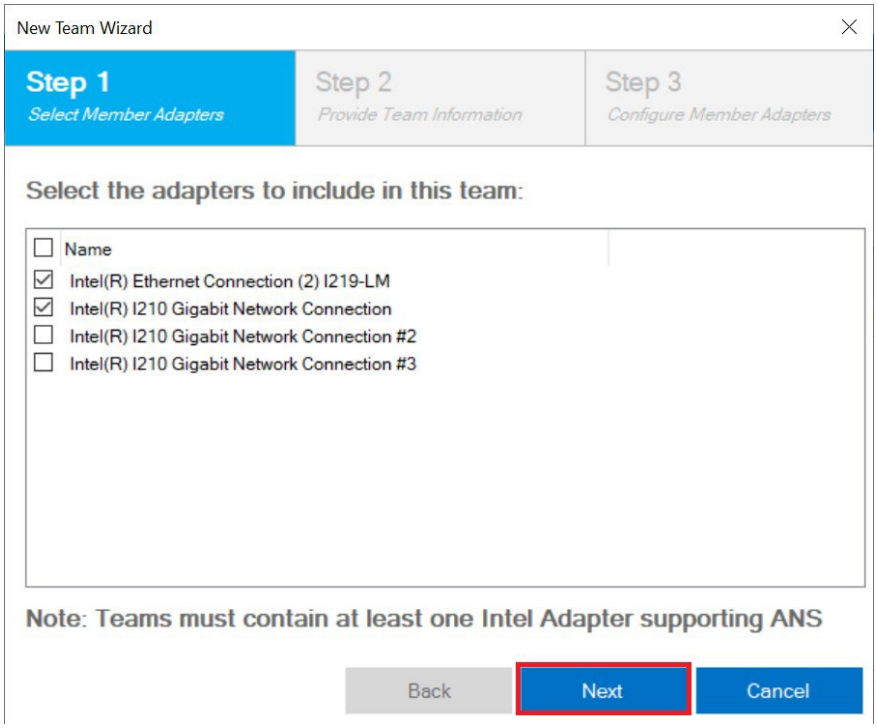
1. Run **Intel® PROSet Adapter Configuration Utility** as administrator.



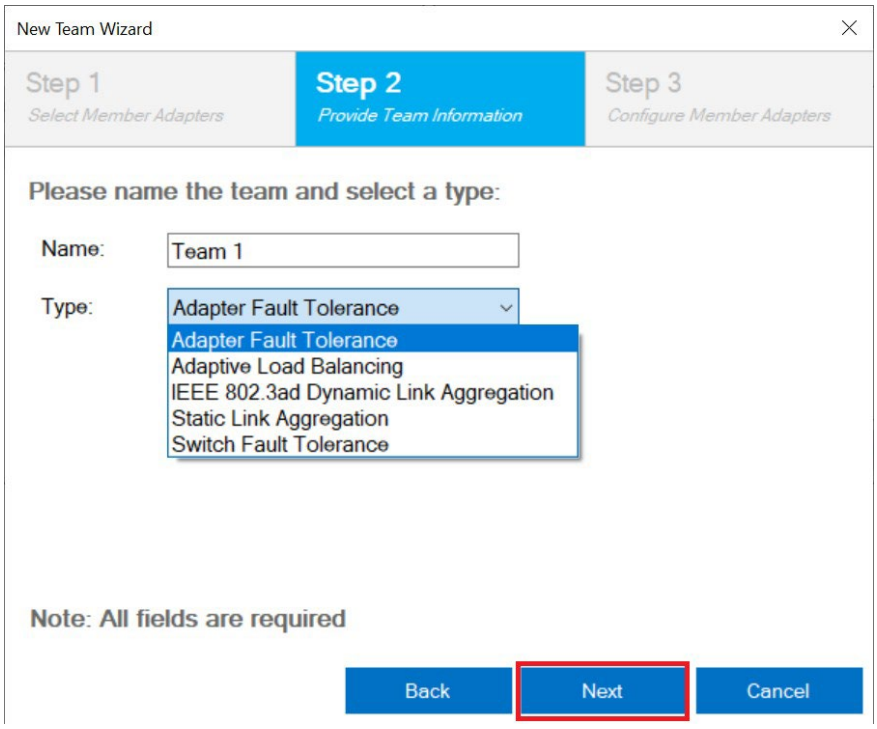
2. In the **Teaming/VLANs** tab, click **Create Team**.




3. Select the adapter to include in this team and click **Next**.
An Intel® ANS team can contain a maximum of eight members.



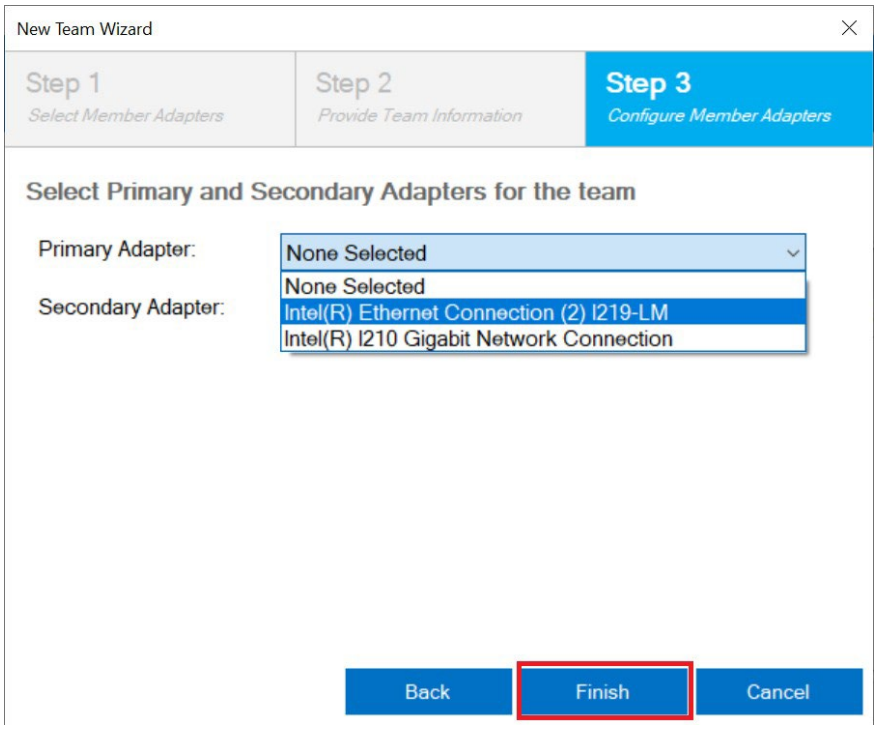
4. Name the team and select a team type. Click **Next** to continue.



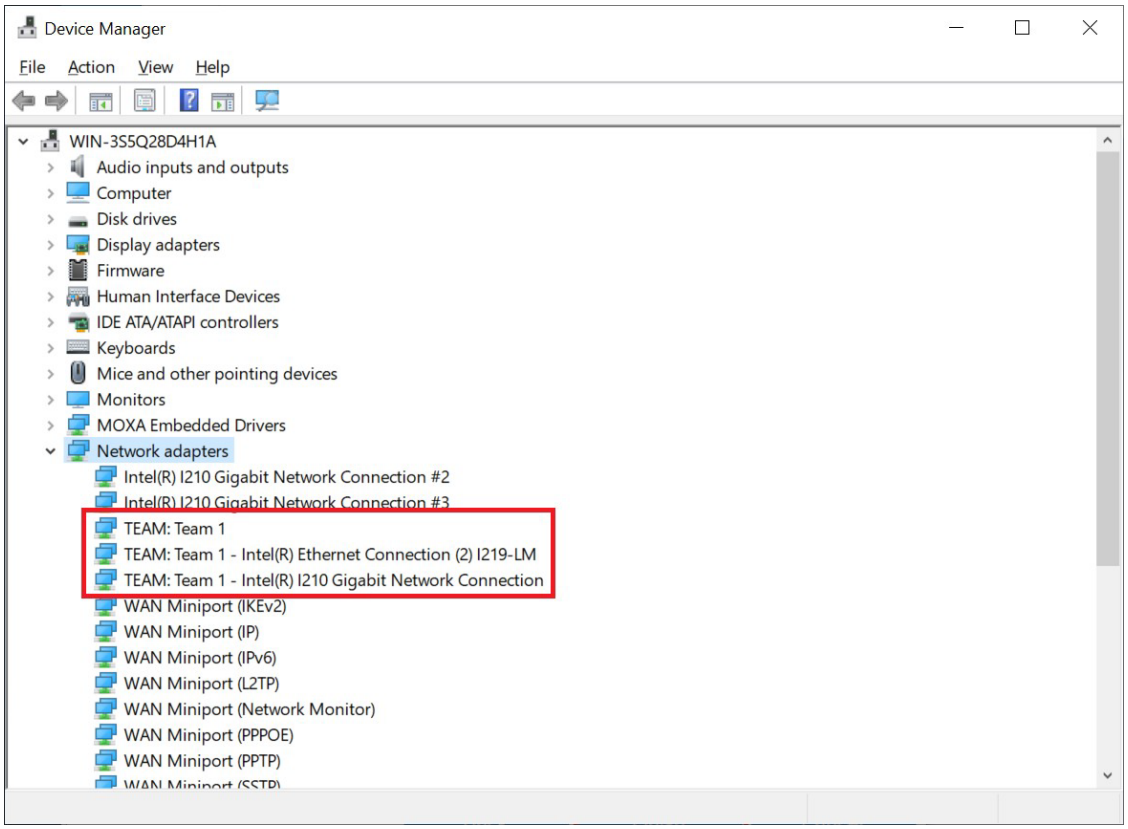

## NOTE

You cannot use an Intel® AMT enabled adapter in a Dynamic Link Aggregation (DLA) team or in a Static Link Aggregation (SLA) team.

5. Select the primary and secondary adapters for the team and click **Finish** to create an Intel® Net team.



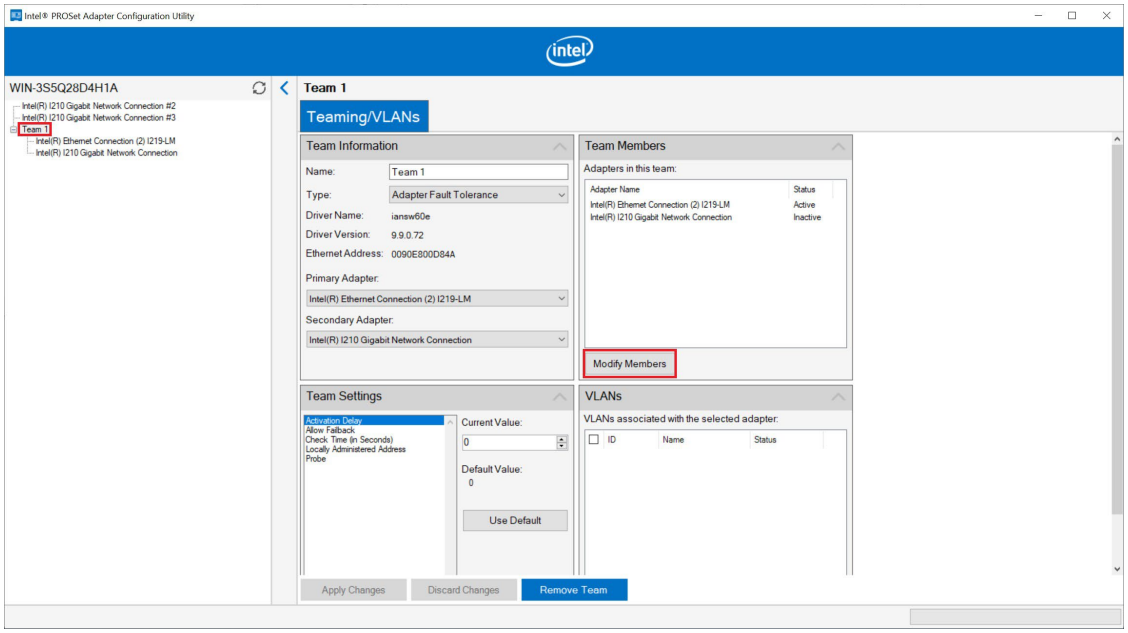
6. Check the **Network adapters** in the **Windows Device Manager**.



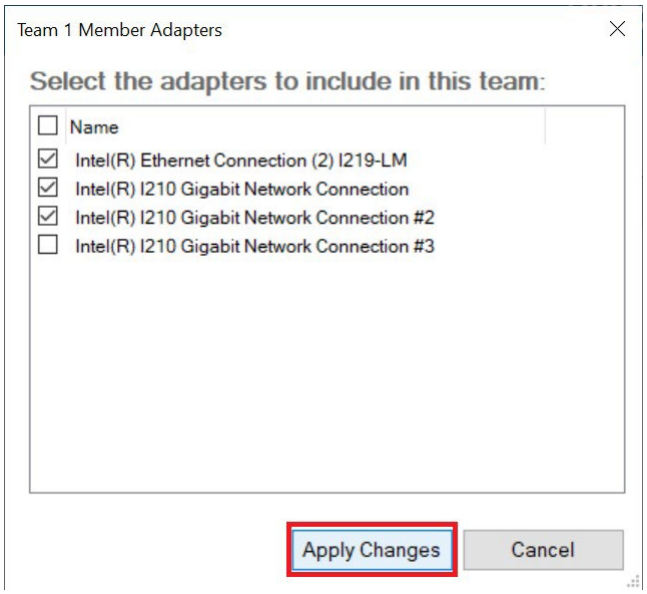

# Modifying an Intel® Net Team Member

## Adding an Intel® Net Team Member

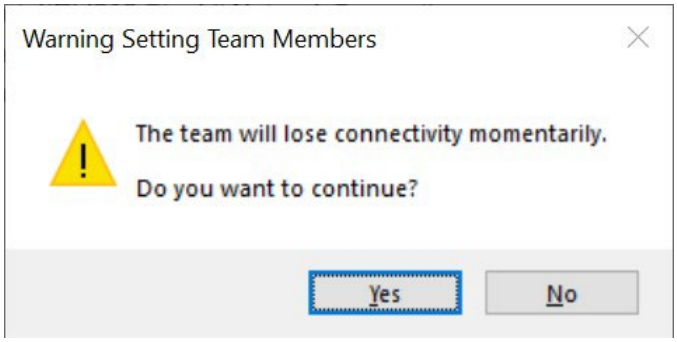
1. Select an Intel® Net team and click **Modify Members**.



2. Select the adapters to include in this team and click **Apply Changes**.



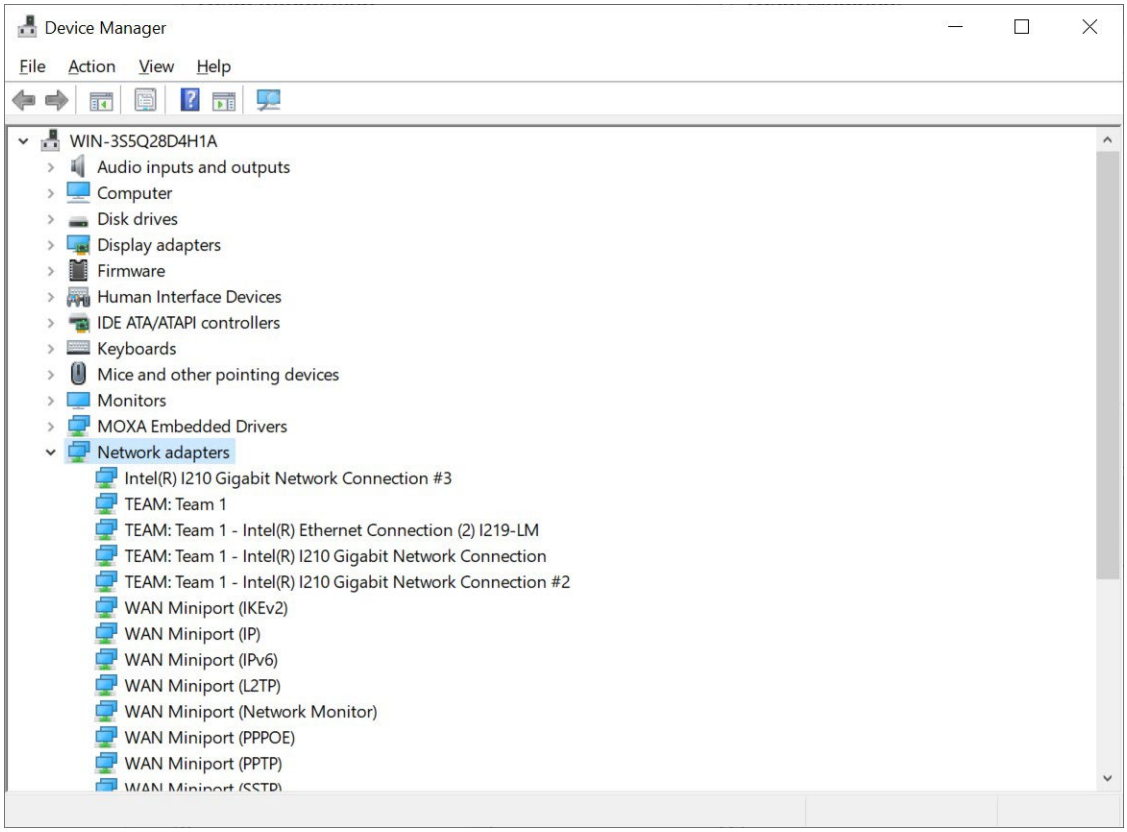

3. Click **Yes** to continue.




**NOTE**

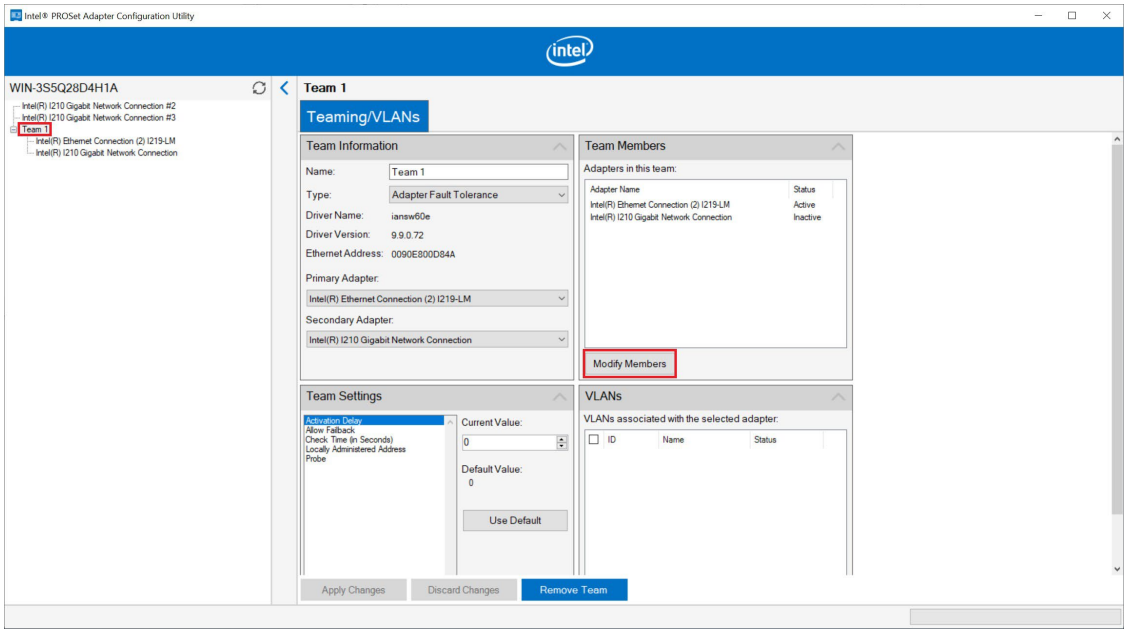
Modifying team members will cause the members to momentarily lose connectivity.

4. Check the **Network adapters** in the **Windows Device Manager**.

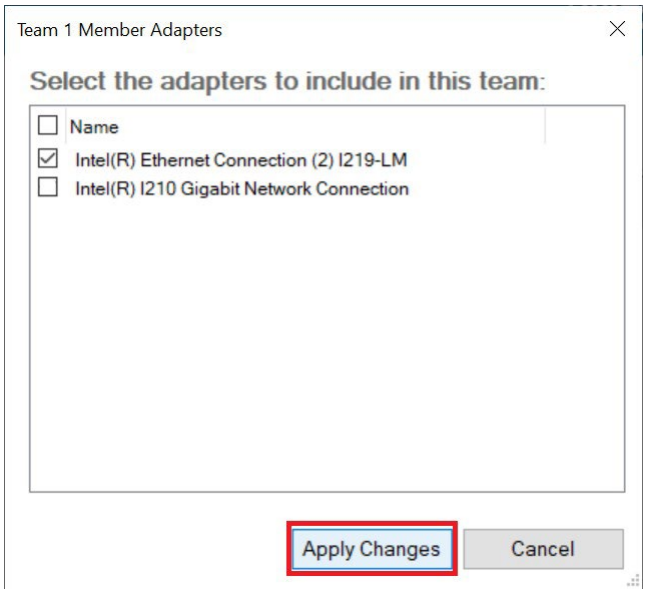



## Removing an Intel® Net Team Member

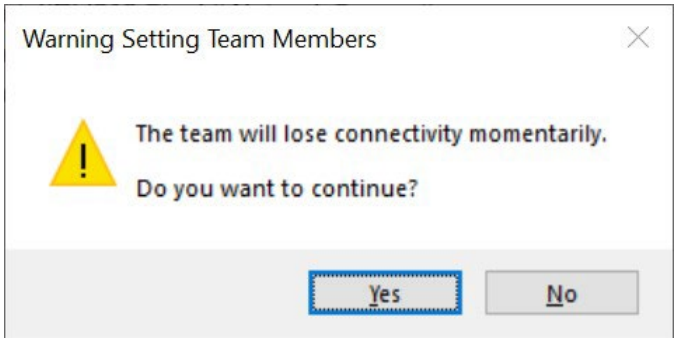
1. Select an Intel® Net team and click **Modify Members**.



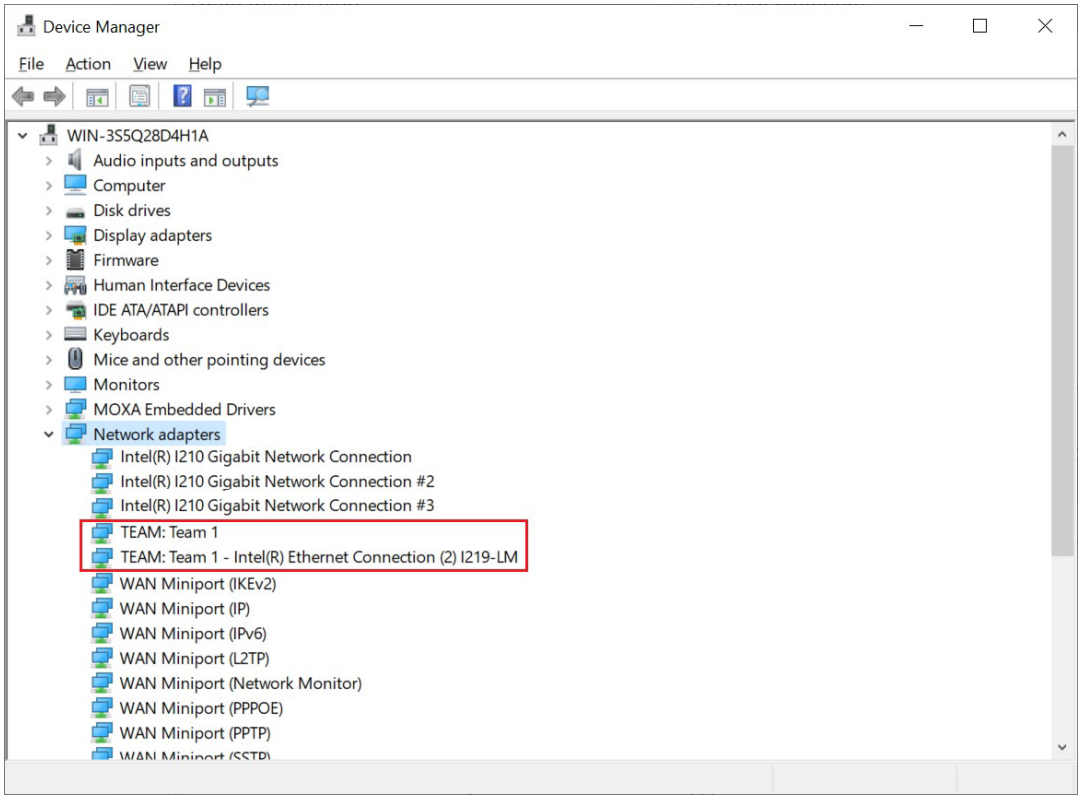
2. Uncheck the adapters you want to remove in this team and click **Apply Changes**.




3. Click **Yes** to continue.

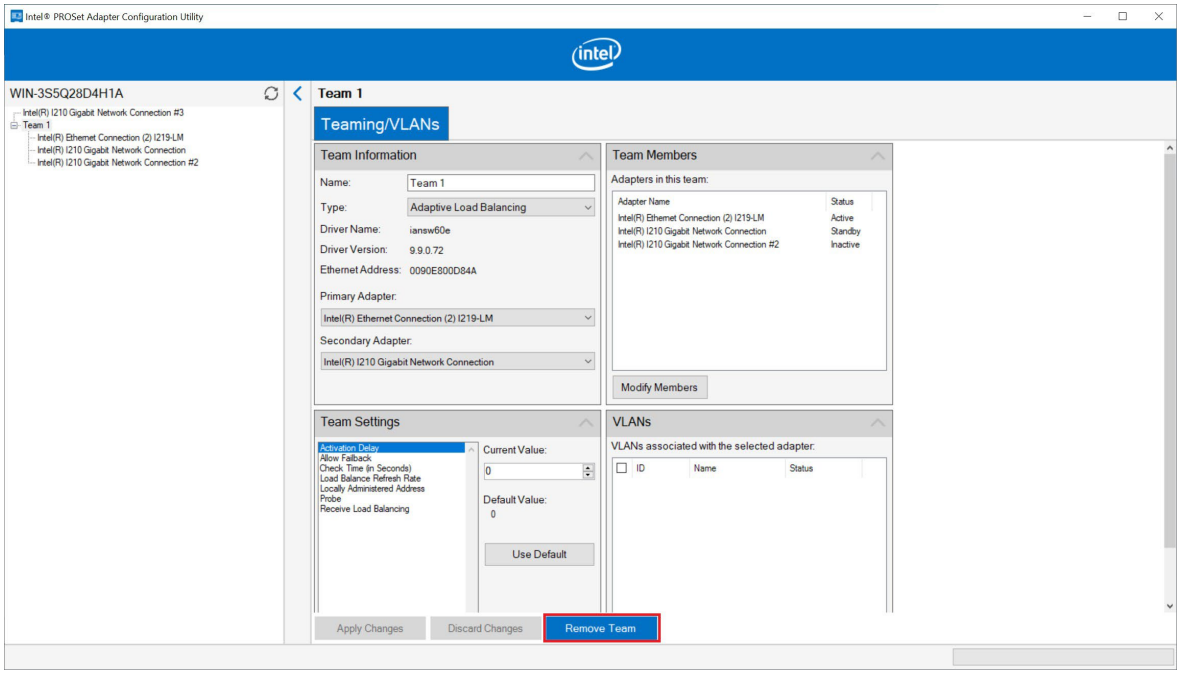



4. Check the **Network adapters** in the **Windows Device Manager**.



# Removing an Intel® Net Team

Select an Intel® Net team and click **Remove Team** to remove the specified Intel® ANS team.



# 5. Unified Write Filter

Unified Write Filter (UWF) is an optional Windows 10 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 school, 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.

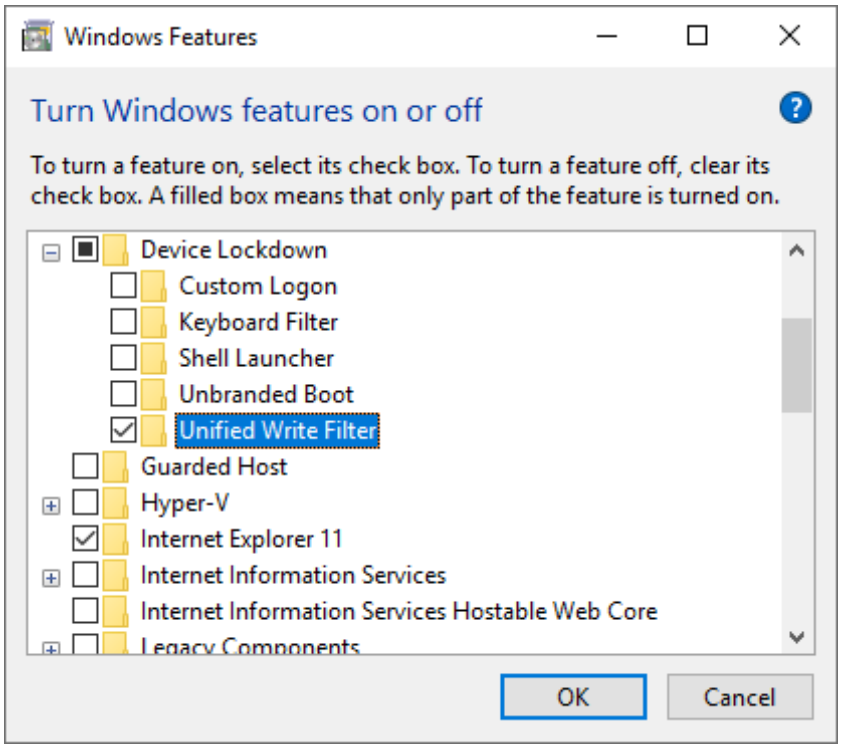
The first time you enable UWF on your device, UWF makes the following changes to your system to improve its performance:

- **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.**

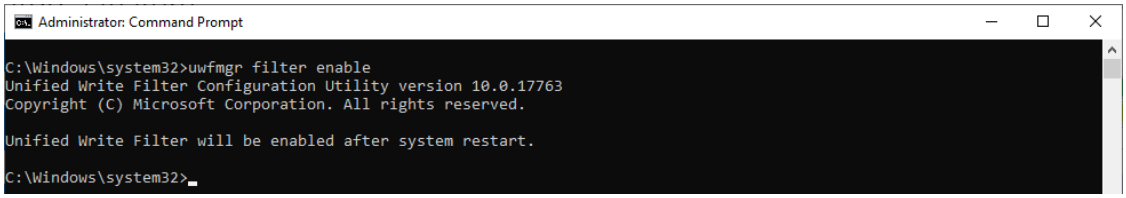
After UWF is enabled, you can select a drive that you want to protect and start using UWF. UWF can help you manage PCs and devices remotely using WMI.

# Turning on UWF on a Running PC

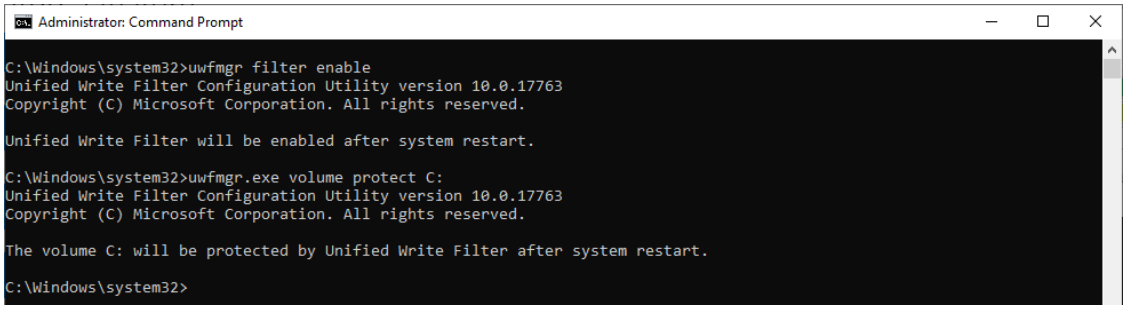
- Install UWF.
  a. In the Windows **Start** window, type **Turn Windows features on or off**.
  b. Open the **Windows Features** window and expand the **Device Lockdown** node.
  c. Select **Unified Write Filter** and click **OK**.
  d. Windows searches for the required files and displays a progress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



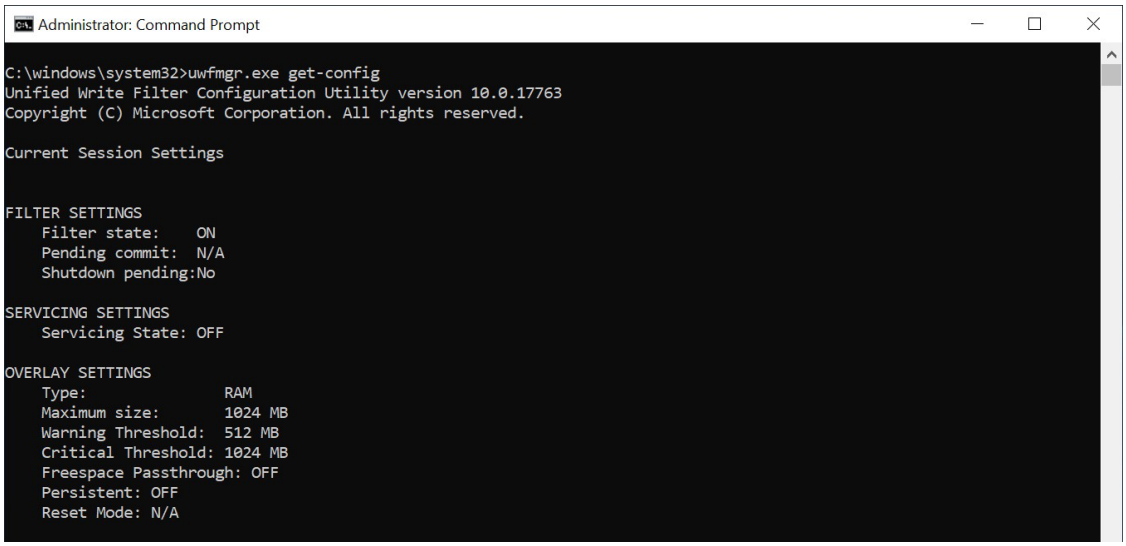

- Enable write protection for a drive:
  **cmd uwfmgr.exe volume protect C:**




- 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.

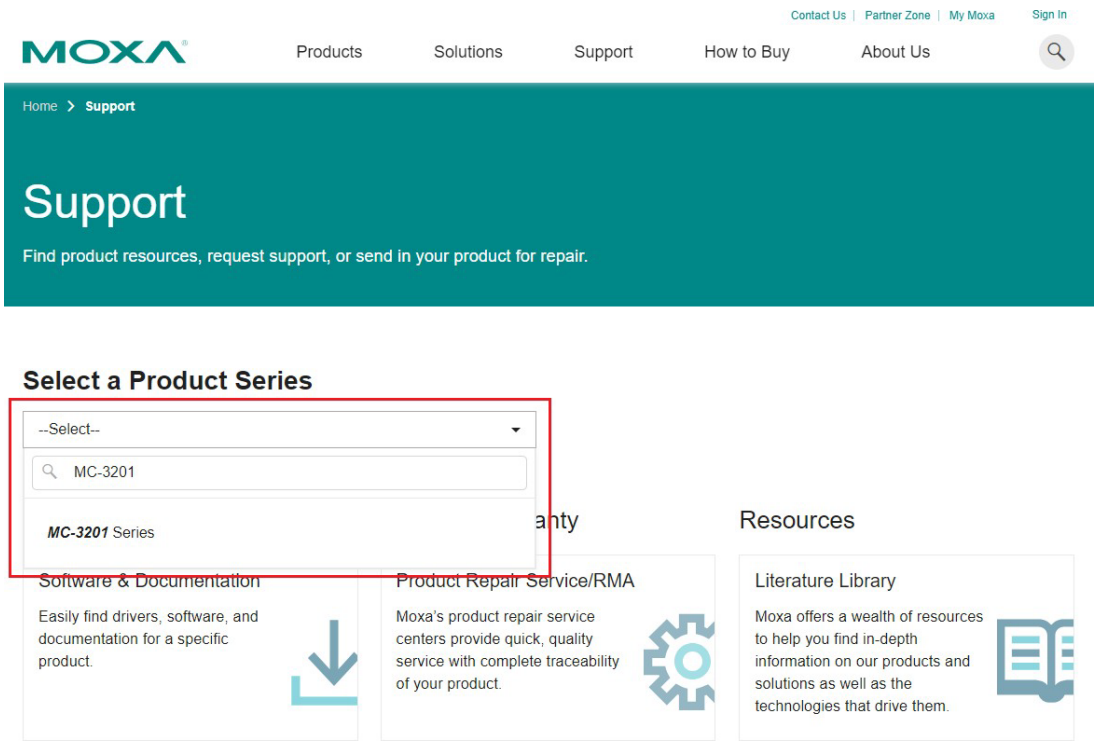
## IMPORTANT!

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.

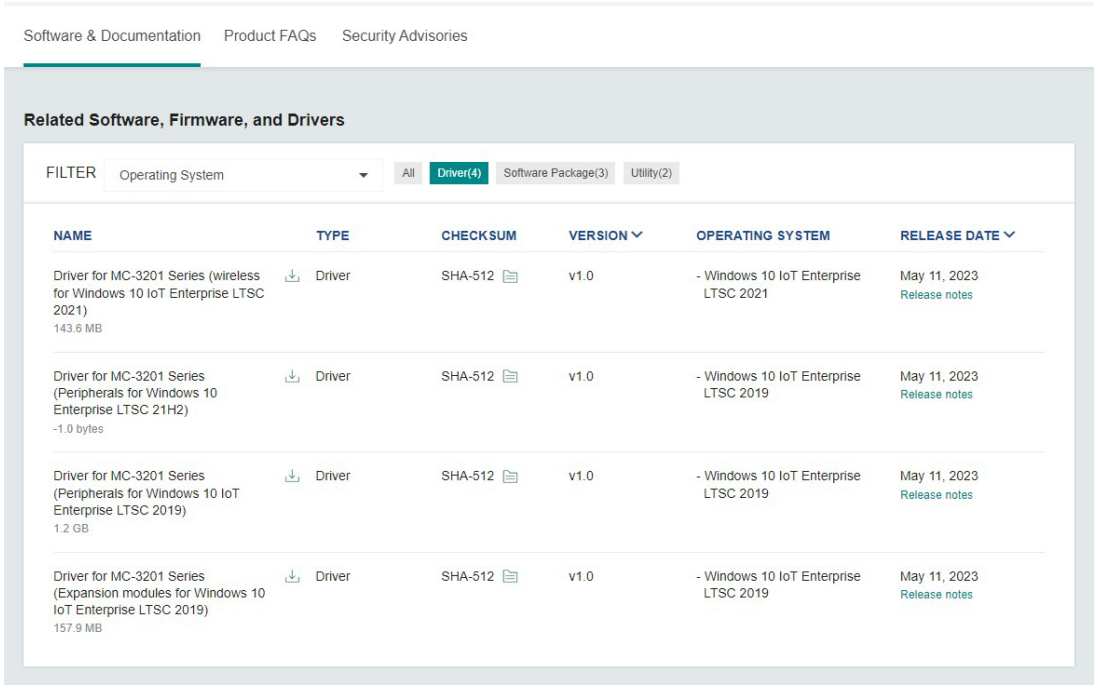
After the device is restarted, UWF maintains configuration settings for the current session in a registry. UWF automatically excludes these registry entries from its filter. Static configuration changes do not take effect until after a device restarts; the changes are saved in registry entries for use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

# 6. 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).




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



# 7. Utility

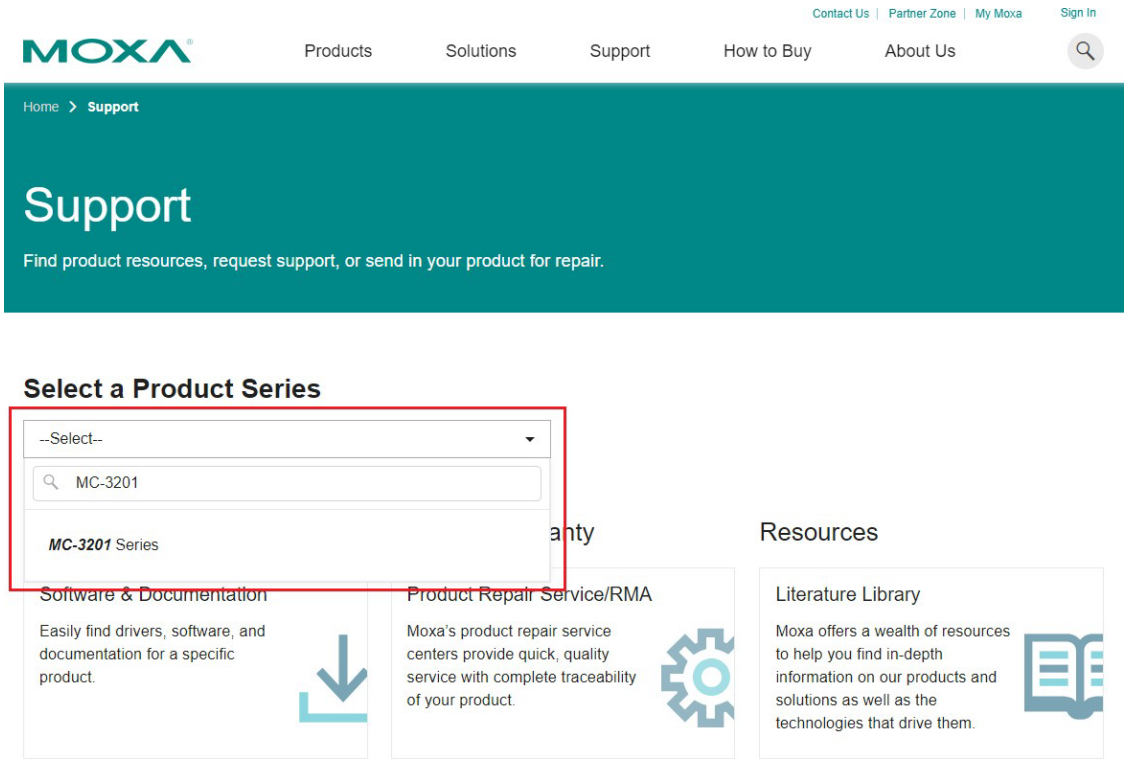
This chapter describes the usage of the following:

- **Moxa IO Controller**
- **Serial Interface**
- **Moxa Sort Net Name**
- **Moxa Audio Configuration**
- **Moxa MCU Firmware Upgrade**
- **Moxa Panel Configuration**

# Where to Find the Utility

The utilities will be preinstalled in the device if the Windows 10 OS is provided by Moxa. If you install Windows 10 by yourself, go to https://www.moxa.com/en/support to download the utilities.

On the support page, search for your device (e.g., MC-3201).



From the **Software & Documentation** page. filter by **Utility** and download the installation ***.zip** file.

**MC-3201 Series**

Compact computers with 11th Gen Intel® Core™ processor, designed for IIoT, AI, and marine applications, -20 to 55°C operating temperature

GO TO PRODUCT PAGE  Save

Software & Documentation  Product FAQs  Security Advisories

**Related Software, Firmware, and Drivers**

FILTER  Operating System  All  Driver(4)  Software Package(3)  Utility(2)

| NAME | TYPE | CHECKSUM | VERSION | OPERATING SYSTEM | RELEASE DATE |
|---|---|---|---|---|---|
| Utility for MC-3201 Series (Windows 10 IoT Enterprise LTSC 2021) 5.9 MB | Utility | SHA-512 | v1.0 | - Windows 10 IoT Enterprise LTSC 2021 | May 11, 2023 Release notes |
| Utility for MC-3201 Series (Windows 10 IoT Enterprise LTSC 2019) 5.1 MB | Utility | SHA-512 | v1.0 | - Windows 10 IoT Enterprise LTSC 2019 | May 11, 2023 Release notes |

# Dependent Packages

- Dependent packages must be installed before the utility is installed; you will need to install the **dependent packages** to ensure the smooth operation of the utility.

  Use the following links 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

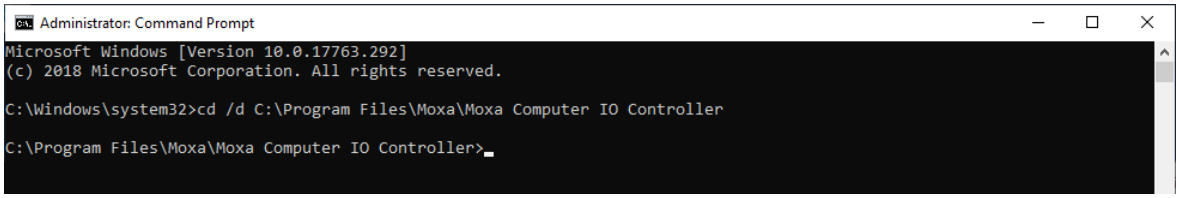
Moxa IO Controller Utility is used to control the peripheral I/Os as well as the interfaces of expansion modules on the device.

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 PCIE Slot Power Status (only for MPC-3XX1)
- Setting the PCIE Reset Pin Status (only for MPC-3XX1)

You can use the preinstalled utility or download and install the utility from https://www.moxa.com/en/support.

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.

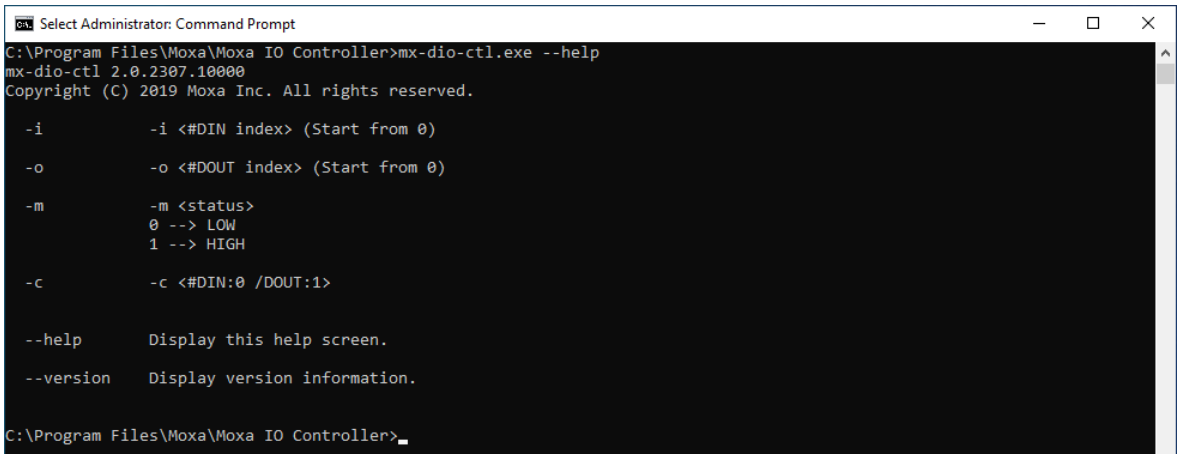


# Setting the DIO Status

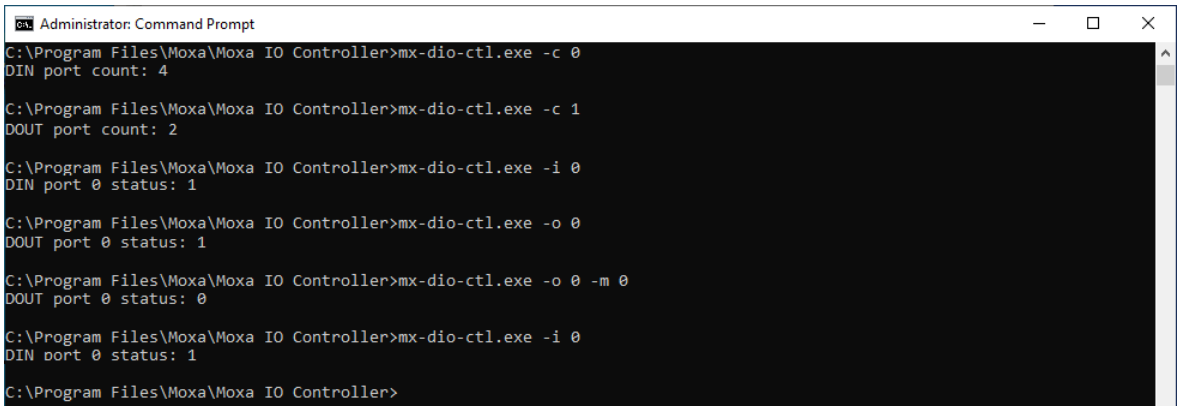
Run the **mx-dio-ctl --help** command to see the instructions on using this utility and follow them to get or set the DIO status.

**IMPORTANT!**

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




**Example:**

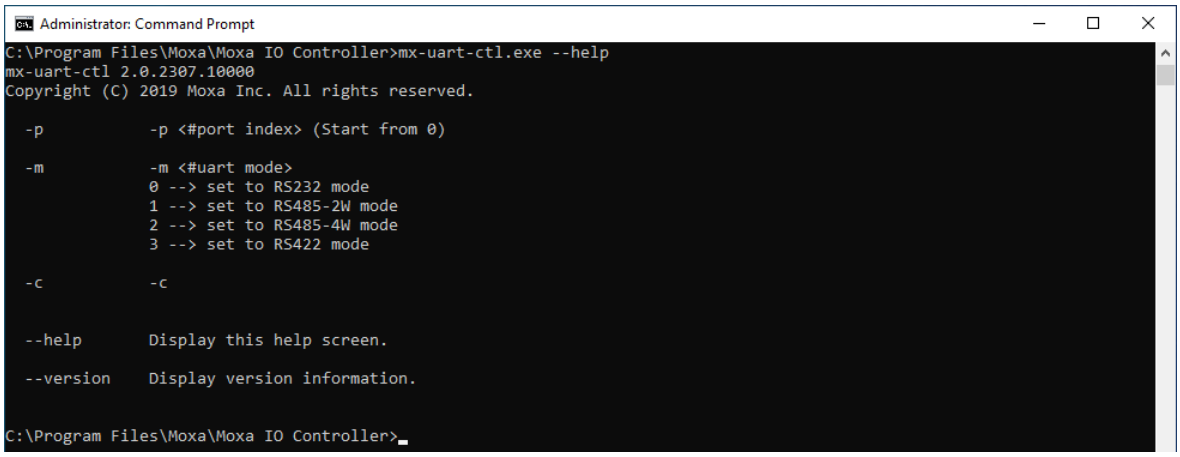


# Setting the UART Mode

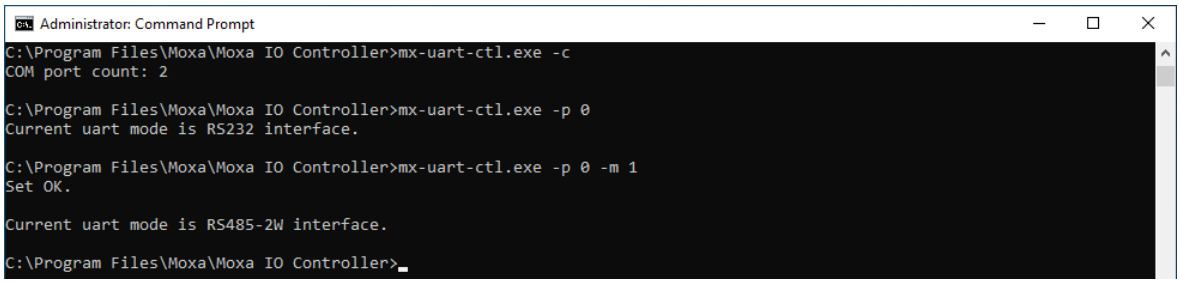
Run the **mx-uart-ctl --help** command to see instructions on using this utility and follow the onscreen instructions to get or set the UART mode.

**IMPORTANT!**

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




**Example:**

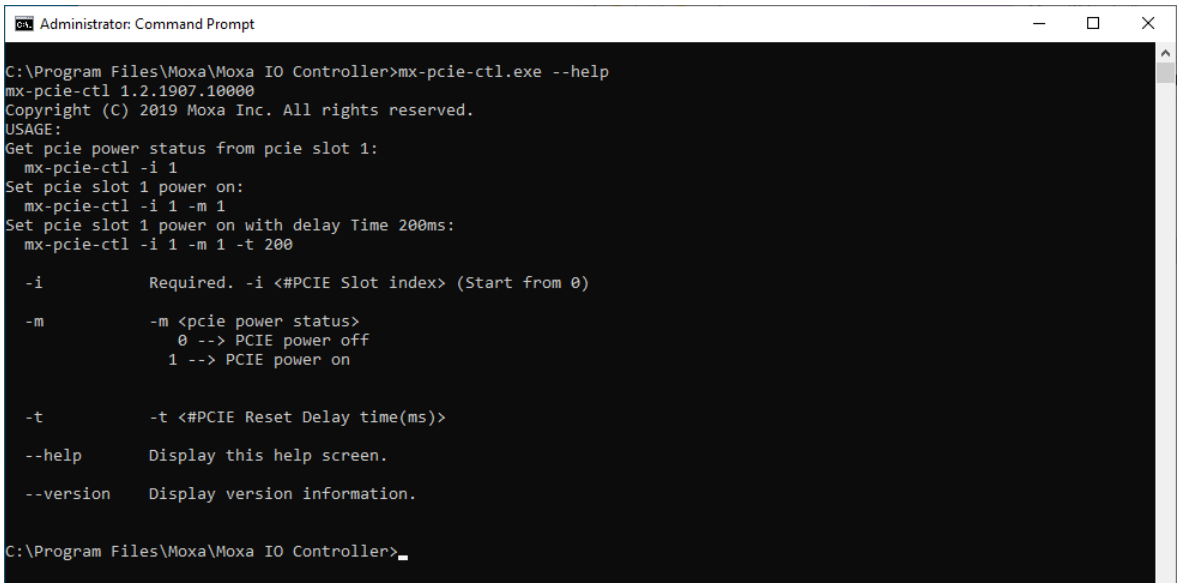


# Setting PCIE Slot Power Status (only for MPC-3XX1 Series)

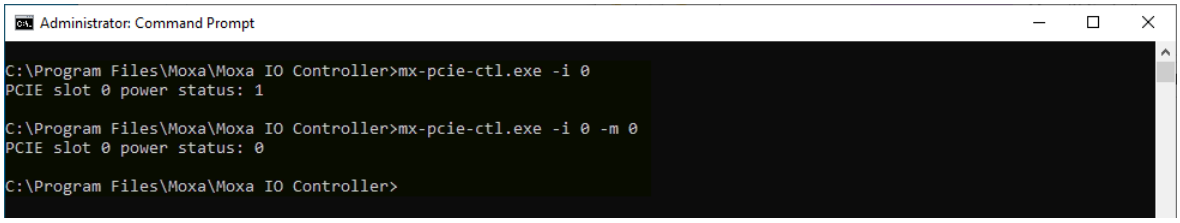
Run the **mx-pcie-ctl --help** command to see instructions on using this utility and follow the onscreen instructions to get or set the status of the PCIE slot power.

## IMPORTANT!

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




**Example:**

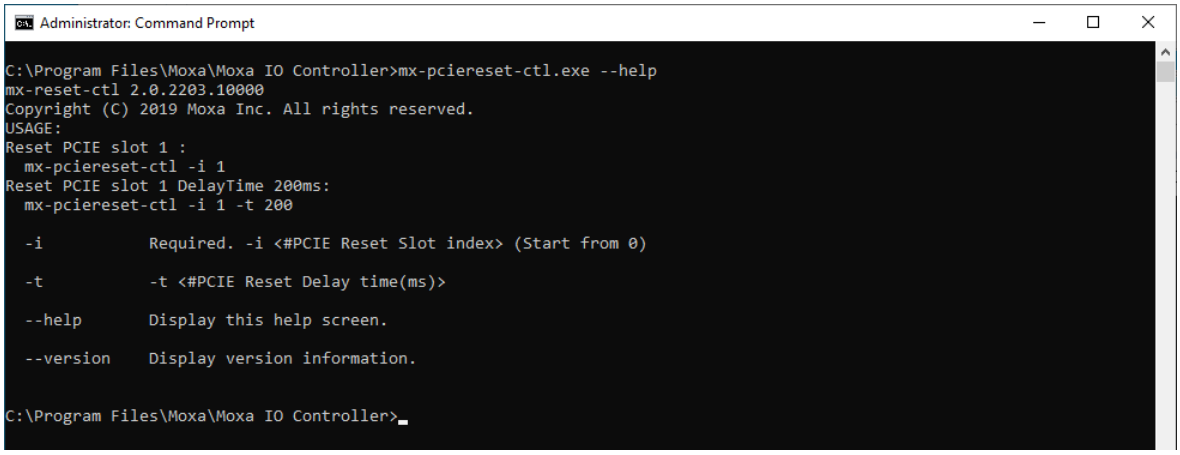


# Setting PCIE Reset Pin Status (only for MPC-3XX1 Series)

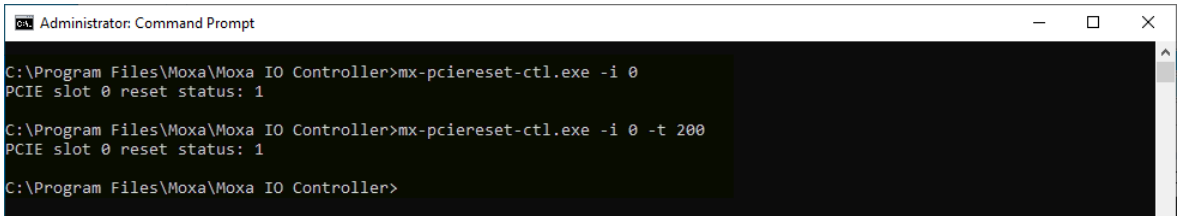
Run the **mx-pciereset-ctl --help** command to see instructions on using this utility and follow the onscreen instructions to get or set the PCIE reset pin status and delay time .

## IMPORTANT!

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




**Example:**

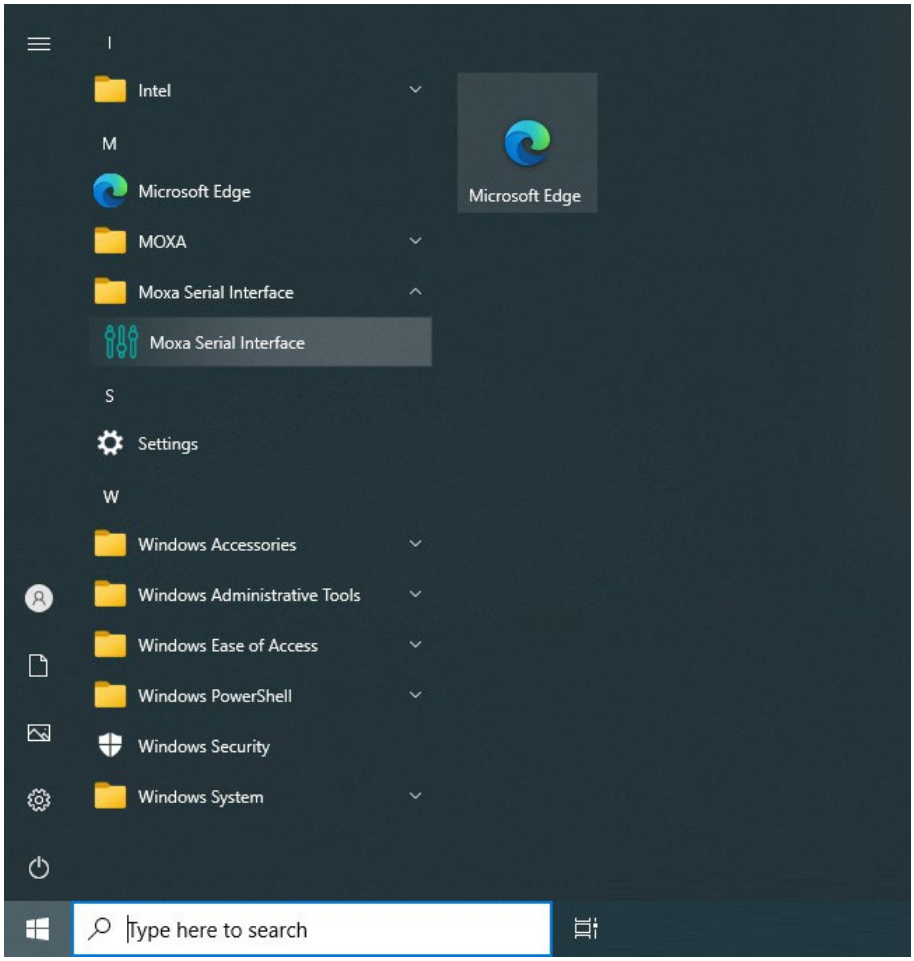


# Moxa Serial Interface

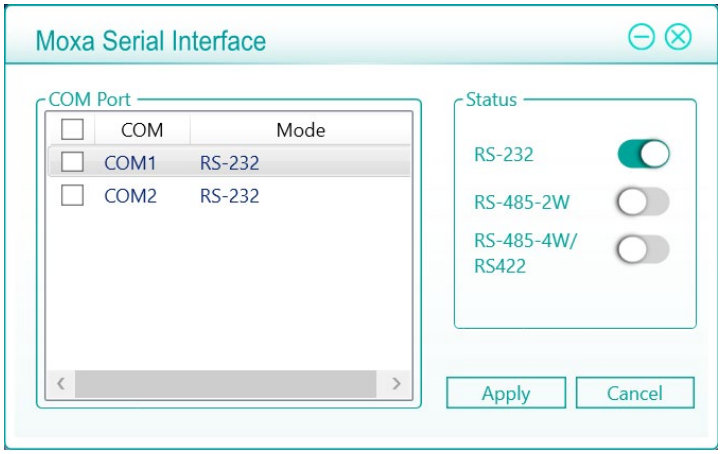
In this section, we describe how to use the Moxa Serial Interface utility to set the UART mode in your computer's serial interface.

## Setting the Serial Port Mode

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



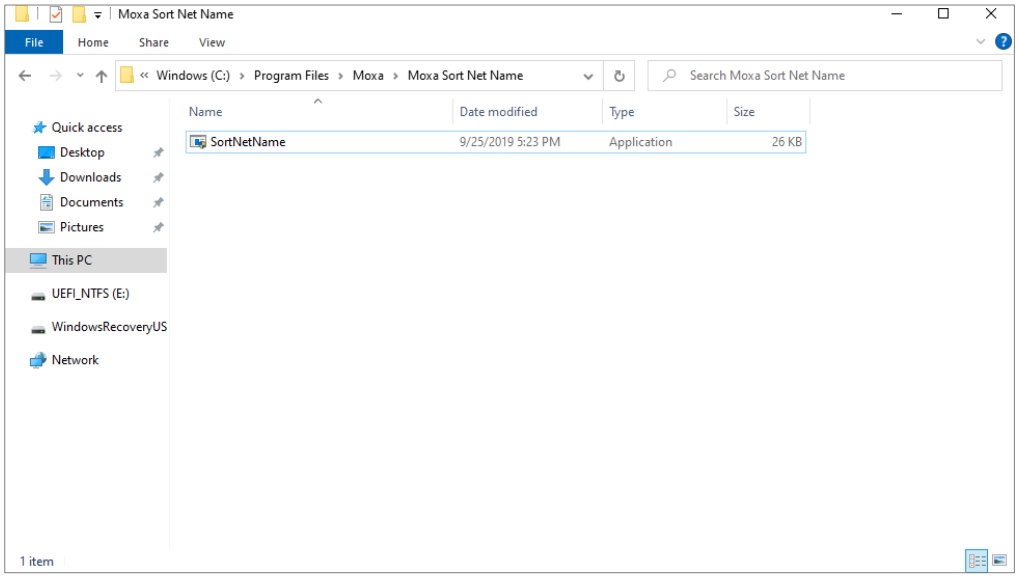

3. Select the target COM port and UART mode and click **Apply** to save the settings.



# Moxa Sort Net Name

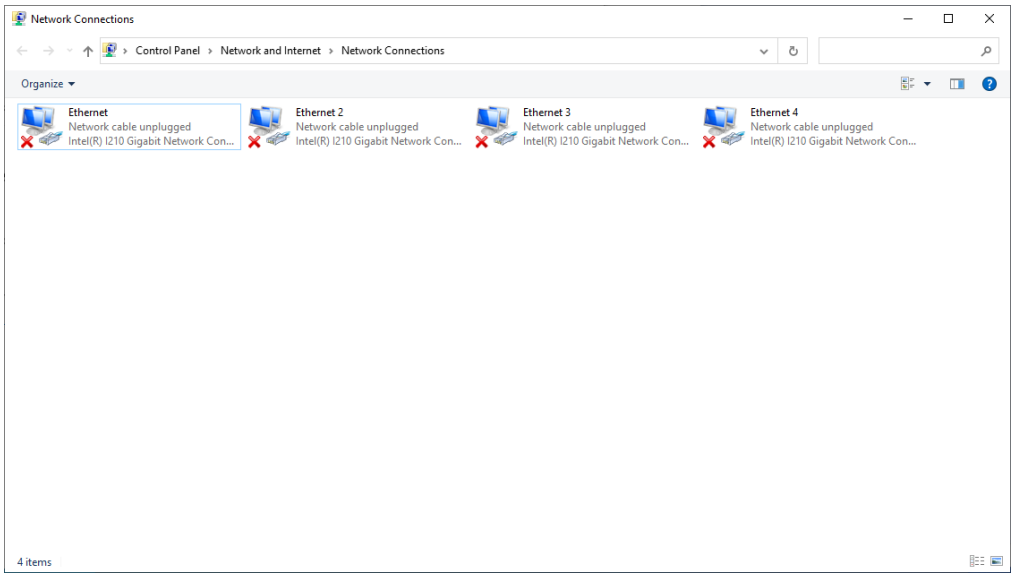
This section explains how to use the **Moxa Sort Net Name** utility to rename Ethernet adapters. This utility helps map the physical LAN port order on the chassis to the corresponding adapter names in the system.

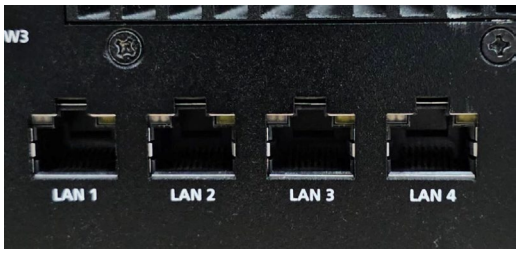
1. Use the pre-installed utility or install the **MoxaSortNetName** utility from the Moxa support page.
2. The initial order of network names may be random.
3. After the installation process is complete, run the **SortNetName.exe** from **C:\Program Files\Moxa\Moxa Sort Net Name** as an Administrator.




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).



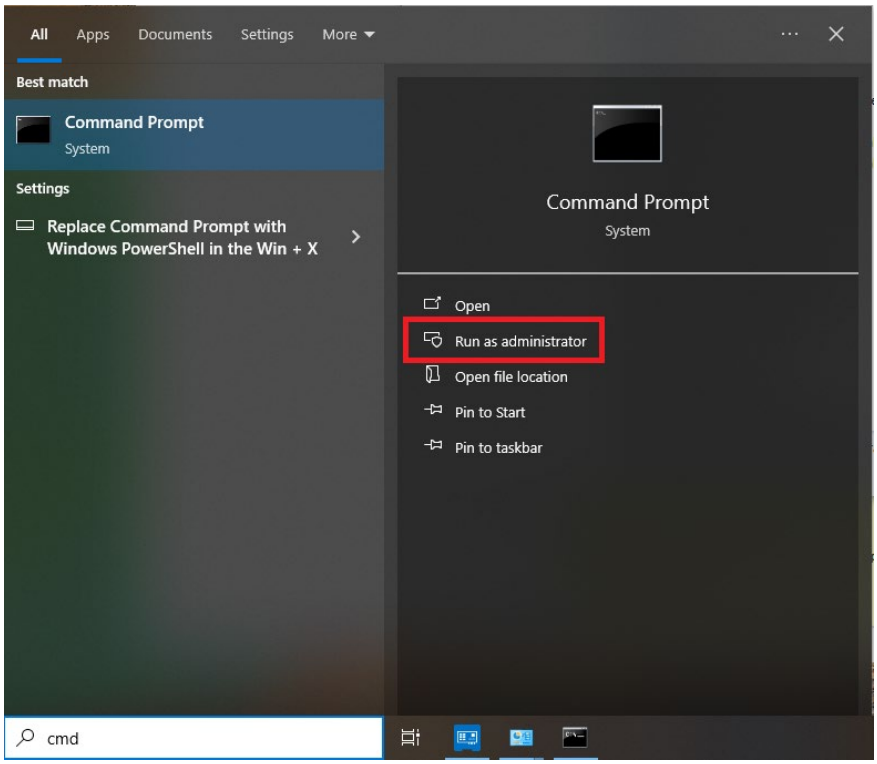



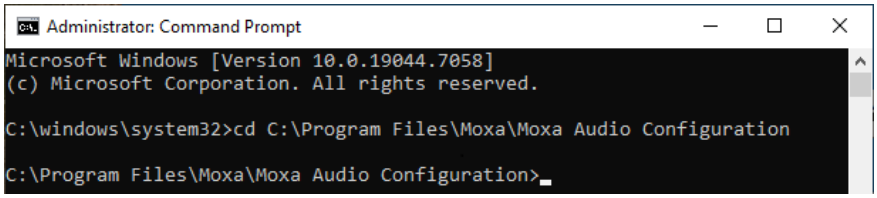
# Moxa Audio Configuration (only for MPC-3XX1 Series)

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

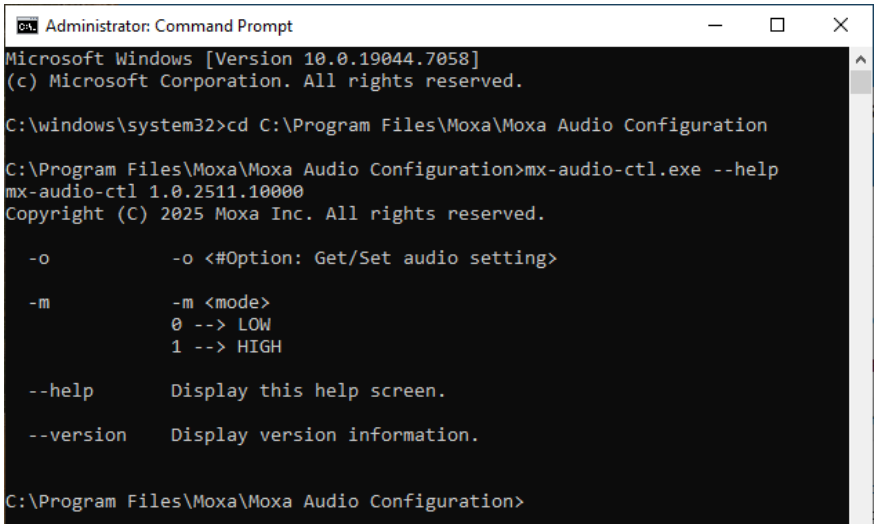
To run the utility, do the following:

2. Open the Command Prompt as Administrator and enter **C:\ProgramFiles\Moxa\Moxa Audio Configuration**.



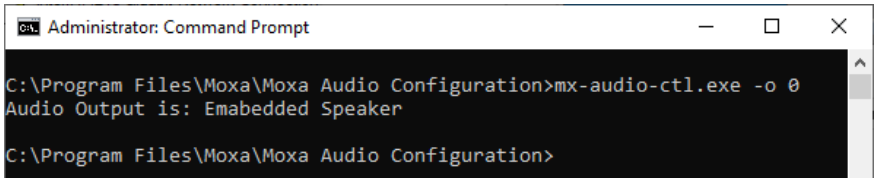

3. Run the **mx-audio-ctl --help** command to see instructions on using this utility and follow the onscreen instructions to get or set the audio ouput.



## Usage

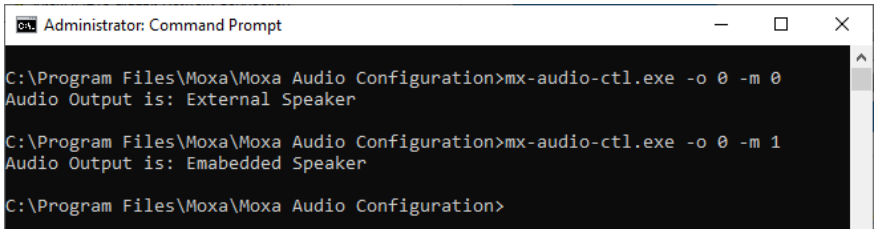
1. Run the following command to check the audio output.

```
mx-audio-ctl.exe -o 0
```




2. Run the following command to switch the audio output.

```
mx-audio-ctl.exe -o 0 -m <0:External Speaker, 1:Embedded Speaker>
```

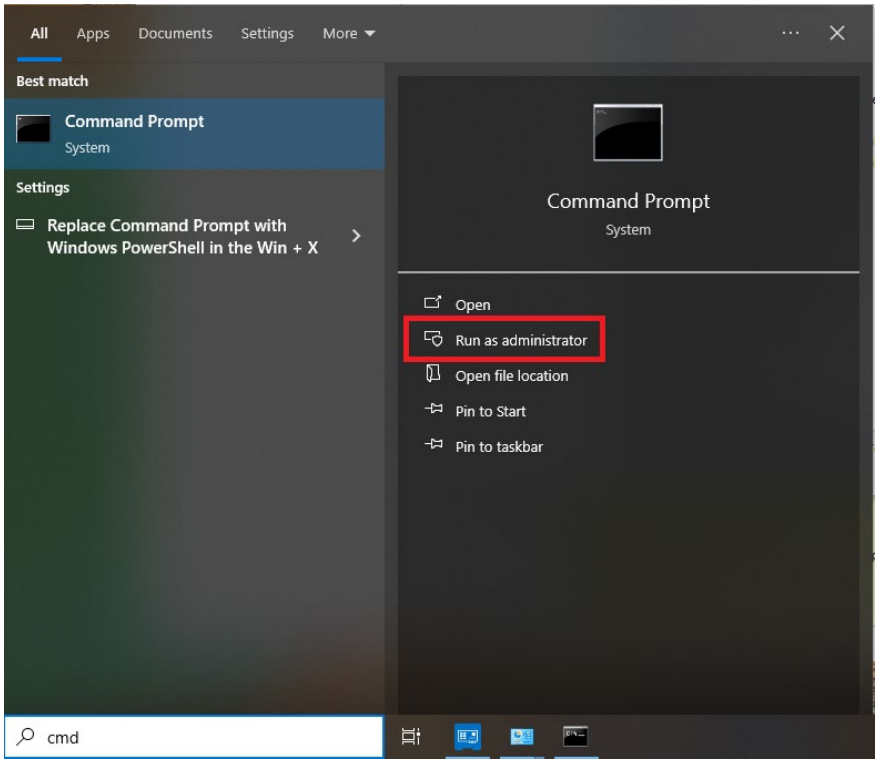


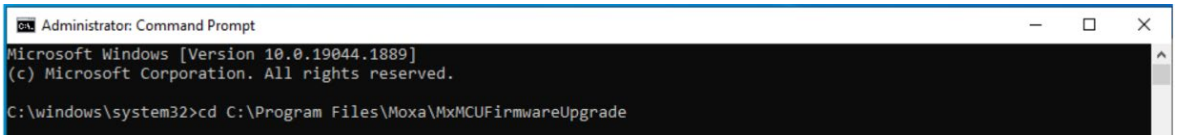

# Moxa MCU Firmware Upgrade

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

To run the utility, do the following:

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



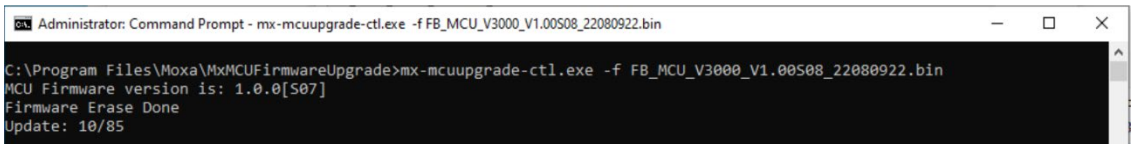



# Upgrade the MCU Firmware

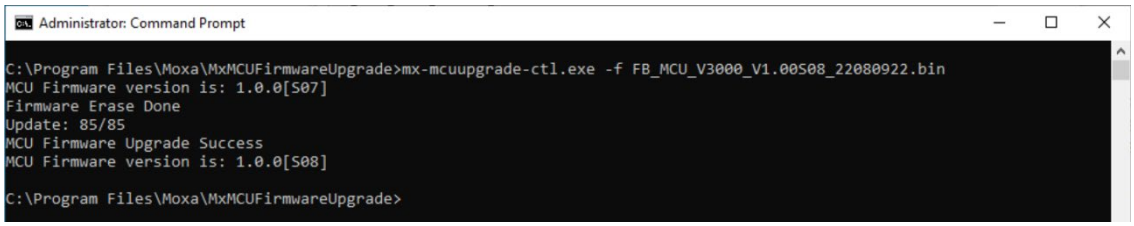
1. Download the lastest MCU firmware upgrade file from Moxa's support website at https://www.moxa.com/en/support.
2. Copy the update files in **C:\Program Files\Moxa\MxMCUFirmwareUpgrade**.
3. Run the following command as Administrator to upgrade MCU firmware in Command Prompt.

```
mx-mcuupgrade-ctl.exe -f <file_name>
```


4. Waiting for update



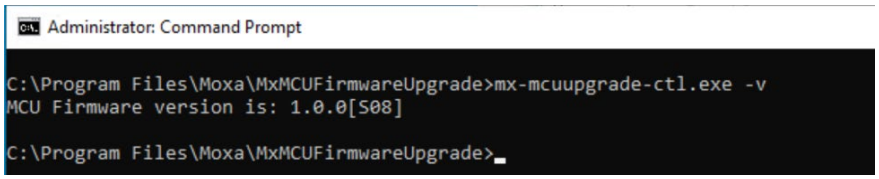

5. Update completed




# Check MCU firmware version

Run the following command to check current MCU firmware version.

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

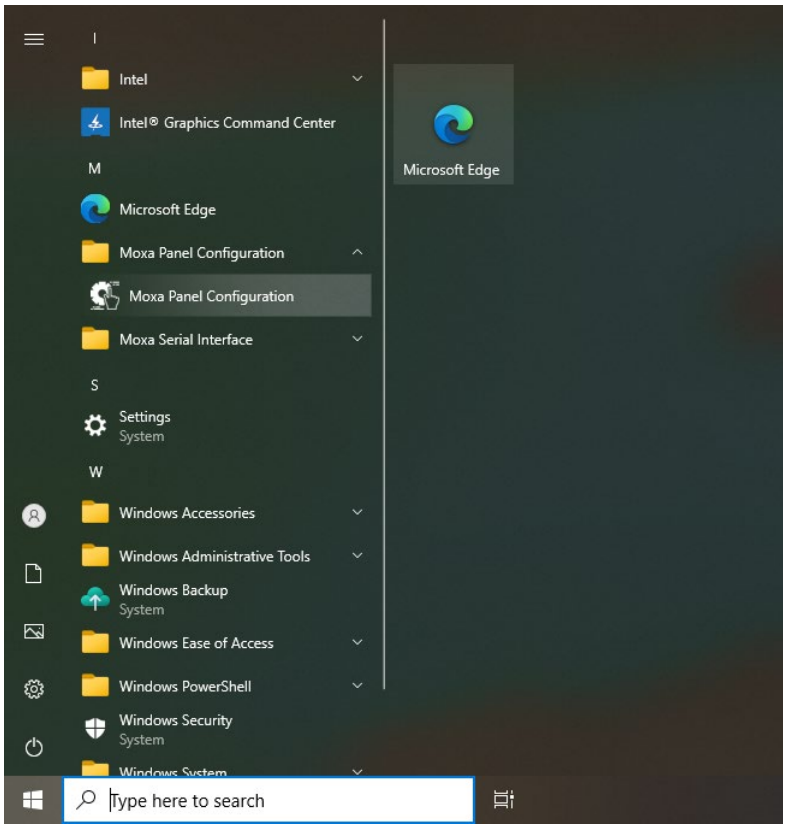


# Moxa Panel Configuration

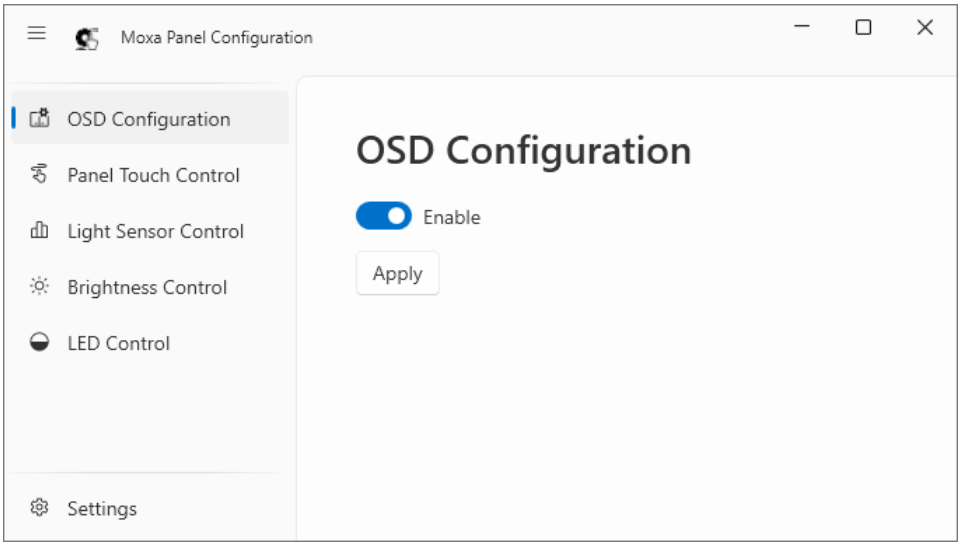
This toolkit provides utilities for performing panel-related operations, including both graphical user interface (GUI) tools and command-line tools.

## GUI utility

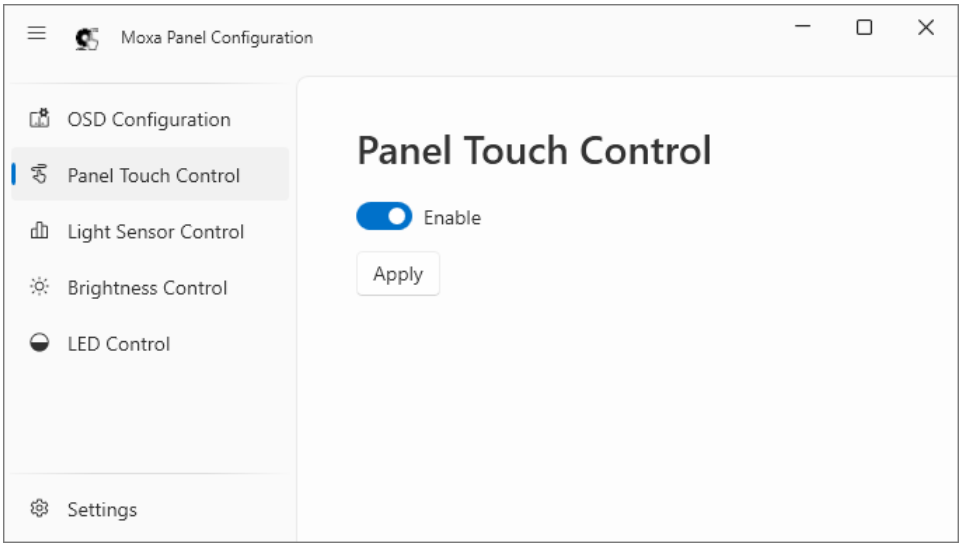
1. Install the Moxa Panel Configuration utility.
2. From the Windows Start menu, run the **Moxa Panel Configuration utility**.



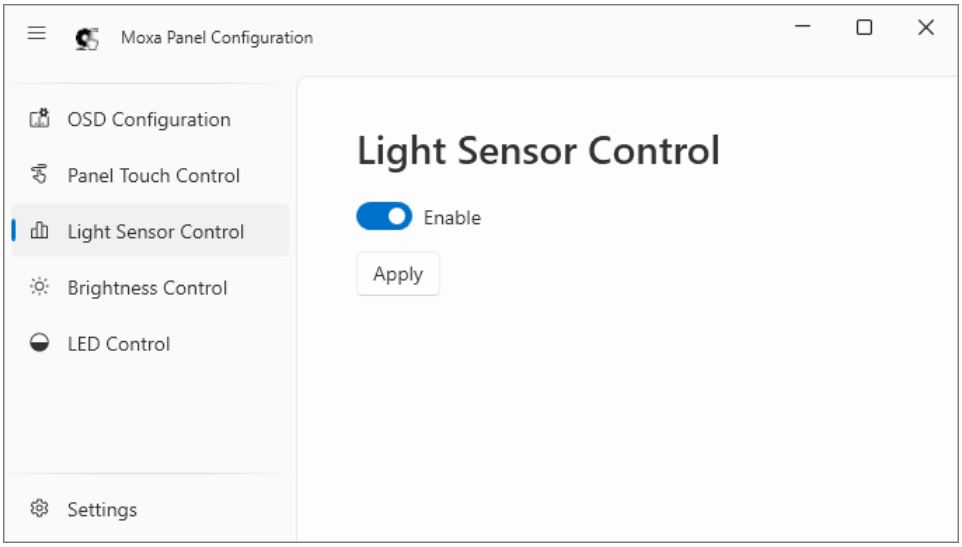

- **OSD configuration:** Configure whether a window appears when the brightness adjustment button is pressed. (Enabled by default)



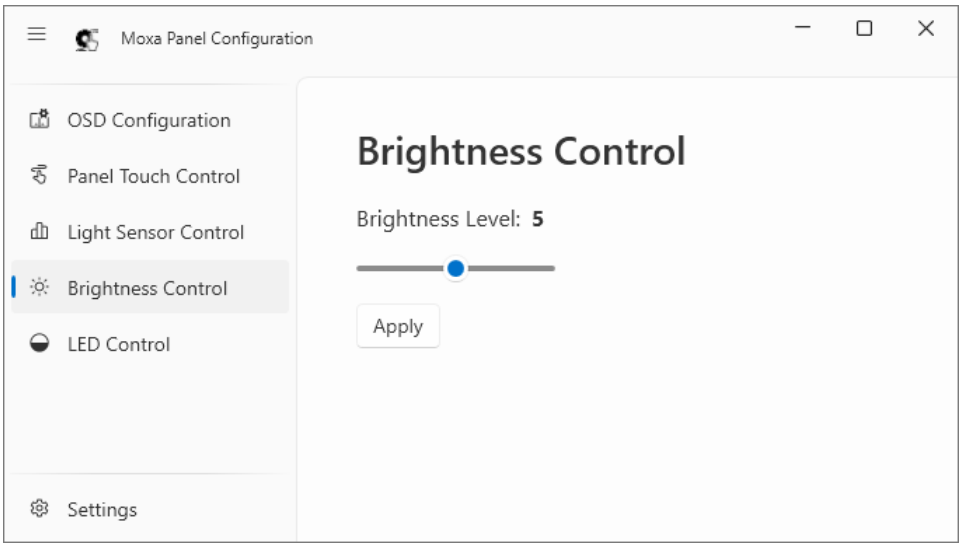
- **Panel Touch Control:** Enable or disable the touchscreen function. (Enabled by default)



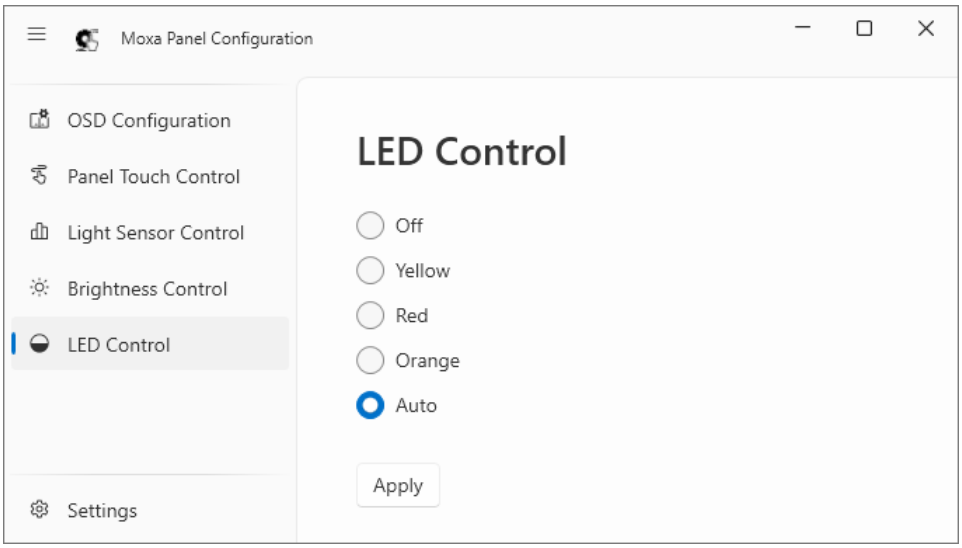

- **Light Sensor Control:** Enable or disable the light sensor. (Enabled by default)




- **Brighting Control:** Adjust the screen brightness. (Level 5 by default)

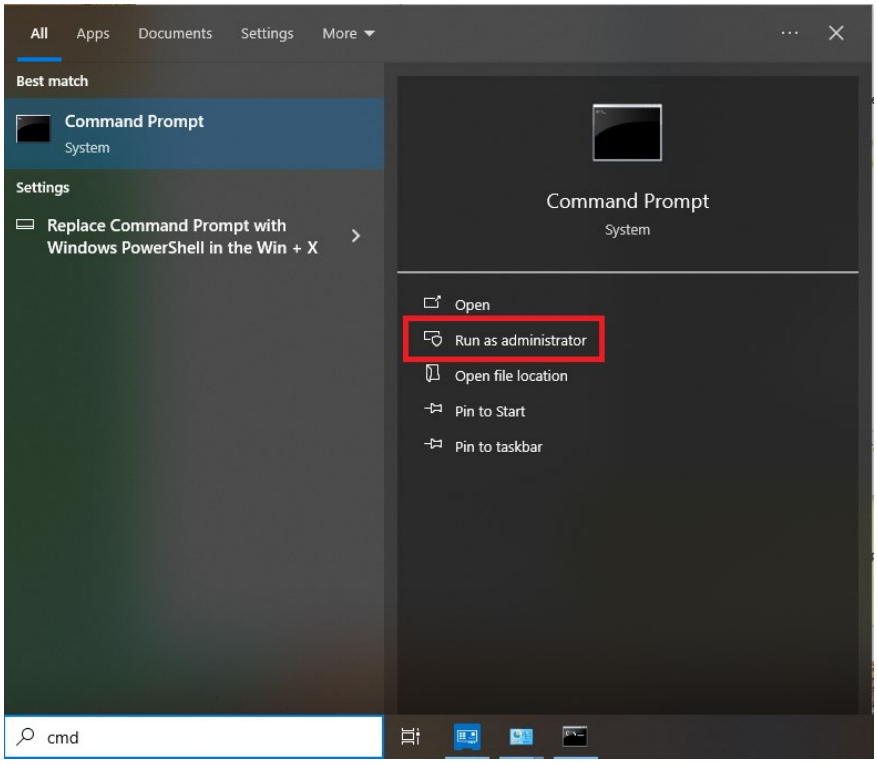


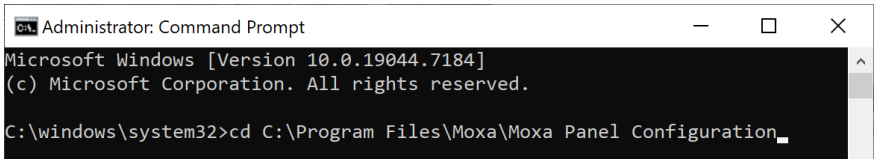
- **LED Control:** Control the LED status. (Auto by default) (Only for MPC-3XX1)



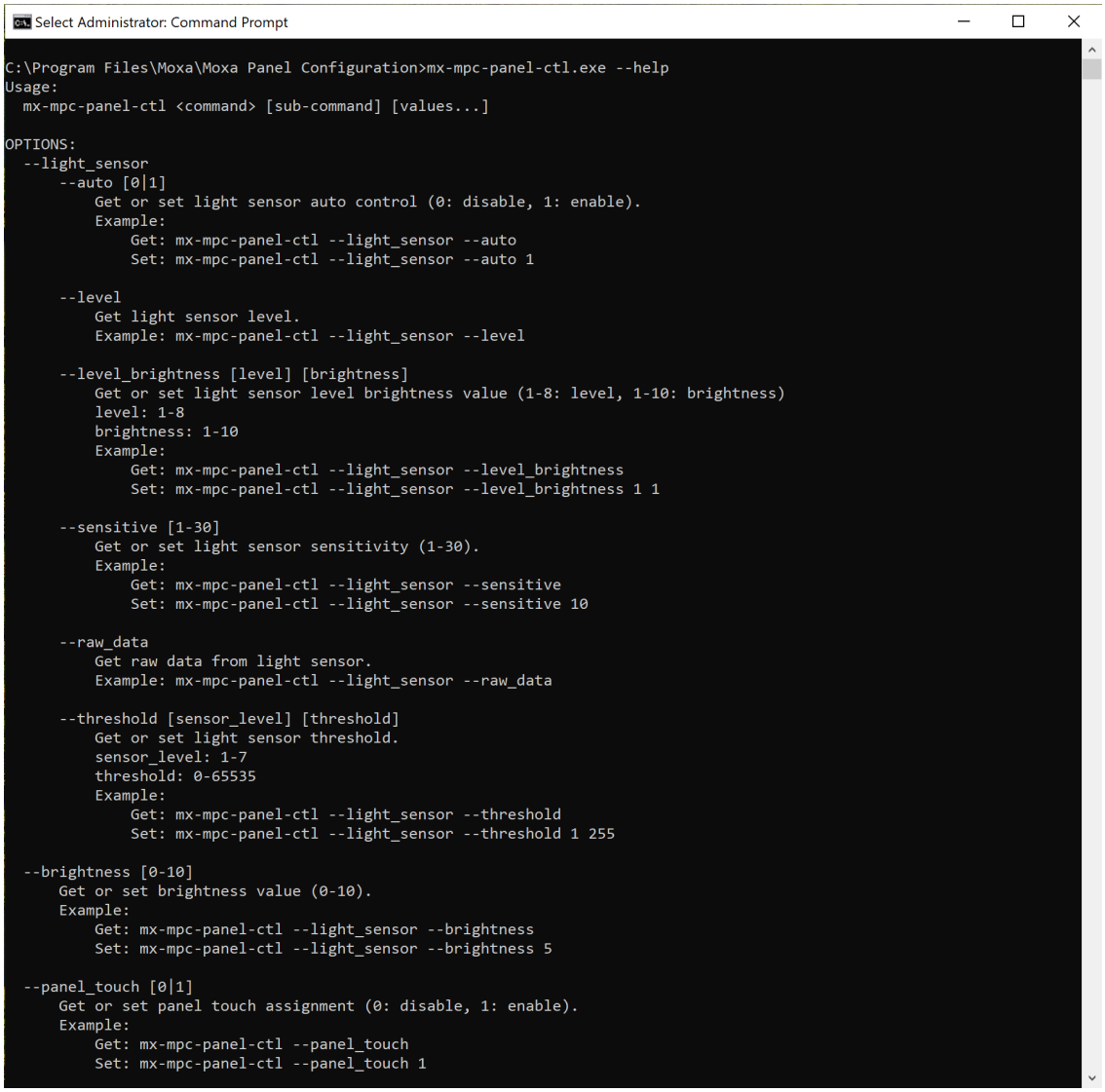

# Command Line Utility

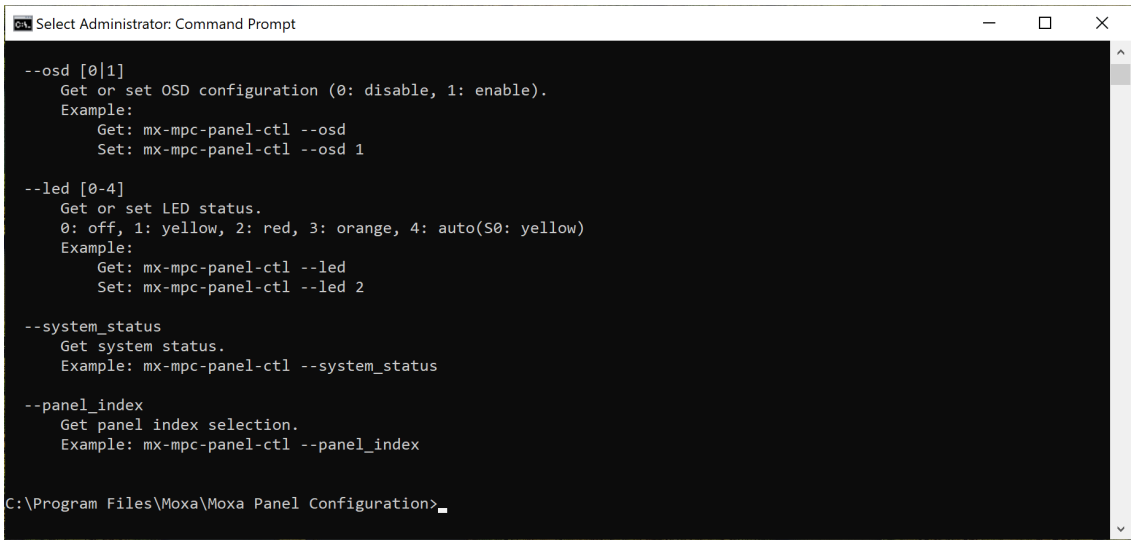
1. Open the Command Prompt as Administrator and enter **C:\ProgramFiles\Moxa\Moxa Panel Configuration**.



2. Run the **mx-mpc-panel-ctl --help** command to see instructions on using this utility and follow the onscreen instructions to adjust the panel settings.



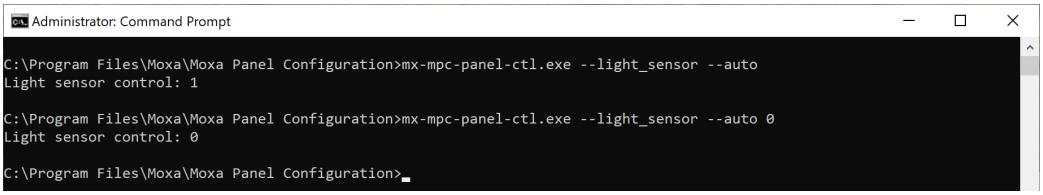



➢ **Light sensor**

Provides light sensor-related functions and can be used with sub-options such as `--brightness`, `--auto`, and `--threshold`.

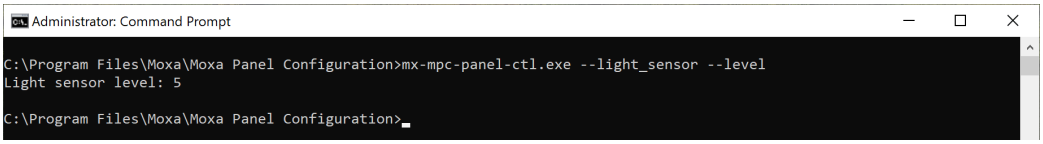
❒ **Auto**Get or set light sensor auto control (default value: 1 (enable))

```
mx-mpc-panel-ctl.exe --light_sensor --auto [0: disable | 1: enable]
```




❒ **Level**Get light sensor level

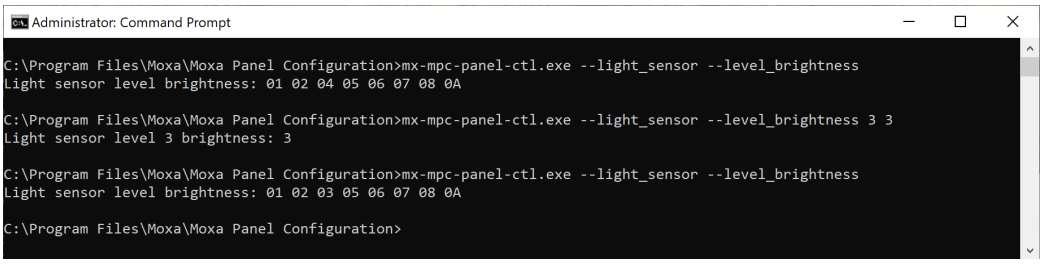
```
mx-mpc-panel-ctl.exe --light_sensor --level
```




❒ Level brightnessGet or set the brightness value for each light sensor level (default value as following fields)

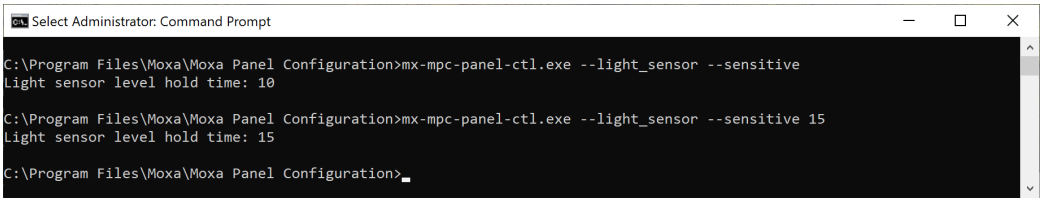
| level 1 | level 2 | level 3 | level 4 | level 5 | level 6 | level 7 | level 8 |
|---|---|---|---|---|---|---|---|
| 01 | 02 | 04 | 05 | 06 | 07 | 08 | 0A |

```
mx-mpc-panel-ctl.exe --light_sensor --level_brightness [level: 1-8]
[brightness: 1-10]
```



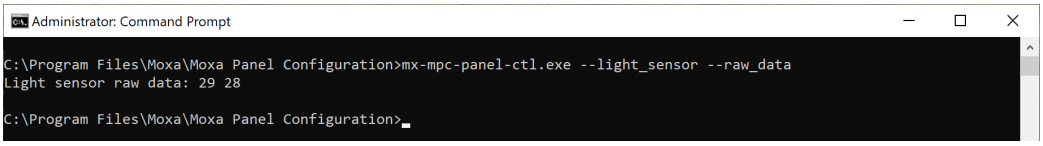

❒ **Sensitive**Get or set light sensor sensitivity (default value: 10)

```
mx-mpc-panel-ctl.exe --light_sensor --sensitive [value: 1-30]
```




❒ **Raw data**Get raw data from light sensor

```
mx-mpc-panel-ctl.exe --light_sensor --raw_data
```

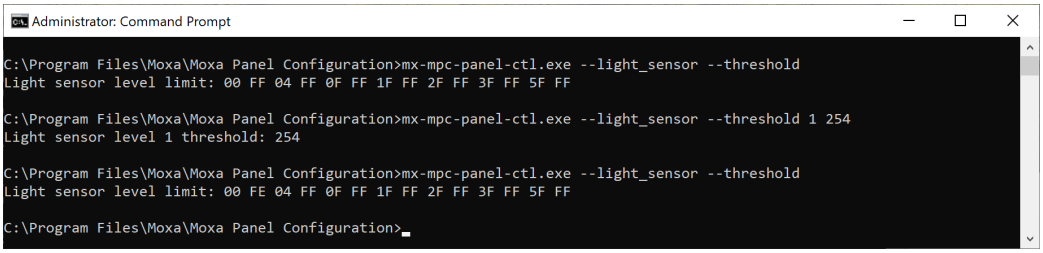


❒ **Threshold**Get or set light sensor threshold (default value as following fields)

The `threshold` parameter is a decimal value composed of the high byte and low byte of the level.

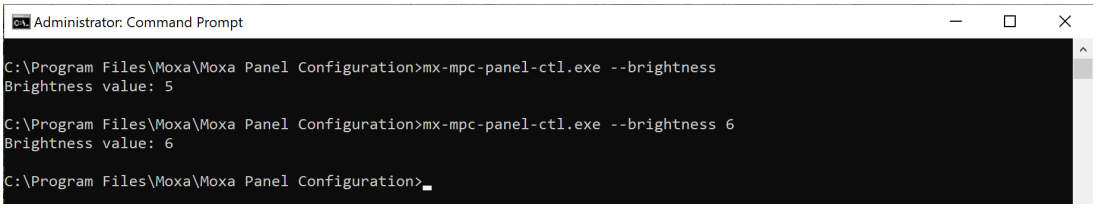
| level 1 DH | level 1 DL | level 2 DH | level 2 DL | level 3 DH | level 3 DL | level 4 DH | level 4 DL | level 5 DH | level 5 DL | level 6 DH | level 6 DL | level 7 DH | level 7 DL |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 00 | FF | 01 | FF | 03 | FF | 05 | FF | 1F | FF | 2F | FF | 3F | FF |

```
mx-mpc-panel-ctl.exe --light_sensor --threshold [sensor_level: 1-7]
[threshold: 0-65535]
```



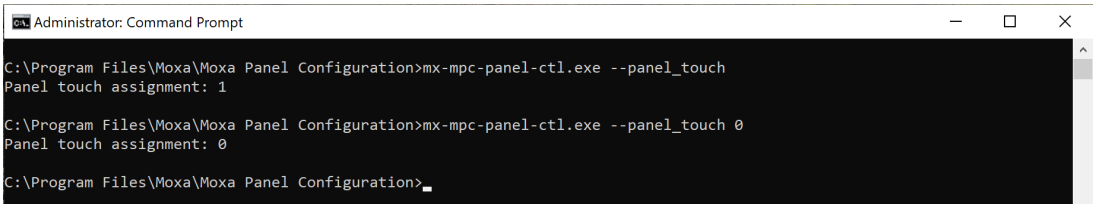

➢ **Brightness**Get or set panel brightness value (default value: 5)

```
mx-mpc-panel-ctl.exe --brightness [value: 0-10]
```



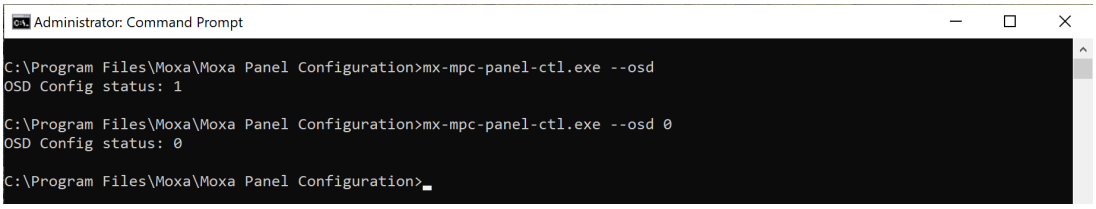

➢ **Panel touch**Get or set panel touch assignment (default value: 1 (enable))

```
mx-mpc-panel-ctl.exe --panel_touch [0: disable | 1: enable]
```



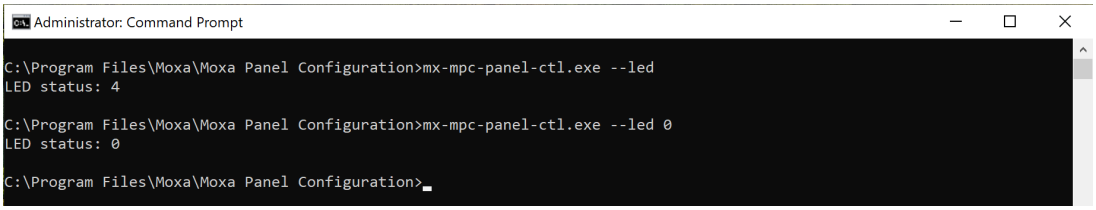

➢ **OSD**Get or set OSD configuration (default value: 1 (enable))

```
mx-mpc-panel-ctl.exe --osd [0: disable | 1: enable]
```




➢ **LED**Get or set LED status (default value: 4 (auto))

```
mx-mpc-panel-ctl.exe --led [0: off, 1: yellow, 2: red, 3: orange, 4:
auto(S0: yellow)]
```



- **System status**Get system status. The output is provided as a decimal value. When converted to binary, each bit corresponds to the following fields.

| bit0 | bit1 | bit2 | bit3 | bit4 | bit5 | bit6 | bit7 | bit8 |
|---|---|---|---|---|---|---|---|---|
| reserve | 0: Backlight Power OFF,<br>1: Backlight Power ON | 0: No display output,<br>1: Display output | reserve | 0: Light Sensor Error,<br>1: Light Sensor Normal | reserve | reserve | reserve | reserve |

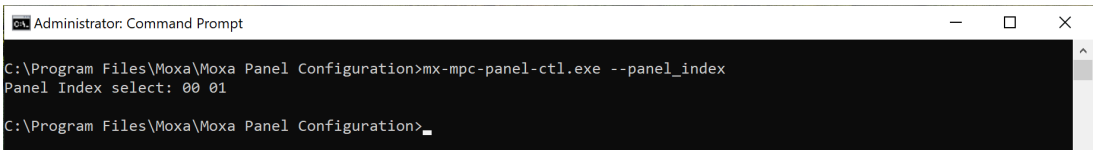
```
mx-mpc-panel-ctl.exe --system_status
```


- **Panel index**Get panel index selection

  First value: Screen size — **00**: 10-inch, **01**: 12-inch

  Second value: Screen brightness — **00**: Standard brightness, **01**: High brightness

```
mx-mpc-panel-ctl.exe --panel_index
```

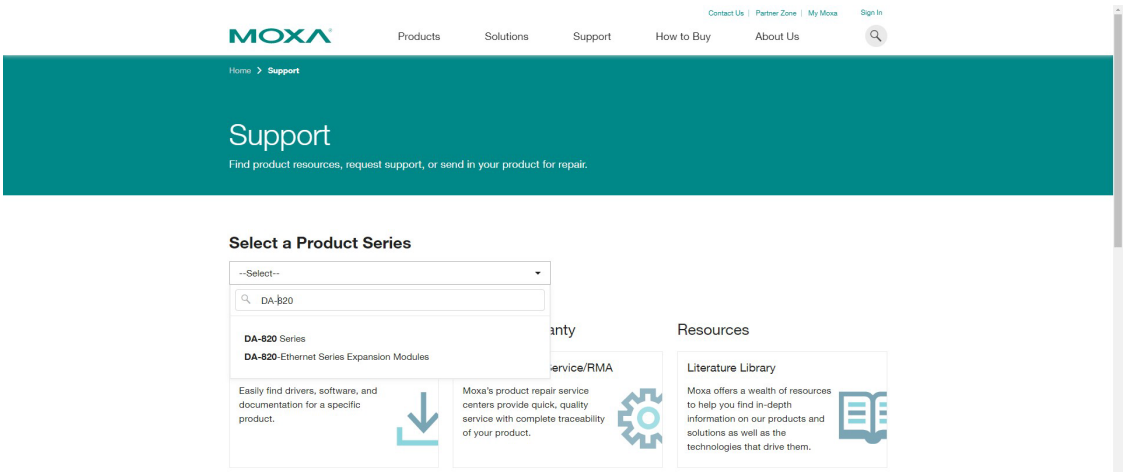


# 8. IO Control API

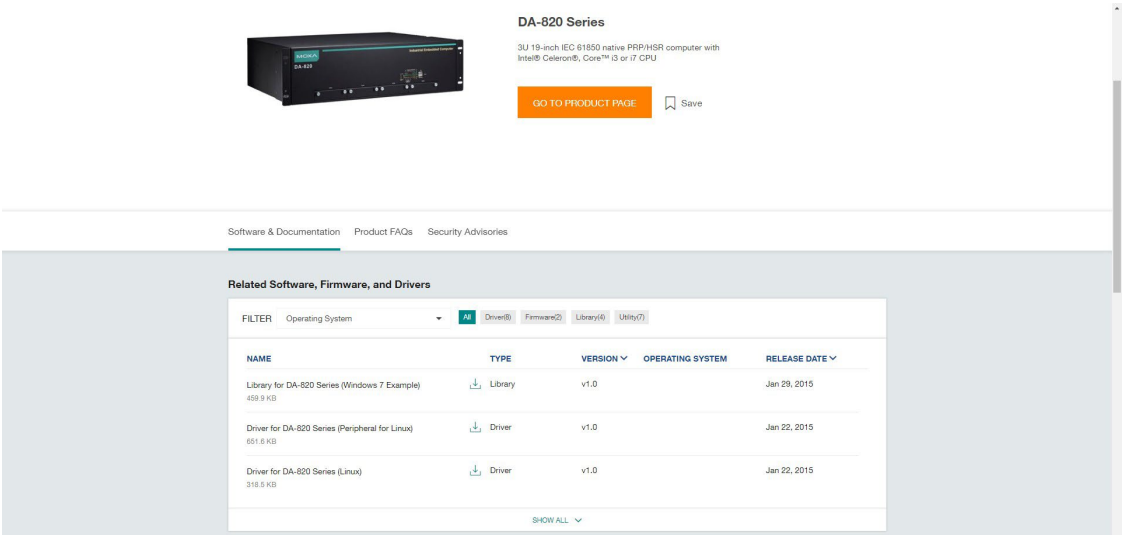
This chapter describes how to use the IO Control API.

# Downloading the API

1. Go to https://www.moxa.com/en/support.
2. Select your product series (e.g., 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 |

# mxpcie (only for MPC-3XX1 Series)

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

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

## 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 (only for MPC-3XX1 Series)

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 |

# 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 seconds.

**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 |

# 9. System Backup Restore

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

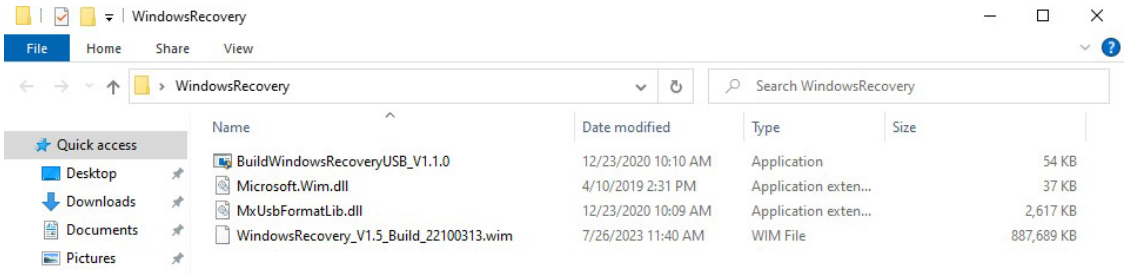
- **WindowsRecovery**

# WindowsRecovery

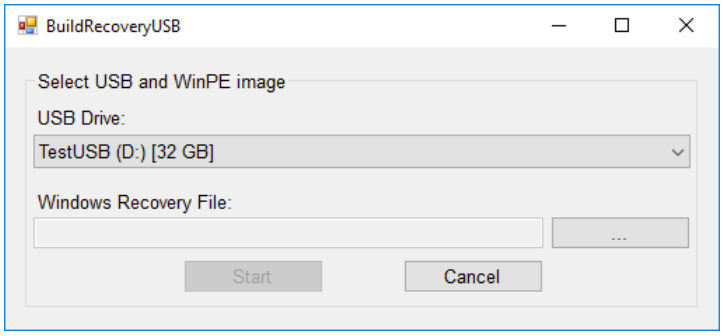
WindowsRecovery is an OS image backup and restore program for system deployment, backup, and recovery. You will first need to create a WindowsRecovery USB disk. This WindowsRecovery disk can only be used to boot a **UEFI BIOS** machine. This chapter describes the setup process of the Windows Recovery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

## Preparing the USB device

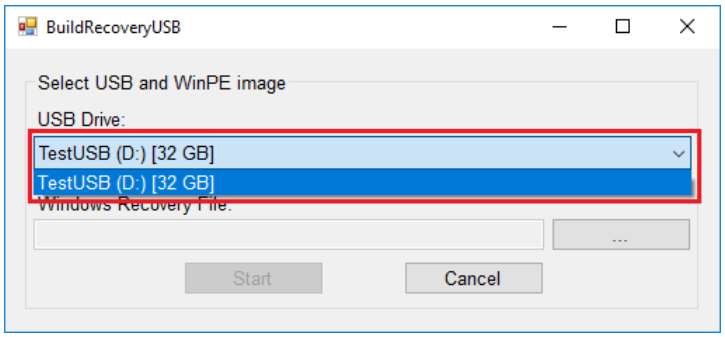
1. Contact a Moxa technical staff and get the required file.



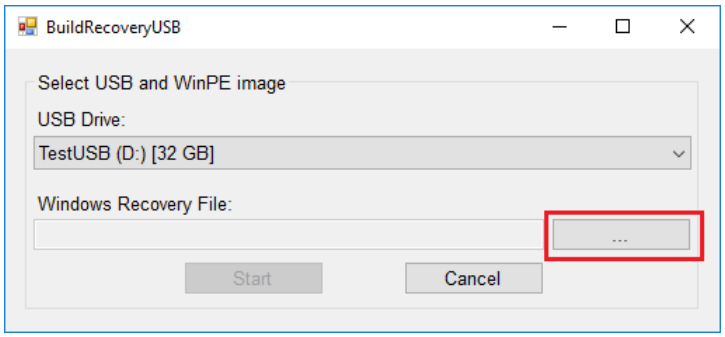

2. Run the **BuildWindowsRecoveryUSB_V1.2.0.exe**.

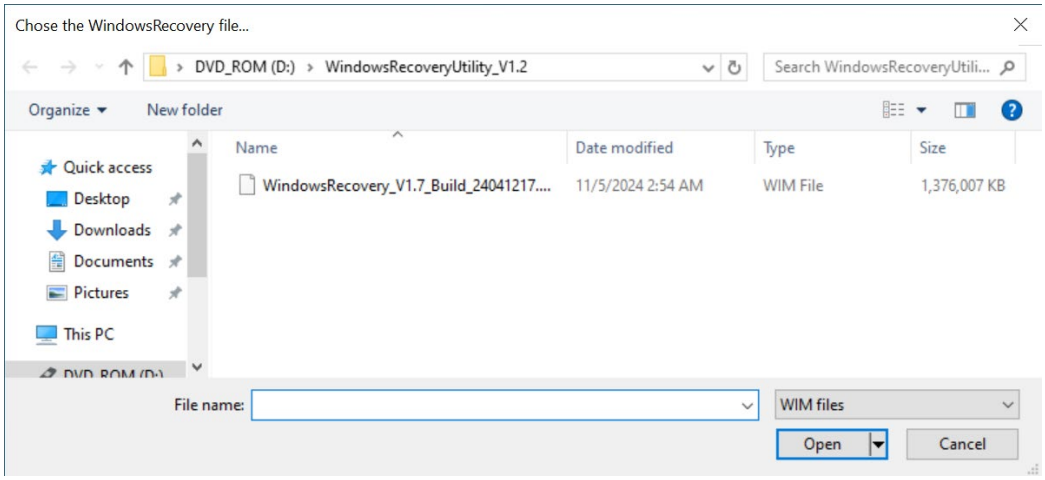



3. Select the USB drive to format.

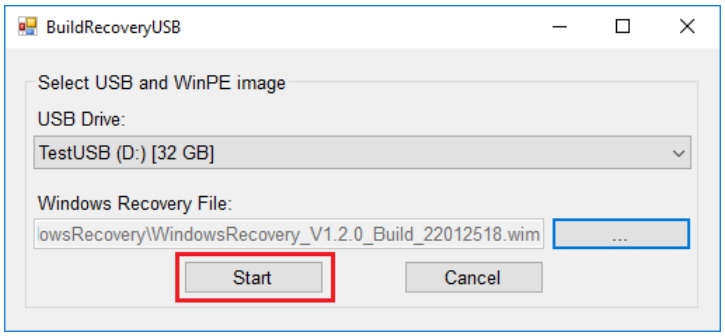


4. Click **...** to select **.wim** file from the folder.

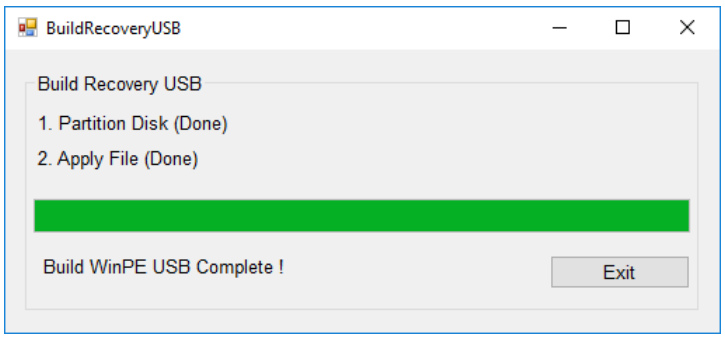


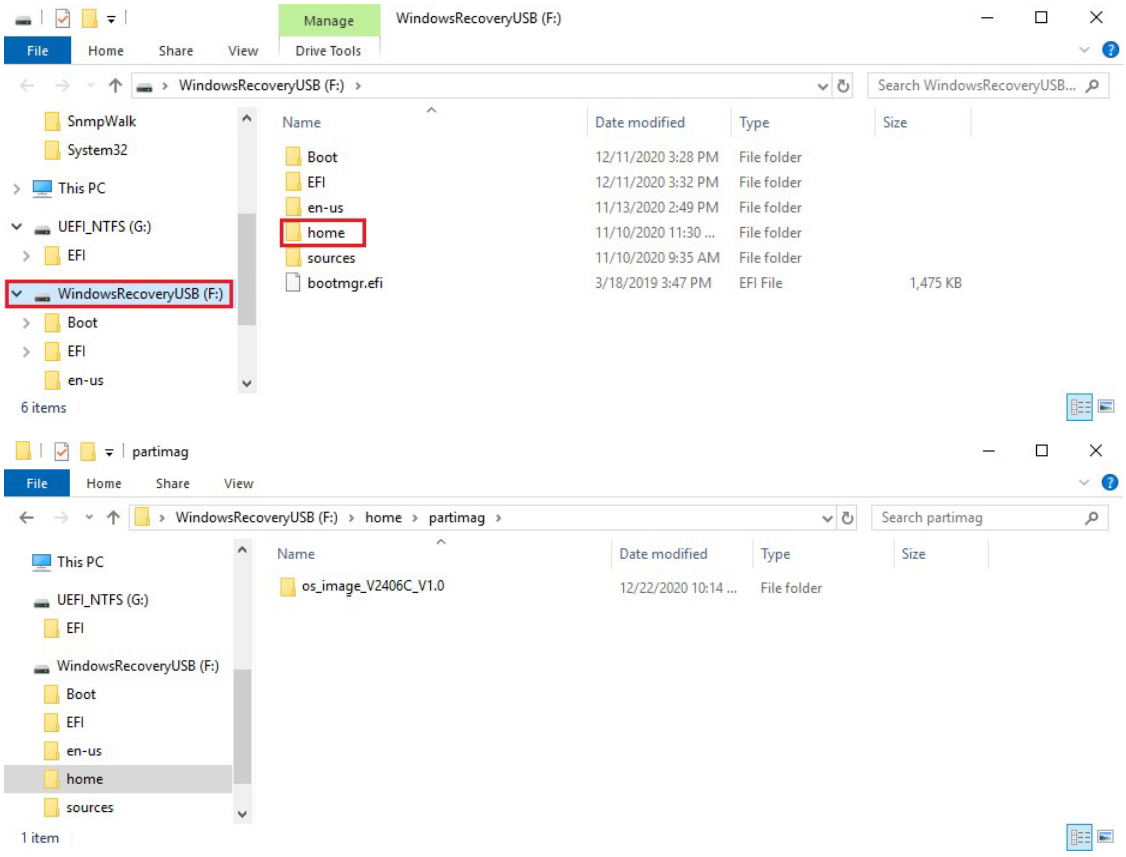

5. Click **Start** and make sure the selected USB can be formatted. Click **Yes** to start creating the recovery USB.



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.

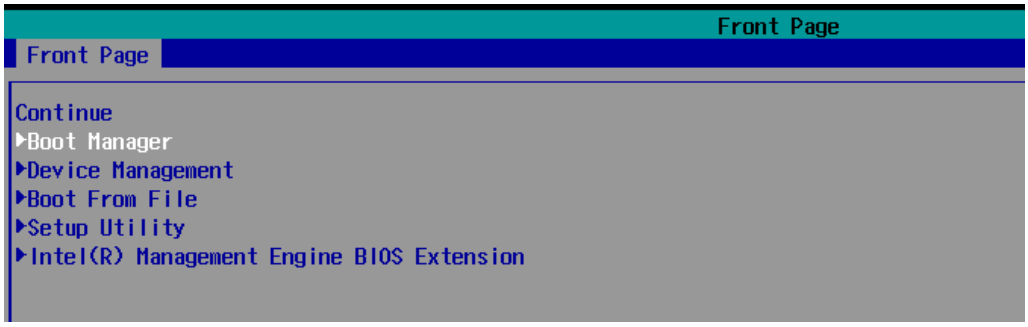



7. To create a recovery USB disk with the Windows 10 image, copy the **os_image_ModelName** directory to the **\home\partimag** folder in the USB drive.

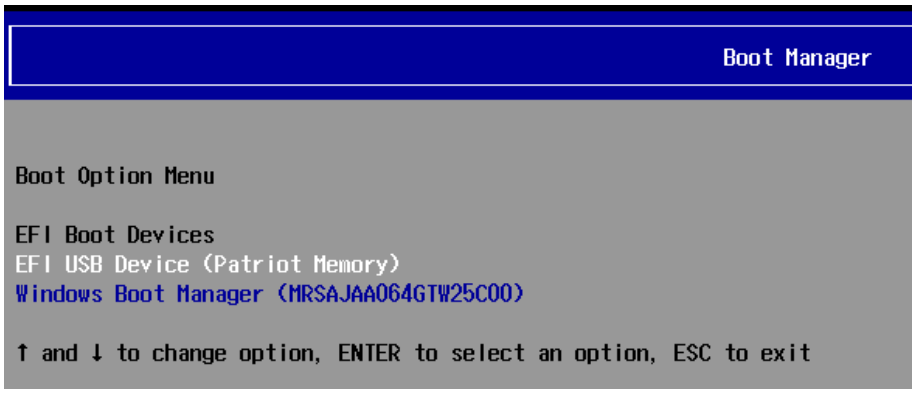


# Booting From the USB Disk

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

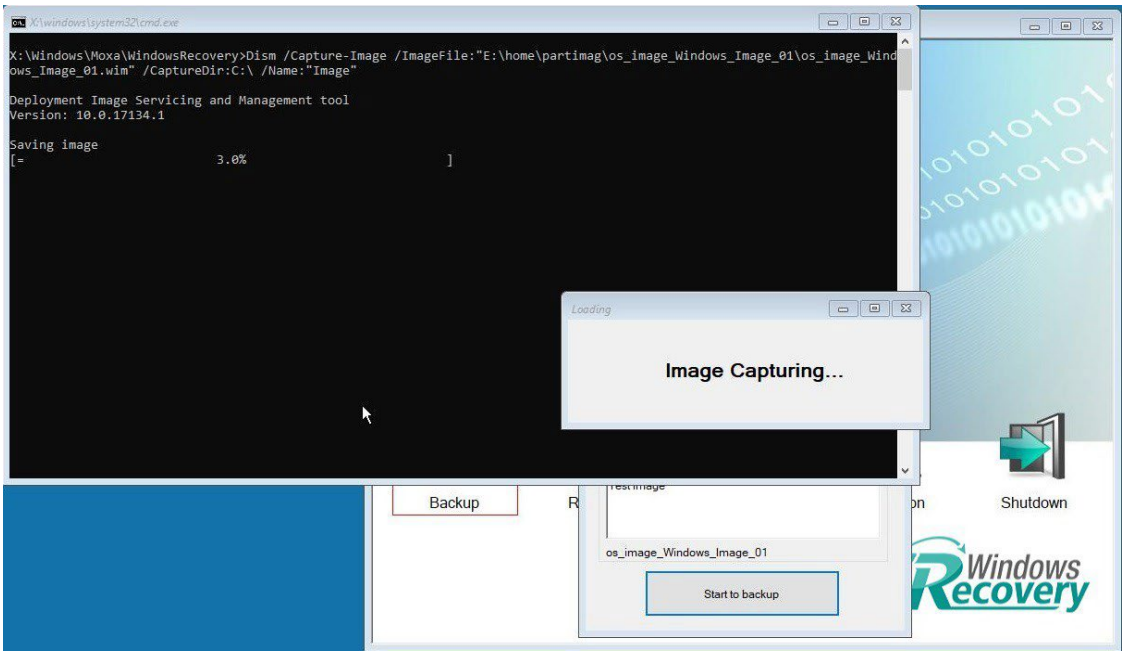
To back up the image from the USB disk, run **Windows Preinstallation Environment(WinPE)** and the **Windows Recovery utility** will display. Follow these steps.

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.

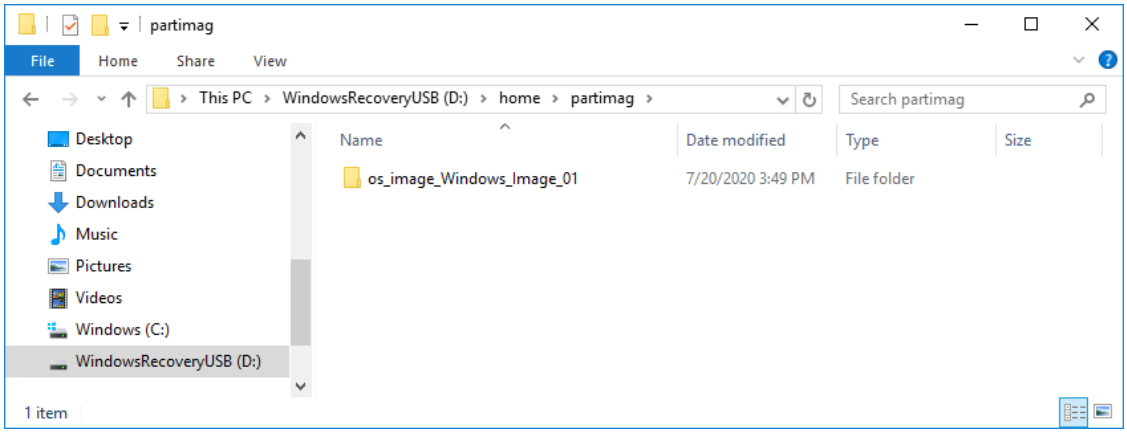

4. Wait for the backup process to complete.



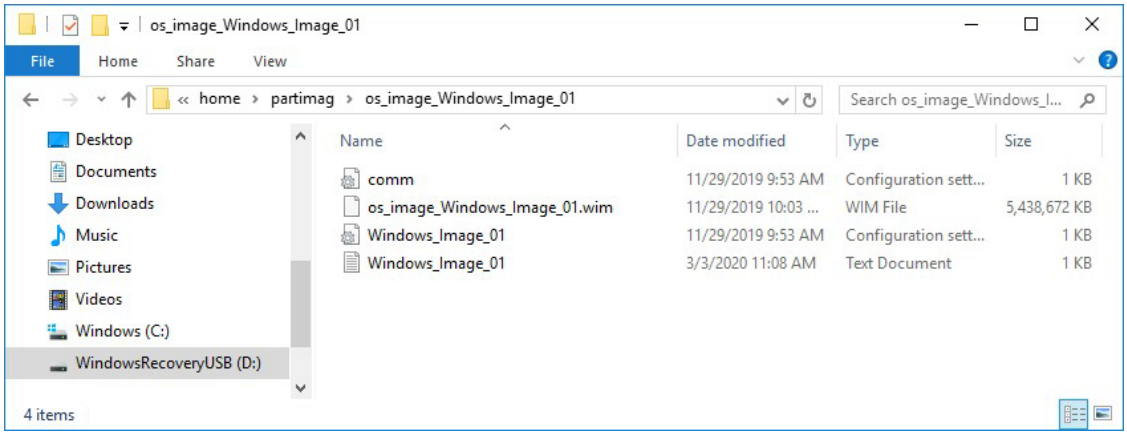
5. When the process is done, click **OK**.


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, we recommend restoring the image to the destination drive using 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.