Proactive Monitoring Software
User Manual

Version 3.0, June 2025

www.moxa.com/products

Proactive Monitoring Software 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

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

Installing Moxa Proactive Monitoring Software

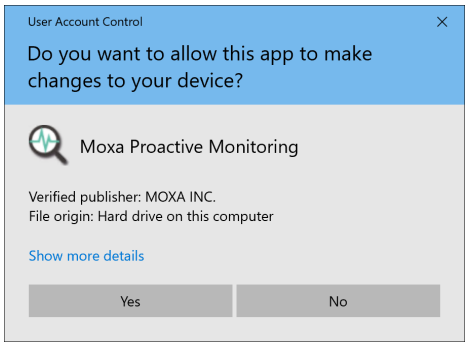
This section provides detailed instructions on installing Proactive Monitoring v3.0 and dependent software.

Prerequisites

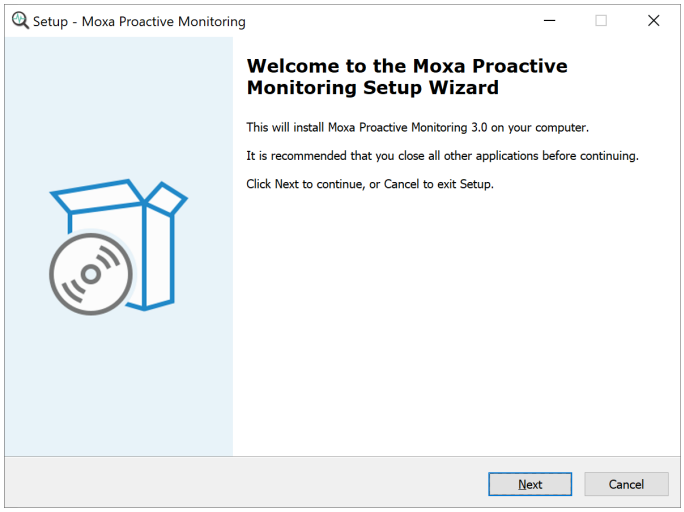
1. Download and install the necessary **drivers** for your target platform from the **Moxa official website**. The drivers are **Moxa-certified versions** that have undergone rigorous system testing before being officially released.
2. Verify that the target platform has the following software and tools required by **Proactive Monitoring V3.0**:
   a. Moxa GeneralIO
   b. Microsoft Visual C++ 2022 x64 Redistributable
   c. .NET 8 Runtime

To install the Proactive Monitoring software, do the following:

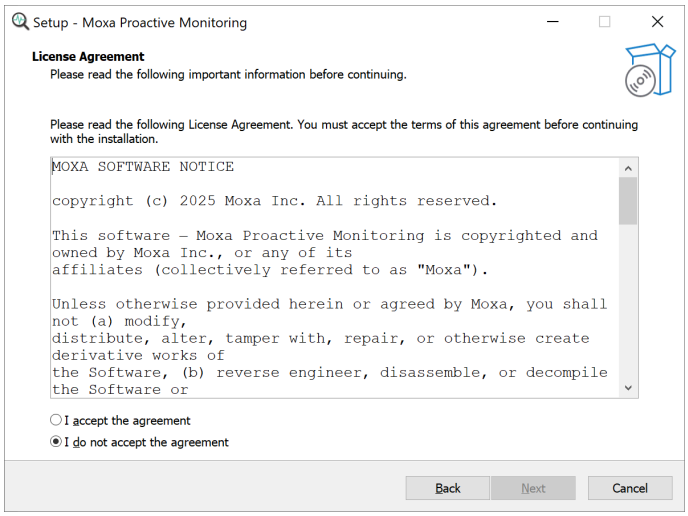
1. Click on **MoxaProactiveMonitoring_V3.0.exe** file to open the file and then click **Yes** to continue the installation.



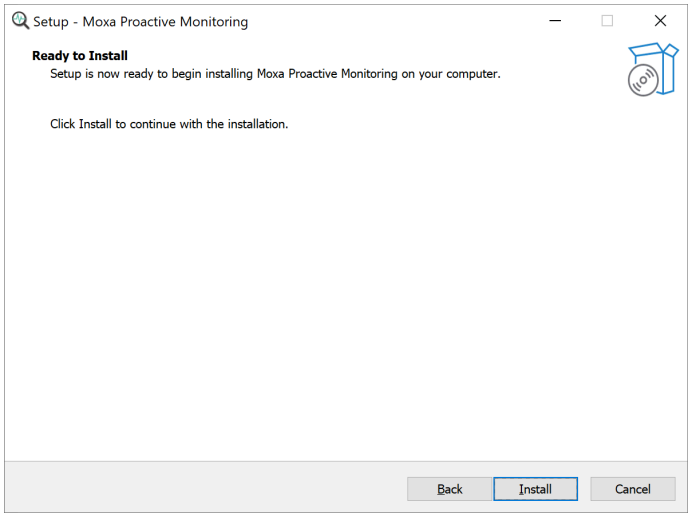
2. In the "Welcome..." window, click **Next** to proceed with the installation.
   If you want to quit the installation process, click **Cancel**.



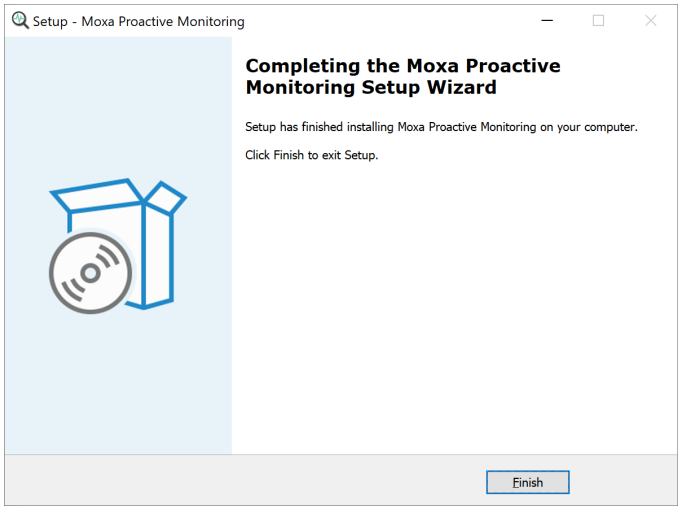
3. In the "License Agreement" window, select **I accept the agreement** and click **Next** to continue with the installation.



4. Click Install to start installing the software.



5. Wait for the installation to complete and click **Finish** to exit the installer.



Monitoring the System Status

Proactive Monitoring provides real-time system monitoring capabilities to detect the status of key resources such as CPU, memory, hard drive, network, and power. This helps users understand the system's operational status. Furthermore, it integrates the detected status information into Windows Management Instrumentation (WMI), allowing users to easily access the data through native Windows tools like PowerShell, WMIC, and other applications and monitoring tools. The following sections will detail the key system resources and their statuses that can be monitored.

CPU

Monitors CPU usage to ensure your system is running smoothly and helps avoid overload.

WMI Location

Namespace	Class	Property	Description	Data Type	Update Frequency
root\MoxaIPC	PM_CPU	Name	WMI Instance Name, default is "Total"	String	10 sec
root\MoxaIPC	PM_CPU	ProcessorCount	Number of logical processors	Int	10 sec
root\MoxaIPC	PM_CPU	Utilization	Overall CPU Utilization, value range is 0% - 100%	Int	10 sec

Disk

Monitors the usage and status of disk slots in the system, confirming if a hard drive is inserted into the slot and its health status.

WMI Location

Namespace	Class	Property	Description	Data Type	Update Frequency
root\MoxaIPC	PM_Disk	EraseCount	The number of times the hard drive has been written to. A high number of writes may lead to hard drive data corruption. If -1 is returned, it indicates that reading the write count is not supported.	Int	15 sec
root\MoxaIPC	PM_Disk	HealthStatus	Hard drive health status; returns the status according to the following configurations: "Good" - Hard drive is normal. "Warning" - Hard drive has issues. "No Disk" - No hard drive is currently inserted in this Disk Slot.	String	15 sec
root\MoxaIPC	PM_Disk	IsExist	Disk Slot has a hard drive inserted; returns the status according to the following configurations: "True" - A hard drive is currently inserted in this Disk Slot. "False" - No hard drive is currently inserted in this Disk Slot.	Bool	15 sec
root\MoxaIPC	PM_Disk	LocationID	Disk Slot's SCSI Location ID on the system.	Int	15 sec
root\MoxaIPC	PM_Disk	Model	Hard drive model information; returns Null if no hard drive is inserted.	String	15 sec
root\MoxaIPC	PM_Disk	Name	Disk Slot's SCSI location name on the system.	String	15 sec
root\MoxaIPC	PM_Disk	PortNumber	Disk Slot number on the device.	Int	15 sec
root\MoxaIPC	PM_Disk	SerialNumber	Disk Serial Number; returns Null if no hard drive is inserted	String	15 sec
root\MoxaIPC	PM_Disk	SlotStatus	Disk Slot status; returns the status according to the following configurations: "Active" - A hard drive is detected in the Disk Slot and it is in a normal state. "No Disk" - No hard drive was inserted in this Disk Slot before the service started. "Unplugged" - The hard drive in this Disk Slot was removed after the service started.	String	15 sec

Memory

Monitors memory usage to ensure your system runs smoothly and helps avoid overload.

WMI Location

Namespace	Class	Property	Description	Data Type	Update Frequency
root\MoxaIPC	PM_Memory	AvailableSize	The remaining available capacity of Memory, in MB	Int	60 sec
root\MoxaIPC	PM_Memory	Name	The name of the WMI Instance; Default is "Physical Memory"	String	60 sec
root\MoxaIPC	PM_Memory	TotalSize	The total capacity of Memory, in MB	Int	60 sec
root\MoxaIPC	PM_Memory	Utilization	Memory usage, the value range is 0% - 100%	Int	60 sec

Network

Monitors network usage, connection speed, and connection status to ensure the system runs smoothly and helps avoid overload.

WMI Location

Namespace	Class	Property	Description	Data Type	Update Frequency
root\MoxaIPC	PM_Network	ConnectionStatus	Network Adapter connection status Display "Connected" or "Not Connected"	String	30 sec
root\MoxaIPC	PM_Network	Description	The name of the Network Adapter displayed in Device Manager	String	30 sec
root\MoxaIPC	PM_Network	LinkSpeed	The current connection speed of the Network Adapter Displays "10", "100", or "1000"	Int	30 sec
root\MoxaIPC	PM_Network	Name	Connection ID of Network Adapter	String	30 sec
root\MoxaIPC	PM_Network	PNPDeviceID	Network Adapter's PNPDeviceID	String	30 sec
root\MoxaIPC	PM_Network	Utilization	Network Adapter usage, value range The range is 0% - 100%	Int	30 sec

Partition

Monitors the usage and remaining available space in system partitions to ensure that the system has sufficient storage space.

WMI Location

Namespace	Class	Property	Description	Data Type	Update Frequency
root\MoxaIPC	PM_Partition	AvailableSize	The remaining available capacity of the partition, in MB	Int	60 sec
root\MoxaIPC	PM_Partition	DriveType	Drive Type	String	60 sec
root\MoxaIPC	PM_Partition	Name	Drive Letter	String	60 sec
root\MoxaIPC	PM_Partition	TotalSize	The total capacity of the Partition, in MB	Int	60 sec
root\MoxaIPC	PM_Partition	Utilization	Partition usage, the value range is 0% - 100%	Int	60 sec

Power Indicator

Monitors the usage and status of the power modules to ensure uninterrupted system operation.

WMI Location

Namespace	Class	Property	Description	Data Type	Update Frequency
root\MoxaIPC	PM_PowerIndicator	Name	The name of the WMI Instance Default is "Power Indicator"	String	60 sec
root\MoxaIPC	PM_PowerIndicator	PowerModule1	Status of Power module 1 If the feature is supported and working properly, it will show "Active" If this feature is supported and has failed, "Failed" will be displayed. If this feature is not supported, null is displayed.	String	60 sec
root\MoxaIPC	PM_PowerIndicator	PowerModule2	Status of Power module 2 If the feature is supported and working properly, it will show "Active" If this feature is supported and has failed, "Failed" will be displayed. If this feature is not supported, null is displayed.	String	60 sec
root\MoxaIPC	PM_PowerIndicator	Support	Is this feature supported?	Bool	60 sec

Processor

Monitors the usage of the logical processor to ensure system runs smoothly and helps avoid overload.

WMI Location

Namespace	Class	Property	Description	Data Type	Update Frequency
root\MoxaIPC	PM_Processor	Name	Processor ID	String	10 sec
root\MoxaIPC	PM_Processor	Utilization	Monitor logical processor usage to ensure system runs smoothly and avoid overload.	Int	10 sec

RAID

Monitors the settings and status of Intel® RST RAID and Microsoft SW RAID to ensure normal system operation.

WMI Location

Namespace	Class	Property	Description	Data Type	Update Frequency
root\MoxaIPC	PM_RAID	Mode	RAID mode	String	10 sec
root\MoxaIPC	PM_RAID	Name	RAID volume name	String	10 sec
root\MoxaIPC	PM_RAID	RedundancyStatus	RAID volume redundancy status	String	10 sec
root\MoxaIPC	PM_RAID	Type	Intel RST RAID Microsoft SW RAID	String	10 sec

RAID Disk

Monitor the usage and status of the disks that make up the Intel® RST RAID to ensure that the disks are operating normally.

WMI Location

Namespace	Class	Property	Description	Data Type	Update Frequency
root\MoxaIPC	PM_RAIDDisk	Firmware	Disk Firmware Version	String	10 sec
root\MoxaIPC	PM_RAIDDisk	Mode	Disk model name	String	10 sec
root\MoxaIPC	PM_RAIDDisk	Name	Intel RST disk ID	String	10 sec
root\MoxaIPC	PM_RAIDDisk	Port	Intel RST RAID port	Int	10 sec
root\MoxaIPC	PM_RAIDDisk	RAIDVolume	Intel RST RAID volume name	String	10 sec
root\MoxaIPC	PM_RAIDDisk	SerialNumber	Disk serial number	String	10 sec
root\MoxaIPC	PM_RAIDDisk	Size	Disk capacity	String	10 sec
root\MoxaIPC	PM_RAIDDisk	Status	Disk current status If the disk status is normal, it will display "Normal" If the Disk status is abnormal, "Missing" will be displayed.	String	10 sec

Serial Port

Monitors the usage and related settings of the serial ports.

WMI Location

Namespace	Class	Property	Description	Data Type	Update Frequency
root\MoxaIPC	PM_SerialPort	BaudRate	Baud rate	Int	60 sec
root\MoxaIPC	PM_SerialPort	BitsPerByte	Bits per byte	Int	60 sec
root\MoxaIPC	PM_SerialPort	InstanceName	Device Manager instance name	String	60 sec
root\MoxaIPC	PM_SerialPort	IsBusy	Serial port status	Bool	60 sec
root\MoxaIPC	PM_SerialPort	Name	Serial port name	String	60 sec
root\MoxaIPC	PM_SerialPort	StopBits	Stop bits	Int	60 sec

Temperature

Monitors the temperature at specific locations on the system.

WMI Location

Namespace	Class	Property	Description	Data Type	Update Frequency
root\MoxaIPC	PM_Temperature	Name	Detected temperature item name	String	10 sec
root\MoxaIPC	PM_Temperature	Support	Is this feature supported?	Bool	10 sec
root\MoxaIPC	PM_Temperature	Temperature	Temperature in °C	Int	10 sec

Voltage

Monitors the voltage at a specific location in the system

WMI Location

Namespace	Class	Property	Description	Data Type	Update Frequency
root\MoxaIPC	PM_Voltage	Name	Detected temperature item name	String	10 sec
root\MoxaIPC	PM_Voltage	Support	Is this feature supported?	Bool	10 sec
root\MoxaIPC	PM_Voltage	Voltage	Temperature in °C	Double	10 sec

Accessing WMI Windows Management Instrumentation (WMI)

WMI is a powerful framework for managing and querying Windows system information. You can access WMI using PowerShell as well as third-party WMI Explorer tools.

Accessing WMI Using PowerShell

PowerShell provides a built-in command to query WMI classes and instances.

`Get-WmiObject cmdlet` (**`Get-CimInstance`** in newer scripts)

Launching PowerShell

Press Windows + R keys, type **`powershell`**, and press **Enter**.

Alternatively, search for "PowerShell" in the Start Menu and select "Run as administrator" to obtain the necessary permissions.

Querying WMI Classes

Use the **`Get-WmiObject cmdlet`** command to query system information.

PowerShell

```
Get-WmiObject -Namespace "root\MoxaIPC" -Class "PM_Network"
```

Detailed information about the network is displayed.

```
PS C:\windows\system32> Get-WmiObject -Namespace "root\MoxaIPC" -Class "PM_Network"


__GENUS          : 2
__CLASS          : PM_Network
__SUPERCLASS     :
__DYNASTY        : PM_Network
__RELPATH        : PM_Network.Name="Ethernet 2"
__PROPERTY_COUNT : 6
__DERIVATION     : {}
__SERVER         : WINDOWS-4HQELG4
__NAMESPACE      : ROOT\MoxaIPC
__PATH           : \\WINDOWS-4HQELG4\ROOT\MoxaIPC:PM_Network.Name="Ethernet 2"
ConnectionStatus : Not Connected
Description      : Intel(R) I210 Gigabit Network Connection
LinkSpeed        : 0
Name             : Ethernet 2
PNPDeviceID      : PCI\VEN_8086&DEV_1533&SUBSYS_00000000&REV_03\4&4997D4D&0&00E3
Utilization      : 0
PSComputerName   : WINDOWS-4HQELG4

__GENUS          : 2
__CLASS          : PM_Network
__SUPERCLASS     :
__DYNASTY        : PM_Network
__RELPATH        : PM_Network.Name="Ethernet"
__PROPERTY_COUNT : 6
__DERIVATION     : {}
__SERVER         : WINDOWS-4HQELG4
__NAMESPACE      : ROOT\MoxaIPC
__PATH           : \\WINDOWS-4HQELG4\ROOT\MoxaIPC:PM_Network.Name="Ethernet"
ConnectionStatus : Connected
Description      : Intel(R) Ethernet Connection (13) I219-LM
LinkSpeed        : 1000
Name             : Ethernet
PNPDeviceID      : PCI\VEN_8086&DEV_15FB&SUBSYS_00008086&REV_20\3&11583659&1&FE
Utilization      : 0
PSComputerName   : WINDOWS-4HQELG4
```

Filtering the Results

You can use the **`Where-Object`** parameter to filter the results.

PowerShell

```
Get-WmiObject -Namespace "root\MoxaIPC" -Class "PM_Network" | Where-Object
{$_.Name -eq "Ethernet"}
```

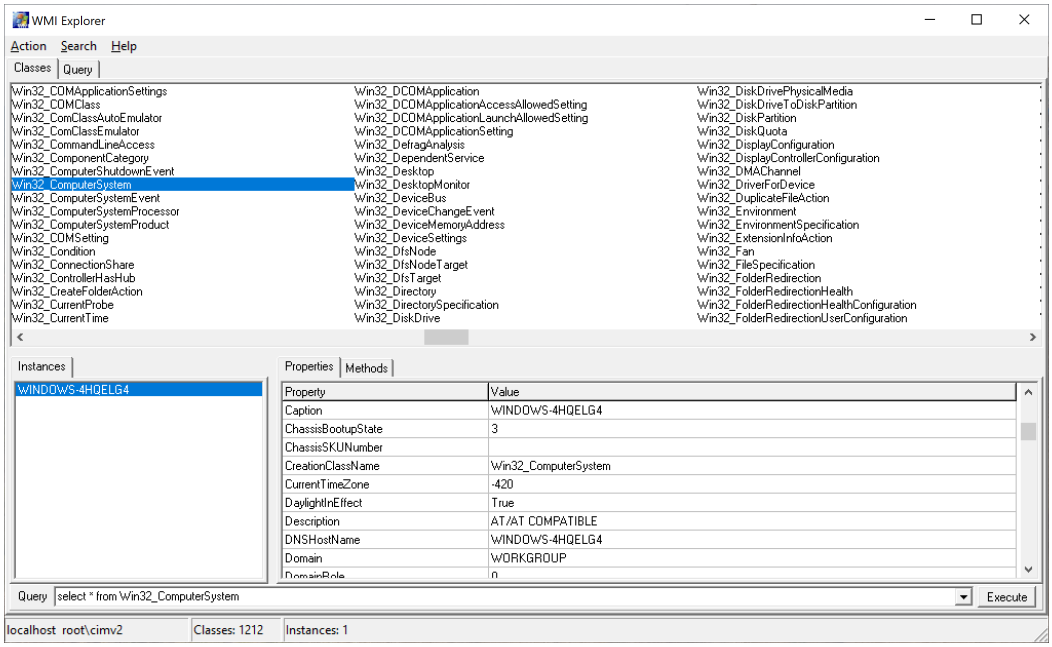
```
PS C:\windows\system32> Get-WmiObject -Namespace "root\MoxaIPC" -Class "PM_Network" | Where-Object {$_.Name -eq "Ethernet"}


__GENUS          : 2
__CLASS          : PM_Network
__SUPERCLASS     :
__DYNASTY        : PM_Network
__RELPATH        : PM_Network.Name="Ethernet"
__PROPERTY_COUNT : 6
__DERIVATION     : {}
__SERVER         : WINDOWS-4HQELG4
__NAMESPACE      : ROOT\MoxaIPC
__PATH           : \\WINDOWS-4HQELG4\ROOT\MoxaIPC:PM_Network.Name="Ethernet"
ConnectionStatus : Connected
Description      : Intel(R) Ethernet Connection (13) I219-LM
LinkSpeed        : 1000
Name             : Ethernet
PNPDeviceID      : PCI\VEN_8086&DEV_15FB&SUBSYS_00008086&REV_20\3&1158365981&1&FE
Utilization      : 0
PSComputerName   : WINDOWS-4HQELG4
```

Accessing WMI via a Third-party WMI Explorer

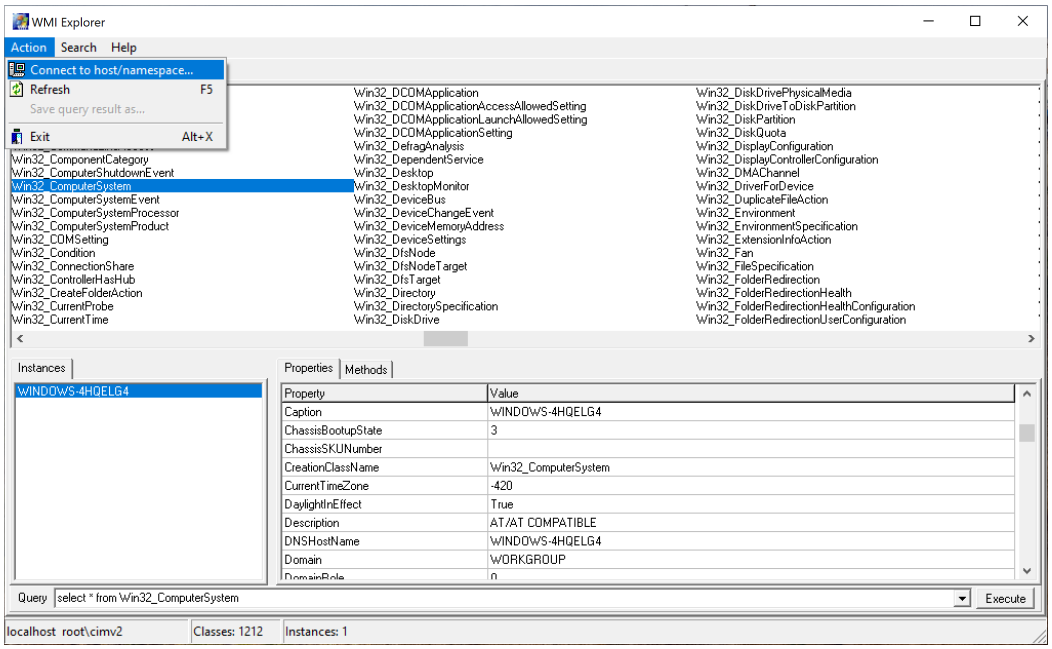
WMI explorer tools provide a graphical interface, making it easy to explore the complete range of WMI management classes, objects, and their properties without needing to write scripts.

The following WMI explorer tool is used in the instructions.
https://www.ks-soft.net/hostmon.eng/wmi/index.htm

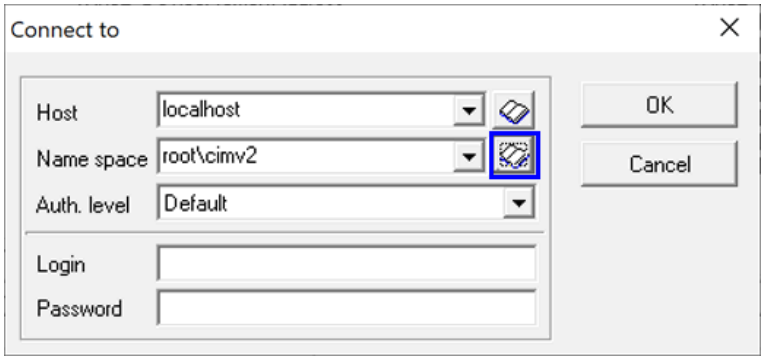


Reading Data via a WMI Explorer

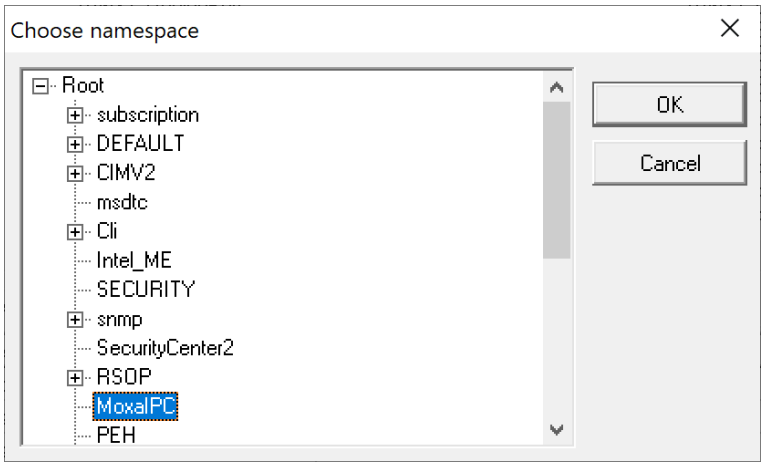
1. Open the WMI explorer, click on the **Action** menu and select **Connect to host/namespace...**.



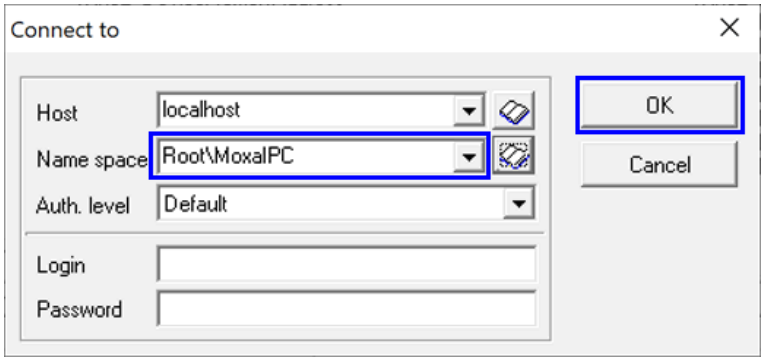
2. Click the book icon corresponding to **Namespace**.



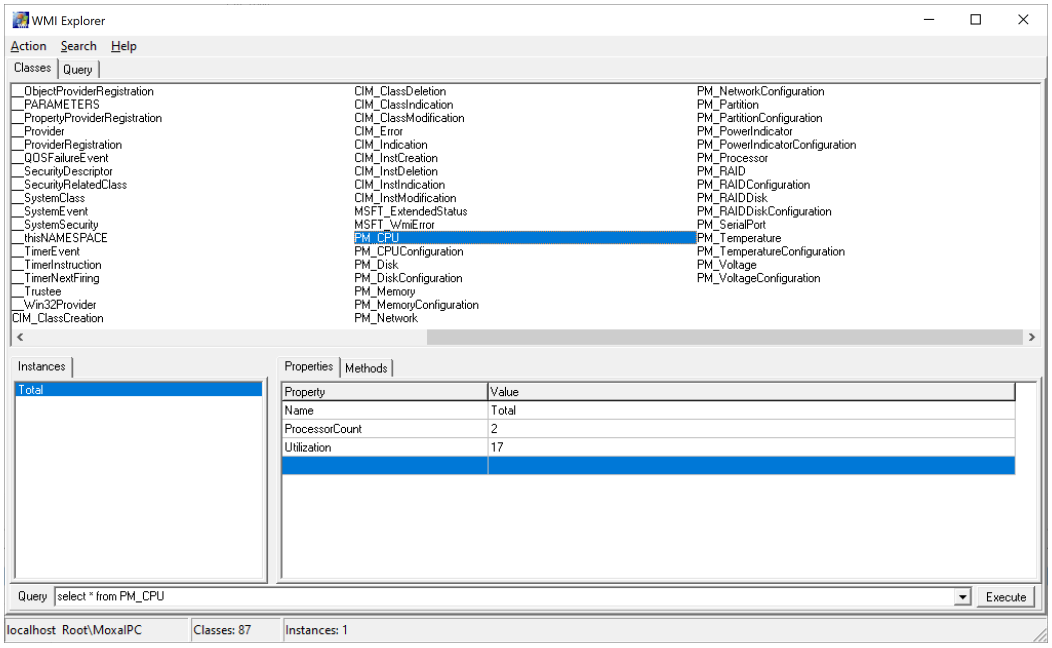
3. Select **MoxaIPC** and click **OK** to confirm.



4. Confirm that the Namespace location has been modified and click **OK**.

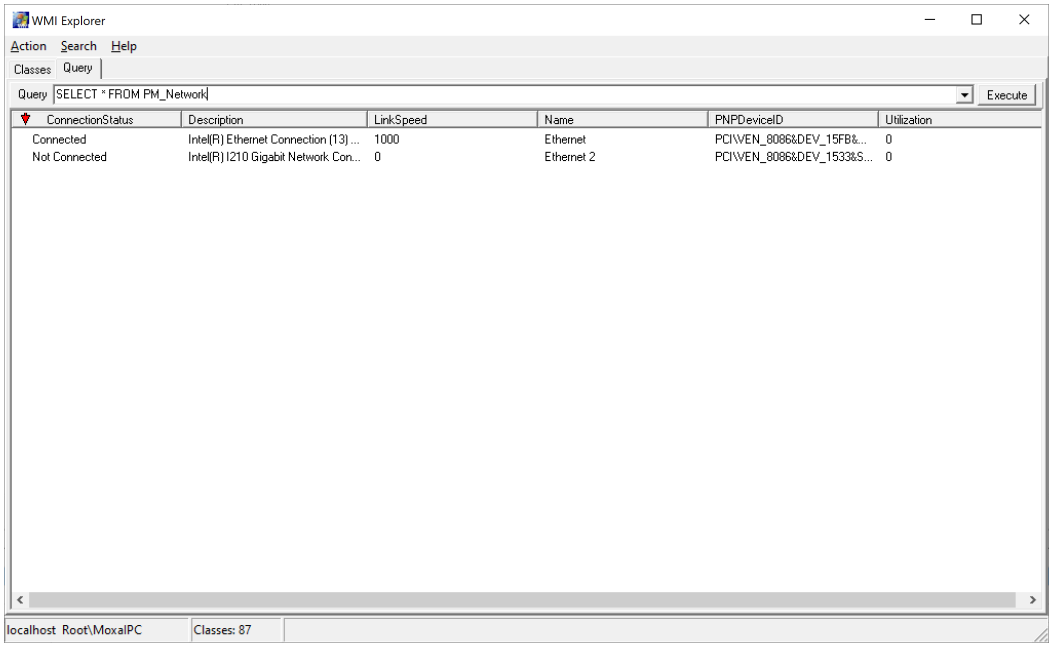


You will now be able to read Proactive Monitoring related data via the WMI explorer.



Reading Data via Query Command in WMI Explorer

Select the **Query** tab to read data using the WQL syntax as shown in the following example:



Setting Up System Alerts

Proactive Monitoring can monitor various system statuses in real-time, including CPU usage, memory status, and disk space. In addition to its monitoring capabilities, Proactive Monitoring also features an **alert mechanism** that allows users to set up alert rules based on conditions and thresholds. When a system state exceeds a predefined threshold, the software will automatically trigger an alert and notify users through various methods, ensuring potential issues can be addressed promptly, thereby enhancing system stability and reliability. The alert mechanisms include running a specified program in the background or writing to the event log.

CPU Configuration

WMI Location

Namespace	Class	Property	Description	Data Type	Notes
root\MoxaIPC	PM_CPUConfiguration	EnableEventLog	Launch Windows Event Log	Bool	
root\MoxaIPC	PM_CPUConfiguration	EnableGracePeriod	Launch Grace Period	Bool	Needs to be used with Grace Period
root\MoxaIPC	PM_CPUConfiguration	EnableLowerAlarm	Launch Lower Alarm	Bool	Needs to be used with Lower threshold to monitor CPU Utilization
root\MoxaIPC	PM_CPUConfiguration	EnableSNMPTrap	Launch SNMP Trap	Bool	
root\MoxaIPC	PM_CPUConfiguration	EnableUpperAlarm	Launch Upper Alarm	Bool	Needs to be used with Upper threshold to monitor CPU Utilization.
root\MoxaIPC	PM_CPUConfiguration	GracePeriod	When the monitored value exceeds the Threshold, the Alarm will only be actually triggered after the Grace Period count is reached.	Int	
root\MoxaIPC	PM_CPUConfiguration	LowerAlarmProgramPath	When the Lower Alarm is triggered, specify the program to execute.	String	Run in the background.
root\MoxaIPC	PM_CPUConfiguration	LowerThreshold	Lower threshold	Int	Range: 0~100
root\MoxaIPC	PM_CPUConfiguration	Name	The Instance name corresponds to the detection items in PM_CPU.	String	
root\MoxaIPC	PM_CPUConfiguration	UpperAlarmProgramPath	When the Upper Alarm is triggered, specify the program to execute.	String	Run in the background.
root\MoxaIPC	PM_CPUConfiguration	UpperThreshold	Upper threshold	Int	Range: 0~100

Example:

PowerShell

```
$cpuconfig = (Get-WmiObject -Namespace "root/MoxaIPC" -Class
"PM_CPUConfiguration")

Set-WmiInstance -Path $cpuconfig -Arguments @{EnableEventLog=$true;
EnableGracePeriod=$true; EnableSNMPTrap=$true;
GracePeriod=5;EnableUpperAlarm=$true; UpperThreshold=50;
UpperAlarmProgramPath="C:\Windows\System32\notepad.exe"}
```

Result:

```
PS C:\windows\system32> $cpuconfig = (Get-WmiObject -Namespace "root/MoxaIPC" -Class "PM_CPUConfiguration")
PS C:\windows\system32> Set-WmiInstance -Path $cpuconfig -Arguments @{EnableEventLog=$true; EnableGracePeriod
=$true; EnableSNMPTrap=$true; GracePeriod=5;EnableUpperAlarm=$true; UpperThreshold=50; UpperAlarmProgramPath=
"C:\Windows\System32\notepad.exe"}


__GENUS               : 2
__CLASS               : PM_CPUConfiguration
__SUPERCLASS          :
__DYNASTY             : PM_CPUConfiguration
__RELPATH             : PM_CPUConfiguration.Name="Total"
__PROPERTY_COUNT      : 12
__DERIVATION          : {}
__SERVER              : WINDOWS-4HQELG4
__NAMESPACE           : ROOT\MoxaIPC
__PATH                : \\WINDOWS-4HQELG4\ROOT\MoxaIPC:PM_CPUConfiguration.Name="Total"
AlarmBackgroudWoker   : False
EnableEventLog        : True
EnableGracePeriod     : True
EnableLowerAlarm      : False
EnableSNMPTrap        : True
EnableUpperAlarm      : True
GracePeriod           : 5
LowerAlarmProgramPath :
LowerThreshold        : 0
Name                  : Total
UpperAlarmProgramPath : C:\Windows\System32\notepad.exe
UpperThreshold        : 50
PSComputerName        : WINDOWS-4HQELG4
```

Disk Configuration

WMI Location

Namespace	Class	Property	Description	Data Type	Notes
root\MoxaIPC	PM_DiskConfiguration	AlarmProgramPath	When the Alarm is triggered, specify the program to execute.	String	Run in the background.
root\MoxaIPC	PM_DiskConfiguration	EnableAlarm	Enable Alarm	Bool	Monitoring Disk Health Status
root\MoxaIPC	PM_DiskConfiguration	EnableEventLog	Launch Windows Event Log.	Bool	
root\MoxaIPC	PM_DiskConfiguration	EnableSNMPTrap	Enable SNMP Trap.	Bool	
root\MoxaIPC	PM_DiskConfiguration	Name	The Instance name corresponds to the monitoring items in PM_Disk.	String	

Example:

PowerShell

```
$diskconfig = (Get-WmiObject -Namespace "root/MoxaIPC" -Class
"PM_DiskConfiguration" -Filter "Name='Bus 0, Port 0, Id 0'")

Set-WmiInstance -Path $diskconfig -Arguments @{EnableEventLog=$true;
EnableAlarm=$true;}
```

Result:

```
PS C:\windows\system32> $diskconfig = (Get-WmiObject -Namespace "root/MoxaIPC" -Class "PM_DiskConfiguration"
-Filter "Name='Bus 0, Port 0, Id 0'")
PS C:\windows\system32> Set-WmiInstance -Path $diskconfig -Arguments @{EnableEventLog=$true; EnableAlarm=$Tru
e;}


__GENUS             : 2
__CLASS             : PM_DiskConfiguration
__SUPERCLASS        :
__DYNASTY           : PM_DiskConfiguration
__RELPATH           : PM_DiskConfiguration.Name="Bus 0, Port 0, Id 0"
__PROPERTY_COUNT    : 6
__DERIVATION        : {}
__SERVER            : WINDOWS-4HQELG4
__NAMESPACE         : ROOT\MoxaIPC
__PATH              : \\WINDOWS-4HQELG4\ROOT\MoxaIPC:PM_DiskConfiguration.Name="Bus 0, Port 0, Id 0"
AlarmBackgroudWoker :
AlarmProgramPath    :
EnableAlarm         : True
EnableEventLog      : True
EnableSNMPTrap      : False
Name                : Bus 0, Port 0, Id 0
PSComputerName      : WINDOWS-4HQELG4
```

Memory Configuration

WMI Location

Namespace	Class	Property	Description	Data Type	Notes
root\MoxaIPC	PM_MemoryConfiguration	EnableEventLog	Enable Windows Event Log	Bool	
root\MoxaIPC	PM_MemoryConfiguration	EnableGracePeriod	Enable Grace Period	Bool	Needs to be used with Grace Period
root\MoxaIPC	PM_MemoryConfiguration	EnableLowerAlarm	Enable Lower Alarm	Bool	Needs to be used with Lower threshold to monitor Memory Utilization.
root\MoxaIPC	PM_MemoryConfiguration	EnableSNMPTrap	Enable SNMP Trap	Bool	
root\MoxaIPC	PM_MemoryConfiguration	EnableUpperAlarm	Enable Upper Alarm	Bool	Needs to be used with Upper threshold to monitor Memory Utilization.
root\MoxaIPC	PM_MemoryConfiguration	GracePeriod	When the monitored value exceeds the Threshold, the Alarm will only be actually triggered after the Grace Period count is reached	Int	

Namespace	Class	Property	Description	Data Type	Notes
root\MoxaIPC	PM_MemoryConfiguration	LowerAlarmProgramPath	When the Lower Alarm is triggered, specify the program to execute	String	Run in the background
root\MoxaIPC	PM_MemoryConfiguration	LowerThreshold	Lower threshold	Int	Range : 0~100
root\MoxaIPC	PM_MemoryConfiguration	Name	The name of the instance will correspond to the monitoring item in PM_Memory	String	
root\MoxaIPC	PM_MemoryConfiguration	UpperAlarmProgramPath	When the Upper Alarm is triggered, specify the program to be executed	String	Run in the background
root\MoxaIPC	PM_MemoryConfiguration	UpperThreshold	Upper threshold	Int	Range : 0~100

Example:

PowerShell

```
$memoryconfig = (Get-WmiObject -Namespace "root/MoxaIPC" -Class
"PM_MemoryConfiguration")

Set-WmiInstance -Path $memoryconfig -Arguments @{EnableEventLog=$true;
EnableUpperAlarm=$true; UpperThreshold=50;}
```

Result:

```
PS C:\windows\system32> $memoryconfig = (Get-WmiObject -Namespace "root/MoxaIPC" -Class "PM_MemoryConfigurati
on")
PS C:\windows\system32> Set-WmiInstance -Path $memoryconfig -Arguments @{EnableEventLog=$true; EnableUpperAla
rm=$True; UpperThreshold=50;}


__GENUS                : 2
__CLASS                : PM_MemoryConfiguration
__SUPERCLASS           :
__DYNASTY              : PM_MemoryConfiguration
__RELPATH              : PM_MemoryConfiguration.Name="Physical Memory"
__PROPERTY_COUNT       : 12
__DERIVATION           : {}
__SERVER               : WINDOWS-4HQELG4
__NAMESPACE            : ROOT\MoxaIPC
__PATH                 : \\WINDOWS-4HQELG4\ROOT\MoxaIPC:PM_MemoryConfiguration.Name="Physical Memory"
AlarmBackgroudWoker    : False
EnableEventLog         : True
EnableGracePeriod      : False
EnableLowerAlarm       : False
EnableSNMPTrap         : False
EnableUpperAlarm       : True
GracePeriod            : 0
LowerAlarmProgramPath  :
LowerThreshold         : 0
Name                   : Physical Memory
UpperAlarmProgramPath  :
UpperThreshold         : 50
PSComputerName         : WINDOWS-4HQELG4
```

Network Configuration

WMI Location

Namespace	Class	Property	Description	Data Type	Notes
root\MoxaIPC	PM_NetworkConfiguration	AlarmProgramPath	When the alarm is triggered, specify the program to be executed	String	Run in the background
root\MoxaIPC	PM_NetworkConfiguration	EnableAlarm	Start Alarm	Bool	Monitor Network Adapter Connection Status
root\MoxaIPC	PM_NetworkConfiguration	EnableEventLog	Enable Windows Event Log	Bool	
root\MoxaIPC	PM_NetworkConfiguration	EnableSNMPTrap	Enable SNMP Trap	Bool	
root\MoxaIPC	PM_NetworkConfiguration	Name	The name of the Instance will correspond to the monitoring item in PM_Network	String	

Example:

PowerShell

```
$networkconfig = (Get-WmiObject -Namespace "root/MoxaIPC" -Class
"PM_NetworkConfiguration" -Filter "Name='Ethernet'")

Set-WmiInstance -Path $networkconfig -Arguments @{EnableEventLog=$true;
EnableAlarm=$true;}
```

Result:

```
PS C:\windows\system32> $networkconfig = (Get-WmiObject -Namespace "root/MoxaIPC" -Class "PM_NetworkConfigura
tion" -Filter "Name='Ethernet'")
PS C:\windows\system32> Set-WmiInstance -Path $networkconfig -Arguments @{EnableEventLog=$true; EnableAlarm=$
true;}


__GENUS             : 2
__CLASS             : PM_NetworkConfiguration
__SUPERCLASS        :
__DYNASTY           : PM_NetworkConfiguration
__RELPATH           : PM_NetworkConfiguration.Name="Ethernet"
__PROPERTY_COUNT    : 6
__DERIVATION        : {}
__SERVER            : WINDOWS-4HQELG4
__NAMESPACE         : ROOT\MoxaIPC
__PATH              : \\WINDOWS-4HQELG4\ROOT\MoxaIPC:PM_NetworkConfiguration.Name="Ethernet"
AlarmBackgroudWoker : False
AlarmProgramPath    :
EnableAlarm         : True
EnableEventLog      : True
EnableSNMPTrap      : False
Name                : Ethernet
PSComputerName      : WINDOWS-4HQELG4
```

Partition Configuration

WMI Location

Namespace	Class	Property	Description	Data Type	Notes
root\MoxaIPC	PM_PartitionConfiguration	EnableEventLog	Enable Windows Event Log	Bool	
root\MoxaIPC	PM_PartitionConfiguration	EnableGracePeriod	Enable Grace Period	Bool	Needs to be used with Grace Period
root\MoxaIPC	PM_PartitionConfiguration	EnableLowerAlarm	Enable Lower Alarm	Bool	Need to be used with Lower threshold to monitor Partition Utilization
root\MoxaIPC	PM_PartitionConfiguration	EnableSNMPTrap	Enable SNMP Trap	Bool	
root\MoxaIPC	PM_PartitionConfiguration	EnableUpperAlarm	Enable Upper Alarm	Bool	Need to be used with Upper threshold to monitor Partition Utilization
root\MoxaIPC	PM_PartitionConfiguration	GracePeriod	When the monitored value exceeds the Threshold, the Alarm will only be actually triggered after the Grace Period count is reached	Int	
root\MoxaIPC	PM_PartitionConfiguration	LowerAlarmProgramPath	When the Lower Alarm is triggered, specify the program to execute	String	Run in the background
root\MoxaIPC	PM_PartitionConfiguration	LowerThreshold	Lower threshold	Int	Range : 0~100
root\MoxaIPC	PM_PartitionConfiguration	Name	The name of the Instance corresponds to the monitoring item in PM_Partition	String	
root\MoxaIPC	PM_PartitionConfiguration	UpperAlarmProgramPath	When the Upper Alarm is triggered, specify the program to be executed	String	Run in the background
root\MoxaIPC	PM_PartitionConfiguration	UpperThreshold	Upper threshold	Int	Range : 0~100

Example:

PowerShell

```
$partitionconfig = (Get-WmiObject -Namespace "root/MoxaIPC" -Class
"PM_PartitionConfiguration" -Filter "Name='C'")

Set-WmiInstance -Path $partitionconfig -Arguments @{EnableEventLog=$true;
EnableUpperAlarm=$true; UpperThreshold=50;}
```

Result:

```
PS C:\windows\system32> $partitionconfig = (Get-WmiObject -Namespace "root/MoxaIPC" -Class "PM_PartitionConfi
guration" -Filter "Name='C'")
PS C:\windows\system32> Set-WmiInstance -Path $partitionconfig -Arguments @{EnableEventLog=$true; EnableUpper
Alarm=$true; UpperThreshold=50;}


__GENUS                : 2
__CLASS                : PM_PartitionConfiguration
__SUPERCLASS           :
__DYNASTY              : PM_PartitionConfiguration
__RELPATH              : PM_PartitionConfiguration.Name="C"
__PROPERTY_COUNT       : 12
__DERIVATION           : {}
__SERVER               : WINDOWS-4HQELG4
__NAMESPACE            : ROOT\MoxaIPC
__PATH                 : \\WINDOWS-4HQELG4\ROOT\MoxaIPC:PM_PartitionConfiguration.Name="C"
AlarmBackgroudWoker    : False
EnableEventLog         : True
EnableGracePeriod      : False
EnableLowerAlarm       : False
EnableSNMPTrap         : False
EnableUpperAlarm       : True
GracePeriod            : 0
LowerAlarmProgramPath  :
LowerThreshold         : 0
Name                   : C
UpperAlarmProgramPath  :
UpperThreshold         : 50
PSComputerName         : WINDOWS-4HQELG4
```

Power Indicator Configuration

WMI Location

Namespace	Class	Property	Description	Data Type	Notes
root\MoxaIPC	PM_PowerIndicatorConfiguration	AlarmProgramPath	When the alarm is triggered, specify the program to be executed	String	Run in the background
root\MoxaIPC	PM_PowerIndicatorConfiguration	EnableAlarm	Start Alarm	Bool	Monitor Power Module 1 Status and Power Module 2 Status
root\MoxaIPC	PM_PowerIndicatorConfiguration	EnableEventLog	Enable Windows Event Log	Bool	
root\MoxaIPC	PM_PowerIndicatorConfiguration	EnableSNMPTrap	Enable SNMP Trap	Bool	
root\MoxaIPC	PM_PowerIndicatorConfiguration	Name	The name of the instance corresponds to the monitoring item in PM_PowerIndicator	String	

Example:

PowerShell

```
$powerindicatorconfig = (Get-WmiObject -Namespace "root/MoxaIPC" -Class
"PM_PowerIndicatorConfiguration")

Set-WmiInstance -Path $powerindicatorconfig -Arguments @{EnableEventLog=$true;
EnableAlarm=$true;}
```

Result:

```
PS C:\windows\system32> $powerindicatorconfig = (Get-WmiObject -Namespace "root/MoxaIPC" -Class "PM_PowerIndi
catorConfiguration")
PS C:\windows\system32> Set-WmiInstance -Path $powerindicatorconfig -Arguments @{EnableEventLog=$true; Enable
Alarm=$true;}


__GENUS             : 2
__CLASS             : PM_PowerIndicatorConfiguration
__SUPERCLASS        :
__DYNASTY           : PM_PowerIndicatorConfiguration
__RELPATH           : PM_PowerIndicatorConfiguration.Name="Power Indicator"
__PROPERTY_COUNT    : 6
__DERIVATION        : {}
__SERVER            : WINDOWS-4HQELG4
__NAMESPACE         : ROOT\MoxaIPC
__PATH              : \\WINDOWS-4HQELG4\ROOT\MoxaIPC:PM_PowerIndicatorConfiguration.Name="Power Indicator"
AlarmBackgroudWoker : False
AlarmProgramPath    :
EnableAlarm         : True
EnableEventLog      : True
EnableSNMPTrap      : False
Name                : Power Indicator
PSComputerName      : WINDOWS-4HQELG4
```

RAID Configuration

WMI Location

Namespace	Class	Property	Description	Data Type	Notes
root\MoxaIPC	PM_RAIDConfiguration	AlarmProgramPath	When the alarm is triggered, specify the program to be executed	String	Run in the background
root\MoxaIPC	PM_RAIDConfiguration	EnableAlarm	Start Alarm	Bool	Monitoring Redundancy Statistics
root\MoxaIPC	PM_RAIDConfiguration	EnableEventLog	Enable Windows Event Log	Bool	
root\MoxaIPC	PM_RAIDConfiguration	Name	The name of the instance will correspond to the monitoring item in PM_RAID	String	

Example:

PowerShell

```
$raidconfig = (Get-WmiObject -Namespace "root/MoxaIPC" -Class
"PM_RAIDConfiguration" -Filter "Name='Volume1'")

Set-WmiInstance -Path $raidconfig -Arguments @{EnableEventLog=$true;
EnableAlarm=$true;}
```

Result:

```
PS C:\windows\system32> $raidconfig = (Get-WmiObject -Namespace "root/MoxaIPC" -Class "PM_RAIDConfiguration" -Filter "Na
me='Volume1'")
PS C:\windows\system32> Set-WmiInstance -Path $raidconfig -Arguments @{EnableEventLog=$true; EnableAlarm=$true;}


__GENUS          : 2
__CLASS          : PM_RAIDConfiguration
__SUPERCLASS     :
__DYNASTY        : PM_RAIDConfiguration
__RELPATH        : PM_RAIDConfiguration.Name="Volume1"
__PROPERTY_COUNT : 4
__DERIVATION     : {}
__SERVER         : WINDOWS-UG5IVVO
__NAMESPACE      : ROOT\MoxaIPC
__PATH           : \\WINDOWS-UG5IVVO\ROOT\MoxaIPC:PM_RAIDConfiguration.Name="Volume1"
AlarmProgramPath :
EnableAlarm      : True
EnableEventLog   : True
Name             : Volume1
PSComputerName   : WINDOWS-UG5IVVO
```

RAID Disk Configuration

WMI Location

Namespace	Class	Property	Description	Data Type	Notes
root\MoxaIPC	PM_RAIDDiskConfiguration	AlarmProgramPath	When the alarm is triggered, specify the program to be executed	String	Run in the background
root\MoxaIPC	PM_RAIDDiskConfiguration	EnableAlarm	Start Alarm	Bool	Monitor Disk Status
root\MoxaIPC	PM_RAIDDiskConfiguration	EnableEventLog	Enable Windows Event Log	Bool	
root\MoxaIPC	PM_RAIDDiskConfiguration	Name	The name of the instance will correspond to the monitoring item in PM_RAIDDisk	String	

Example:

PowerShell

```
$raiddiskconfig = (Get-WmiObject -Namespace "root/MoxaIPC" -Class
"PM_RAIDDiskConfiguration" -Filter "Name='2-2-0-0'")

Set-WmiInstance -Path $raiddiskconfig -Arguments @{EnableEventLog=$true;
EnableAlarm=$true;}
```

Result:

```
PS C:\windows\system32> $raiddiskconfig = (Get-WmiObject -Namespace "root/MoxaIPC" -Class "PM_RAIDDiskConfiguration" -Fi
lter "Name='2-2-0-0'")
PS C:\windows\system32> Set-WmiInstance -Path $raiddiskconfig -Arguments @{EnableEventLog=$true; EnableAlarm=$true;}


__GENUS          : 2
__CLASS          : PM_RAIDDiskConfiguration
__SUPERCLASS     :
__DYNASTY        : PM_RAIDDiskConfiguration
__RELPATH        : PM_RAIDDiskConfiguration.Name="2-2-0-0"
__PROPERTY_COUNT : 4
__DERIVATION     : {}
__SERVER         : WINDOWS-UG5IVVO
__NAMESPACE      : ROOT\MoxaIPC
__PATH           : \\WINDOWS-UG5IVVO\ROOT\MoxaIPC:PM_RAIDDiskConfiguration.Name="2-2-0-0"
AlarmProgramPath :
EnableAlarm      : True
EnableEventLog   : True
Name             : 2-2-0-0
PSComputerName   : WINDOWS-UG5IVVO
```

Temperature Configuration

WMI Location

Namespace	Class	Property	Description	Data Type	Notes
root\MoxaIPC	PM_TemperatureConfiguration	EnableEventLog	Enable Windows Event Log	Bool	
root\MoxaIPC	PM_TemperatureConfiguration	EnableGracePeriod	Enable Grace Period	Bool	Needs to be used with Grace Period
root\MoxaIPC	PM_TemperatureConfiguration	EnableLowerAlarm	Enable Lower Alarm	Bool	Need to be used with Lower threshold to monitor Temperature
root\MoxaIPC	PM_TemperatureConfiguration	EnableSNMPTrap	Enable SNMP Trap	Bool	
root\MoxaIPC	PM_TemperatureConfiguration	EnableUpperAlarm	Enable Upper Alarm	Bool	Need to be used with Upper threshold to monitor Temperature
root\MoxaIPC	PM_TemperatureConfiguration	GracePeriod	When the monitored value exceeds the Threshold, the Alarm will only be actually triggered after the Grace Period count is reached	Int	
root\MoxaIPC	PM_TemperatureConfiguration	LowerAlarmProgramPath	When the Lower Alarm is triggered, specify the program to execute	String	Run in the background
root\MoxaIPC	PM_TemperatureConfiguration	LowerThreshold	Lower threshold	Int	
root\MoxaIPC	PM_TemperatureConfiguration	Name	The name of the instance will correspond to the monitoring item in PM_Temperature	String	
root\MoxaIPC	PM_TemperatureConfiguration	UpperAlarmProgramPath	When the Upper Alarm is triggered, specify the program to be executed	String	Run in the background
root\MoxaIPC	PM_TemperatureConfiguration	UpperThreshold	Upper threshold	Int	

Example:

PowerShell

```
$temperatureconfig = (Get-WmiObject -Namespace "root/MoxaIPC" -Class
"PM_TemperatureConfiguration" -Filter "Name='CPU'")

Set-WmiInstance -Path $temperatureconfig -Arguments @{EnableEventLog=$true;
EnableUpperAlarm=$true; UpperThreshold=50;}
```

Result:

```
PS C:\windows\system32> $temperatureconfig = (Get-WmiObject -Namespace "root/MoxaIPC" -Class "PM_TemperatureC
onfiguration" -Filter "Name='CPU'")
PS C:\windows\system32> Set-WmiInstance -Path $temperatureconfig -Arguments @{EnableEventLog=$true; EnableUpp
erAlarm=$true; UpperThreshold=50;}


__GENUS                : 2
__CLASS                : PM_TemperatureConfiguration
__SUPERCLASS           :
__DYNASTY              : PM_TemperatureConfiguration
__RELPATH              : PM_TemperatureConfiguration.Name="CPU"
__PROPERTY_COUNT       : 12
__DERIVATION           : {}
__SERVER               : WINDOWS-4HQELG4
__NAMESPACE            : ROOT\MoxaIPC
__PATH                 : \\WINDOWS-4HQELG4\ROOT\MoxaIPC:PM_TemperatureConfiguration.Name="CPU"
AlarmBackgroudWoker    : False
EnableEventLog         : True
EnableGracePeriod      : False
EnableLowerAlarm       : False
EnableSNMPTrap         : False
EnableUpperAlarm       : True
GracePeriod            : 0
LowerAlarmProgramPath  :
LowerThreshold         : 0
Name                   : CPU
UpperAlarmProgramPath  :
UpperThreshold         : 50
PSComputerName         : WINDOWS-4HQELG4
```

Voltage Configuration

WMI Location

Namespace	Class	Property	Description	Data Type	Notes
root\MoxaIPC	PM_VoltageConfiguration	EnableEventLog	Enable Windows Event Log	Bool	
root\MoxaIPC	PM_VoltageConfiguration	EnableGracePeriod	Enable Grace Period	Bool	Needs to be used with Grace Period
root\MoxaIPC	PM_VoltageConfiguration	EnableLowerAlarm	Enable Lower Alarm	Bool	Needs to be used with Lower threshold to monitor Voltage
root\MoxaIPC	PM_VoltageConfiguration	EnableSNMPTrap	Enable SNMP Trap	Bool	
root\MoxaIPC	PM_VoltageConfiguration	EnableUpperAlarm	Enable Upper Alarm	Bool	Needs to be used with Upper threshold to monitor Voltage.
root\MoxaIPC	PM_VoltageConfiguration	GracePeriod	When the monitored value exceeds the Threshold, the Alarm will only be actually triggered after the Grace Period count is reached	Int	
root\MoxaIPC	PM_VoltageConfiguration	LowerAlarmProgramPath	When the Lower Alarm is triggered, specify the program to execute	String	Run in the background

Namespace	Class	Property	Description	Data Type	Notes
root\MoxaIPC	PM_VoltageConfiguration	LowerThreshold	Lower threshold	Int	
root\MoxaIPC	PM_VoltageConfiguration	Name	The name of the instance corresponds to the monitoring item in PM_Voltage	String	
root\MoxaIPC	PM_VoltageConfiguration	UpperAlarmProgramPath	When the Upper Alarm is triggered, specify the program to be executed	String	Run in the background
root\MoxaIPC	PM_VoltageConfiguration	UpperThreshold	Upper threshold	Int	

Example:

PowerShell

```
$voltageconfig = (Get-WmiObject -Namespace "root/MoxaIPC" -Class
"PM_VoltageConfiguration" -Filter "Name='Mainboard'")

Set-WmiInstance -Path $voltageconfig -Arguments @{EnableEventLog=$true;
EnableUpperAlarm=$true; UpperThreshold=6;}
```

Result:

```
PS C:\windows\system32> $voltageconfig = (Get-WmiObject -Namespace "root/MoxaIPC" -Class "PM_VoltageConfigura
tion" -Filter "Name='Mainboard'")
PS C:\windows\system32> Set-WmiInstance -Path $voltageconfig -Arguments @{EnableEventLog=$true; EnableUpperAl
arm=$true; UpperThreshold=6;}


__GENUS                : 2
__CLASS                : PM_VoltageConfiguration
__SUPERCLASS           :
__DYNASTY              : PM_VoltageConfiguration
__RELPATH              : PM_VoltageConfiguration.Name="Mainboard"
__PROPERTY_COUNT       : 11
__DERIVATION           : {}
__SERVER               : WINDOWS-4HQELG4
__NAMESPACE            : ROOT\MoxaIPC
__PATH                 : \\WINDOWS-4HQELG4\ROOT\MoxaIPC:PM_VoltageConfiguration.Name="Mainboard"
AlarmBackgroudWoker    : False
EnableEventLog         : True
EnableGracePeriod      : False
EnableLowerAlarm       : False
EnableUpperAlarm       : True
GracePeriod            : 0
LowerAlarmProgramPath  :
LowerThreshold         : 0
Name                   : Mainboard
UpperAlarmProgramPath  :
UpperThreshold         : 6
PSComputerName         : WINDOWS-4HQELG4
```

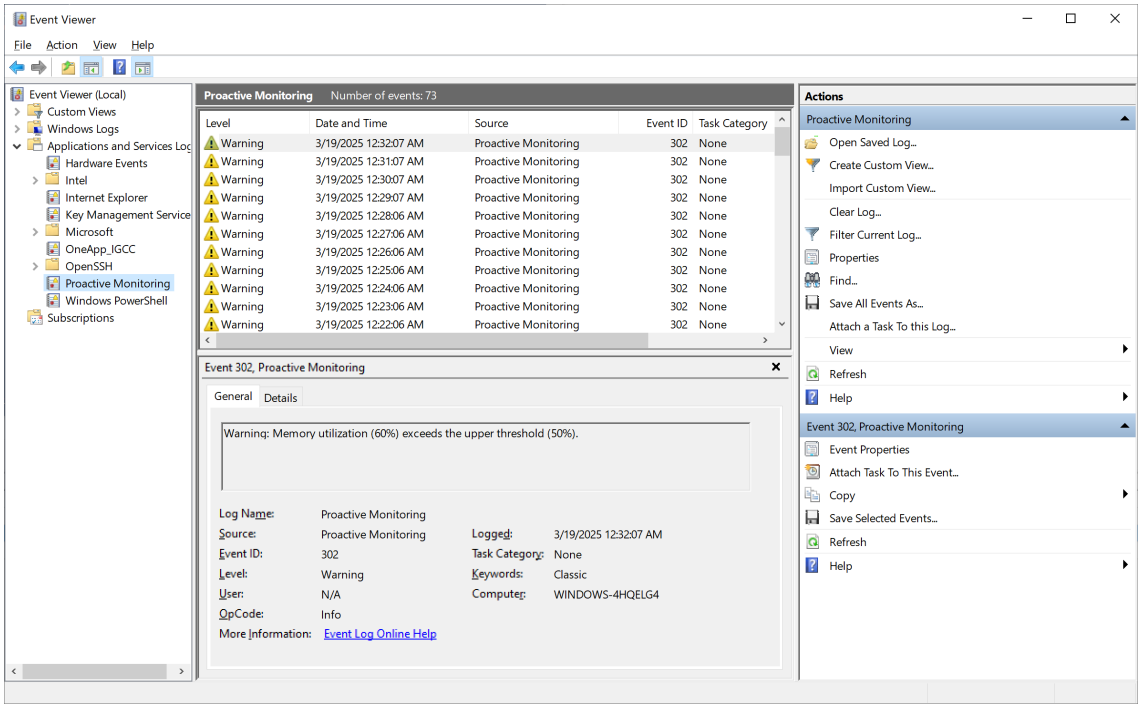
Event Log

Proactive Monitoring provides integrating with Windows Event Log for recording information on the system statuses being monitored and abnormal events in the event log. Proactive Monitoring also defines a custom event viewer path, which allows users to find relevant monitoring records more easily.

Event Viewer Path
Application and Services Log\Proactive Monitoring

To view the events in the event log, do the following:

1. Open the Event Viewer.
2. In the menu tree (on the left panel), expand **Application and Services Log**.
3. Click on **Proactive Monitoring** to view the event records.



Integration of the Proactive Monitoring event logs with the Windows Event Log provides the following benefits:

- Centralized Management: All system events are recorded in the Windows Event Viewer for easy unified management and review.
- History Tracking: Keeping a historical record of system status helps in subsequent problem analysis and tracking.
- Automated Processing: Can be combined with event triggers or custom scripts to implement automated response mechanisms.
- Enhanced Security: By monitoring and recording abnormal events, it helps administrators detect potential security threats in real time.

This integration makes Proactive Monitoring more than just a real-time monitoring tool by adding system maintenance and security management capabilities.

Event ID List:

Event ID	Item	Description
101	CPU	CPU usage is lower than Threshold
102	CPU	CPU usage is higher than Threshold
201	Disk	Disk health status is "Warning"
301	Memory	Memory usage is lower than the threshold
302	Memory	Memory usage is higher than the threshold
401	Network	Network adapter connection status is "Not Connected"
501	Partition	Partition usage is lower than Threshold

Event ID	Item	Description
502	Partition	Partition usage is higher than Threshold
601	Power Indicator	Power Module Status is "Failed"
701	RAID	Intel RST RAID Redundancy Status is abnormal
801	RAID Disk	The disks that make up the Intel RST RAID are abnormal.
901	Temperature	Temperature below Threshold
902	Temperature	Temperature higher than Threshold
1001	Voltage	Temperature below Threshold
1002	Voltage	Temperature higher than Threshold

Overview

The Moxa Proactive Monitoring tools provides APIs to communicate with supported hardware devices. The APIs are developed using the C programming language.

CPU Status

The **CPUStatus** API can get the following CPU information:

GetAverageCpuUsage	Gets the average CPU utilization
GetCpuUsage	Gets the CPU utilization per core

GetAverageCpuUsage

Syntax

```
int GetAverageCpuUsage(
        int *value
    );
```

Description

Retrieves the average usage of CPU.

Parameters

value:
CPU total usage in percent.

Return Value

If the CPU usage is successfully retrieved, return **0**; otherwise, return **-1**.

Error codes

Name	Value	Meaning
Return Value	-1	Failed to retrieve CPU usage.

Requirements

Name	Items
Header	HardwareMonitorApi.h
Library	HardwareMonitorApi.lib
DLL	HardwareMonitorApi.dll

GetCpuUsage

Syntax

```
int GetCpuUsage(
    int index,
    int* value
);
```

Description

Retrieves the usage of each individual CPU logical processors.

Parameters

index:
The index of logical processors. When the input index is **0**, it retrieves the **total CPU utilization**.

value:
Processor total usage in percent.

Return Value

If the processor usage is successfully retrieved, return **0**; otherwise, return **-1**.

Error codes

Name	Value	Meaning
Return Value	-1	Failed to retrieve processor usage.

Requirements

Name	Items
Header	HardwareMonitorApi.h
Library	HardwareMonitorApi.lib
DLL	HardwareMonitorApi.dll

Disk Health Status

GetDiskPort

Syntax

```
int GetDiskPort(
    int index,
    int *value
);
```

Description

Retrieves the port location of the disk in the slot.

Parameters

index:
The index of disk.

value:
The port location of the disk in the slot.

Return Value

If the port location of the disk in the slot is successfully retrieved, return **0**; otherwise, return **-1**.

Error codes

Name	Value	Meaning
Return Value	-1	Failed to retrieve port location of the disk in the slot.

Requirements

Name	Items
Header	HardwareMonitorApi.h
Library	HardwareMonitorApi.lib
DLL	HardwareMonitorApi.dll

GetDiskExist

Syntax

```
int GetDiskExist(
    int index,
    bool *value
);
```

Description

Retrieves whether the disk in the specified slot exists or not.

Parameters

index:
The index of disk slot.

value:
The disk in the specified slot exists or not.

Return Value

If the slot status is successfully retrieved, return **0**; otherwise, return **-1**.

Error codes

Name	Value	Meaning
Return Value	-1	Failed to retrieve the slot status.

Requirements

Name	Items
Header	HardwareMonitorApi.h
Library	HardwareMonitorApi.lib
DLL	HardwareMonitorApi.dll

GetDiskExistInt

Syntax

```
int GetDiskExist(
    int index,
    int *value
);
```

Description

Retrieves whether the disk in the specified slot exists or not.

Parameters

index:
The index of disk slot.

value:
The disk in the specified slot exists or not. If the disk exists, return **1**; if the disk is not present, return **0**.

Return Value

If the slot status is successfully retrieved, return **0**; otherwise, return **-1**.

Error codes

Name	Value	Meaning
Return Value	-1	Failed to retrieve the slot status.

Requirements

Name	Items
Header	HardwareMonitorApi.h
Library	HardwareMonitorApi.lib
DLL	HardwareMonitorApi.dll

GetDiskHealthStatus

Syntax

```
int GetDiskHealthStatus(
    int index,
    int *value
);
```

Description

Retrieves the health status of the specified disk.

Parameters

index:
The index of disk slot.

value:
The health status of the disk:

- If no disk is inserted, return **1**.
- If the disk status is **"Good"**, return **2**.
- If the disk status is **"Warning"**, return **3**.
- If the status is **unknown**, return **-1**.

Return Value

If the disk health status is successfully retrieved, return **0**; otherwise, return **-1**.

Error codes

Name	Value	Meaning
Return Value	-1	Failed to retrieve the disk health status.

Requirements

Name	Items
Header	HardwareMonitorApi.h
Library	HardwareMonitorApi.lib
DLL	HardwareMonitorApi.dll

GetDiskSerialNumber

Syntax

```
int GetDiskSerialNumber(
    int index,
    char *value
);
```

Description

Retrieves the serial number of the specified disk.

Parameters

index:
The index of disk slot.

value:
The serial number of the disk.

Return Value

If the disk serial number of the disk is successfully retrieved, return **0**; otherwise, return **-1**.

Error codes

Name	Value	Meaning
Return Value	-1	Failed to retrieve the disk serial number.

Requirements

Name	Items
Header	HardwareMonitorApi.h
Library	HardwareMonitorApi.lib
DLL	HardwareMonitorApi.dll

GetDiskAvgEraseCount

Syntax

```
int GetDiskAvgEraseCount(
    int index,
    int *value
);
```

Description

Retrieves the erase count of the specified disk.

Parameters

index:
The index of disk slot.

value:
The erase count of the disk.

Return Value

If the disk erase count is successfully retrieved, return **0**; otherwise, return **-1**.

Error codes

Name	Value	Meaning
Return Value	-1	Failed to retrieve the disk erase count.

Requirements

Name	Items
Header	HardwareMonitorApi.h
Library	HardwareMonitorApi.lib
DLL	HardwareMonitorApi.dll

Memory Status

GetMemUsage

Syntax

```
int GetMemUsage(
    int *value
);
```

Description

Retrieves the total usage of memory.

Parameters

value:
Memory total usage in percent.

Return Value

If the memory usage is successfully retrieved, return **0**; otherwise, return **-1**.

Error codes

Name	Value	Meaning
Return Value	-1	Failed to retrieve memory usage.

Requirements

Name	Items
Header	HardwareMonitorApi.h
Library	HardwareMonitorApi.lib
DLL	HardwareMonitorApi.dll

GetMemTotalSize

Syntax

```
int GetMemTotalSize(
    int *value
);
```

Description

Retrieves the total memory size.

Parameters

value:
Total memory size, measured in MB.

Return Value

If the total memory size is successfully retrieved, return **0**; otherwise, return **-1**.

Error codes

Name	Value	Meaning
Return Value	-1	Failed to retrieve total memory size.

Requirements

Name	Items
Header	HardwareMonitorApi.h
Library	HardwareMonitorApi.lib
DLL	HardwareMonitorApi.dll

GetMemAvailSize

Syntax

```
int GetMemAvailSize(
    int *value
);
```

Description

Retrieves the available memory size.

Parameters

value:
Available memory size, measured in MB.

Return Value

If the available memory size is successfully retrieved, return **0**; otherwise, return **-1**.

Error codes

Name	Value	Meaning
Return Value	-1	Failed to retrieve available memory size.

Requirements

Name	Items
Header	HardwareMonitorApi.h
Library	HardwareMonitorApi.lib
DLL	HardwareMonitorApi.dll

Ethernet Status

GetEthConnectionID

Syntax

```
int GetEthConnectionID(
    int index,
    char *value
);
```

Description

Retrieves the connection ID of the specified network adapter.

Parameters

index:
The index of network adapter

value:
The connection ID of the network adapter.

Return Value

If the network adapter connection ID is successfully retrieved, return **0**; otherwise, return **-1**.

Error codes

Name	Value	Meaning
Return Value	-1	Failed to retrieve the network adapter connection ID.

Requirements

Name	Items
Header	HardwareMonitorApi.h
Library	HardwareMonitorApi.lib
DLL	HardwareMonitorApi.dll

GetEthDescr

Syntax

```
int GetEthDescr(
    int index,
    char *value
);
```

Description

Retrieves the description of the specified network adapter.

Parameters

index:
The index of network adapter.

value:
The description of the network adapter.

Return Value

If the network adapter description is successfully retrieved, return **0**; otherwise, return **-1**.

Error codes

Name	Value	Meaning
Return Value	-1	Failed to retrieve the network adapter description.

Requirements

Name	Items
Header	HardwareMonitorApi.h
Library	HardwareMonitorApi.lib
DLL	HardwareMonitorApi.dll

GetEthCount

Syntax

```
int GetEthCount(
    int *value
);
```

Description

Retrieves the network adapter count on device.

Parameters

value:
The network adapter count.

Return Value

If the network adapter count is successfully retrieved, return **0**; otherwise, return **-1**.

Error codes

Name	Value	Meaning
Return Value	-1	Failed to retrieve the network adapter count.

Requirements

Name	Items
Header	HardwareMonitorApi.h
Library	HardwareMonitorApi.lib
DLL	HardwareMonitorApi.dll

GetEthSpeed

Syntax

```
int GetEthSpeed(
    int index,
    int *value
);
```

Description

Retrieves the current connection speed, which can be **0**, **10**, **100**, or **1000**.

Parameters

index:
The index of network adapter.

value:
The network adapter connection speed.

Return Value

If the network adapter count is successfully retrieved, return **0**; otherwise, return **-1**.

Error codes

Name	Value	Meaning
Return Value	-1	Failed to retrieve the network adapter connection speed.

Requirements

Name	Items
Header	HardwareMonitorApi.h
Library	HardwareMonitorApi.lib
DLL	HardwareMonitorApi.dll

GetEthLink

Syntax

```
int GetEthLink(
    int index,
    int *value
);
```

Description

Retrieves the network adapter connection status.

Parameters

index:
The index of network adapter.

value:
The connection status of the network adapter. Return **0** for disconnected and **1** for connected.

Return Value

If the network adapter connection status is successfully retrieved, return **0**; otherwise, return **-1**.

Error codes

Name	Value	Meaning
Return Value	-1	Failed to retrieve the network adapter connection status.

Requirements

Name	Items
Header	HardwareMonitorApi.h
Library	HardwareMonitorApi.lib
DLL	HardwareMonitorApi.dll

GetEthUsage

Syntax

```
int GetEthUsage(
    int index,
    int *value
);
```

Description

Retrieves the network adapter utilization.

Parameters

index:
The index of network adapter.

value:
The network adapter utilization.

Return Value

If the network adapter utilization is successfully retrieved, return **0**; otherwise, return **-1**.

Error codes

Name	Value	Meaning
Return Value	-1	Failed to retrieve the network adapter utilization.

Requirements

Name	Items
Header	HardwareMonitorApi.h
Library	HardwareMonitorApi.lib
DLL	HardwareMonitorApi.dll

Partition Status

GetPartitionUsage

Syntax

```
int GetPartitionUsage(
    int index,
    int *value
);
```

Description

Retrieves the utilization of the specified partition.

Parameters

index:
The index of partition.

value:
The utilization of partition.

Return Value

If the partition utilization is successfully retrieved, return **0**; otherwise, return **-1**.

Error codes

Name	Value	Meaning
Return Value	-1	Failed to retrieve the partition utilization.

Requirements

Name	Items
Header	HardwareMonitorApi.h
Library	HardwareMonitorApi.lib
DLL	HardwareMonitorApi.dll

GetPartitionTotalSize

Syntax

```
int GetPartitionTotalSize(
    int index,
    int *value
);
```

Description

Retrieves the total capacity of the specified partition.

Parameters

index:
The index of partition.

value:
The total capacity of the partition.

Return Value

If the total capacity of partition is successfully retrieved, return **0**; otherwise, return **-1**.

Error codes

Name	Value	Meaning
Return Value	-1	Failed to retrieve the total capacity of partition.

Requirements

Name	Items
Header	HardwareMonitorApi.h
Library	HardwareMonitorApi.lib
DLL	HardwareMonitorApi.dll

GetPartitionName

Syntax

```
int GetPartitionName(
    int index,
    char *value
);
```

Description

Retrieves the label name of specified partition.

Parameters

index:
The index of partition.

value:
The label name of partition.

Return Value

If the partition label is successfully retrieved, return **0**; otherwise, return **-1**.

Error codes

Name	Value	Meaning
Return Value	-1	Failed to retrieve the partition label.

Requirements

Name	Items
Header	HardwareMonitorApi.h
Library	HardwareMonitorApi.lib
DLL	HardwareMonitorApi.dll

GetPartitionAvailSize

Syntax

```
int GetPartitionAvailSize(
    int index,
    int *value
);
```

Description

Retrieves the available capacity of the specified partition.

Parameters

index:
The index of partition.

value:
The available capacity of partition.

Return Value

If the available capacity is successfully retrieved, return **0**; otherwise, return **-1**.

Error codes

Name	Value	Meaning
Return Value	-1	Failed to retrieve the available capacity of the specified partition.

Requirements

Name	Items
Header	HardwareMonitorApi.h
Library	HardwareMonitorApi.lib
DLL	HardwareMonitorApi.dll

Mainboard Status

GetPwrStatus

Syntax

```
int GetPwrStatus(
    int index,
    int *value
);
```

Description

Retrieves the power status of specified power module.

Parameters

index:
The index of power module.

value:
Retrieve the power status of the specified power module. Return **0** if the power module has failed, and **1** if it is active.

Return Value

If the power status of power module is successfully retrieved, return **0**; otherwise, return **-1**.

Error codes

Name	Value	Meaning
Return Value	-1	Failed to retrieve the power status of power module.

Requirements

Name	Items
Header	HardwareMonitorApi.h
Library	HardwareMonitorApi.lib
DLL	HardwareMonitorApi.dll

GetPwrIndicator

Syntax

```
int GetPwrIndicator(
    int *value
);
```

Description

Retrieves the power status of the power module.

Parameters

value:
The power status of the power module with the following return values:

- **0:** Not supported.
- **1:** Module1 is Active and Module2 is Failed.
- **2:** Module1 is Failed and Module2 is Active.
- **3:** Both Module1 and Module2 are Active.

Return Value

If the power indicator value is successfully retrieved, return **0**; otherwise, return **-1**.

Error codes

Name	Value	Meaning
Return Value	-1	Failed to retrieve the power indicator value.

Requirements

Name	Items
Header	HardwareMonitorApi.h
Library	HardwareMonitorApi.lib
DLL	HardwareMonitorApi.dll

RAID Status

GetIntelRaidDiskCount

Syntax

```
int GetIntelRaidDiskCount(
    int raidIndex,
    int *value
);
```

Description

Retrieves the disk count of specified Intel® RAID volume.

Parameters

raidIndex:
The index of Intel® RAID volume.

value:
The disk count of Intel® RAID volume.

Return Value

If the disk count of Intel® RAID volume is successfully retrieved, return **0**; otherwise, return **-1**.

Error codes

Name	Value	Meaning
Return Value	-1	Failed to retrieve the disk count of Intel® RAID volume.

Requirements

Name	Items
Header	HardwareMonitorApi.h
Library	HardwareMonitorApi.lib
DLL	HardwareMonitorApi.dll

GetIntelRaidDiskSerialNumber

Syntax

```
int GetIntelRaidDiskSerialNumber(
    int raidIndex,
    int diskIndex,
    char *value
);
```

Description

Retrieves the serial number of the specified disk within the designated Intel® RAID volume.

Parameters

raidIndex:
The index of Intel® RAID volume.

diskIndex:
The index of disk on the Intel® RAID volume.

value:
The serial number of disk.

Return Value

If the serial number of the disk is successfully retrieved, return **0**; otherwise, return **-1**.

Error codes

Name	Value	Meaning
Return Value	-1	Failed to retrieve the serial number of disk on Intel® RAID volume.

Requirements

Name	Items
Header	HardwareMonitorApi.h
Library	HardwareMonitorApi.lib
DLL	HardwareMonitorApi.dll

GetIntelRaidDiskPort

Syntax

```
int GetIntelRaidDiskPort(
    int raidIndex,
    int diskIndex,
    int *value
);
```

Description

Retrieves the port number of the specified disk within the designated Intel® RAID volume.

Parameters

raidIndex:
The index of Intel® RAID volume.

diskIndex:
The index of disk on the Intel® RAID volume.

value:
The port number of disk.

Return Value

If the port number of disk is successfully retrieved, return **0**; otherwise, return **-1**.

Error codes

Name	Value	Meaning
Return Value	-1	Failed to retrieve the port number of disk.

Requirements

Name	Items
Header	HardwareMonitorApi.h
Library	HardwareMonitorApi.lib
DLL	HardwareMonitorApi.dll

GetIntelRaidDiskStatus

Syntax

```
int GetIntelRaidDiskStatus(
    int raidIndex,
    int diskIndex,
    char *value
);
```

Description

Retrieves the RAID status of the specified disk within the designated Intel® RAID volume.

Parameters

raidIndex:
The index of Intel® RAID volume.

diskIndex:
The index of disk on the Intel® RAID volume.

value:
The RAID status of disk.

Return Value

If the RAID status of disk is successfully retrieved, return **0**; otherwise, return **-1**.

Error codes

Name	Value	Meaning
Return Value	-1	Failed to retrieve the RAID status of disk.

Requirements

Name	Items
Header	HardwareMonitorApi.h
Library	HardwareMonitorApi.lib
DLL	HardwareMonitorApi.dll

GetRaidCount

Syntax

```
int GetRaidCount(
    int *value
);
```

Description

Retrieves the count of Intel® RAID and Software RAID arrays on the system.

Parameters

value:
The count of Intel® RAID and Software RAID arrays.

Return Value

If the count of RAID arrays is successfully retrieved, return **0**; otherwise, return **-1**.

Error codes

Name	Value	Meaning
Return Value	-1	Failed to retrieve the count of RAID arrays.

Requirements

Name	Items
Header	HardwareMonitorApi.h
Library	HardwareMonitorApi.lib
DLL	HardwareMonitorApi.dll

GetRaidMode

Syntax

```
int GetRaidMode(
    int index,
    int *value
);
```

Description

Retrieves the RAID mode of Intel® RAID and Software RAID arrays.

Parameters

index:
The index of RAID arrays on the system.

value:
The power status of power module.

The RAID mode of the RAID arrays with the following return values:

- **0:** Intel® RAID **- RAID 0**; Software RAID **- Stripe**
- **1:** Intel® RAID **- RAID 1**; Software RAID **- Mirror**
- **2:** Software RAID **- Spanned**
- **5:** Intel® RAID **- RAID 5**; Software RAID **- RAID-5**
- **10:** Intel® RAID **- RAID 10**

Return Value

If the RAID mode of RAID arrays is successfully retrieved, return **0**; otherwise, return **-1**.

Error codes

Name	Value	Meaning
Return Value	-1	Failed to retrieve the RAID mode of RAID arrays.

Requirements

Name	Items
Header	HardwareMonitorApi.h
Library	HardwareMonitorApi.lib
DLL	HardwareMonitorApi.dll

GetRaidRedundancyStatus

Syntax

```
int GetRaidRedundancyStatus(
    int index,
    int *value
);
```

Description

Retrieves the redundancy status of specified disk.

Parameters

index:
The index of disk.

value:
The redundancy status of disk.

Return Value

If the redundancy status of disk is successfully retrieved, return **0**; otherwise, return **-1**.

Error codes

Name	Value	Meaning
Return Value	-1	Failed to retrieve the redundancy status of disk.

Requirements

Name	Items
Header	HardwareMonitorApi.h
Library	HardwareMonitorApi.lib
DLL	HardwareMonitorApi.dll

GetRaidVolumeName

Syntax

```
int GetRaidVolumeName(
    int index,
    char *value
);
```

Description

Retrieves the RAID volume name of specified disk.

Parameters

index:
The index of disk.

value:
The RAID volume name of disk.

Return Value

If the RAID volume name of disk is successfully retrieved, return **0**; otherwise, return **-1**.

Error codes

Name	Value	Meaning
Return Value	-1	Failed to retrieve the RAID volume name of disk.

Requirements

Name	Items
Header	HardwareMonitorApi.h
Library	HardwareMonitorApi.lib
DLL	HardwareMonitorApi.dll

GetIntelRaidCount

Syntax

```
int GetIntelRaidCount(
    int *value
);
```

Description

Retrieves the count of Intel® RAID arrays.

Parameters

value:
The count of Intel® RAID arrays.

Return Value

If the count of Intel® RAID arrays is successfully retrieved, return **0**; otherwise, return **-1**.

Error codes

Name	Value	Meaning
Return Value	-1	Failed to retrieve the count of Intel® RAID arrays.

Requirements

Name	Items
Header	HardwareMonitorApi.h
Library	HardwareMonitorApi.lib
DLL	HardwareMonitorApi.dll

GetIntelRaidStatus

Syntax

```
int GetIntelRaidStatus(
    int index,
    int *value
);
```

Description

Retrieves the status of the disk on the Intel® RAID volume.

Parameters

index:
The index of disk.

value:
Retrieve the status of the disk on the Intel® RAID with the following return values:

- **0:** Failed
- **1:** Normal
- **2:** Rebuilding
- **3:** General Migration
- **-1:** Unknown

Return Value

If the status of the disk is successfully retrieved, return **0**; otherwise, return **-1**.

Error codes

Name	Value	Meaning
Return Value	-1	Failed to retrieve the status of the disk.

Requirements

Name	Items
Header	HardwareMonitorApi.h
Library	HardwareMonitorApi.lib
DLL	HardwareMonitorApi.dll

GetIntelRaidVolumeName

Syntax

```
int GetIntelRaidVolumeName(
    int index,
    char *value
);
```

Description

Retrieves the Intel® RAID volume name of specified disk.

Parameters

index:
The index of disk.

value:
The Intel® RAID volume name of disk.

Return Value

If the Intel® RAID volume name of disk is successfully retrieved, return **0**; otherwise, return **-1**.

Error codes

Name	Value	Meaning
Return Value	-1	Failed to retrieve the Intel® RAID volume name of disk.

Requirements

Name	Items
Header	HardwareMonitorApi.h
Library	HardwareMonitorApi.lib
DLL	HardwareMonitorApi.dll

Serial Port Status

GetUartCount

Syntax

```
int GetUartCount(
    int *value
);
```

Description

Retrieves the count of UART port.

Parameters

value:
The count of UART port.

Return Value

If the UART port count is successfully retrieved, return **0**; otherwise, return **-1**.

Error codes

Name	Value	Meaning
Return Value	-1	Failed to retrieve the count of UART port.

Requirements

Name	Items
Header	HardwareMonitorApi.h
Library	HardwareMonitorApi.lib
DLL	HardwareMonitorApi.dll

GetUartStatus

Syntax

```
int GetUartStatus(
    int index,
    int *value
);
```

Description

Retrieves the status of specified UART port.

Parameters

index:
The index of UART port.

value:
The status of UART port with the following return values:

0: Normal
1: Busy

Return Value

If the status of UART port is successfully retrieved, return **0**; otherwise, return **-1**.

Error codes

Name	Value	Meaning
Return Value	-1	Failed to retrieve the status of UART port.

Requirements

Name	Items
Header	HardwareMonitorApi.h
Library	HardwareMonitorApi.lib
DLL	HardwareMonitorApi.dll

GetTemperature

Syntax

```
int GetTemperature(
    int index,
    int *value
);
```

Description

Retrieves the temperature of CPU, mainboard or memory.

Parameters

index:
The index of disk.

value:
The RAID volume name of disk.

Return Value

If the RAID volume name of disk is successfully retrieved, return **0**; otherwise, return **-1**.

Error codes

Name	Value	Meaning
Return Value	-1	Failed to retrieve the RAID volume name of disk.

Requirements

Name	Items
Header	HardwareMonitorApi.h
Library	HardwareMonitorApi.lib
DLL	HardwareMonitorApi.dll

Overview

Proactive Monitoring provides Simple Network Management Protocol (SNMP) functionality and supports SNMP v2 and v3 versions. With this function, administrators can remotely monitor device status and integrate them into the existing network management systems. SNMP v2 provides basic monitoring capabilities, while SNMP v3 provides additional security enhancements that support encryption and user-authentication mechanisms, making network management more secure and reliable.

Through these methods, you can effectively use scripted and graphical methods to collect system information and choose the most appropriate tool based on your needs.

MIB File

Management Information Base (MIB) is a database that describes manageable objects on network devices. It defines various parameters and states of devices in a hierarchical structure, allowing the management system to query or set these parameters through SNMP.

Proactive Monitoring provides its own MIB file, which can be found at:

Path
C:\Program Files\Moxa\Moxa Proactive Monitoring\SNMP\PROACTIVE-MONITORING-MIB.my

This MIB file allows users to easily integrate Proactive Monitoring into existing SNMP management systems and query system status and event information.

OIDs

OID	Name	Data Type	Access	Description
.1.3.6.1.4.1.8691.17.2.1.1	cpuLogicCount	Int	read-only	The number of CPU logic processors.
.1.3.6.1.4.1.8691.17.2.1.2	cpuTotalUsage	Int	read-only	Current overall CPU utilization, measured in percentage, with a range from 0% to 100%.
.1.3.6.1.4.1.8691.17.2.1.3	cpuTemperature	Int	read-only	Current CPU temperature, measured in Celsius.
.1.3.6.1.4.1.8691.17.2.1.4	cpuVoltage	Int	read-only	Current CPU voltage, measured in mV.
.1.3.6.1.4.1.8691.17.2.1.5.1.1	logicProcessorIndex	Int	read-only	Each logical processor is assigned a unique value. This value ranges from 1 to the total number of logical processors on the managed device
.1.3.6.1.4.1.8691.17.2.1.5.1.2	logicProcessorUsage	Int	read-only	Current usage of each logical processor, measured in percentage, ranges from 0% to 100%.
.1.3.6.1.4.1.8691.17.2.2.1	memoryUsage	Int	read-only	Current usage of memory, measured in percentage, ranges from 0% to 100%.
.1.3.6.1.4.1.8691.17.2.2.2	memoryTotalSize	Int	read-only	Total memory capacity, measured in MB.
.1.3.6.1.4.1.8691.17.2.2.3	memoryAvailableSize	Int	read-only	Available memory size, measured in MB.

OID	Name	Data Type	Access	Description
.1.3.6.1.4.1.8691.17.2.2.4	dramVoltage	Int	read-only	Current DRAM voltage, measured in mV.
.1.3.6.1.4.1.8691.17.2.3.1.1.1	partitionStatusIndex	Int	read-only	A unique value for each partition on the managed device. The ranges of this value from 1 to the number of partition on the managed device.
.1.3.6.1.4.1.8691.17.2.3.1.1.2	partitionFriendlyName	String	read-only	The partition friendly name on the managed device. The value of this object should be the drive letter. (e.g., 'C' on the Windows system)
.1.3.6.1.4.1.8691.17.2.3.1.1.3	partitionUsage	Int	read-only	Current partition utilization, measured in percentage, ranges from 0% to 100%.
.1.3.6.1.4.1.8691.17.2.3.1.1.4	partitionTotalSize	Int	read-only	Current partition total size, measured in MB.
.1.3.6.1.4.1.8691.17.2.3.1.1.5	partitionAvailableSize	Int	read-only	Available partition size, measured in MB.
.1.3.6.1.4.1.8691.17.2.4.1	mainboardPowerIndicator	Int	read-only	The mainboard power status. A 'notsupport(0)' is returned when the managed device does not support the power indicator function. The 'pwr1(1)' is returned when only power 1 is activated. The 'pwr2(2)' is returned when only power 2 is activated. A 'dualpower(3)' is returned when both powers are activated.
.1.3.6.1.4.1.8691.17.2.4.2	mainboardTemperature	Int	read-only	Current mainboard temperature, measured in Celsius.
.1.3.6.1.4.1.8691.17.2.4.3	mainboardVoltage	Int	read-only	Current mainboard voltage, measured in mV.
.1.3.6.1.4.1.8691.17.2.4.4	mainboardPower1Status	Int	read-only	The mainboard power 1 status. The 'fail(0)' is returned when power 1 is deactivated. The 'activated(1)' is returned when power 1 is activated. The 'notsupport(2)' is returned when the managed device does not support the power indicator.
.1.3.6.1.4.1.8691.17.2.4.5	mainboardPower2Status	Int	read-only	The mainboard power 2 status. The 'fail(0)' is returned when power 1 is deactivated. The 'activated(1)' is returned when power 1 is activated. The 'notsupport(2)' is returned when the managed device does not support the power indicator.
.1.3.6.1.4.1.8691.17.2.5.1.1.1	portStatusIndex	Int	read-only	A unique value for each serial port on the managed device. The ranges of this value from 1 to the number of serial port on the managed device.
.1.3.6.1.4.1.8691.17.2.5.1.1.2	serialPortFriendlyName	String	read-only	The serial port friendly name on the managed device.

OID	Name	Data Type	Access	Description
.1.3.6.1.4.1.8691.17.2.5.1.1.3	portStatusValue	Int	read-only	The serial port status of managed device. If the serial port is available, the value 'available(0)' is returned. If the serial port is in use, the value 'inuse(1)' is returned.
.1.3.6.1.4.1.8691.17.2.6.1.1.1	networkAdapterIndex	Int	read-only	A unique value for each network adapter on the managed device. The ranges of this value from 1 to the number of network adapters on the managed device.
.1.3.6.1.4.1.8691.17.2.6.1.1.2	networkFriendlyName	String	read-only	The network adapter friendly name on the managed device.
.1.3.6.1.4.1.8691.17.2.6.1.1.3	connectionStatus	Int	read-only	The network connection status. If the network cable unplugged, the value 'disconnected(0)' is returned. If the network cable plugged, the value 'connected(1)' is returned.
.1.3.6.1.4.1.8691.17.2.6.1.1.4	linkSpeed	Int	read-only	The current network link speed. The unit of value is 'M'. Represent its status to '10M' or '100M' and '1000M'.
.1.3.6.1.4.1.8691.17.2.6.1.1.5	networkUtilization	Int	read-only	Current network adapter utilization, measured in percentage, ranges from 0% to 100%.
.1.3.6.1.4.1.8691.17.2.7.1.1.1	diskStatusIndex	Int	read-only	A unique value for each disk on the managed device. The ranges of this value from 1 to the number of disk on the managed device.
.1.3.6.1.4.1.8691.17.2.7.1.1.2	diskSlotStatus	Int	read-only	The slot status on the managed device. If the disk doesn't insert into the slot when the Proactive Monitoring service start, the value 'nodisk(1)' is returned. If the disk insert into the slot, the value 'good(2)' is returned. If the disk had been unplugged from slot, the value 'unplugged(3)' is returned.
.1.3.6.1.4.1.8691.17.2.7.1.1.3	diskPort	Int	read-only	The disk port number on the managed device. The ranges of this value from 0 to the port number of disk on the managed device.
.1.3.6.1.4.1.8691.17.2.7.1.1.4	diskSerialNumber	String	read-only	The unique serial number of each disk.

OID	Name	Data Type	Access	Description
.1.3.6.1.4.1.8691.17.2.7.1.1.5	diskHealthStatus	Int	read-only	The current disk health status. If the disk doesn't insert into the slot, the value 'nodisk(1)' is returned. If the disk work properly, the value 'normal(2)' is returned. If the disk reallocated sectors count, current pending sector count and uncorrectable sector count which is more than 1, the value 'error(3)' is returned.
.1.3.6.1.4.1.8691.17.2.7.1.1.6	diskAvgEraseCount	Int	read-only	The current disk Avg Erase Count. If the disk doesn't support avg erase count, the value '-1' is returned.
.1.3.6.1.4.1.8691.17.2.9.1.1.1	raidStatusIndex	Int	read-only	A unique value for each RAID on the managed device. The ranges of this value from 1 to the number of RAID on the managed device.
.1.3.6.1.4.1.8691.17.2.9.1.1.2	raidMode	Int	read-only	The raidMode. The value is (0,1,5,10).
.1.3.6.1.4.1.8691.17.2.9.1.1.3	raidRedundancyStatus	Int	read-only	The RAID Redundancy Status.
.1.3.6.1.4.1.8691.17.2.9.1.1.4	raidVolumeName	String	read-only	The volume name of the RAID.
.1.3.6.1.4.1.8691.17.2.9.2.1.1	raidDiskStatusIndex	Int	read-only	A unique value for each disk in RAID on the managed device. The ranges of this value from 1 to the number of disk in specific RAID on the managed device.
.1.3.6.1.4.1.8691.17.2.9.2.1.2	raidDiskPort	Int	read-only	The disk port number on the managed device. The ranges of this value from 0 to the port number of disk on the managed device.
.1.3.6.1.4.1.8691.17.2.9.2.1.3	raidDiskSerialNumber	Int	read-only	The unique serial number of each disk.
.1.3.6.1.4.1.8691.17.2.9.2.1.4	raidDiskStatus	Int	read-only	The disk Status in specific RAID.
.1.3.6.1.4.1.8691.17.2.9.2.1.5	raidDiskRaidVolumeName	String	read-only	The RAID volume name of the RAID disk.

Windows SNMP Service (v2)

Windows SNMP Service is a built-in service of the Windows operating system that allows system administrators to monitor and manage network devices using the SNMP protocol. This service can receive SNMP queries and return system status information, making it suitable for network monitoring and automated management environments.

Installing Windows SNMP Service (v2)

To install the SNMP service on Windows, use the following PowerShell commands:

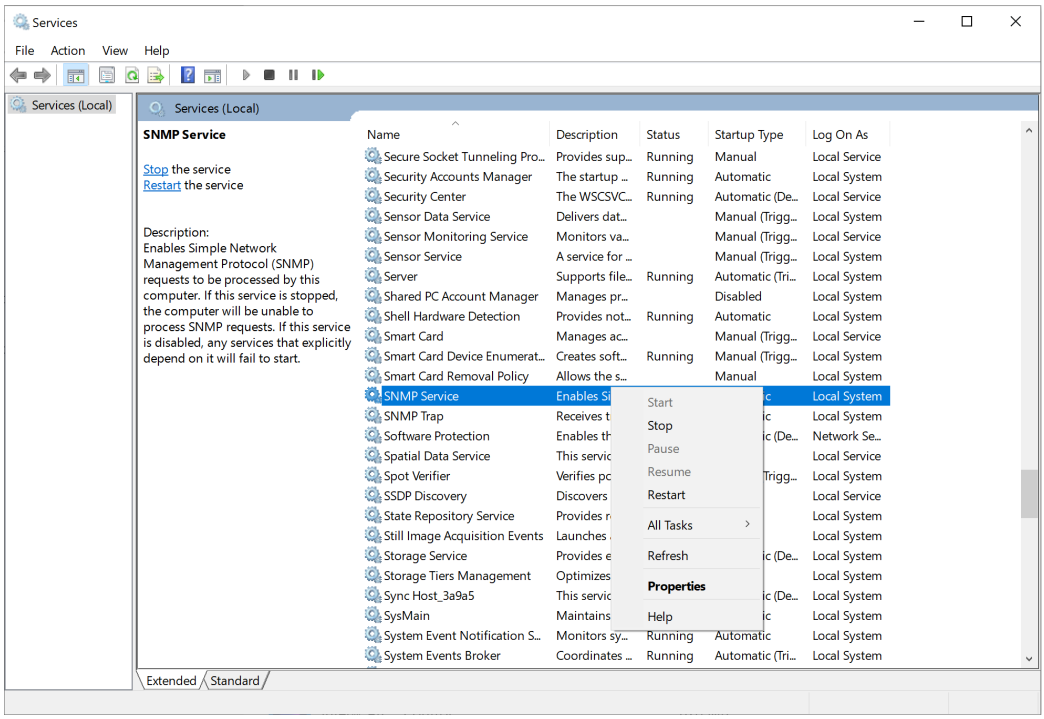
```
Add-WindowsCapability -Online -Name "SNMP.Client~~~~0.0.1.0"
Add-WindowsCapability -Online -Name "WMI-SNMP-Provider.Client~~~~0.0.1.0"
```

Code block 1 PowerShell

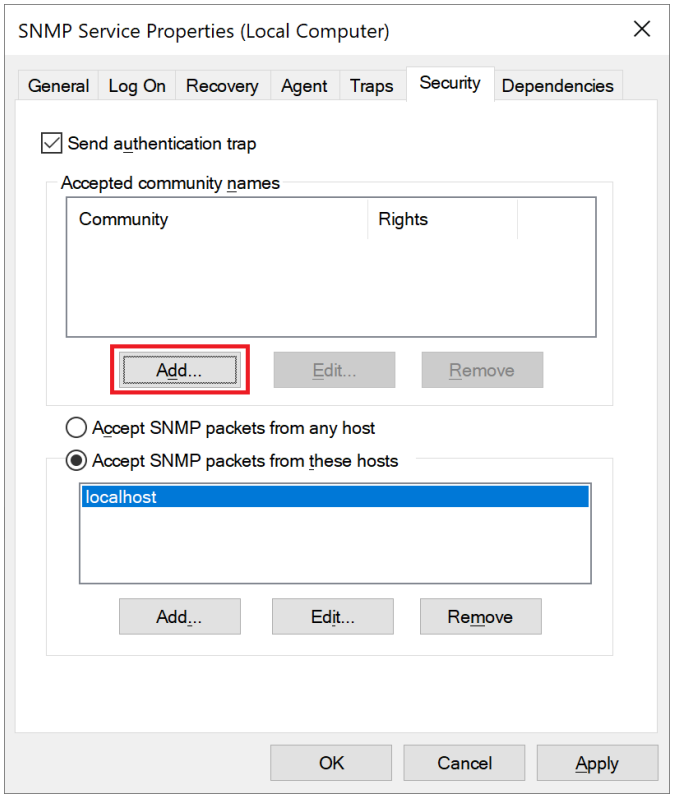
These commands will install the SNMP Client and WMI SNMP Provider, allowing the system to access SNMP information through WMI further enhancing the monitoring and management capabilities of the device.

Setting Up the Windows SNMP Service

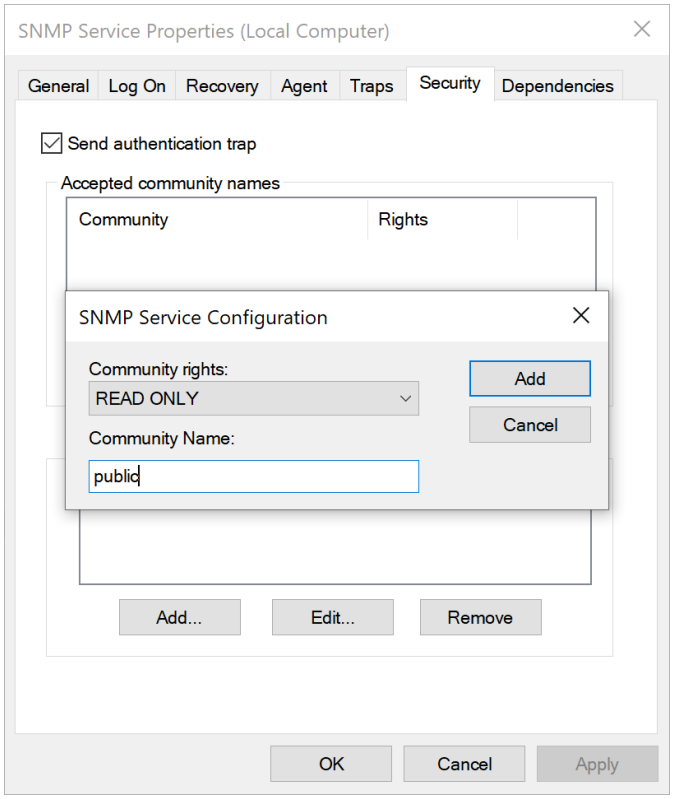
1. In the Windows **Services** page, right-click on the **SNMP Service** and select **Properties**.



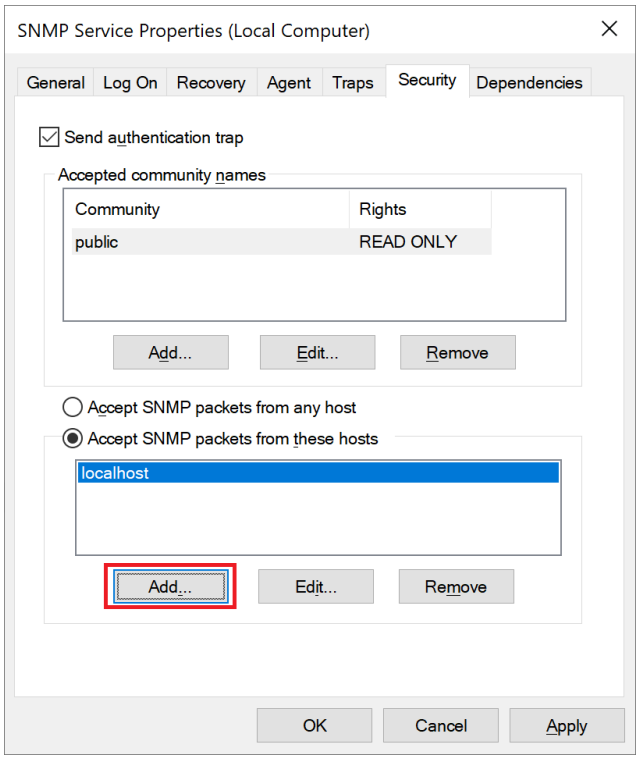
2. Select the **Security** tab and click **Add**.



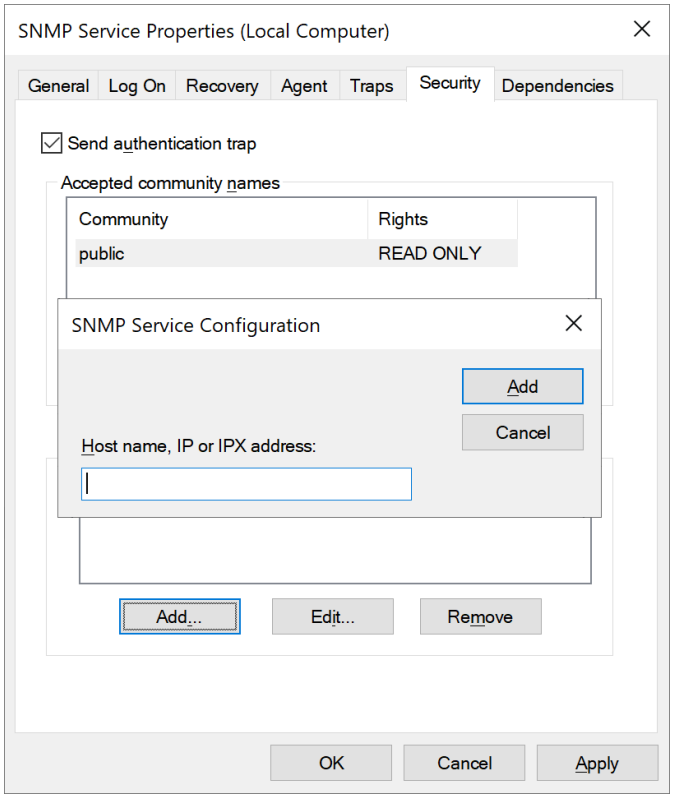
3. Enter community name and then click **Add**.



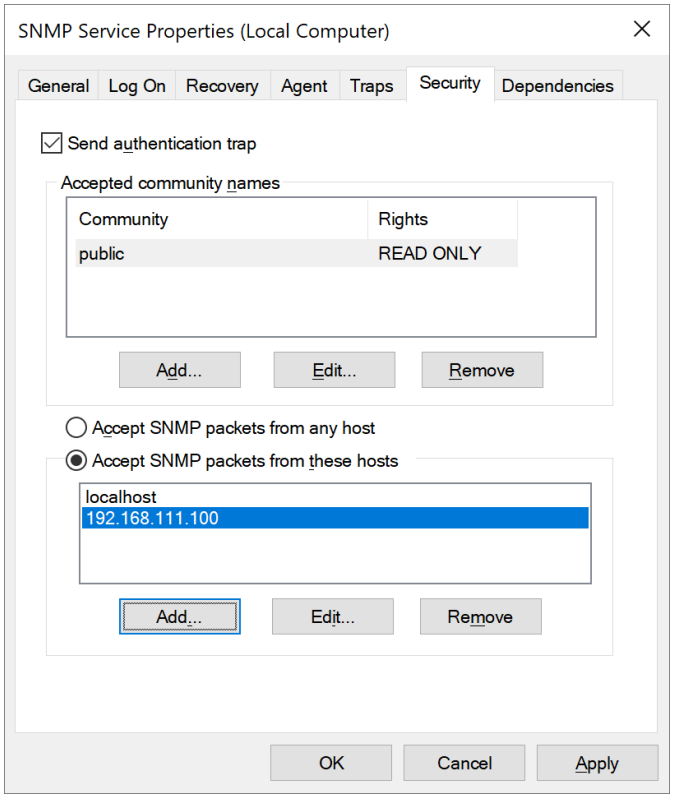
4. Click **Add**.



5. Enter the IP address that can access this device.



6. Press the **OK** to complete the setup.



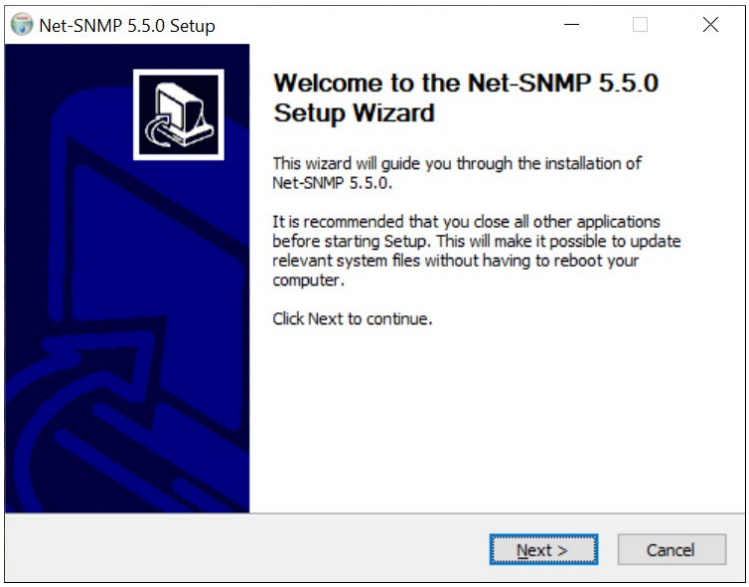
Net-SNMP Service (v2 & v3)

Net-SNMP is an open source SNMP tool suite that provides a rich set of command line tools and libraries to manage and monitor SNMP-enabled devices. It supports multiple platforms, including Windows, allowing users to easily run SNMP queries, set traps, and monitor device status in a Windows environment.

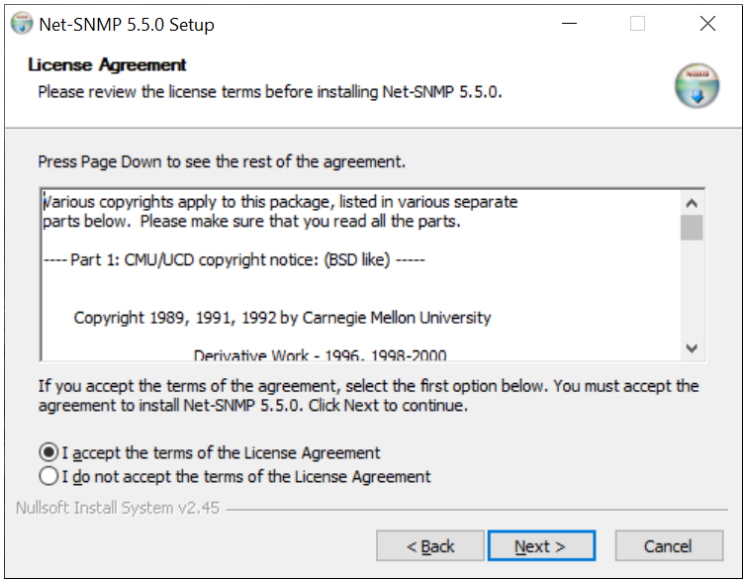
It is particularly worth mentioning that Net-SNMP supports SNMP v3, which means that users can use more secure communication mechanisms such as user authentication, encryption and access control to protect sensitive data, providing higher security and reliability.

Installing the Net-SNMP Service

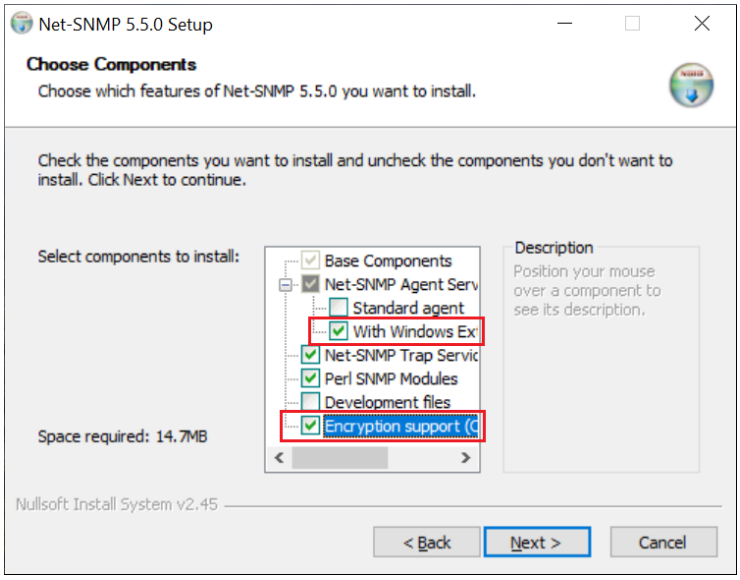
1. Install Net-SNMP, click Next to start the installation.



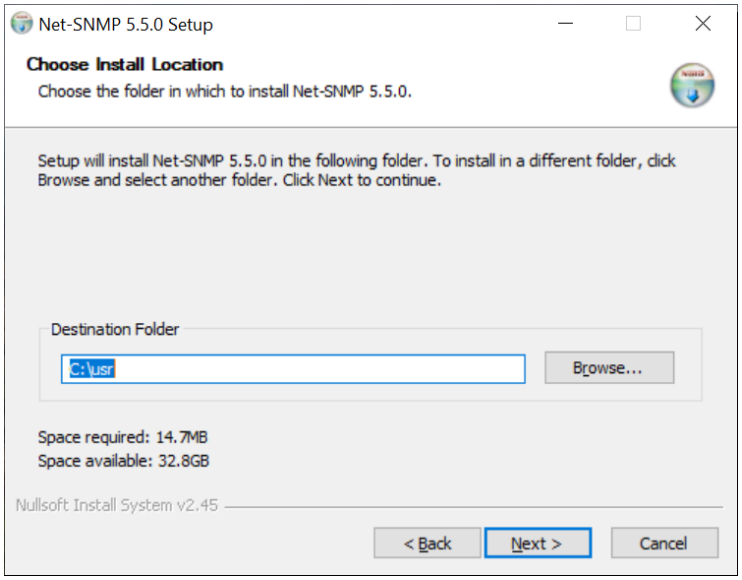
2. After agreeing to the License, select Next to continue the installation.



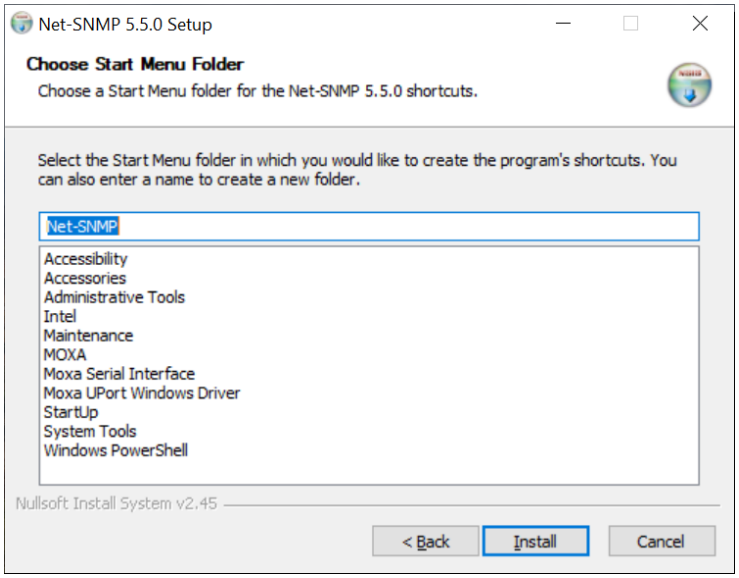
3. Check With Windows Extension DLL support and Encryption support (OpenSSL), and select Next to continue the installation.



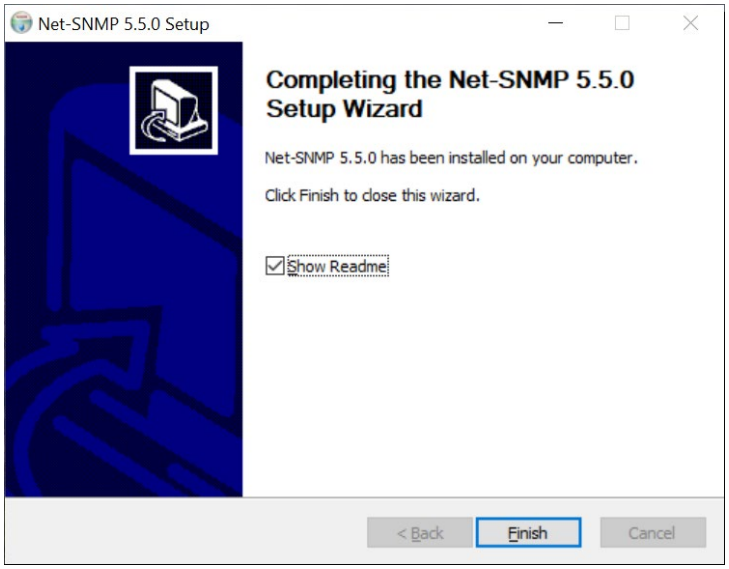
4. Confirm the installation folder and select Next to continue the installation.



5. Select Install

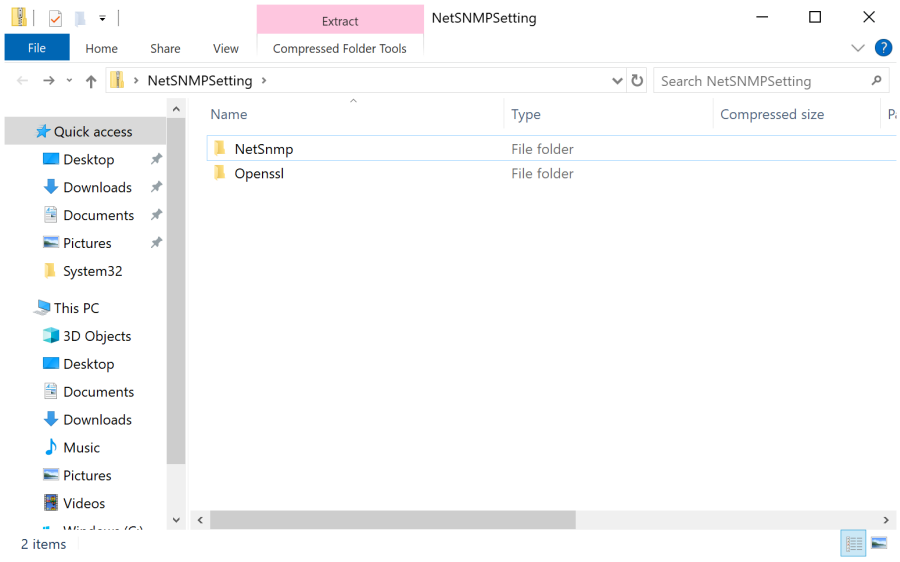


6. Select Finish to complete the installation

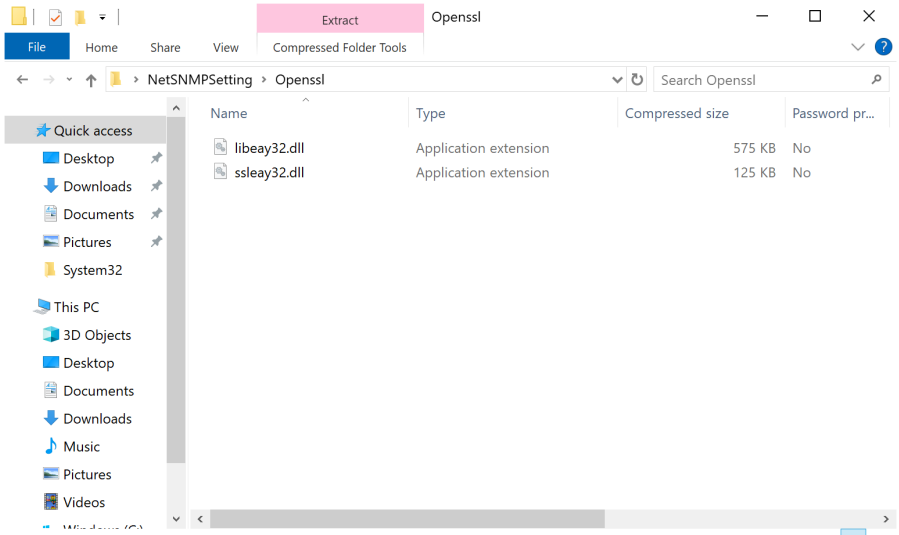


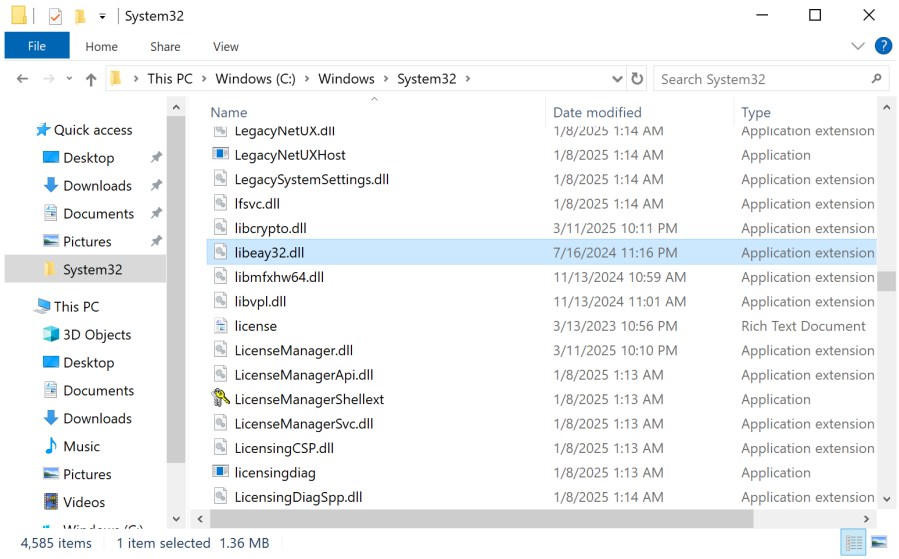
Installing NetSNMPSetting

1. Extract the contents of the NetSNMPSetting.zip to a local system.

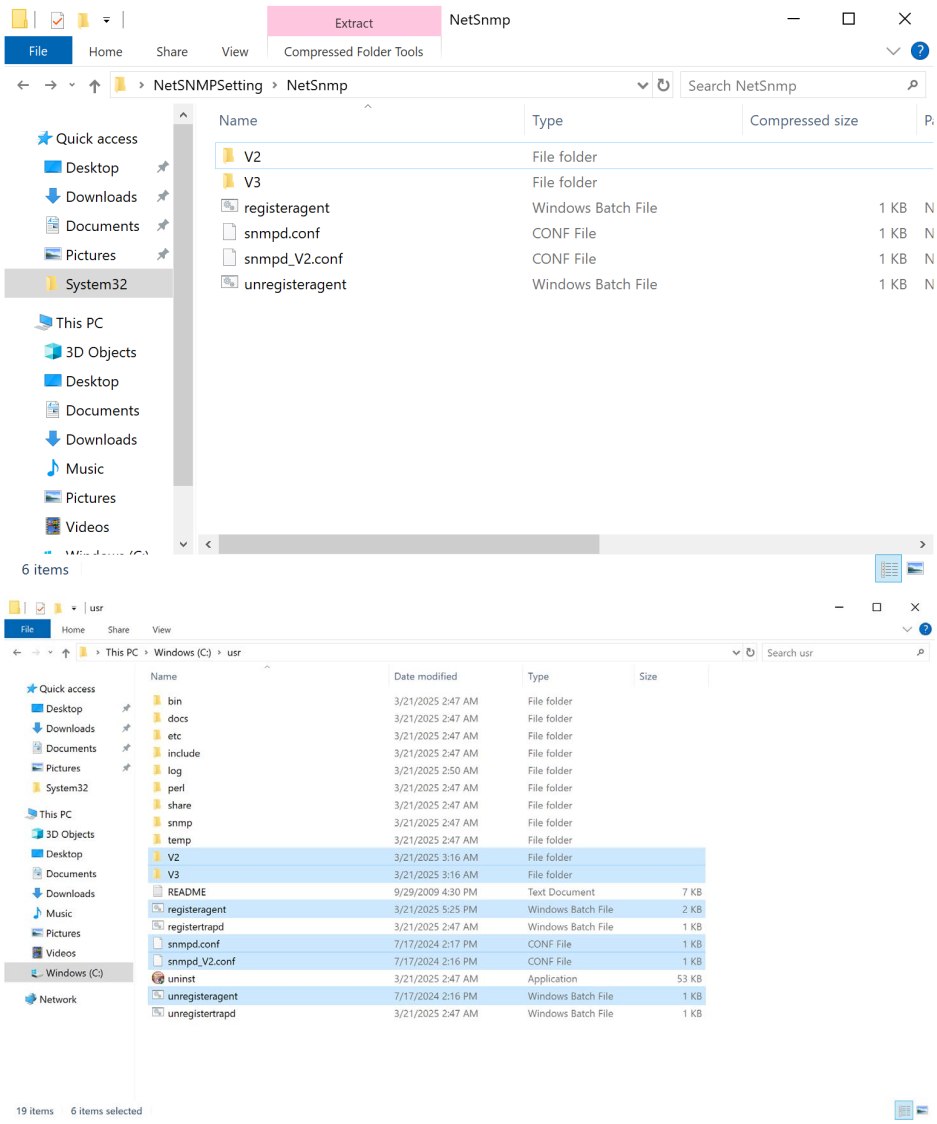


2. Copy the files in the Openssl folder to the C:\Windows\System32 folder.





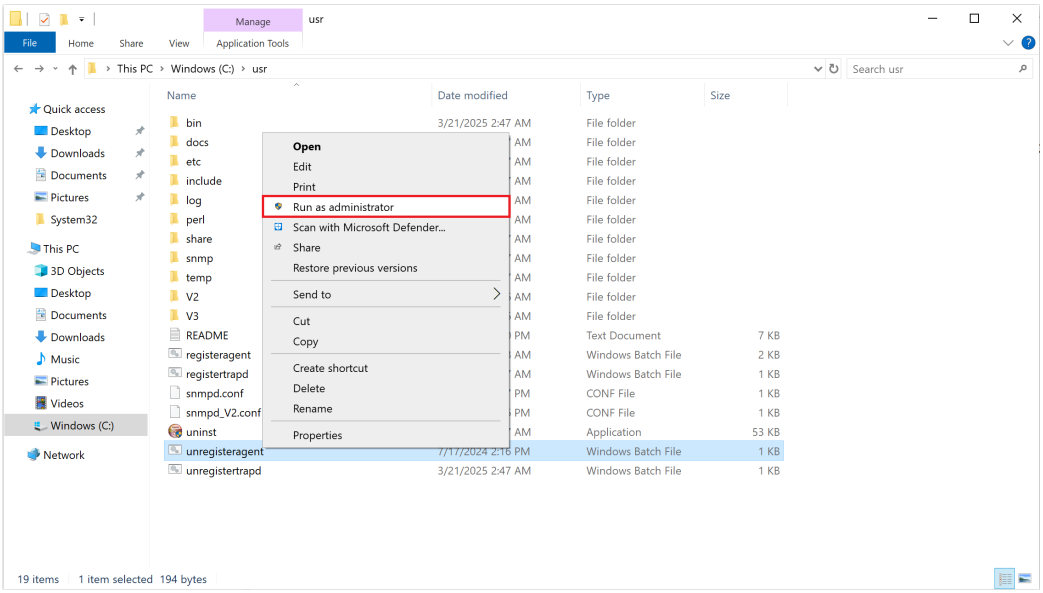
3. Copy the files in the NetSnmp folder to the C:\usr folder.



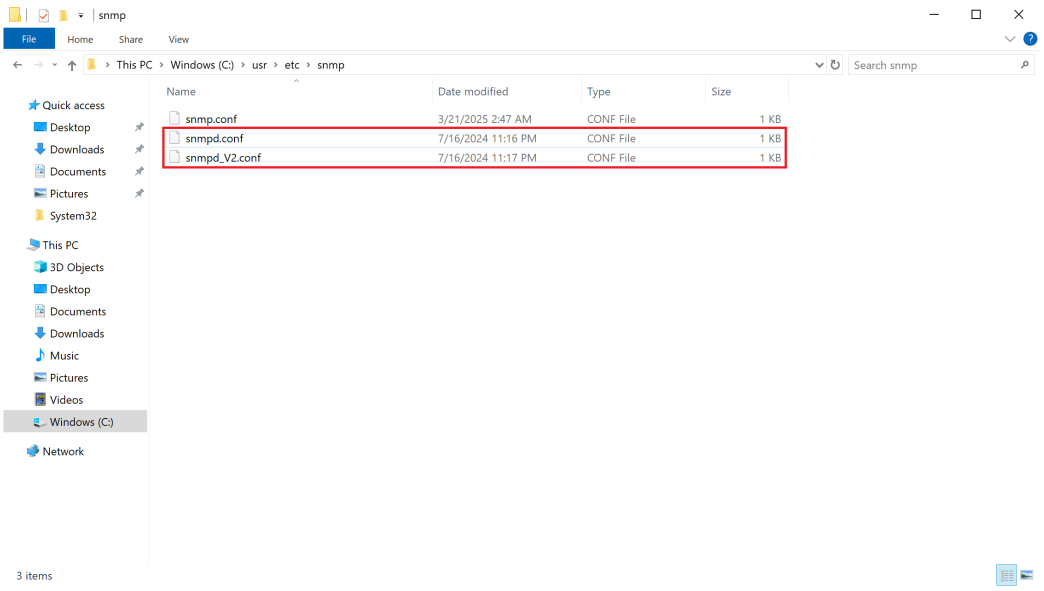
Setting Up Net-SNMP V2 Agent

1. If you have registered Net-SNMP Service before, please unregister it first. Right-click C:\usr\unregisteragent.bat and select "Run as administrator";

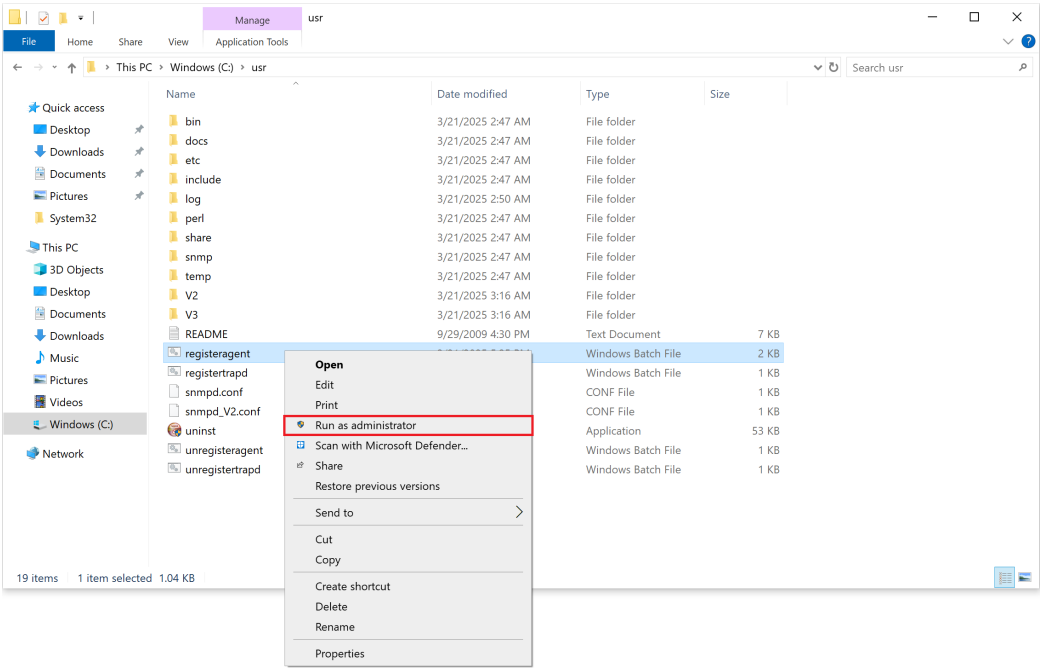
   If you have not registered before, you do not need to perform this step.



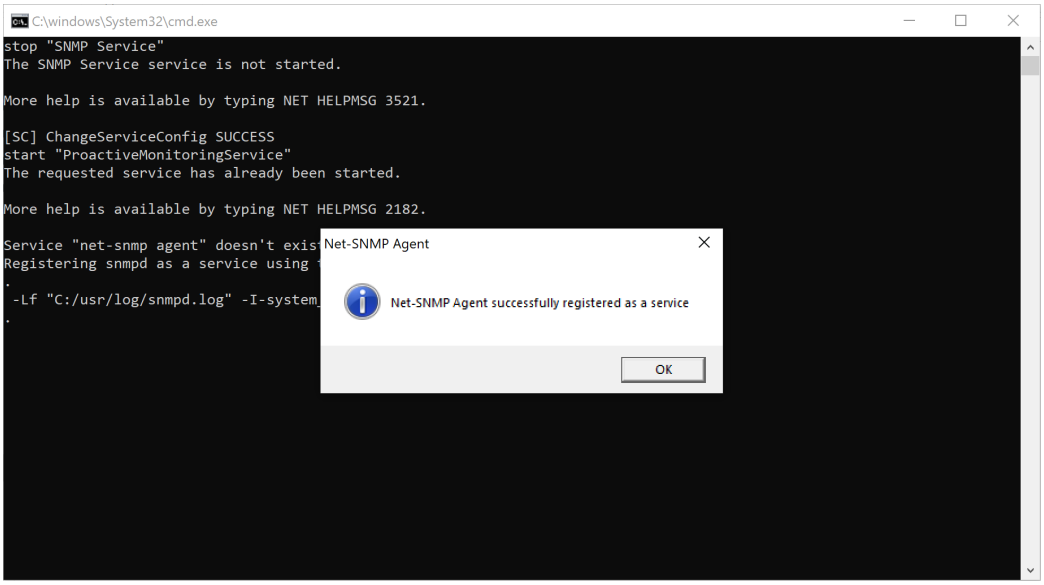
2. Rename C:\usr\etc\snmp\snmpd.conf to snmpd_V3.conf, and rename C:\usr\etc\snmp\snmpd_V2.conf to snmpd.conf.



3. Right click on C:\usr\registeragent.bat and select "Run as administrator".



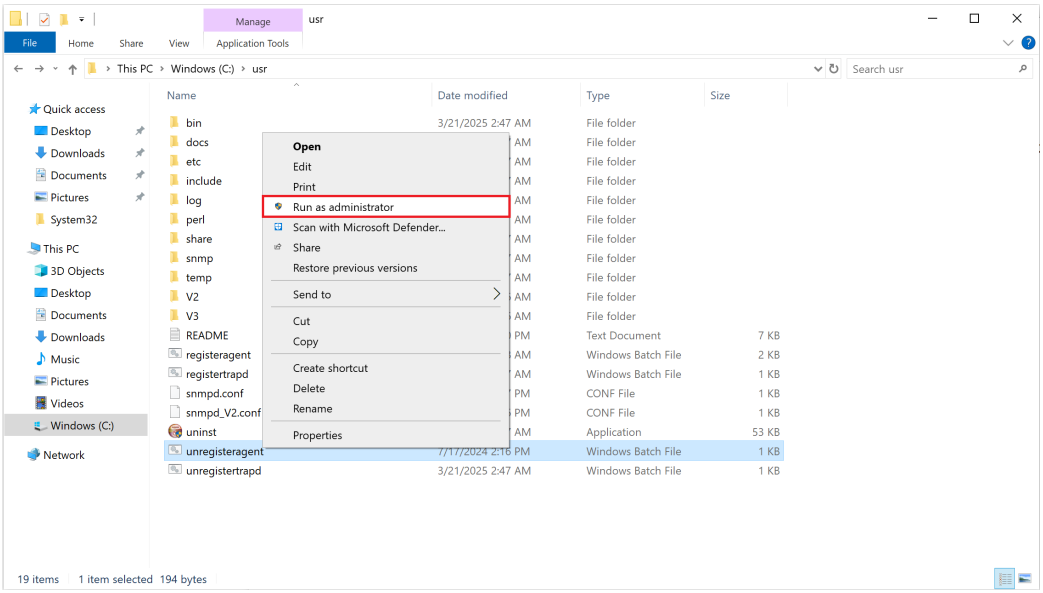
4. Press OK to complete the setting.



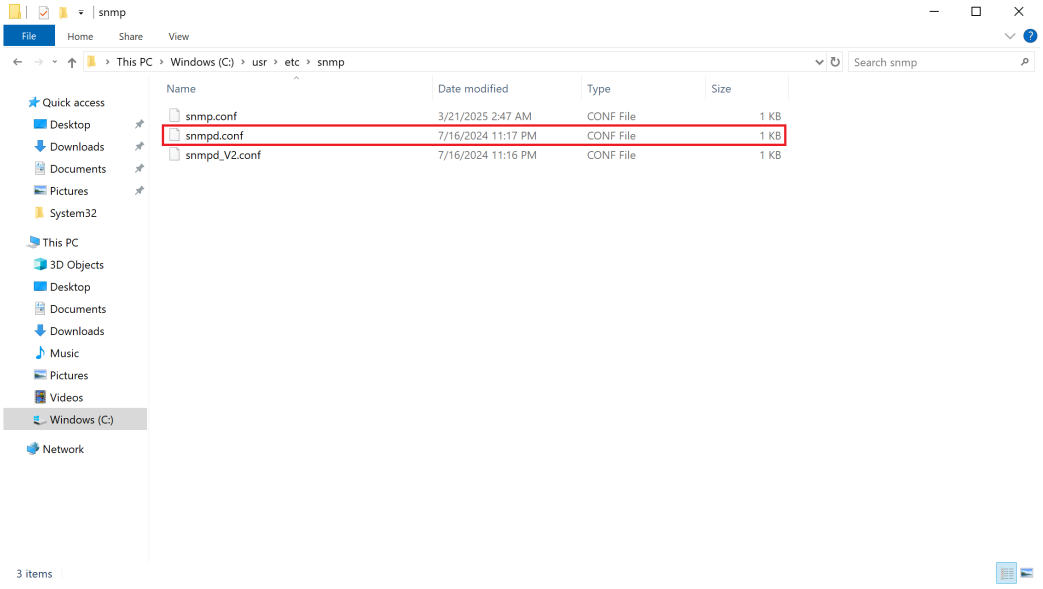
Setting Up the Net-SNMP V3 Agent

1. If you have registered Net-SNMP Service before, please unregister it first. Right-click C:\usr\unregisteragent.bat and select "Run as administrator";

   If you have not registered before, you do not need to perform this step.



2. Confirm that the content of C:\usr\etc\snmp\snmpd.conf is in V3 format.



3. Open **C:\usr\etc\snmp\snmpd.conf** and then change the user settings

   a. Set User createUser <username> <Auth Type> <Password> <encryption> <encryption Key>

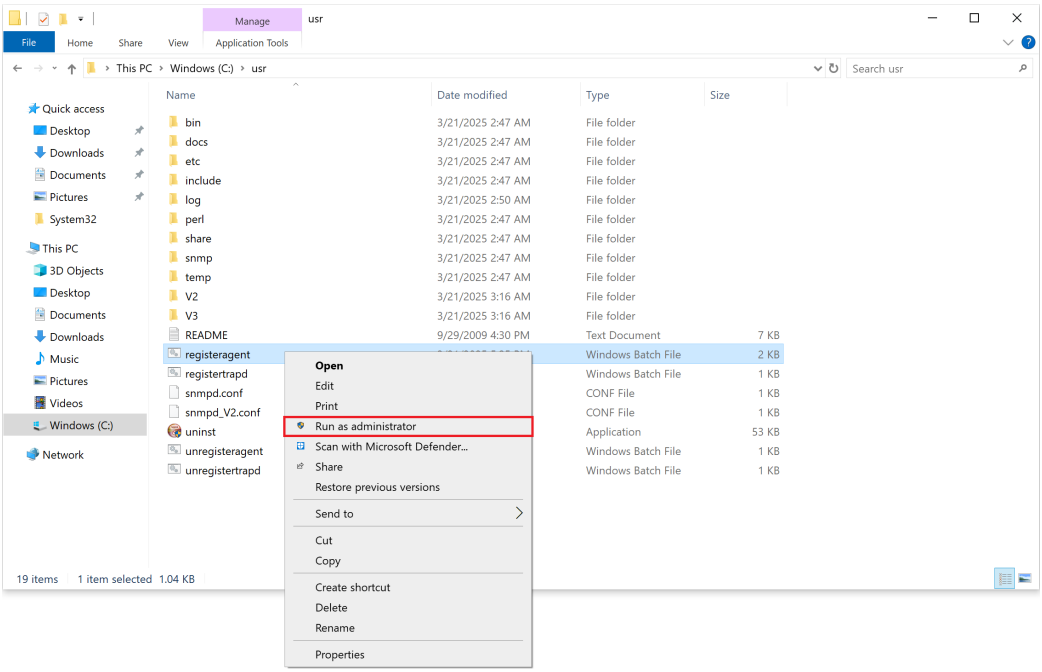
   b. Set Permission rwuser <username> <Security Level> or rouser <username>

   c. Set trapsess -v 3 -l <Security Level> -u <username> -e <EngineID> -a <Auth Type> -A <Password> <destination IP>

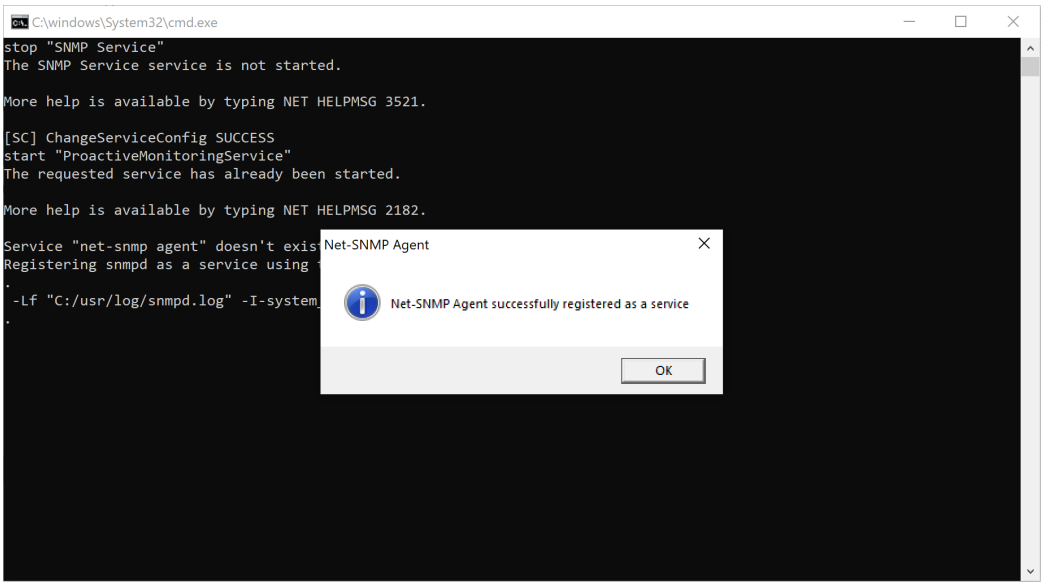
   For more information, please refer to the link:
   http://www.net-snmp.org/docs/man/snmpd.examples.html

```
trap2sink 127.0.0.1:162 public
oldEngineID 0x80001f88800b190000d299045c00000000
createUser testuser MD5 12345678 DES 23456789
rwuser testuser authpriv .1
authtrapenable 1
trapsess -v 3 -l authPriv -u testuser -e 0x80001f88800b190000d299045c00000000 -a MD5 -A 12345678 -x DES -X 23456789 127.0.0.1:162
iquerySecName testuser
linkUpDownNotifications yes
defaultMonitors yes
```

4. Right click on C:\usr\registeragent.bat and select "Run as administrator".



5. Press OK to complete the setting.

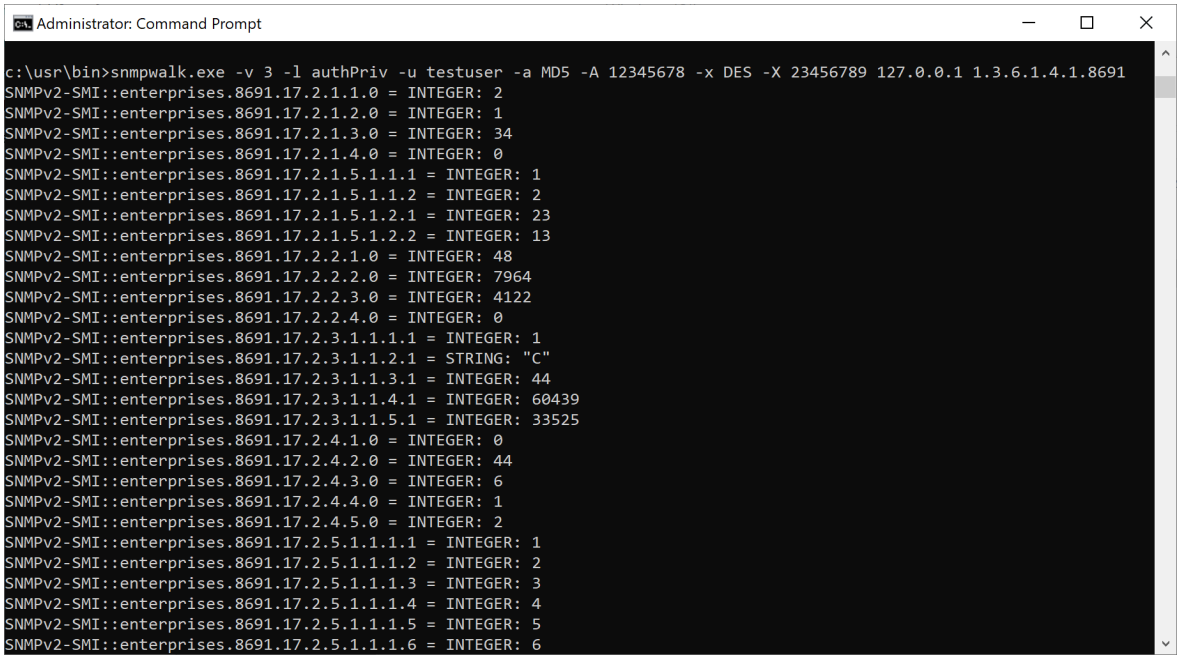


Querying Data With Net-SNMP

SNMP V3 Query

To query a SNMP V3 value, open cmd and run the following command:

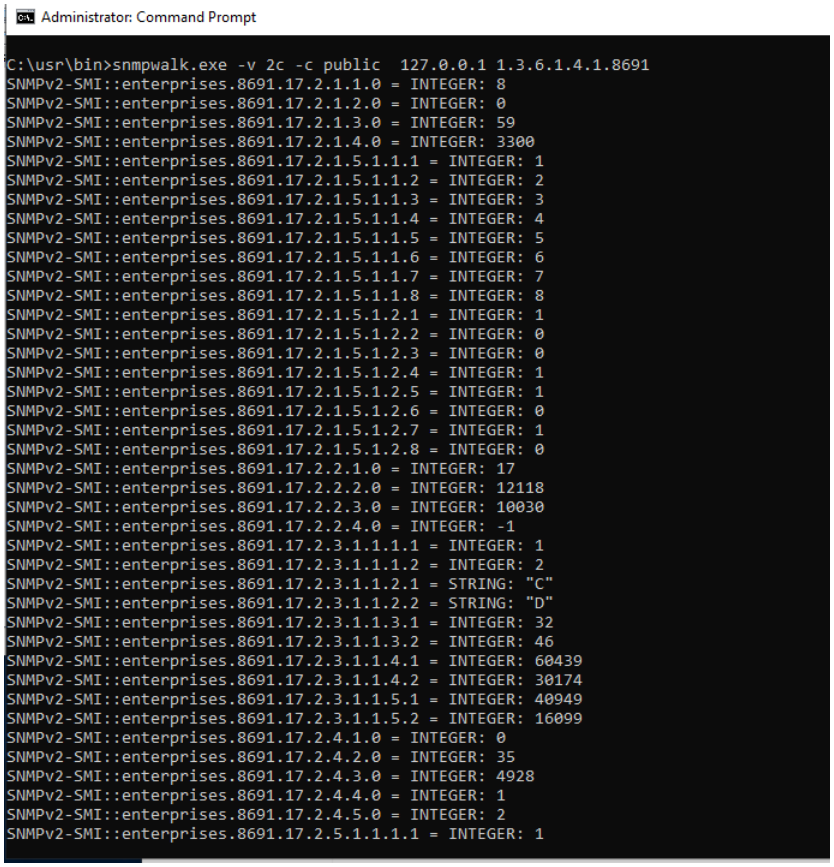
```
snmpwalk.exe -v 3 -l authPriv -u <username> -a MDS -A <password> -x DES -X
<encryption Key> <target Device IP> <Proactive Monitoring OID>
```



SNMP V2 Query

To query a SNMP V2 value, open cmd, and run the following command:

```
snmpwalk.exe -v 2c -c public <target Device IP> <Proactive Monitoring OID>
```



SNMP Trap

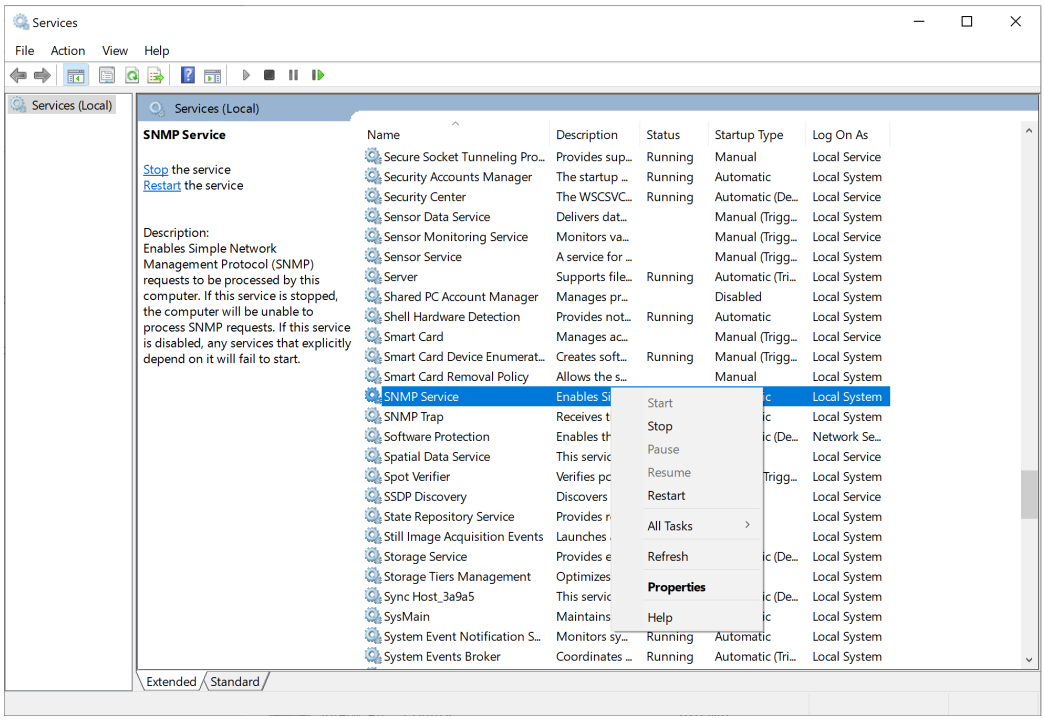
Proactive Monitoring provides SNMP Trap function. To enable this function, you need to set the relevant fields of WMI Configuration.

SNMP Trap Support List

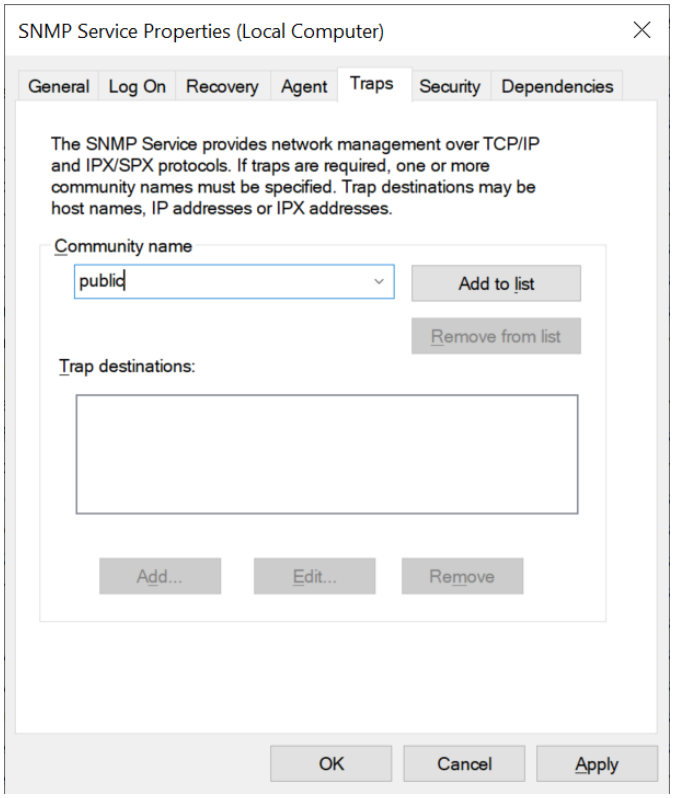
Monitor Type	Monitor Item	Configuration
CPU	Utilization	EnableLowerAlarm LowerThreshold EnableUpperAlarm UpperThreshold EnableSNMPTrap
Disk	Health Status	EnableAlarm EnableSNMPTrap
Memory	Utilization	EnableLowerAlarm LowerThreshold EnableUpperAlarm UpperThreshold EnableSNMPTrap
Network	Connection Status	EnableAlarm EnableSNMPTrap
Partition	Utilization	EnableLowerAlarm LowerThreshold EnableUpperAlarm UpperThreshold EnableSNMPTrap
Power Indicator	Power Module Status	EnableAlarm EnableSNMPTrap
Temperature	Temperature	EnableLowerAlarm LowerThreshold EnableUpperAlarm UpperThreshold EnableSNMPTrap

Setting Up SNMP Trap with Windows SNMP Service

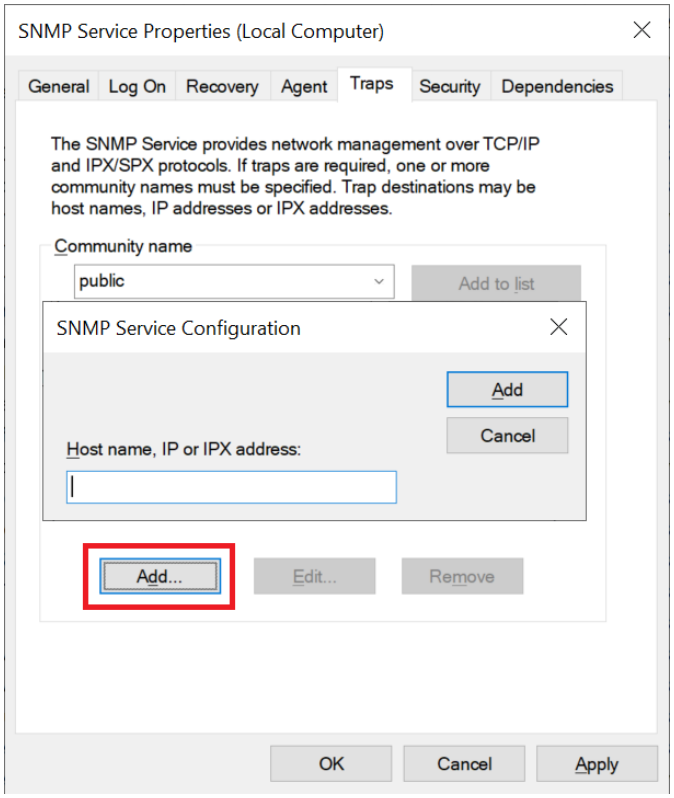
1. In the Windows **Services** page, right-click on the **SNMP Service** and select **Properties**.



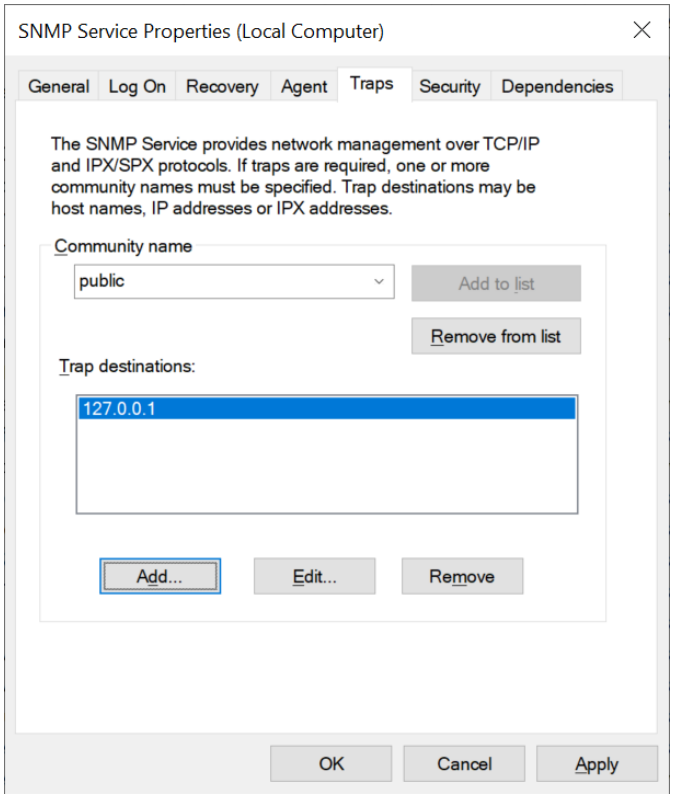
2. Enter the **Community name** and click the "**Add to list**" button.



3. Click "**Add**" button and enter the trap destinations.

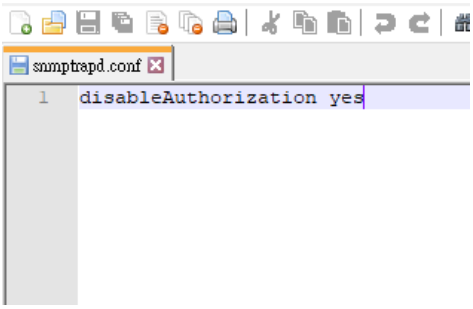


4. Press the **OK** to complete the setup.



Setting Up SNMP Trap with Net-SNMP Agent

1. Copy the snmptrapd.conf file to C:\usr\snmp folder.



2. Open cmd and switch to C:\usr\bin and execute the following command:

```
snmpdtrapd.exe -C c "<snmptrapd.conf path>" -f -Le -d
```

3. Receive SNMP Trap when alarm occurs