SIEMENS

SIMATIC

SIMATIC Automation Tool V4.0 SP2 User Guide

Application Manual

V4.0 SP2, 03/2021

A5E45044277-AF

Legal information

Warning notice system

This manual contains notices you have to observe in order to ensure your personal safety, as well as to prevent damage to property. The notices referring to your personal safety are highlighted in the manual by a safety alert symbol, notices referring only to property damage have no safety alert symbol. These notices shown below are graded according to the degree of danger.

⚠ DANGER

indicates that death or severe personal injury **will** result if proper precautions are not taken.

⚠ WARNING

indicates that death or severe personal injury **may** result if proper precautions are not taken.

⚠ CAUTION

indicates that minor personal injury can result if proper precautions are not taken.

NOTICE

indicates that property damage can result if proper precautions are not taken.

If more than one degree of danger is present, the warning notice representing the highest degree of danger will be used. A notice warning of injury to persons with a safety alert symbol may also include a warning relating to property damage.

Qualified Personnel

The product/system described in this documentation may be operated only by **personnel qualified** for the specific task in accordance with the relevant documentation, in particular its warning notices and safety instructions. Qualified personnel are those who, based on their training and experience, are capable of identifying risks and avoiding potential hazards when working with these products/systems.

Proper use of Siemens products

Note the following:

⚠ WARNING

Siemens products may only be used for the applications described in the catalog and in the relevant technical documentation. If products and components from other manufacturers are used, these must be recommended or approved by Siemens. Proper transport, storage, installation, assembly, commissioning, operation and maintenance are required to ensure that the products operate safely and without any problems. The permissible ambient conditions must be complied with. The information in the relevant documentation must be observed.

Trademarks

All names identified by ® are registered trademarks of Siemens AG. The remaining trademarks in this publication may be trademarks whose use by third parties for their own purposes could violate the rights of the owner.

Disclaimer of Liability

We have reviewed the contents of this publication to ensure consistency with the hardware and software described. Since variance cannot be precluded entirely, we cannot guarantee full consistency. However, the information in this publication is reviewed regularly and any necessary corrections are included in subsequent editions.

Siemens AG
Digital Industries
Postfach 48 48
90026 NÜRNBERG
GERMANY

A5E45044277-AF
Ⓟ 02/2021 Subject to change

Table of contents

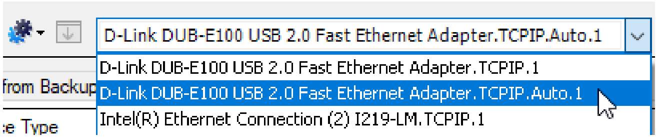
Getting started with the SIMATIC Automation Tool 1

Activate the 21-day trial

If you haven't purchased a license for the SIMATIC Automation Tool, activate the free 21-day trial license. You can explore all the features of SIMATIC Automation Tool with the trial license for 21 days. Learn more (Page 101).

Select your network interface

Start by selecting the network interface that is connected to your PROFINET network. Select the adapter with the ".Auto." in the name. This selection enables the SIMATIC Automation Tool to find all devices on the network. Learn more (Page 312).



D-Link DUB-E100 USB 2.0 Fast Ethernet Adapter.TCPIP.Auto.1

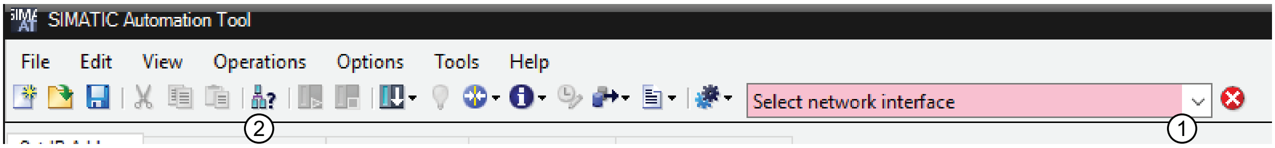
D-Link DUB-E100 USB 2.0 Fast Ethernet Adapter.TCPIP.1

D-Link DUB-E100 USB 2.0 Fast Ethernet Adapter.TCPIP.Auto.1

Intel(R) Ethernet Connection (2) I219-LM.TCPIP.1

Scan your network

Click the toolbar button to scan your PROFINET network for all devices. This scan sends a DCP broadcast command to the network. The SIMATIC Automation Tool displays all responding devices in the Device table (Page 21). The network scan does not find devices behind routers. You must insert these devices into the Device table manually. If you see errors on the network scan, refer to Troubleshooting Event Log messages (Page 125).



① Select network interface

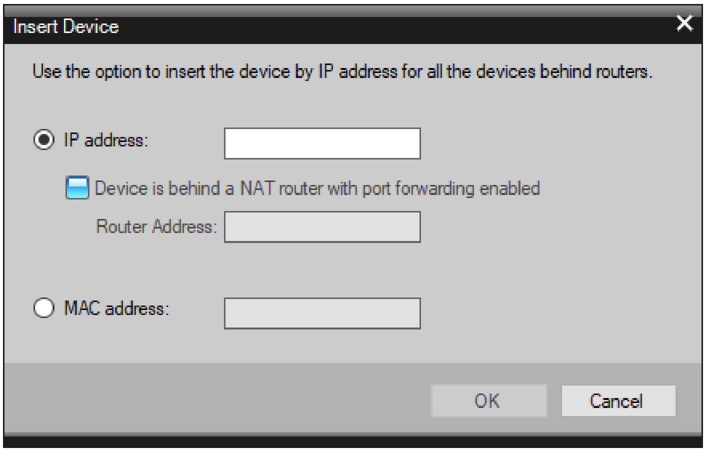
② Scan the network

Inserting devices behind routers

Select the "Edit > Insert > Device" menu command to insert devices behind routers into the Device table. The network scan does not find devices behind routers because they do not respond to the DCP broadcast command. To insert a device behind a router you must insert them by entering the device's IP address. Learn more (Page 25).

The SIMATIC Automation Tool displays devices behind routers in blue text in the Device table

Advanced topics: Using NAT routers and limitations (Page 114), Troubleshooting (Page 125)



Use the option to insert the device by IP address for all the devices behind routers.

● IP address:

□ Device is behind a NAT router with port forwarding enabled

Router Address:

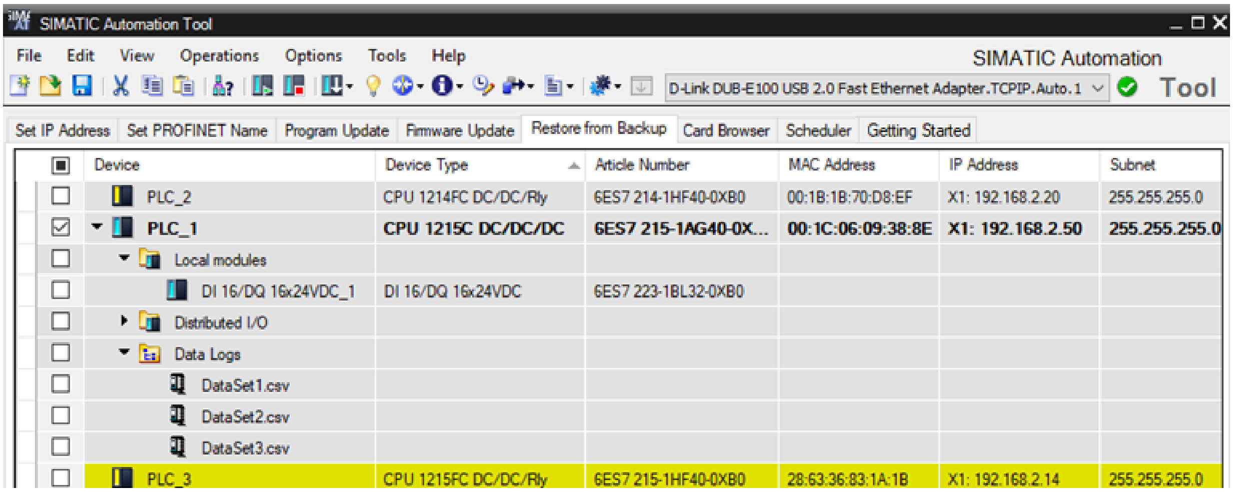
○ MAC address:

Working with the Device table

The Device table shows the connected devices. For each device, the SIMATIC Automation Tool displays columns of data about each device. To perform an operation, select the device by clicking the checkbox on the left. Then use the toolbar or main menu to select the operation you wish to perform. You can select multiple devices and SIMATIC Automation Tool performs the operation to all the devices that you selected.

If a CPU is protected, enter a CPU password for the device. Most operations within the SIMATIC Automation Tool require read and write privileges (Page 308) for the CPU.

Advanced topics: Sorting, Filtering, Configuring columns (Page 117), Supported devices (Page 117)



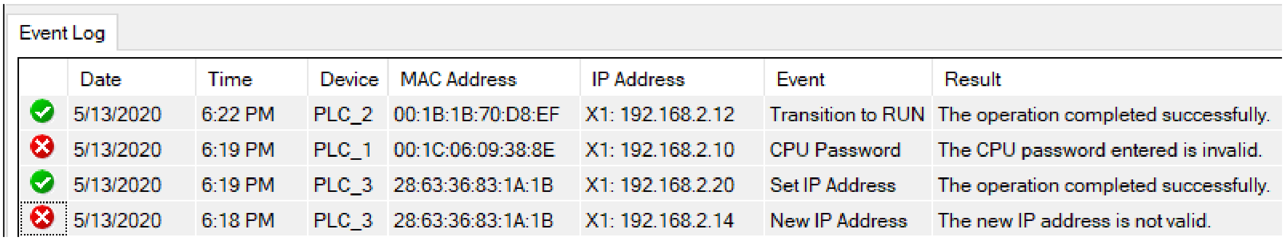
D-Link DUB-E100 USB 2.0 Fast Ethernet Adapter.TCPIP.Auto.1

▣	Device	Device Type	Article Number	MAC Address	IP Address	Subnet
□	PLC_2	CPU 1214FC DC/DC/Rly	6ES7 214-1HF40-0XB0	00:1B:1B:70:D8:EF	X1: 192.168.2.20	255.255.255.0
☑	**PLC_1**	**CPU 1215C DC/DC/DC**	**6ES7 215-1AG40-0X...**	**00:1C:06:09:38:8E**	**X1: 192.168.2.50**	**255.255.255.0**
□	Local modules					
□	DI 16/DQ 16x24VDC_1	DI 16/DQ 16x24VDC	6ES7 223-1BL32-0XB0			
□	Distributed I/O					
□	Data Logs					
□	DataSet1.csv					
□	DataSet2.csv					
□	DataSet3.csv					
□	PLC_3	CPU 1215FC DC/DC/Rly	6ES7 215-1HF40-0XB0	28:63:36:83:1A:1B	X1: 192.168.2.14	255.255.255.0

Understanding the Event Log

Located at the bottom of the SIMATIC Automation Tool is the Event Log. When you select devices and perform operations, the Event Log shows the results of the selected operation to each device. If a CPU is password protected, you must enter a CPU password with at least read and write privileges (Page 308) to perform most of the operations within SIMATIC Automation Tool.

Advanced topics: Types of Event Log messages, configuring columns, saving and clearing events (Page 123), Settings (Page 87)

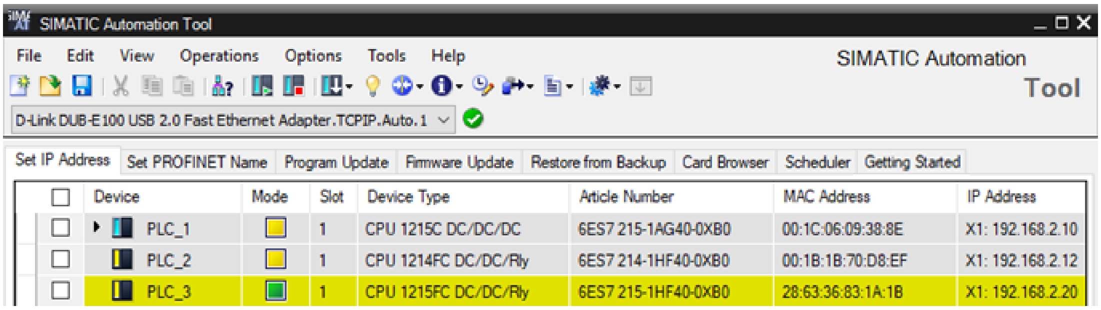


	Date	Time	Device	MAC Address	IP Address	Event	Result
✓	5/13/2020	6:22 PM	PLC_2	00:1B:1B:70:D8:EF	X1: 192.168.2.12	Transition to RUN	The operation completed successfully.
✕	5/13/2020	6:19 PM	PLC_1	00:1C:06:09:38:8E	X1: 192.168.2.10	CPU Password	The CPU password entered is invalid.
✓	5/13/2020	6:19 PM	PLC_3	28:63:36:83:1A:1B	X1: 192.168.2.20	Set IP Address	The operation completed successfully.
✕	5/13/2020	6:18 PM	PLC_3	28:63:36:83:1A:1B	X1: 192.168.2.14	New IP Address	The new IP address is not valid.

What can the SIMATIC Automation Tool do? 2

2.1 Overview of device operations

When you scan the network (Page 21), you see the devices on the network.



□	Device	Mode	Slot	Device Type	Article Number	MAC Address	IP Address
□	PLC_1	🟨	1	CPU 1215C DC/DC/DC	6ES7 215-1AG40-0XB0	00:1C:06:09:38:8E	X1: 192.168.2.10
□	PLC_2	🟨	1	CPU 1214FC DC/DC/Rly	6ES7 214-1HF40-0XB0	00:1B:1B:70:D8:EF	X1: 192.168.2.12
□	PLC_3	🟩	1	CPU 1215FC DC/DC/Rly	6ES7 215-1HF40-0XB0	28:63:36:83:1A:1B	X1: 192.168.2.20

Device operations

Using the SIMATIC Automation Tool, you can perform many useful operations:

- Identify devices (Page 27)
- Show device diagnostics (Page 28)
- Read Service Data from CPUs (Page 29)
- Set the CPU operating mode (Page 30)
- Set the CPU time (Page 31)
- Set the IP address (Page 32)
- Set the PROFINET name (Page 33)
- Reset communication parameters (Page 35)
- Reset devices to factory defaults (Page 36)
- Reset device memory (Page 38)
- Format memory card in a CPU (Page 39)
- Update device firmware (Page 40)
- Update the device program (Page 48)
- Read or delete Data Logs from a CPU (Page 56)
- Back up and restore a CPU or HMI (Page 57)
- Schedule device operations (Page 106)
- Browse the SIMATIC Memory Card of a CPU (Page 103)

Some device operations require an Advanced license (Page 101).

Device catalog access

The Help menu of the SIMATIC Automation Tool gives you access to the Device Catalog (Page 95). This Microsoft Excel file shows the supported devices and versions. For each device, the Device Catalog shows the supported operations.

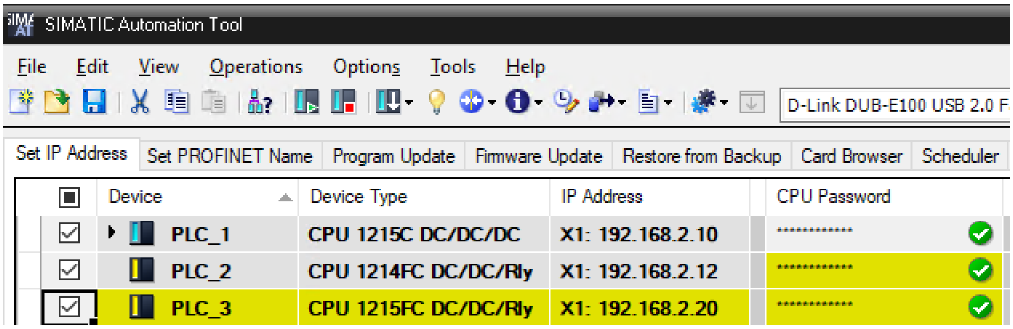
2.2 Sample first operation with the SIMATIC Automation Tool

Example: Setting the device time on three CPUs

As an example, consider a case where you want to set the device time for the CPUs on your network. The network scan finds three devices, which already have IP addresses. For this operation, you need to click one of the first five tabs of the Device table.

To set the device time to the time of your programming device follow these steps:

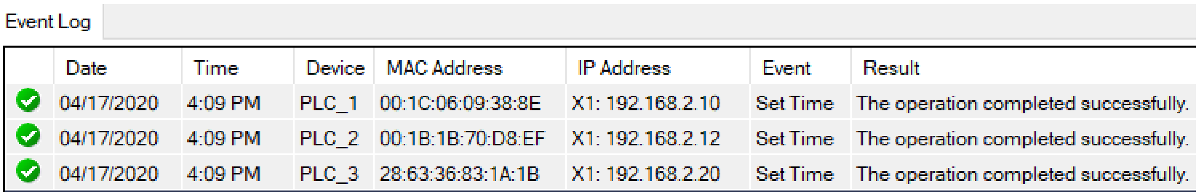
1. If you see a lock symbol by a device, enter a valid password (Page 308) in the "CPU Password" column. The password must provide the write privilege.
2. Select the check box for the three PLCs.



▣	Device	Device Type	IP Address	CPU Password	
☑	PLC_1	CPU 1215C DC/DC/DC	X1: 192.168.2.10	*************	✓
☑	PLC_2	CPU 1214FC DC/DC/Rly	X1: 192.168.2.12	*************	✓
☑	PLC_3	CPU 1215FC DC/DC/Rly	X1: 192.168.2.20	*************	✓

3. Click the Set Time button or select the Operations > Set Time menu command

The Event Log shows you the result of the operation.


Event Log

	Date	Time	Device	MAC Address	IP Address	Event	Result
✓	04/17/2020	4:09 PM	PLC_1	00:1C:06:09:38:8E	X1: 192.168.2.10	Set Time	The operation completed successfully.
✓	04/17/2020	4:09 PM	PLC_2	00:1B:1B:70:D8:EF	X1: 192.168.2.12	Set Time	The operation completed successfully.
✓	04/17/2020	4:09 PM	PLC_3	28:63:36:83:1A:1B	X1: 192.168.2.20	Set Time	The operation completed successfully.

For help with understanding the Event Log messages, refer to Troubleshooting Event Log messages (Page 125)

STEP 7 configuration requirements 3

3.1 STEP 7 requirements for setting an IP address

You can use the SIMATIC Automation Tool to set the IP address (Page 32) of a device. For the IP address to persist after a power cycle, you must enable "IP address is set directly at the device" in the device configuration of the STEP 7 project. You must compile the STEP 7 project and download the project to the target device.

If the STEP 7 project in the device does not enable this setting, then you can set a temporary IP address with the SIMATIC Automation Tool. The device must be in STOP mode and the address that you set is temporary until the next power cycle.

Note

Temporary (Emergency) IP addresses

For more information on temporary (emergency) IP addresses, refer to this FAQ (https://support.industry.siemens.com/cs/ww/en/view/97649773).

Configuring the STEP 7 project to enable setting an IP address

To enable the SIMATIC Automation Tool to set an IP address, configure the device configuration as follows:

1. Open the device configuration for the CPU.
2. Click the PROFINET interface that you use for the SIMATIC Automation Tool connection.
3. On the **Properties** tab, click the **General** tab to view the options for **Ethernet addresses**.

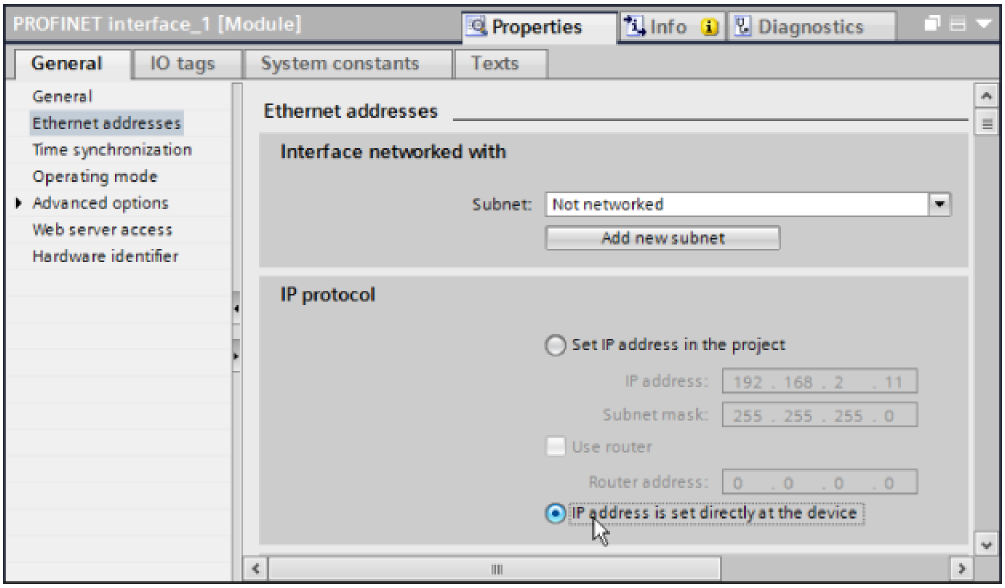
4. Click the **"IP address is set directly at the device"** option. This option has had different names in different releases of STEP 7. Possible names for this field are:

   – IP address is set directly at the device

   – Set IP address on the device

   – Set IP address using a different method

   This selection allows the SIMATIC Automation Tool to assign an IP address. For devices with multiple PROFINET interfaces such as the S7-1500 CPU, you can configure one interface or all interfaces to allow IP address changes



Subnet: Not networked

○ Set IP address in the project

IP address: 192 . 168 . 2 . 11

Subnet mask: 255 . 255 . 255 . 0

□ Use router

Router address: 0 . 0 . 0 . 0

● IP address is set directly at the device

5. Save your project and download the project to the device.

If the STEP 7 project in the device does not enable this setting, then you can set an IP address with the SIMATIC Automation Tool. The device must be in STOP mode and the address that you set is temporary until the next power cycle.

Direct connection requirement for setting an IP address

Devices that the SIMATIC Automation Tool identified on a network scan have a direct connection to the SIMATIC Automation Tool. Devices behind a router that you inserted into the SIMATIC Automation Tool do not have a direct connection. Set IP Address requires a direct connection

3.2 STEP 7 requirements for setting a PROFINET name

You can use the SIMATIC Automation Tool to set the PROFINET name of a device. To set the PROFINET name, you must enable "PROFINET device name is set directly at the device" in the device configuration of the STEP 7 project. You must compile the STEP 7 project and download the project to the target device before you can use the SIMATIC Automation Tool to set a device's PROFINET name.

Configuring the STEP 7 project to enable setting the PROFINET name

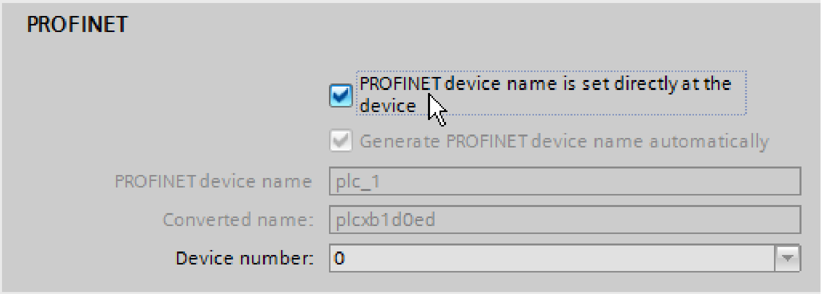
To enable the SIMATIC Automation Tool to set the PROFINET name, configure the device configuration as follows:

1. Open the device configuration for the CPU.
2. Click the PROFINET interface that you use for the SIMATIC Automation Tool connection.
3. On the **Properties** tab, click the **General** tab to view the options for **Ethernet addresses**.
4. On the **Ethernet addresses** options, click the **"PROFINET device name is set directly at the device"** option. This option has had different names in different releases of STEP 7. Possible names for this file are:

   – PROFINET device name is set directly at the device

   – Set PROFINET device name on the device

   This selection allows the SIMATIC Automation Tool to assign a PROFINET station name. For devices with multiple PROFINET interfaces such as the S7-1500 CPU, you can configure all interfaces to enable PROFINET name changes or you can configure only one interface to allow PROFINET name changes.



PROFINET

☑ PROFINET device name is set directly at the device

☑ Generate PROFINET device name automatically

PROFINET device name: plc_1

Converted name: plcxb1d0ed

Device number: 0

5. Save your project and download the project to the device.

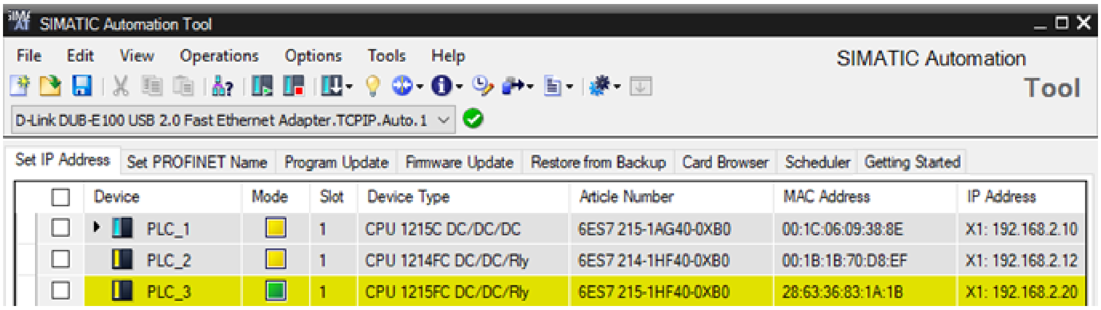
Direct connection requirement for setting a PROFINET name

Devices that the SIMATIC Automation Tool identified on a network scan have a direct connection to the SIMATIC Automation Tool. Devices behind a router that you inserted into the SIMATIC Automation Tool do not have a direct connection. Set PROFINET name (Page 33) requires a direct connection.

Device operations 4

4.1 Scanning a network

To scan the network, click the Scan button on the toolbar . Alternatively, you can select the "Operations > Scan Network" menu command. The scan detects devices connected directly to the network through an Ethernet interface either on the device or on a connected CM or CP module (Page 113). The Device table shows the scanned devices on the network:



□	Device	Mode	Slot	Device Type	Article Number	MAC Address	IP Address
□	PLC_1	🟨	1	CPU 1215C DC/DC/DC	6ES7 215-1AG40-0XB0	00:1C:06:09:38:8E	X1: 192.168.2.10
□	PLC_2	🟨	1	CPU 1214FC DC/DC/Rly	6ES7 214-1HF40-0XB0	00:1B:1B:70:D8:EF	X1: 192.168.2.12
□	PLC_3	🟩	1	CPU 1215FC DC/DC/Rly	6ES7 215-1HF40-0XB0	28:63:36:83:1A:1B	X1: 192.168.2.20

If the Device table includes devices that are no longer present on the network, the SIMATIC Automation Tool displays the device row data in italics. You can also manually insert a device (Page 25) into the Device table.

Note

The SIMATIC Automation Tool is an offline tool

The SIMATIC Automation Tool does not update device data continuously. The SIMATIC Automation Tool displays device information at the point of time that you scanned or refreshed (Page 122) your communications network or at the point of time when you inserted devices. The TIA Portal or Web server, for example, could change device data since your last scan.

When you scan the network, the SIMATIC Automation Tool retains the device data from the last scan until it updates the device data from the new scan. If a scan fails for a device, the device displays the data from the previous scan and the Event Log displays an error.

Device table conventions

Row icons help you identify the Device table rows:

Device is unknown or not supported. The SIMATIC Automation Tool displays the device data in gray text.

Fail-Safe device is unknown or not fully supported. The SIMATIC Automation Tool displays the device data in gray text.

PROFINET device

PROFINET Fail-Safe device

PROFINET HMI device

PROFINET Fail-Safe HMI device

Distributed I/O PROFINET device

Distributed I/O PROFINET Fail-Safe device

Folder containing PROFINET devices

Folder containing PROFIBUS devices

Folder containing PROFINET AS-i devices

Folder containing Data Log or Recipe data

Folder containing local I/O or distributed I/O modules

Data Log data

Recipe data

Standard device identity problem

Fail-Safe device identity problem

Note

TIA Portal online connections to devices

If a device has an online connection in the TIA Portal, the SIMATIC Automation Tool might not be able to read information from the device. The SIMATIC Automation Tool reports an error in the Event Log (Page 125), which states that the SIMATIC Automation Tool could not determine the identity of the device.

To be able to read the device, go offline in the TIA Portal.

Click the check box next to a device to select it. The SIMATIC Automation Tool displays device text in black for devices you have not selected and in **bold black** when you have selected them.

You can enter text in cells with a light gray background. You cannot enter text in cells with a dark gray background. A dark gray cell indicates that the SIMATIC Automation Tool does not support the operation for that device type/firmware version.

You can find handy tips for working with the Device table in Device table power features (Page 117).

Device identities

A device identity problem can occur, for example, if the TIA Portal makes modifications to a device since the last network scan. A device identity problem can also occur when a device fails.

The SIMATIC Automation Tool does not perform any device operations on a device with an identity problem.

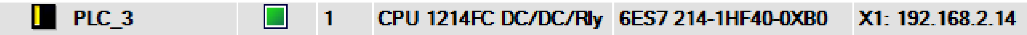
You must scan the network to resolve the device identity problem.

If you replace a failed device with a new device and assign the same IP address, you must scan the network to resolve the device identity problem.

Fail-Safe devices

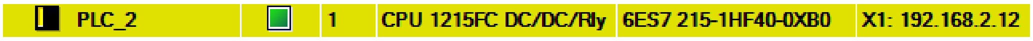
When you change the identity of a device or safety program status from the SIMATIC Automation Tool, the tool makes the changes without a new network scan. For example, if you download a new firmware version from the SIMATIC Automation Tool, the SIMATIC Automation Tool updates the device variables to the new values. The SIMATIC Automation Tool is an approved tool for operating safety devices and can handle safety state changes.

If you have a fail-safe CPU in your network, but you have not downloaded a safety program (Page 314) to it, the device row cells appear in the color gray:



PLC_3	🟩	1	CPU 1214FC DC/DC/Rly	6ES7 214-1HF40-0XB0	X1: 192.168.2.14

If you have downloaded a safety program to a fail-safe CPU, the information fields for the device appear in yellow:



PLC_2	🟩	1	CPU 1215FC DC/DC/Rly	6ES7 215-1HF40-0XB0	X1: 192.168.2.12

Yellow cell coloring for the following user-entry fields for fail-safe CPUs have the following meanings:

- CPU Password, Password in Program File: Password is the safety password (Page 308)
- Program Update Folder, Backup File: Selection is a safety program (Page 314)

Devices connected through CPUs, CMs, and CPs

For devices that do not have an IP address, refer to Setting an IP address (Page 32) for setting the IP address, Subnet, and Gateway. To communicate to a CPU though a CM (Communications Module) or CP (Communications Processor), refer to Working with devices connected to CMs or CPs (Page 113).

Devices connected through IP address routers and NAT routers

A network scan does not find devices connected through an IP address router or NAT router. You must insert these devices (Page 25). The Device table shows these devices in blue text.

Effect of network scan on Device table

If a device at a MAC address already exists in the table, a network scan updates the IP address, Subnet, and Gateway fields for that Device table row. All other data fields remain the same. If the network scan finds a new device, then the SIMATIC Automation Tool creates a new row for the device.

Indication of CPU or CM/CP connection

The IP address column indicates the interface connection for CPUs:

- X1, X2: Ethernet interface on the CPU
- CM: Ethernet interface on a Communications Module connected to the CPU
- CP: Ethernet interface on a Communications Processor connected to the CPU

Password identification

If the device is protected (Page 308) at any protection level, then the SIMATIC Automation Tool enables the password field. Until you enter a valid password for the device, the SIMATIC Automation Tool displays the lock icon 🔒 beside the device name.

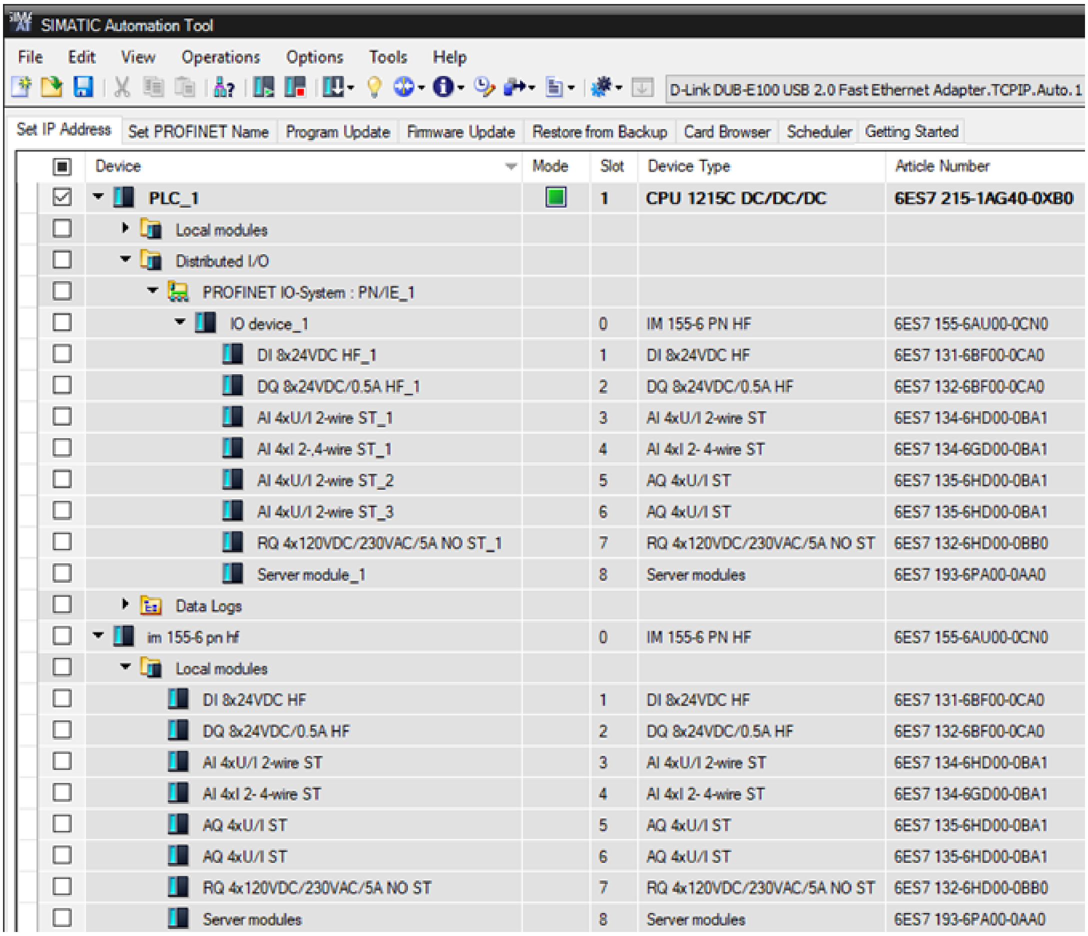
PROFINET I/O

PROFINET I/O devices can appear twice in the Device table:

- As a root device if the SIMATIC Automation Tool finds the device on a network scan
- As distributed I/O if the CPU is configured to use this device as remote I/O

You can perform a firmware update from either location in the Device tree.

Example: IM 155-6 PN HF



■	Device	Mode	Slot	Device Type	Article Number
☑	**PLC_1**	🟩	**1**	**CPU 1215C DC/DC/DC**	**6ES7 215-1AG40-0XB0**
☐	Local modules				
☐	Distributed I/O				
☐	PROFINET IO-System : PN/IE_1				
☐	IO device_1		0	IM 155-6 PN HF	6ES7 155-6AU00-0CN0
☐	DI 8x24VDC HF_1		1	DI 8x24VDC HF	6ES7 131-6BF00-0CA0
☐	DQ 8x24VDC/0.5A HF_1		2	DQ 8x24VDC/0.5A HF	6ES7 132-6BF00-0CA0
☐	AI 4xU/I 2-wire ST_1		3	AI 4xU/I 2-wire ST	6ES7 134-6HD00-0BA1
☐	AI 4xI 2-,4-wire ST_1		4	AI 4xI 2- 4-wire ST	6ES7 134-6GD00-0BA1
☐	AI 4xU/I 2-wire ST_2		5	AQ 4xU/I ST	6ES7 135-6HD00-0BA1
☐	AI 4xU/I 2-wire ST_3		6	AQ 4xU/I ST	6ES7 135-6HD00-0BA1
☐	RQ 4x120VDC/230VAC/5A NO ST_1		7	RQ 4x120VDC/230VAC/5A NO ST	6ES7 132-6HD00-0BB0
☐	Server module_1		8	Server modules	6ES7 193-6PA00-0AA0
☐	Data Logs				
☐	im 155-6 pn hf		0	IM 155-6 PN HF	6ES7 155-6AU00-0CN0
☐	Local modules				
☐	DI 8x24VDC HF		1	DI 8x24VDC HF	6ES7 131-6BF00-0CA0
☐	DQ 8x24VDC/0.5A HF		2	DQ 8x24VDC/0.5A HF	6ES7 132-6BF00-0CA0
☐	AI 4xU/I 2-wire ST		3	AI 4xU/I 2-wire ST	6ES7 134-6HD00-0BA1
☐	AI 4xI 2- 4-wire ST		4	AI 4xI 2- 4-wire ST	6ES7 134-6GD00-0BA1
☐	AQ 4xU/I ST		5	AQ 4xU/I ST	6ES7 135-6HD00-0BA1
☐	AQ 4xU/I ST		6	AQ 4xU/I ST	6ES7 135-6HD00-0BA1
☐	RQ 4x120VDC/230VAC/5A NO ST		7	RQ 4x120VDC/230VAC/5A NO ST	6ES7 132-6HD00-0BB0
☐	Server modules		8	Server modules	6ES7 193-6PA00-0AA0

Software Controller CPUs

For the SIMATIC Automation Tool to be able to communicate with Software Controller CPUs, your programming device must connect to the Software Controller using the Ethernet interface. Typically, this interface is X1 or X2. Depending on the device, the interface label can be PROFINET (LAN) or PN/IE (LAN). The SIMATIC Automation Tool communicates to a Software Controller using X2 and to an Open Controller using X1. The device catalog (Page 95) lists the Software Controller CPUs that the SIMATIC Automation Tool supports and the supported device operations.

Network scan errors

If a communication error occurs with a device during a network scan, the SIMATIC Automation Tool adds an entry to the Event Log (Page 123). The SIMATIC Automation Tool displays scan errors in the Event Log (Page 125) in the order they occur.

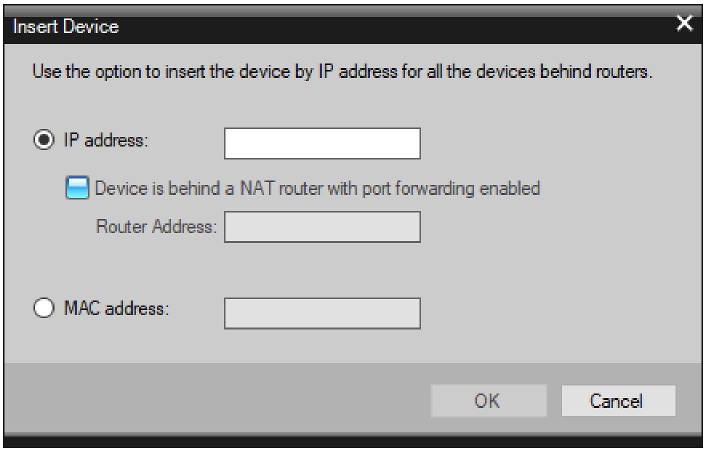
4.2 Inserting one or more devices

Inserting a single device including a device behind a router

You can insert a device into the Device table. You can only add a device that is unique from any other device in the Device table. The SIMATIC Automation Tool supports devices behind routers, including NAT routers with port forwarding enabled.

To insert a device, follow these steps:

1. Select the "Insert > Device" menu command from the Edit menu.



Insert Device

Use the option to insert the device by IP address for all the devices behind routers.

● IP address:

□ Device is behind a NAT router with port forwarding enabled

Router Address:

○ MAC address:

2. From the "Insert Device" dialog, enter either an IP address or MAC address for the device. The address you enter must not correspond to the address for an existing device on your network.
   – For an IP address that is behind a NAT router, select the corresponding check box and enter the NAT router address.
   – For a MAC address, use a ":" as the separator, not a "-".

The SIMATIC Automation Tool validates the device data. If successful, the SIMATIC Automation Tool inserts the device into the Device table. If unsuccessful, the Event Log displays an error message.

Inserting multiple devices including devices behind a router

You can enter multiple devices into the Device table. As with a single device, each device must be unique.

To insert multiple devices, follow these steps

1. Select the "Insert > Multiple Devices" menu command from the Edit menu.
2. From the "Insert Device" dialog, enter devices one at a time by either IP address or MAC address. The address you enter must not correspond to the address for an existing device on the device's network. Note that you cannot mix IP address entry and MAC address entry on the Insert device dialog. You can only insert multiple devices by IP address or by MAC address.
   - For each IP address that is behind a NAT router, select the corresponding check box and enter the NAT router address.
   - For each MAC address, use a ":" as the separator, not a "-".

The SIMATIC Automation Tool attempts to insert each device into the Device table. For each entry that is successful, the SIMATIC Automation Tool inserts the device into the Device table. For each entry that is unsuccessful, the Event Log displays an error message.

Inserting devices from tables in Microsoft Excel

You can also insert multiple devices including their NAT router addresses from an Excel worksheet. The SIMATIC Automation Tool supports the following operations:

- Pasting a single column of device IP addresses into the "Insert Device" dialog
- Pasting two columns of data from Microsoft Excel into the "Insert Device" dialog, where the first column contains device IP addresses and the second column contains corresponding NAT router IP addresses. Empty cells in the NAT router IP address column indicate the device is not behind a NAT router.
- Dragging and dropping a .csv file into the "Insert Device" dialog, where the .csv file contains one or two columns of IP addresses for devices and NAT routers.

The same rules for device insertion apply when using data from Microsoft Excel.

Validating the device data for uniqueness and successful communication

The SIMATIC Automation Tool rejects an attempt to insert a device that does not have a unique address or a valid IP address. Uniqueness follows these rules:

- The MAC address is unique from any other MAC address
- The IP address, if not behind a router, is unique from any other IP address on the main network.
- The IP address, if behind a router, is unique from any other IP addresses on that router.

When you enter a unique IP address or unique MAC address, the SIMATIC Automation Tool attempts to communicate with the address you provided. If communication is successful, the

SIMATIC Automation Tool inserts the device into the Device table. If communication is not successful, the SIMATIC Automation Tool informs you that the device does not exist on the network.

If the device is behind a router, the SIMATIC Automation Tool displays the device name in blue.

4.3 Identifying devices

For devices at the root level in your Device table, the Identify operation helps you physically locate devices on the network.

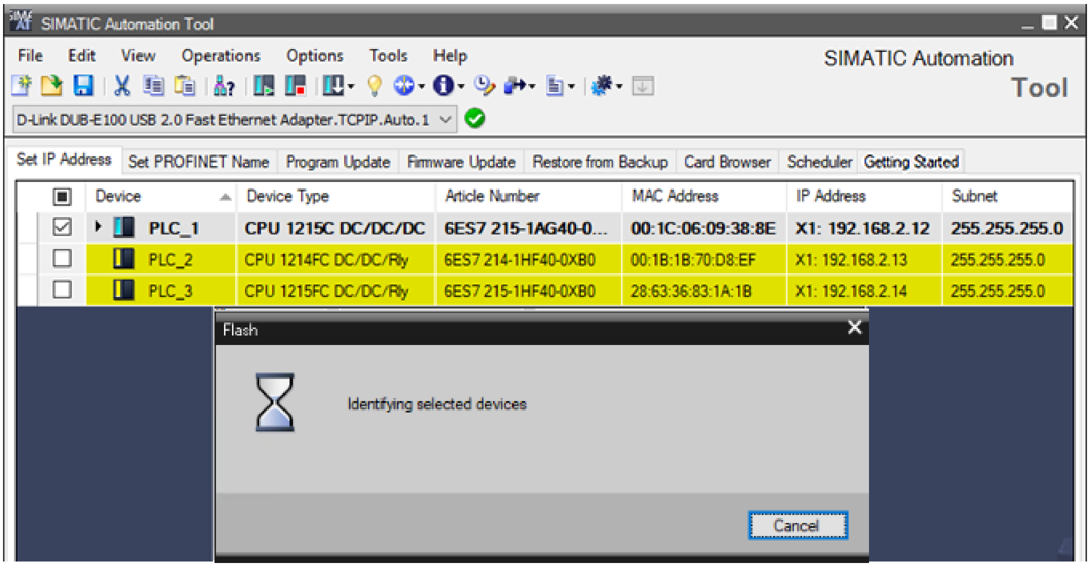
Locate a device by flashing LEDs or HMI screens

You can use the Identify operation in RUN mode and STOP mode. To identify devices, follow these steps:

1. Select one or more devices to include in the operation. You can use the Devices check box at the top of the Device table to select or deselect all devices. Alternatively, you can use the "View > Select" menu command to access the "Select Row(s)" and "Deselect Row(s)" commands.
2. Select the "Operations > Identify" menu command or click the toolbar button 💡 for identifying selected devices.

CPUs, SCALANCE, and other devices flash their LEDs to show their location. HMI devices perform a screen flash.

Flashing continues until you click the Cancel button:



D-Link DUB-E100 USB 2.0 Fast Ethernet Adapter.TCPIP.Auto.1

■	Device	Device Type	Article Number	MAC Address	IP Address	Subnet
☑	PLC_1	CPU 1215C DC/DC/DC	6ES7 215-1AG40-0...	00:1C:06:09:38:8E	X1: 192.168.2.12	255.255.255.0
□	PLC_2	CPU 1214FC DC/DC/Rly	6ES7 214-1HF40-0XB0	00:1B:1B:70:D8:EF	X1: 192.168.2.13	255.255.255.0
□	PLC_3	CPU 1215FC DC/DC/Rly	6ES7 215-1HF40-0XB0	28:63:36:83:1A:1B	X1: 192.168.2.14	255.255.255.0

Identifying selected devices

Identifying unsupported devices

The Identify operation is a MAC address-based operation that uses DCP (Discovery and Configuration Protocol). The SIMATIC Automation Tool can use the DCP-MAC address operations for all directly-connected network devices.

Unsupported devices might not flash LEDs or HMI screens based on the hardware configuration of the device.

By default, the SIMATIC Automation Tool displays unsupported devices. You can disable this option in the General settings (Page 81).

4.4 Showing device diagnostics

CPU diagnostics contain an entry for each diagnostic event. Each entry includes the date and time the event occurred, an event category, and an event description. The entries are in chronological order with the most recent event at the top. When the log is full, a new event replaces the oldest event in the log. When power is lost, the events are saved.

To show CPU diagnostics, follow these steps:

1. Select one or more CPUs in the Device table. If you want to deselect all devices first, you can deselect the Devices check box at the top of the Device table. Alternatively, you can also use the right-click shortcut menu or the "View > Select" menu to access the "Deselect Row(s)" command.
2. For each selected CPU, enter a password, if used, in the "CPU Password" column that provides read access (Page 308).
3. Select the "Operations > Diagnostics > Show Diagnostics" menu command. Alternatively, click the "Show device diagnostics" toolbar button ℹ and select "Show Diagnostics" from the button drop-down menu.

The SIMATIC Automation Tool then displays a dialog that includes the diagnostic buffers of the selected CPUs. You can select a CPU from the device list to see the diagnostic buffer for that CPU.

Note

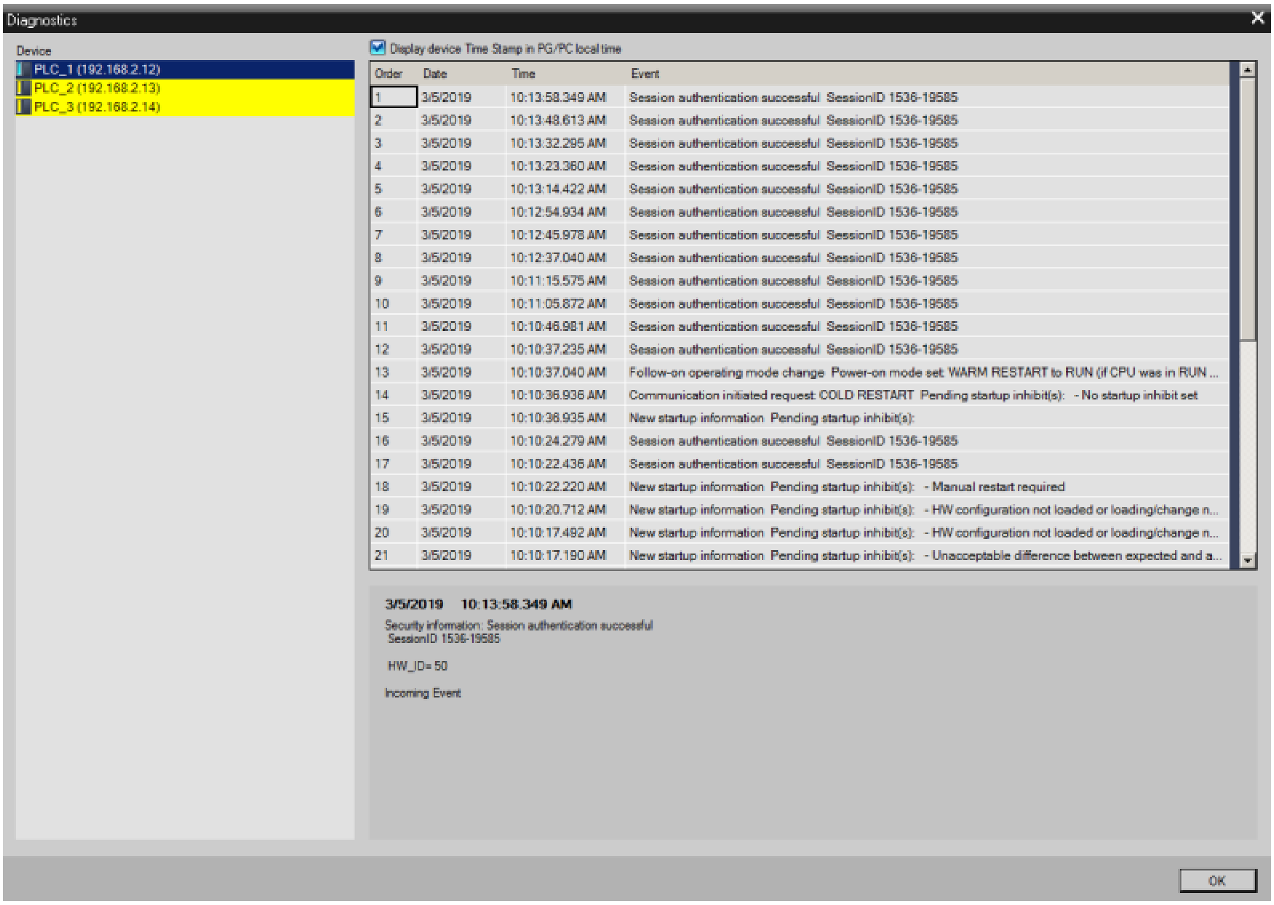
The diagnostic buffer displays the events in the order they occurred, regardless of the date and time of the event. Setting the CPU time to a time in the past has no effect on the event order.

Example diagnostic buffer

The diagnostic buffer contains the following types of entries:

- System diagnostic event (each CPU error and module error)
- CPU state changes (each power up, transition to STOP, and transition to RUN)

You can use the "Display device Time Stamp in PG/PC local time" check box to view time stamps in local time or UTC time (Coordinated Universal Time).



☑ Display device Time Stamp in PG/PC local time

Order	Date	Time	Event
1	3/5/2019	10:13:58.349 AM	Session authentication successful SessionID 1536-19585
2	3/5/2019	10:13:48.613 AM	Session authentication successful SessionID 1536-19585
3	3/5/2019	10:13:32.295 AM	Session authentication successful SessionID 1536-19585
4	3/5/2019	10:13:23.360 AM	Session authentication successful SessionID 1536-19585
5	3/5/2019	10:13:14.422 AM	Session authentication successful SessionID 1536-19585
6	3/5/2019	10:12:54.934 AM	Session authentication successful SessionID 1536-19585
7	3/5/2019	10:12:45.978 AM	Session authentication successful SessionID 1536-19585
8	3/5/2019	10:12:37.040 AM	Session authentication successful SessionID 1536-19585
9	3/5/2019	10:11:15.575 AM	Session authentication successful SessionID 1536-19585
10	3/5/2019	10:11:05.872 AM	Session authentication successful SessionID 1536-19585
11	3/5/2019	10:10:46.981 AM	Session authentication successful SessionID 1536-19585
12	3/5/2019	10:10:37.235 AM	Session authentication successful SessionID 1536-19585
13	3/5/2019	10:10:37.040 AM	Follow-on operating mode change Power-on mode set: WARM RESTART to RUN (if CPU was in RUN ...
14	3/5/2019	10:10:36.936 AM	Communication initiated request: COLD RESTART Pending startup inhibit(s): - No startup inhibit set
15	3/5/2019	10:10:36.935 AM	New startup information Pending startup inhibit(s):
16	3/5/2019	10:10:24.279 AM	Session authentication successful SessionID 1536-19585
17	3/5/2019	10:10:22.436 AM	Session authentication successful SessionID 1536-19585
18	3/5/2019	10:10:22.220 AM	New startup information Pending startup inhibit(s): - Manual restart required
19	3/5/2019	10:10:20.712 AM	New startup information Pending startup inhibit(s): - HW configuration not loaded or loading/change n...
20	3/5/2019	10:10:17.492 AM	New startup information Pending startup inhibit(s): - HW configuration not loaded or loading/change n...
21	3/5/2019	10:10:17.190 AM	New startup information Pending startup inhibit(s): - Unacceptable difference between expected and a...

3/5/2019 10:13:58.349 AM

Security information: Session authentication successful
SessionID 1536-19585

HW_ID= 50

Incoming Event

4.5 Reading a device's service data

If a CPU enters a defective state, the CPU saves fault information that you can upload to your programming device. You can send this service data to Siemens customer support for diagnosis.

You can read service data when the CPU is in STOP or RUN mode. The service data contains multiple files that are compressed into a single .zip file with a file name based on the PLC name and MAC address. A unique number in parentheses is appended to the file name to avoid duplicate file names.

You configure or accept the default service data path from the Service Data settings (Page 85).

To read service data from selected CPUs, follow these steps:

1. Select one or more devices to include in the operation. You can use the Devices check box at the top of the Device table to select or deselect all devices. Alternatively, you can use the right-click shortcut menu or the "View > Select" menu command to access the "Select Row(s)" and "Deselect Row(s)" commands.
2. For each selected CPU that is protected, enter a CPU password (Page 308) that provides read access in the "CPU Password" column of the currently open tab.

3. Select the "Operations > Diagnostics > Read Service Data" command to start the operation. Alternatively, click the "Show device diagnostics" toolbar button ℹ and select "Read Service Data" from the button drop-down menu.
4. Click the "Continue" button on the "Upload Service Data" dialog.

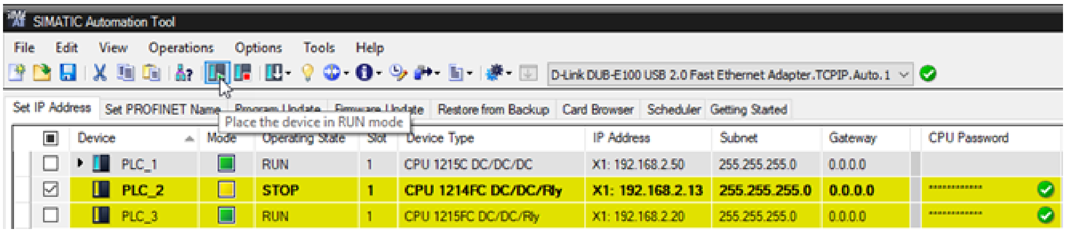
The SIMATIC Automation Tool reads service data from the selected CPUs and stores the files in the service data folder (Page 85). The Event Log shows the results of the operation.

> ⚠ **WARNING**
>
> **Service Data is clear text**
>
> A malicious user could read the service data files to obtain status and configuration details about the control system. The CPU stores the service data files in clear text, which is unencrypted. A CPU password can control access to this information.
>
> Operating a process or machine with compromised data could affect the operation of an online process or machine. Unexpected operation of a process or machine could result in death or injury to personnel and/or property damage.
>
> Use the TIA Portal device configuration to set up CPU protection with a strong password. Strong passwords are at least ten characters in length, mix letters, numbers, and special characters, are not words that can be found in a dictionary, and are not names or identifiers that can be derived from personal information. Keep the password secret and change it frequently.

4.6 Setting a CPU's operating mode

To change the operating mode for a device, follow these steps:

1. Select one or more devices to include in the operation. You can use the Devices check box at the top of the Device table to select or deselect all devices. Alternatively, you can use the right-click shortcut menu or the "View > Select" menu command to access the "Select Row(s)" and "Deselect Row(s)" commands.
2. For each selected CPU, enter a password, if used, in the "CPU Password" column of the currently open tab.
3. Set the operating mode to either RUN mode or STOP mode:
   - – Select RUN from the Operations menu or click the RUN toolbar button. A valid program must exist in the CPU before it can enter RUN mode.
   - – Select STOP from the Operations menu or click the STOP toolbar button.



■	Device	Mode	Operating State	Slot	Device Type	IP Address	Subnet	Gateway	CPU Password
□	PLC_1	🟩	RUN	1	CPU 1215C DC/DC/DC	X1: 192.168.2.50	255.255.255.0	0.0.0.0	
☑	PLC_2	🟨	STOP	1	CPU 1214FC DC/DC/Rly	X1: 192.168.2.13	255.255.255.0	0.0.0.0	************ ✓
□	PLC_3	🟩	RUN	1	CPU 1215FC DC/DC/Rly	X1: 192.168.2.20	255.255.255.0	0.0.0.0	************ ✓

You must confirm a prompt to change the operating mode. After you confirm the prompt, sets the selected CPUs to RUN or STOP mode. The SIMATIC Automation Tool does not change the operating mode without confirmation.

The Mode and Operating state columns in the Device table indicate the current CPU state:

- Yellow: STOP mode
- Green: RUN mode
- Red: CPU fault

The Event Log below the Device table shows the results of the operation.

4.7 Setting a CPU's time

The Time button sets the time for selected CPUs to the time of your programming device. Time transformation information for time zone and daylight saving time is not changed and must be modified in the TIA Portal project.

> ⚠ **WARNING**
>
> **Changing the CPU time of day could disrupt process operation**
>
> Changing the CPU time of day could cause process disruption to STEP 7 programs that execute program logic based on the time of day.
>
> Unexpected operation of a process or machine could result in death or injury to personnel and/or property damage.
>
> Ensure that changing the time of day does not cause unwanted effects in the STEP 7 program.

You must have write access (Page 308) to a CPU to set the CPU time.

To set the CPU time to the programming device time, follow these steps:

1. Select one or more devices to include in the operation. You can use the Devices check box at the top of the Device table to select or deselect all devices. Alternatively, you can use the right-click shortcut menu or the "View > Select" menu command to access the "Select Row(s)" and "Deselect Row(s)" commands.
2. For each selected CPU, enter a password, if used, in the "CPU Password" column of the currently open tab.
3. Select the "Operations > Set Time" menu command or click the "Set the selected devices time" toolbar button:
4. Click the "Continue" button on the "Set Time" dialog box.

The SIMATIC Automation Tool sets the system time on the selected devices to your current programming device time. The Event Log shows the results of the operation.

4.8 Setting an IP address

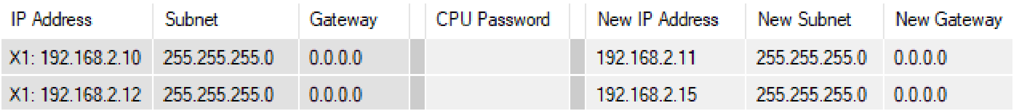
To be able to set a device IP address that persists after a power cycle of the device, the device configuration in the STEP 7 project must specify that the IP address is set at the device (Page 17).

If the STEP 7 project in the device does not enable this setting, then you can set a temporary IP address with the SIMATIC Automation Tool. The device must be in STOP mode and the address that you set is temporary until the next power cycle.

Setting the IP address on supported devices

To set the IP address for a device, follow these steps:

1. Click the "Set IP Address" tab.
2. Select one or more devices to include in the operation. You can use the Devices check box at the top of the Device table to select or deselect all devices. Alternatively, you can use the right-click shortcut menu or the "View > Select" menu command to access the "Select Row(s)" and "Deselect Row(s)" commands.
3. Enter values in the "New IP Address", "New Subnet", and "New Gateway" columns. Note that you do not enter a communication interface such as "X1" or "X2" when you enter the New IP Address.



IP Address	Subnet	Gateway	CPU Password	New IP Address	New Subnet	New Gateway
X1: 192.168.2.10	255.255.255.0	0.0.0.0		192.168.2.11	255.255.255.0	0.0.0.0
X1: 192.168.2.12	255.255.255.0	0.0.0.0		192.168.2.15	255.255.255.0	0.0.0.0

   If you enter invalid syntax, the SIMATIC Automation Tool displays the field in red. Correct any errors if necessary.

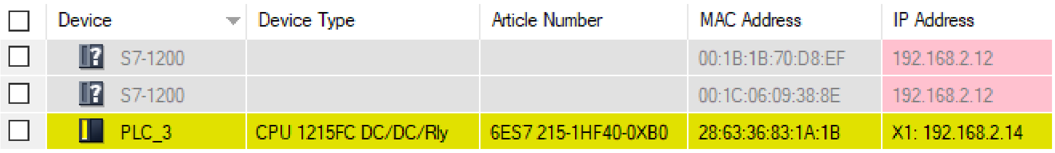
4. Select Update from the Operations menu or click the Update button on the toolbar and select "Set IP address" from the drop-down menu.

The Update operation sets the IP, subnet, and gateway addresses in the selected devices.

The Event Log below the Device table shows the results of this operation.

Duplicate IP addresses

When two or more devices have the same IP address, the IP Address cells appear in red as shown in the following image:



☐	Device	Device Type	Article Number	MAC Address	IP Address
☐	S7-1200			00:1B:1B:70:D8:EF	192.168.2.12
☐	S7-1200			00:1C:06:09:38:8E	192.168.2.12
☐	PLC_3	CPU 1215FC DC/DC/Rly	6ES7 215-1HF40-0XB0	28:63:36:83:1A:1B	X1: 192.168.2.14

You can select devices with duplicate IP addresses, update the IP addresses, and correct the network problem. Only the following operations on devices with duplicate IP addresses are possible:

- Delete
- Set IP address

- Set PROFINET name
- Identify devices

No other device operations are possible for devices that have duplicate IP addresses.

Setting the IP address on unsupported devices

Setting the IP address is a MAC address-based operation that uses DCP (Discovery and Configuration Protocol). The SIMATIC Automation Tool can use the DCP-MAC address operations for all directly-connected network devices.

Unsupported devices might not accept a change based on the hardware configuration of the device.

By default, the SIMATIC Automation Tool displays unsupported devices. You can disable this option in the General settings (Page 81)

Devices behind routers

You cannot use the Set IP address command to set the IP address of a device behind a router (Page 114). You must use the Insert Device command (Page 25).

IP address after a power cycle

The device configuration of the STEP 7 project (Page 17) determines how the device uses the IP address that you set in the SIMATIC Automation Tool. When you set an IP address for a device in the SIMATIC Automation Tool, the next power cycle for that device has the following consequences:

STEP 7 device configuration setting	IP address after power cycle
IP address is set directly at the device (or equivalent)	The IP address that you set in the SIMATIC Automation Tool is the device IP address
Set IP address in the project (or equivalent)	The IP address in the device configuration of the STEP 7 project is the IP address

Note

Temporary (Emergency) IP addresses

For more information on temporary (emergency) IP addresses, refer to this FAQ (https://support.industry.siemens.com/cs/ww/en/view/97649773).

4.9 Setting a PROFINET name

To change a PROFINET device name, the device configuration in the downloaded TIA Portal project (Page 17) must support this change.

Valid PROFINET names follow the standard DNS (Domain Name System) naming conventions. If you enter a name that is invalid, the SIMATIC Automation Tool converts it to a valid PROFINET name. You can see the converted name in the PROFINET Converted Name column.

PROFINET name rules

The maximum number of characters for the device name is 63. Valid characters are the lower case letters "a" through "z", the digits 0 through 9, the hyphen character (minus sign), and the period character.

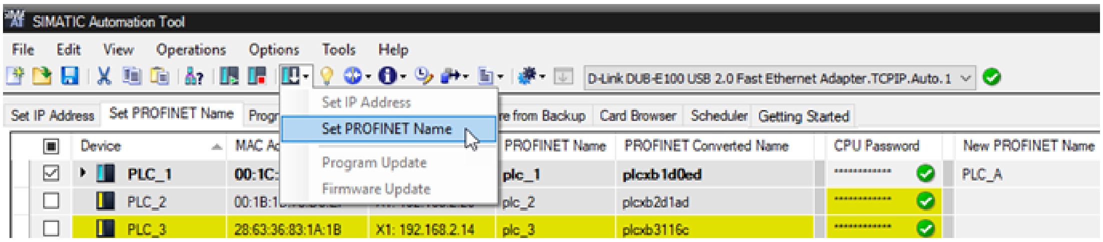
Invalid names

- The name must not have the format n.n.n.n where n is a value of 0 through 999.
- You cannot begin the name with the string port-nnn or the string port-nnnnnnn, where n is a digit 0 through 9. For example, "port-123" and "port-123-45678" are illegal names.
- A name cannot start or end with a hyphen "-" or period "." character.

Setting the PROFINET name

To set the PROFINET name for one or more devices, follow these steps:

1. Click the "Set PROFINET Name" tab.
2. Select one or more devices to include in the operation. You can use the Device check box at the top of the Device table to select or deselect all devices. Alternatively, you can use the right-click shortcut menu or the "View > Select" menu command to access the "Select Row(s)" and "Deselect Row(s)" commands.
3. Enter a new PROFINET name in the "New PROFINET Name" column.
4. Select Update from the Operations menu or click the Update button on the toolbar and select "PROFINET Name" from the button drop-down menu.



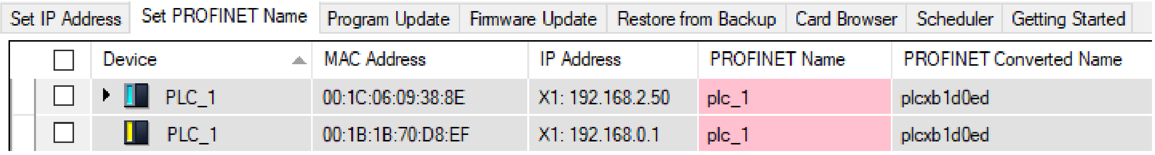
☐	Device	MAC Ad		PROFINET Name	PROFINET Converted Name	CPU Password	New PROFINET Name
☑	PLC_1	00:1C:		plc_1	plcxb1d0ed	************ ✓	PLC_A
☐	PLC_2	00:1B:1		plc_2	plcxb2d1ad	************ ✓	
☐	PLC_3	28:63:36:83:1A:1B	X1: 192.168.2.14	plc_3	plcxb3116c	************ ✓	

The Update operation sets new PROFINET names in the selected devices. If you enter an invalid PROFINET name according to the PROFINET name rules, the SIMATIC Automation Tool corrects the name to a valid name. The column "PROFINET Converted Name" shows the converted name.

The Event Log below the Device table shows the results of this operation.

Duplicate PROFINET names

When two or more devices have duplicate PROFINET names, the SIMATIC Automation Tool indicates the duplicates with red text. The SIMATIC Automation Tool supports full functionality for these devices and displays all other information:



☐	Device	MAC Address	IP Address	PROFINET Name	PROFINET Converted Name
☐	PLC_1	00:1C:06:09:38:8E	X1: 192.168.2.50	plc_1	plcxb1d0ed
☐	PLC_1	00:1B:1B:70:D8:EF	X1: 192.168.0.1	plc_1	plcxb1d0ed

Setting the PROFINET name on unsupported devices

MAC address-based operations use the DCP (Discovery and Configuration Protocol). DCP is an Ethernet standard. The SIMATIC Automation Tool can use the DCP-MAC address operations Scan Entire Network, Identify, Update IP address, and Update PROFINET name for all directly-connected network devices (CPUs, HMIs, decentralized I/O, and other devices).

Select the unsupported device row, enter new data in the appropriate column, and update unsupported device PROFINET names in the same way that you update supported devices.

Unsupported devices might not accept a change based on the hardware configuration of the device.

By default, the SIMATIC Automation Tool displays unsupported devices. You can disable this option in the General settings (Page 81).

4.10 Resetting a device

4.10.1 Resetting communication parameters

The SIMATIC Automation Tool supports the DCP "Reset Communication Parameters" command for PROFINET devices. The PROFINET standard defines the "Reset Communication Parameters" command to set devices similar to an "out of the box state". Specifically, the command resets the following values for the communication interfaces and ARs (Application Relationships):

Parameter	Value
NameOfStation	"" (empty string)
IP suite parameters	0.0.0.0
DHCP parameters, if available	Factory values
P Dev parameters: • PD IR Data • PD Port Data Adjust • PD Interface • MRP Data Adjust • others	Factory values
Parameters from MIB-II adjusted by SNMP, for example: • sysContact • sysName • sysLocation	Factory values

To reset communication parameters for PROFINET devices, follow these steps:

1. Select the devices in the Device table that you want to reset.
2. Select the "Operations > Reset > Reset Communication Parameters" menu command.

The SIMATIC Automation Tool sends the "Reset Communication Parameters" DCP command to the devices. The Event Log displays a message for devices that do not support the DCP reset command. The Event Log also displays error messages for devices that return errors.

Note

You must configure CPUs as IO devices in the STEP 7 device configuration to enable the CPU to reset communication parameters. The IO device settings are in the Operating mode group of the PROFINET interface settings in the STEP 7 device configuration.

4.10.2 Resetting devices to factory defaults

You can reset selected devices to factory defaults, except for the IP address. The device retains the existing IP address to preserve your network IP assignments.

You can only reset a CPU to factory defaults through the CPU network interface. You cannot reset a CPU to factory defaults through a CM or CP interface.

If you have a chain communication topology and the Communications settings (Page 82) enable multi-threading, be aware of the risk of communication disruption with this operation.

Note

Fail-Safe devices

If a fail-safe CPU is protected, you must enter the safety password (Page 308) in the "CPU Password" column to reset a fail-safe device to factory defaults.

You must confirm an additional prompt and reselect your device if the program in the F-CPU is a safety program (Page 314).

Reset to factory defaults requests for fail-safe devices are placed in the safety-relevant operation queue and only single-thread sequential processing is allowed.

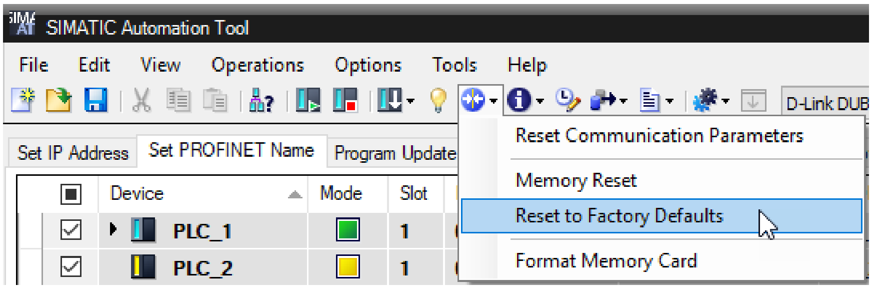
⚠ **WARNING**

Verify that the device is not actively running a process before you reset the device to factory defaults

A Reset to factory defaults operation causes the CPU to go to STOP mode, which could affect the operation of an online process or machine. Unexpected operation of a process or machine could result in death or injury to personnel and/or property damage.

To reset selected devices to factory defaults, follow these steps:

1. Select one or more devices to include in the operation. You can use the Devices check box at the top of the Device table to select or deselect all devices. Alternatively, you can use the right-click shortcut menu or the "View > Select" menu command to access the "Select Row(s)" and "Deselect Row(s)" commands.
   For each selected CPU, enter a password, if used, in the "CPU Password" column of the currently open tab. Reset to factory defaults is a safety-relevant operation. You must enter the safety password (Page 308) for a protected fail-safe device.
2. Select the "Operations > Reset > Reset to Factory Defaults" menu command to start the operation. Alternatively, click the "Reset" toolbar button and select "Reset to Factory Defaults" from the drop-down menu.
   For fail-safe CPUs, the SIMATIC Automation Tool displays a dialog for additional confirmation. Select the device, devices, or all devices that you want to reset to factory defaults.



3. Click the "Continue" button on the "Reset to Factory" dialog.
   The SIMATIC Automation Tool resets the selected devices to factory defaults.

Note

Reset to Factory operation does not clear SIMATIC memory card

If you have a SIMATIC memory card in a CPU, a "Reset to Factory" operation does not clear the contents. If you do not have a SIMATIC memory card in a CPU, "Reset to Factory" clears the program in the internal load memory of the CPU.

The Event Log below the Device table shows the results of the operation.

Effect of resetting a CPU to factory defaults on IP address

Before you reset a CPU to factory defaults, the STEP 7 project in the CPU has a setting in the device configuration for how the IP address is set (Page 17). In short, the setting is one of the following:

- Set IP address in the project (or equivalent)
- IP address is set directly at the device (or equivalent)

In addition, a user might have updated the device IP address (Page 32) with the SIMATIC Automation Tool. In that case, the user-updated device IP address is currently in the CPU.

The IP address after you reset the CPU to factory defaults and after a subsequent CPU power cycle depends on several factors. The following table describes various possibilities for the IP address of the device.

IP address in the CPU	CPU has SIMATIC memory card?	IP address after program update	IP address after CPU power cycle after reset to factory defaults
Device configuration: Set IP address in project	Yes	IP address is the IP address from the project on the SIMATIC memory card.	IP address is the IP address from the project on the SIMATIC memory card.
	No	IP address is the IP address of the device before you reset to factory defaults.	IP address is the IP address of the device before you reset to factory defaults.
Temporary IP address, for example from Set IP address operation (Page 32)	Yes	IP address is the IP address from the project on the SIMATIC memory card.	IP address is the IP address from the project on the SIMATIC memory card.
	No	IP address is the IP address of the device before you reset to factory defaults.	IP address is the IP address of the device before you reset to factory defaults.
Device configuration: IP address is set directly at the device	Not applicable	IP address is the IP address of the device before you reset to factory defaults.	IP address is the IP address of the device before you reset to factory defaults.

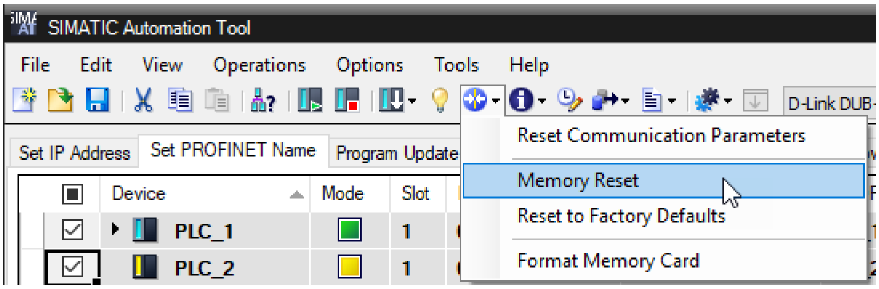
4.10.3 Performing a memory reset on CPUs

You must have write access (Page 308) to a CPU to perform a memory reset.

To perform a memory reset on selected CPUs, follow these steps:

1. Select one or more devices to include in the operation. You can use the Devices check box at the top of the Device table to select or deselect all devices. Alternatively, you can use the right-click shortcut menu or the "View > Select" menu command to access the "Select Row(s)" and "Deselect Row(s)" commands.
2. For each selected CPU, enter a password, if used, in the "CPU Password" column.
3. Select the "Operations > Reset > Memory Reset" command to start the operation. Alternatively, click the "Reset" toolbar button and select "Memory Reset" from the drop-down menu.
4. Click the "Continue" button on the "Memory Reset" dialog box.

The SIMATIC Automation Tool performs a memory reset on the selected devices.



The Event Log below the Device table shows the results of this operation.

4.11 Formatting a memory card

Depending on the device type or device family, SIMATIC memory cards in CPUs can contain the following types of data:

- Load memory of a CPU
- Storage medium for a project
- Firmware backup and update
- Storage medium for the PROFINET device name
- Project transfer from one device to another
- Other files

From the SIMATIC Automation Tool, you can format SIMATIC memory card through the CPU network interface. You cannot format it through a CM or CP interface.

Note

Use only Siemens software to format SIMATIC memory cards

If you use a SIMATIC memory card for non-SIMATIC purposes or you format it incorrectly, the internal structure of the SIMATIC memory card is overwritten. The structure is not recoverable and the SIMATIC memory card becomes unusable for SIMATIC devices.

Do not use SIMATIC memory cards for non-SIMATIC-related purposes and do not format SIMATIC memory cards with third-party devices or Windows tools.

If you have a chain communication topology and the Communications settings (Page 82) enable multi-threading, be aware of the risk of communication disruption with this operation.

⚠ **WARNING**

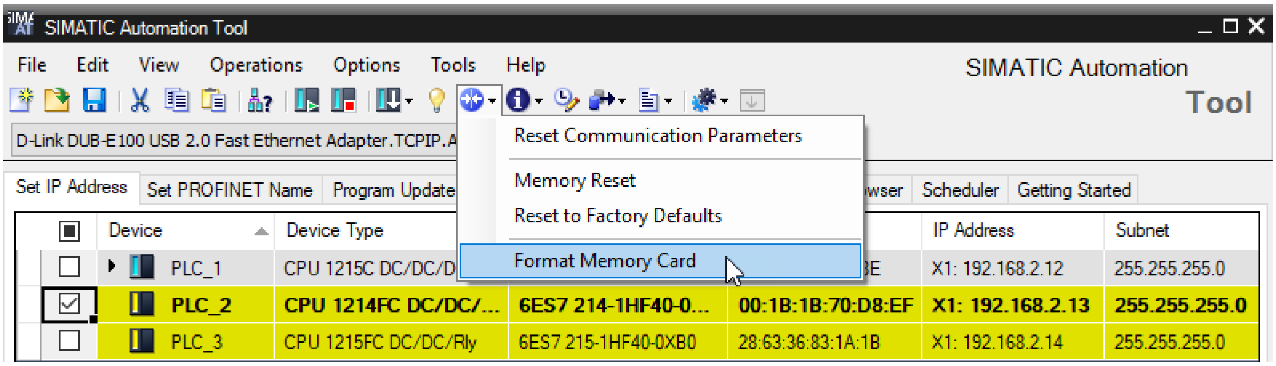
Verify that the device is not actively running a process before formatting a memory card

Formatting a SIMATIC memory card causes a CPU to go to STOP mode, which could affect the operation of an online process or machine. Unexpected operation of a process or machine could result in death or injury to personnel and/or property damage.

To format SIMATIC memory cards on selected devices, follow these steps:

1. Select one or more devices to include in the operation. You can use the Devices check box at the top of the Device table to select or deselect all devices. Alternatively, you can use the right-click shortcut menu or the "View > Select" menu command to access the "Select Row(s)" and "Deselect Row(s)" commands.
2. For each selected CPU, ensure that the SIMATIC memory card is inserted in the CPU.
3. Enter a password, if used, in the "CPU Password" column for each selected CPU.

4. Select the "Operations > Reset > Format Memory Card" command to start the operation. Alternatively, click the "Reset" toolbar button and select "Format Memory Card" from the button drop-down menu.



■	Device	Device Type			IP Address	Subnet
□	PLC_1	CPU 1215C DC/DC/D		3E	X1: 192.168.2.12	255.255.255.0
☑	PLC_2	CPU 1214FC DC/DC/...	6ES7 214-1HF40-0...	00:1B:1B:70:D8:EF	X1: 192.168.2.13	255.255.255.0
□	PLC_3	CPU 1215FC DC/DC/Rly	6ES7 215-1HF40-0XB0	28:63:36:83:1A:1B	X1: 192.168.2.14	255.255.255.0

5. Click the "Continue" button on the "Format Memory Card" dialog box.

The SIMATIC Automation Tool performs the operation for each selected device. Check the Event Log for the results of the operations. Check the CPU diagnostics following the format operation. For device-specific questions, refer to the documentation for the device.

Note

Fail-Safe devices

If a fail-safe CPU is protected, you must enter the safety password (Page 308) in the "CPU Password" column to format a SIMATIC memory card in a fail-safe device.

You must confirm an additional prompt and reselect your device if the program in the F-CPU is a safety program (Page 314).

For fail-safe devices, the SIMATIC Automation Tool places requests to format a SIMATIC memory card in the safety-relevant operation queue and only allows single-thread sequential processing.

Formatting a SIMATIC memory card has no effect on the device IP address.

4.12 Updating device firmware

The SIMATIC Automation Tool can perform firmware updates on a group of devices. The file extension for a firmware update depends on the device type:

- For HMIs, a firmware update file has an .fwf extension.
- For SCALANCE devices, firmware update files have an .lad, .sfw, or .fwl extension.
- For CPUs and other devices, you can use the format of a single .upd file or the older (classic) format which uses three or more separate .upd files.

If you have a chain communication topology and the Communications settings (Page 82) enable multi-threading, be aware of the risk of communication disruption with this operation.

The SIMATIC Automation Tool also provides a two-step firmware update operation (Page 47) that you can use for some devices to minimize disruption to your process.

Note

Configuration of S7-1200 CM communication modules

For an S7-1200 CM module, you must configure the CM module in STEP 7 and download the configuration to the module. You can then use the SIMATIC Automation Tool to update the CM firmware.

CPU firmware downgrade

You can use the SIMATIC Automation Tool to downgrade CPU firmware (load a previous firmware version), but the IP address and program might be erased. In this case, the IP address might be reset to 0.0.0.0 and a new network scan is required to communicate with this device. You must set the IP address to restore your previous network address.

You cannot downgrade the firmware for some devices. Check your device documentation.

Note that programs for one CPU firmware version might not run on another firmware version. The CPU cannot go to RUN mode if the program is incompatible with the firmware version.

If you performed a firmware downgrade and the program is no longer compatible for the new firmware version, you must reset your CPU as follows:

- If your CPU is running from internal load memory, reset the CPU to factory defaults.
- If your CPU is running from external load memory, format the memory card and power cycle the CPU.

After you reset the CPU, download a program that is valid for the firmware version in the CPU.

Determining whether a firmware update is possible

The Device Catalog (Page 95) lists the devices and versions for which the SIMATIC Automation Tool can perform a firmware update. For the SIMATIC Automation Tool to be able to update firmware for devices, the device and version must support firmware update and be one of the following:

- A CPU at the root level in the Device table, which is connected to the SIMATIC Automation Tool through the CPU network interface. The SIMATIC Automation Tool does support firmware update of a CPU through a CM or CP interface.
- An HMI at the top level in the Device table
- A local module of a CPU, including PROFIBUS CM/CP modules
  Local modules for a CPU appear beneath the CPU in the Local Modules folder:
- A distributed device or module within a PROFINET IO-System
  Distributed I/O modules in a PROFINET IO-System appear beneath a PROFINET IO-System folder:

Note that both conditions must be met:

- The Device Catalog indicates that the SIMATIC Automation Tool supports firmware update for the device.
- The device or module is at a supported level in the Device table.

Note that the SIMATIC Automation Tool does not support firmware update for modules in the following IO-Systems:

- PROFIBUS IO-System:
- AS-i IO-System:

Note

Dependencies for local modules of CPUs

You cannot update the firmware for local modules if the CPU does not support firmware update. For example, S7-1200 CPUs with a firmware version earlier than V4.0 do not support firmware update over Ethernet. You cannot use the SIMATIC Automation Tool to update the firmware for local modules connected to these CPUs.

Preparing CPU and module firmware update files for use with the SIMATIC Automation Tool

You can obtain CPU and module firmware update files from the Siemens product support (https://support.industry.siemens.com/cs/us/en/ps) web site.

You can also select a device row and then select "Check for Firmware Updates" from either the Tools menu or the Tools toolbar icon. The SIMATIC Automation Tool launches the device's customer support web page. The Siemens support web page selection corresponds to the article number displayed in a Device table row. For example, a "Check for updates" command on article number 6ES7 215-1HG31-0XB0 links to the corresponding CPU 1215C web support page (https://support.industry.siemens.com/cs/products/6es7215-1hg31-0xb0/cpu-1215c-dcdcrly-14di10do2ai2ao?pid=79072&dtp=Download&mlfb=6ES7215-1HG31-0XB0&lc=en-WW).

For a CPU example, the firmware update file named **6ES7 211-1AE40-0XB0**_V04.00.02**.exe** is only for the **CPU 1211C DC/DC/DC** model. If you use the .upd file within this package for any other S7-1200 CPU model, the update process will fail.

When you execute the update file and extract the files, you see the following set of files and folders.

- File: S7-JOB.S7S
- Folder: FWUPDATE.S7S contains the .upd file.
  - file: **6ES7 211-1AE40-0XB0** V04.00.02**.upd** (.upd file used by the SIMATIC Automation Tool)

For an I/O module example, the firmware update file named **232-4HD32-0XB0**_V203**.exe** is only for the **SM 1232 ANALOG OUTPUT 4AO** module. The self-extracting .exe file contains the file **6ES7 232-4HD32-0XB0** V02.00.03_00.00.00.00.**upd** that is used by the SIMATIC Automation Tool**.**

Note

New format firmware update files

- The self-extracting .exe update package name must refer to the article number of the device that you want to update.
- The extracted .upd file name must match the article number of the device and the firmware version that you want to load.

Note

Old format firmware update files

- The self-extracting .exe update package name must refer to the article number of the device that you want to update.
- Contains three or more files depending on the firmware size.
- Create a folder with any name in the Firmware Update folder (Page 84). You can name the folder with the article number and version number so it will be easier to identify, but you can use any name. The SIMATIC Automation Tool parses all firmware files at startup to confirm exact firmware version numbers.

Copy .upd files to the Firmware Update folder

The new format firmware update single .upd files have the target module model and version numbers in their file names. You can copy multiple .upd files to a single firmware folder and then identify the target module by the .upd file name. Copy all the .upd files you need to the Firmware Update folder (Page 84).

⚠ **WARNING**

Verify that the CPU is not actively running a process before installing firmware updates

Installing a firmware update for a CPU or module causes the CPU to go to STOP mode, which could affect the operation of an online process or machine. Unexpected operation of a process or machine could result in death or injury to personnel and/or property damage.

Preparing HMI firmware update files for use with the SIMATIC Automation Tool

A firmware update file for an HMI file has an .fwf file extension. This file is part of the HMI runtime and operating system that you can copy from the TIA Portal project. See the topic Updating a device's program (Page 48) for instructions on transferring the SIMATIC.HMI file structure from the TIA Portal to a storage medium such as a SIMATIC memory card. After you copy the SIMATIC.HMI file structure to your storage medium, follow these steps to update the firmware for an HMI device:

1. From the SIMATIC.HMI folder on your SIMATIC memory card or other device, start at the "Firmware" subfolder and navigate until you see the file with an .fwf file extension.
2. Copy the .fwf file to your Firmware Update (Page 84) folder.

You can store multiple .fwf files for multiple HMI devices and multiple firmware versions in the Firmware Update folder.

Note

Transfer Settings for an HMI device (SIMATIC Panel)

To communicate with an HMI, you must set the Transfer Settings on the HMI device to PN/IE or Ethernet. You must set the Communications settings (Page 82) for HMI Transfer Channel to the same setting.

Preloading firmware update files

You can preload the latest available firmware update in the "New Firmware Version" column. The SIMATIC Automation Tool examines the firmware update folder. For each device, if the SIMATIC Automation Tool finds corresponding firmware update files that are newer versions than the current version in the device, the SIMATIC Automation Tool loads the "New Firmware Version" column with the newest version that is available.

To preload firmware update files, select the "Tools > Preload Firmware Update Files" menu command. Alternatively, you can click the Tools button ⚙ and select the "Preload Firmware Update Files" command from the shortcut menu.

For each device, the SIMATIC Automation Tool then preloads the "New Firmware Version" column with the newest version in the folder that is newer than the version in the device. If the SIMATIC Automation Tool does not find an update file that is newer than the version currently in the device, it leaves the "New Firmware Version" column empty.

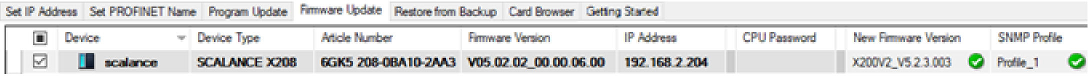
You can copy and paste (Page 120) firmware update file cells to other applicable firmware update cells. Copying and pasting can simplify firmware update operations for multiple devices.

Performing firmware updates for devices

Follow these steps to perform a firmware update operation:

1. Select one or more devices to include in the operation. You can use the Devices check box at the top of the Device table to select or deselect all devices. Alternatively, you can use the right-click shortcut menu or the "View > Select" menu command to access the "Select Row(s)" and "Deselect Row(s)" commands.
2. Click the "Firmware Update" tab.

3. If you did not preload firmware update files, select the firmware update files. For each device row that you select, click the "New Firmware Version" column drop-down list and select a firmware version. The drop-down list shows the names of the files in the Firmware Update folder (Page 84) that correspond to your device selection. If new firmware versions are available in the Firmware Update folder, then these files are available from the "New Firmware Version" drop-down list.
   You can also use the browse [...] button and navigate to a folder on your programming device that contains firmware update files. Select a file to add it to the drop-down list. If the selected file has the same name as one of the files already listed, the SIMATIC Automation Tool adds a number to the new file to make the names unique. To help you identify files, a tooltip displays the entire path and filename.
   If selecting a firmware update file in the old format, select the header.upd file. Do not select one of the other .upd files in the set.
4. For each selected CPU, enter a password, if used, in the "CPU Password" column.
5. For each selected SCALANCE device, select the SNMP profile (Page 88) for your device. An icon next to the SNMP profile indicates whether the SNMP profile name corresponds to a profile name in the profile file in the SNMP Profiles folder (Page 88):
   ✓ SNMP profile does correspond to an SNMP profile in the profile file in the SNMP Profiles folder.
   ✗ SNMP profile does not correspond to an SNMP profile in the profile file in the SNMP Profiles folder.



▣	Device	Device Type	Article Number	Firmware Version	IP Address	CPU Password	New Firmware Version	SNMP Profile
☑	scalance	SCALANCE X208	6GK5 208-0BA10-2AA3	V05.02.02_00.00.06.00	192.168.2.204		X200V2_V5.2.3.003 ✓	Profile_1 ✓

> **Note**
>
> You need to provide a TFTP (Trivial File Transfer Protocol) server to install firmware in SCALANCE devices.

6. Select the "Operations > Update > Firmware Update" menu command to start the operation. Alternatively, click the Update toolbar button and the "Firmware Update" command from the button drop-down menu.

The Event Log below the Device table shows the results of the operation. After a successful firmware update operation, the SIMATIC Automation Tool clears the "New Firmware Version" field.

Fail-Safe device CPU passwords for firmware update

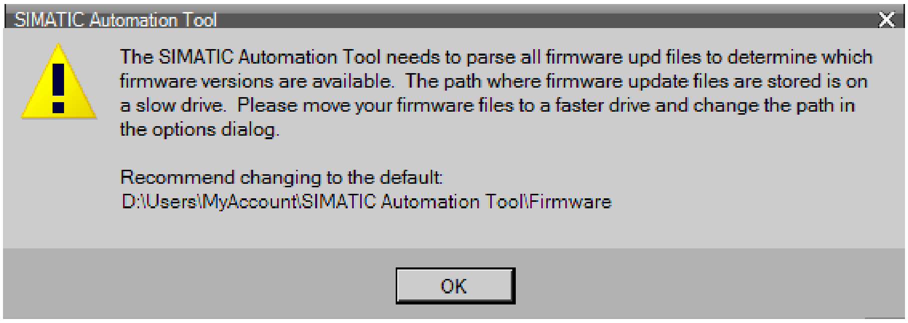
For firmware versions earlier than S7-1200 V4.2 and S7-1500 V2.0, a protected fail-safe CPU requires the safety password (Page 308) for the firmware update operation.

S7-1200 V4.2 or later and S7-1500 V2.0 or later require only a password with the write access level (Page 308).

The SIMATIC Automation Tool does not check the required access level for different firmware versions. It initiates the operation. The device firmware will reject a password if it is insufficient and the Event Log displays a message.

Timeout error message due to slow communication with .upd file storage device

If you see the following error message box, then more than ten seconds have elapsed. The SIMATIC Automation Tool has not completed processing all of the .upd files. The time required to open and scan all the .upd files depends on data access time and the number of .upd files in the folder.



The SIMATIC Automation Tool needs to parse all firmware upd files to determine which firmware versions are available. The path where firmware update files are stored is on a slow drive. Please move your firmware files to a faster drive and change the path in the options dialog.

Recommend changing to the default:
D:\Users\MyAccount\SIMATIC Automation Tool\Firmware

This timeout error can occur when communication with a remote storage device is too slow.

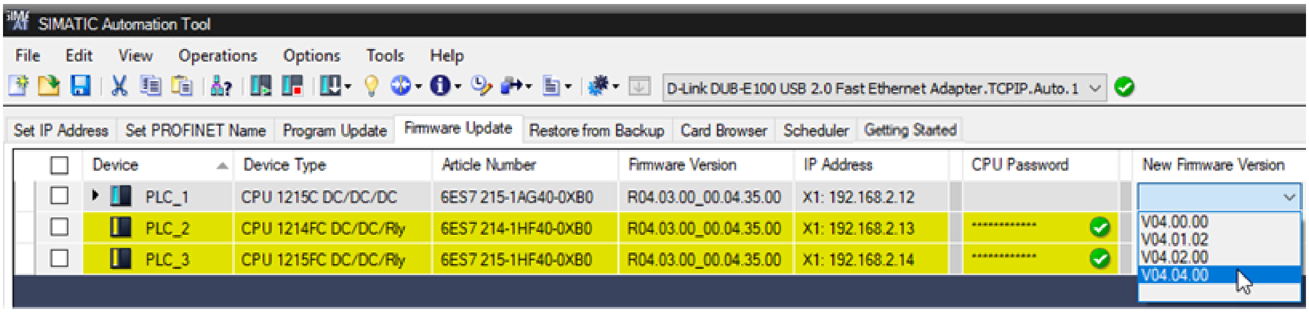
To prevent this problem, copy the firmware update files to a local storage device that provides faster access. Assign this path as the firmware update folder (Page 84) and try the operation again.

Example: Firmware update for a CPU

This example shows how to update the firmware for a single CPU.

To perform the firmware update, follow these steps:

1. Open the drop-down list of available versions from the "New Firmware Version" column. The drop-down list shows all of the available firmware update files in the firmware update folder (Page 84).
2. Select the firmware update version to use. (If you had selected more than one device, then you would choose an update file for each selected device.)



□	Device	Device Type	Article Number	Firmware Version	IP Address	CPU Password	New Firmware Version
□	PLC_1	CPU 1215C DC/DC/DC	6ES7 215-1AG40-0XB0	R04.03.00_00.04.35.00	X1: 192.168.2.12		
□	PLC_2	CPU 1214FC DC/DC/Rly	6ES7 214-1HF40-0XB0	R04.03.00_00.04.35.00	X1: 192.168.2.13	************* ✓	
□	PLC_3	CPU 1215FC DC/DC/Rly	6ES7 215-1HF40-0XB0	R04.03.00_00.04.35.00	X1: 192.168.2.14	************* ✓	

3. Select the "Operations > Update > Firmware Update" menu command to start the operation. Alternatively, click the Update toolbar button followed by the Firmware Update menu command from the drop-down menu.

Note

You cannot update the firmware of some S7-1200 modules with the SIMATIC Automation Tool

Some CPUs do not support firmware update. If a CPU does not support firmware update, you can only update the device with a SIMATIC memory card. You also cannot update the firmware of modules connected to this CPU with the SIMATIC Automation Tool.

Methods for updating firmware other than the SIMATIC Automation Tool

Alternative firmware update methods include the following:

- SIMATIC memory card in applicable devices
- TIA Portal online and diagnostic functions
- Module Information page of a CPU's Web server

Note

HMI operating system and runtime software updates

You can use the SIMATIC Automation Tool program update operation (Page 48) to update the HMI operating system and runtime software. The program update operation updates all data components as necessary for a consistent download.

You can update the HMI operating system (firmware) without updating the runtime software. In versions prior to V3.1 SP2, you could not update just the operating system firmware. You had to perform a program update.

4.12.1 Updating device firmware by two-step method

The standard firmware update operation (Page 40) downloads the firmware update files to the devices that you selected and resets each device. For CPUs, the operation places each CPU into STOP mode before downloading the firmware update file. After the download completes, the firmware update operation then resets the CPUs.

The two-step firmware update operation allows you to download the firmware update files to all the devices that you selected in a single step. You can then activate the firmware update for all the devices you selected in a second step. Activating the firmware update resets the devices. With the two-step method, you can optimize your firmware update procedure and minimize downtime of your process. Some devices do not support a two-step firmware update. The Device Catalog (Page 95) indicates which devices support the two-step method.

For downloading firmware update files to CPUs (the first step), the SIMATIC Automation Tool does not require placing CPUs in STOP mode. The CPUs can continue operating your process. You can select as many devices as you wish and download the firmware files. The SIMATIC Automation Tool displays an Event Log message if a CPU does not support a two-step firmware update.

Performing firmware updates for devices

To perform a two-step firmware update, follow these steps:

1. Follow the steps in Updating device firmware (Page 40) except for the last step.
2. From the "Operations > Update" menu or from the Update toolbar button , select the "Two Step Firmware Update > Download Firmware" menu command.
   If you selected CPUs, you must choose whether to put the CPUs in STOP mode or leave them in their current operating mode. The SIMATIC Automation Tool can download a firmware update file to a CPU regardless of whether the CPU is in RUN mode or STOP mode. A download is faster when the CPU is in STOP mode; however, your process can continue operation when the CPU is in RUN mode.
   The SIMATIC Automation Tool then downloads the firmware update files to all the devices you selected. The Event Log displays errors for any devices that do not support the two-step firmware update, or for any other errors in the operation.
3. After the download completes, you can initiate the second step at any time. From the "Operations > Update" menu or from the Update toolbar button , select the "Two Step Firmware Update > Activate Firmware" menu command.
   The SIMATIC Automation Tool prompts you that it must place CPUs in STOP mode and reset devices. You must confirm that you wish to continue.

The Event Log below the Device table shows the results of the operations. After a successful firmware update operation, the SIMATIC Automation Tool clears the "New Firmware Version" field.

4.13 Updating a device's program

You can update device programs for a CPU through the CPU network interface. You cannot update a CPU program through a CM or CP interface.

You can update device programs (operating system and runtime software) for HMIs through the HMI network interface.

The SIMATIC Automation Tool does not update a device program for a device behind a NAT router if the program update file includes a change in IP address.

Prerequisites

Before you can transfer a program to a CPU or HMI using the SIMATIC Automation Tool, you must have access to the program on one of the following forms of media:

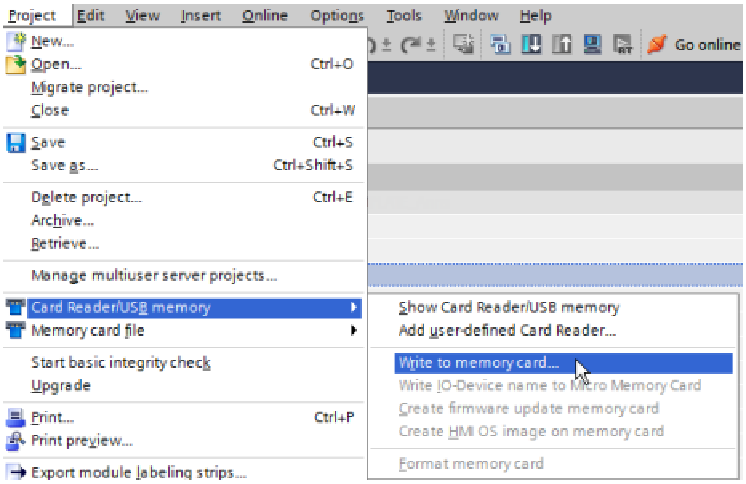
- SIMATIC memory card
- USB flash drive
- Hard drive of your programming device

Example: Preparing a CPU program for use with the SIMATIC Automation Tool

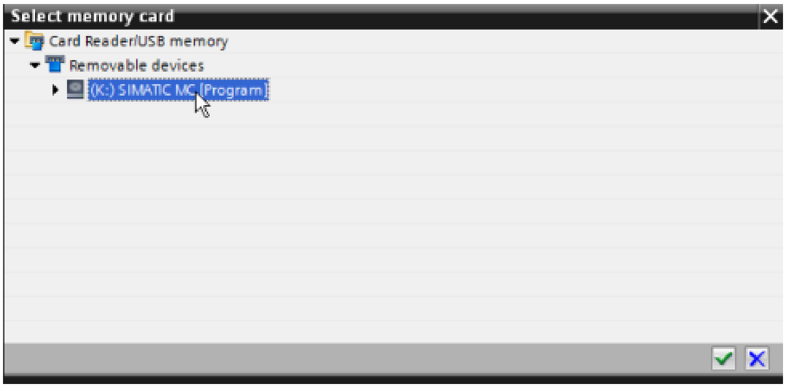
This example uses a SIMATIC memory card for the transfer. You can also transfer to a USB flash drive or Windows folder on your programming device.

To transfer a STEP 7 CPU project to a SIMATIC memory card, follow these steps:

1. Insert a SIMATIC memory card into the card reader for your programming device
2. From STEP 7, select the CPU in the Project tree
3. Select the "Project > Card Reader/USB memory > Write to memory card" menu command:



4. Select your memory card from the dialog:



STEP 7 saves a SIMATIC.S7S folder on your SIMATIC memory card that contains your CPU project. You can also copy the STEP 7 project to the memory card by dragging the project to the memory card in the project tree.

Refer to the STEP 7 Information System (online help) for additional information.

After STEP 7 transfers program data to a storage device, you can use the Windows File Explorer to transfer the program to the folder that is used by the SIMATIC Automation Tool.

Copy the "SIMATIC.S7S" folder for each CPU program

To make a CPU program accessible to the SIMATIC Automation Tool, follow these steps:

1. Create subfolders under the Program Update folder (Page 85). Create one folder for each program and create a folder name that identifies the program. The folder names that you create will appear in the SIMATIC Automation Tool program drop-down list.
2. Use the Windows File Explorer to copy the "SIMATIC.S7S" folder (including all subfolders and files) to each subfolder for each program. You can put a TIA Portal program (a "SIMATIC.S7S" folder) in a zip file archive and extract it to your subfolder location. Note that you update recipes in a separate recipe operation (Page 103).

See the "Example CPU program update" section later in this topic.

Note

STEP 7 program data

The program data is protected. You cannot discover details like the project name or target CPU of a STEP 7 program from the data that is stored in a SIMATIC.S7S folder. You cannot identify one program's SIMATIC.S7S folder from another program's SIMATIC.S7S folder.
You must create and name subfolders under the SIMATIC Automation Tool Program Update folder (Page 85) that identify a program's function or target CPU. Copy a program's SIMATIC.S7S folder into the subfolder that you named. The subfolder names that you create appear in the SIMATIC Automation Tool "Program" column drop-down list and provide the path to the correct SIMATIC.S7S folder.

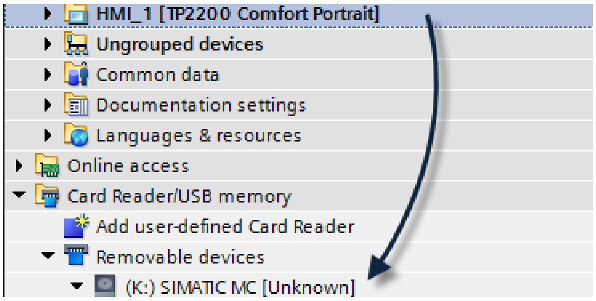
Example: Preparing an HMI operating system and runtime software for use with the SIMATIC Automation Tool

This example uses a SIMATIC memory card for the transfer. You can also transfer to a USB flash drive or Windows folder on your programming device.

HMI devices v14 and higher support saving the operating system and runtime from STEP 7.

To copy the operating system and runtime files for an HMI to a SIMATIC memory card, follow these steps:

1. Insert a SIMATIC memory card into the card reader for your programming device.
2. Expand "Card Reader/USB memory" in the Project tree to show the drive corresponding to your card reader.
3. Select your HMI in the Project tree and drag it to the drive letter of your card reader.



STEP 7 saves a SIMATIC.HMI folder on your SIMATIC memory card that contains your HMI runtime and HMI operating system. HMI updates include the operating system and runtime data. You do not have the option to select a partial update.

After the TIA Portal transfers the SIMATIC.HMI folder to a storage device, use the Windows File Explorer to make the SIMATIC.HMI folder accessible to the SIMATIC Automation Tool:

1. Create a subfolder for the HMI program in the Program Update (Page 85) folder.
2. Copy the SIMATIC.HMI folder to the subfolder.

If you want to use the firmware (operating system) portion of the SIMATIC.HMI folder to update the firmware of an HMI device (Page 40), follow these steps:

1. Navigate through the "Firmware" folder of the SIMATIC.HMI folder until you see an .fwf file.
2. Copy the .fwf file to the Firmware Update (Page 84) folder.

Update CPU programs or HMI operating system and runtime software

If you have a chain communication topology and the Communications settings (Page 82) enable multi-threading, be aware of the risk of communication disruption with this operation.

Note

Fail-Safe devices

If the fail-safe CPU is protected, you must enter the safety password (Page 308) in the "CPU Password" column to update the program in a fail-safe device.

You must confirm an additional prompt for program updates to F-CPUs and reselect your device under the following conditions:

- You are updating a safety program (Page 314) with another safety program
- You are updating a safety program with a standard program
- You are loading a safety program for the first time
- You are updating a standard program that requires the safety password

The SIMATIC Automation Tool places program update requests for fail-safe devices in the F-CPU safety-relevant operation queue. The SIMATIC Automation Tool uses only single-thread sequential processing for the safety-relevant operation queue.

The destination device for a safety program must be a fail-safe CPU.

⚠ **WARNING**

Verify that the device is not actively running a process before updating the program

Installing a new program causes CPUs to go to STOP mode, which could affect the operation of an online process or machine. Unexpected operation of a process or machine could result in death or injury to personnel and/or property damage.

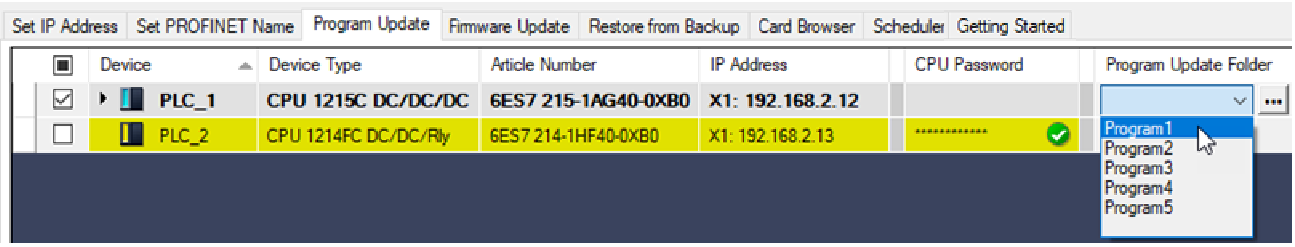
Note

Transfer Settings for an HMI device (SIMATIC Panel)

To communicate with an HMI, you must set the Transfer Settings on the HMI device to PN/IE or Ethernet. You must set the Communications settings (Page 82) for the HMI Transfer Channel to the same setting.

After you have stored programs in the Program Update folder, you can use the SIMATIC Automation Tool to load new programs in one or more devices. To perform a program update, follow these steps:

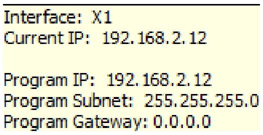
1. Click the "Program Update" tab.
2. Select one or more devices to include in the operation. You can use the Devices check box at the top of the Device table to select or deselect all devices. Alternatively, you can use the right-click shortcut menu or the "View > Select" menu command to access the "Select Row(s)" and "Deselect Row(s)" commands.
3. For each selected device, use the "Program Update Folder" column drop-down list to select a folder name. The drop-down list shows the folders that you created in the program update path.



▣	Device	Device Type	Article Number	IP Address	CPU Password	Program Update Folder
☑	PLC_1	CPU 1215C DC/DC/DC	6ES7 215-1AG40-0XB0	X1: 192.168.2.12		
☐	PLC_2	CPU 1214FC DC/DC/Rly	6ES7 214-1HF40-0XB0	X1: 192.168.2.13	************* ✓	

Program1
Program2
Program3
Program4
Program5

You can also use the browse button ... and navigate to the folder where you have stored a program on your programming device. When you select a program, the SIMATIC Automation Tool adds it to the drop-down list. If the selected file has the same name as one of the files already listed, the SIMATIC Automation Tool adds a number to the new file name to make the names unique. To help you identify files, when you hover over the program name in the Program Update field, a tooltip displays the device interface and IP information, for example:

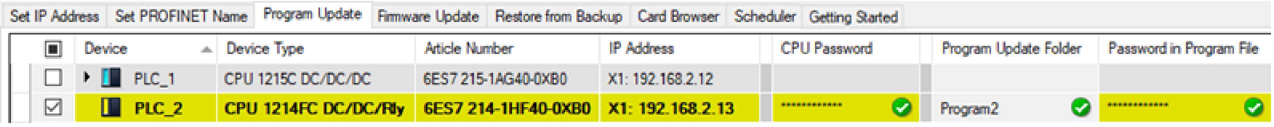


```
Interface: X1
Current IP: 192.168.2.12

Program IP: 192.168.2.12
Program Subnet: 255.255.255.0
Program Gateway: 0.0.0.0
```

If the program file does not contain an IP address, the tooltip displays "Set directly at device" for all IP address fields.
If you selected a safety program, the SIMATIC Automation Tool displays the Program Update Folder cell in yellow. If it is not yellow, it is a standard program.

4. Enter passwords, if used, in the "CPU Password" and "Password in Program File" columns. Program update is a safety-relevant operation. If the device is a fail-safe device, you must enter the safety password (Page 308).



■	Device	Device Type	Article Number	IP Address	CPU Password	Program Update Folder	Password in Program File
□	PLC_1	CPU 1215C DC/DC/DC	6ES7 215-1AG40-0XB0	X1: 192.168.2.12			
☑	PLC_2	CPU 1214FC DC/DC/Rly	6ES7 214-1HF40-0XB0	X1: 192.168.2.13	************* ✓	Program2 ✓	************* ✓

5. Select the "Operations > Update > Program Update" menu command to start the operation. Alternatively, select the Update toolbar button and select "Program Update" from the drop-down menu.

The Event Log below the Device table shows the results of this operation.

Program validation

The SIMATIC Automation Tool verifies the program data before updating the program in a CPU.

If there is an error in the program data, the SIMATIC Automation Tool displays a red "X" icon in the "Program Update Folder" cell. Additional error information is available in a tooltip when you hover over the cell.

Password handling after program update operation

A program file can have a password that might be different from the existing CPU password. When a program file has a password, you must enter the program password in the "Password in Program File" cell to perform a program update. The program password becomes the CPU password after the program update operation completes.

After a successful program update operation, the SIMATIC Automation Tool automatically copies the Password in Program File to the CPU password field and attempts a connection using the new password. The SIMATIC Automation Tool then clears the Password in Program File field and the Program Update Folder field.

If the password you enter in the "Password in Program File" column is not the password configured for the project in STEP 7, the Event Log shows a warning after the operation completes. In this case, the CPU password shows a red 'X' icon that indicates an invalid password.

F-signature validation

A STEP 7 project that contains a safety program has an F-signature that is used to verify the data in a copied program, The F-signature provides an additional level of security for safety programs. After a program update operation, the SIMATIC Automation Tool performs a CRC comparison of the F-Signature in the project to the F-Signature now loaded in the CPU device. An Event Log message (Page 125) shows the result of the CRC comparison.

If the CRC comparison fails, reset the device to factory defaults (Page 36) and repeat the program update.

⚠ **WARNING**

Be sure you load the correct safety program.

Running the wrong program on an F-CPU can affect the operation of a process or machine. Unexpected operation of a process or machine could result in death or injury to personnel and/ or property damage.

Do not attempt to go to RUN mode if you are not sure that you have loaded the correct safety program.

CPU program update rules

The SIMATIC Automation Tool supports the program update operation for standard CPUs and fail-safe CPUs.

Note the following program update rules:

- The firmware version of the CPU hardware must be greater than or equal to the firmware version in the project that you want to load. You can work around this restriction by updating the firmware in the CPU, if possible.
- The Device Catalog shows the devices that support the program update operation. You can access the Device Catalog from the Help menu (Page 92).

Effect of program update on IP address

Before a program update, the STEP 7 project in the CPU has a setting in the device configuration for how the IP address is set (Page 17). In short, the setting is one of the following:

- Set IP address in the project (or equivalent)
- IP address is set directly at the device (or equivalent)

The program that you are updating might have a different setting than the device configuration of the project that is currently in the CPU. In addition, a user might have updated the device IP address (Page 32) with the SIMATIC Automation Tool. In that case, the user-updated device address is currently in the CPU.

The IP address of the device after a program update and after a subsequent CPU power cycle depends on the setting in the new program and the current IP address. The following table describes various possibilities for the IP address of the device.

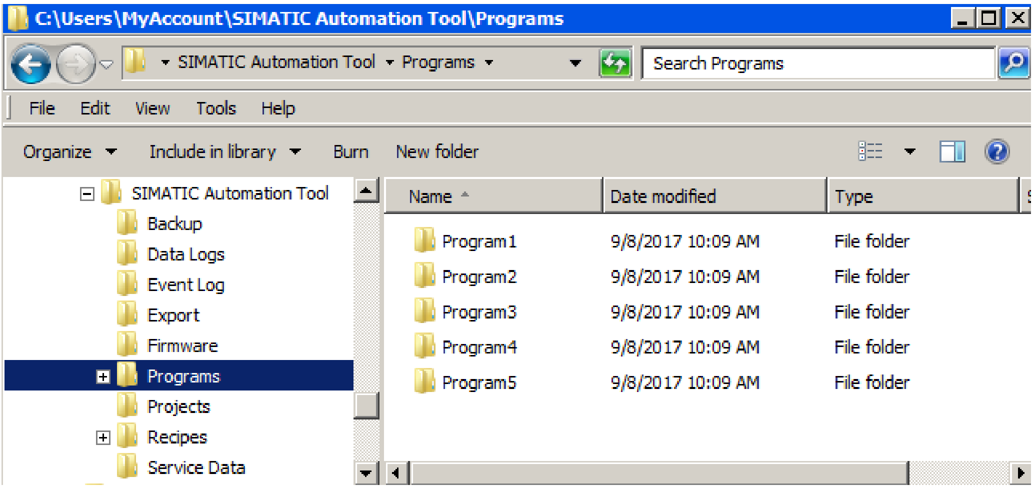
Device configuration in new program for update	IP address after program update	IP address after next CPU power cycle after program update
Set IP address in project	IP address is the IP address from the newly-updated program.	IP address is the IP address from the newly-updated program.
IP address is set directly at the device	IP address is the IP address of the device before the program update.	IP address is the IP address of the device before the program update.

Example: Program update

If you want five different CPU programs available for program update, then you must create and name five folders in the Program Update folder (Page 84). Copy the entire "SIMATIC.S7S" folders to the five corresponding folders.

In this example, the folder names "Program1", "Program2", "Program3", "Program4", and "Program5" identify the available programs. You can use any folder name you want. The folder name could refer to a program function or CPU location.

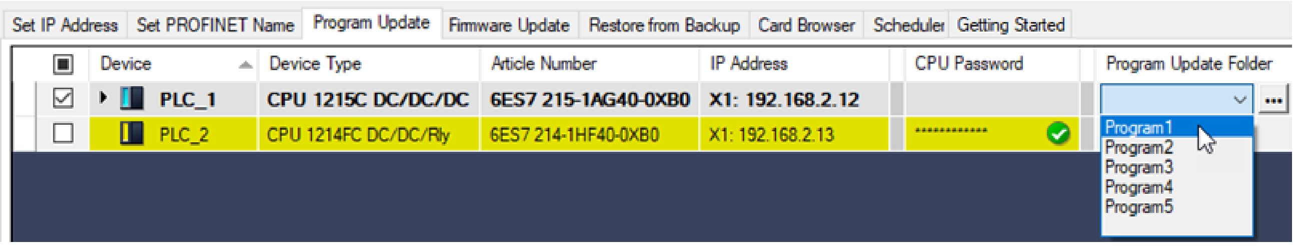
The following image shows the Windows File Explorer view of the subfolders under the Programs folder. You copy the corresponding SIMATIC.S7S program folders to these folders:


C:\Users\MyAccount\SIMATIC Automation Tool\Programs
SIMATIC Automation Tool ▸ Programs ▸
Search Programs
File
Edit
View
Tools
Help
Organize
Include in library
Burn
New folder

- SIMATIC Automation Tool
  - Backup
  - Data Logs
  - Event Log
  - Export
  - Firmware
  - Programs
  - Projects
  - Recipes
  - Service Data

Name	Date modified	Type
Program1	9/8/2017 10:09 AM	File folder
Program2	9/8/2017 10:09 AM	File folder
Program3	9/8/2017 10:09 AM	File folder
Program4	9/8/2017 10:09 AM	File folder
Program5	9/8/2017 10:09 AM	File folder

The following image shows the SIMATIC Automation Tool Program Update tab with the example folder names in the "New Program Update" column drop-down list. You must use the drop-down list in the "New Program Update" column to assign which program to use. If you select more than one CPU row, then you must repeat the process and assign the correct program for each CPU that you selected.



■	Device	Device Type	Article Number	IP Address	CPU Password	Program Update Folder
☑	PLC_1	CPU 1215C DC/DC/DC	6ES7 215-1AG40-0XB0	X1: 192.168.2.12		
☐	PLC_2	CPU 1214FC DC/DC/Rly	6ES7 214-1HF40-0XB0	X1: 192.168.2.13	************ ✓	Program1 Program2 Program3 Program4 Program5

Select the "Operations > Update > Program Update" menu command to start the program update. Alternatively, click the toolbar Update button and select "Program Update" from the drop-down menu.

The process is similar for HMI data. The folder name within a project folder is "SIMATIC.HMI" instead of "SIMATIC.S7S". The procedure is the same.

4.14 Reading and deleting Data Logs

You can read Data Logs from CPUs that have Data Logs on a SIMATIC memory card. The CPU can be in either RUN or STOP mode.

You can only delete Data Logs from CPUs that are in STOP mode. If you attempt to delete Data Logs from multiple CPUs, the SIMATIC Automation Tool checks whether any of the CPUs are in RUN mode. If any of the CPUs are in RUN mode, the SIMATIC Automation Tool prompts you to confirm placing all selected CPUs in STOP mode. If you choose not to allow the change to STOP mode, the SIMATIC Automation Tool stops the delete operation for all CPUs.

The SIMATIC Automation Tool reads and stores Data Logs as .csv (comma-separated values) text files.

You can select multiple data files from one or more CPUs and process all the selected files in a single operation.

The SIMATIC Automation Tool creates a unique folder name for each CPU's Data Log files on your programming device. The folder name is a combination of the CPU name and the MAC address. If you read the same Data Log file twice, the SIMATIC Automation Tool appends a number to the filename to make all filenames unique.

The SIMATIC Automation Tool must have Read access to read Data Log files and Full access (read and write access) to delete Data Log files from a CPU. You might have to enter a password to delete Data Logs. If you do not enter a password, or if the password does not provide write access, the SIMATIC Automation Tool does not delete the data logs for that CPU.

Data Log actions

The SIMATIC Automation Tool provides the following Data Log operations:

- **Read Data Logs:** Reads a copy of selected Data Log file(s) from the CPU and stores it on the programming device. The SIMATIC Automation Tool copies the files to the folder assigned in the Data Logs settings (Page 86).
- **Delete Data Logs:** Deletes selected Data Log files that are stored in a CPU.

To read or delete data log files, follow these steps:

1. Expand a CPU row and make any Data Log folders visible.
2. Expand a Data Log folder and select Data Log files:
3. For each CPU, enter a password, if used, in the "CPU Password" column of the currently open tab.
4. Select the "Operations > Data Logs > Read Data Logs" menu command or the "Operations > Data Logs > Delete Data Logs" menu command. Alternatively, click the toolbar button and select the "Read Data Logs" or "Delete Data Logs" command from the drop-down menu.

The Event Log below the Device table shows the results of your operation.

⚠ **WARNING**

Protect access to Data Log files

Operating a process or machine with compromised data could affect the operation of an online process or machine. Unexpected operation of a process or machine could result in death or injury to personnel and/or property damage.

Take steps to protect Data Log .csv files from being compromised, for example, by limiting network access and by using firewalls.

4.15 Backing up and restoring device data

4.15.1 Backing up a CPU or HMI

Backing up a CPU or HMI creates one or more backup files and stores the files in the backup and restore folder (Page 85).

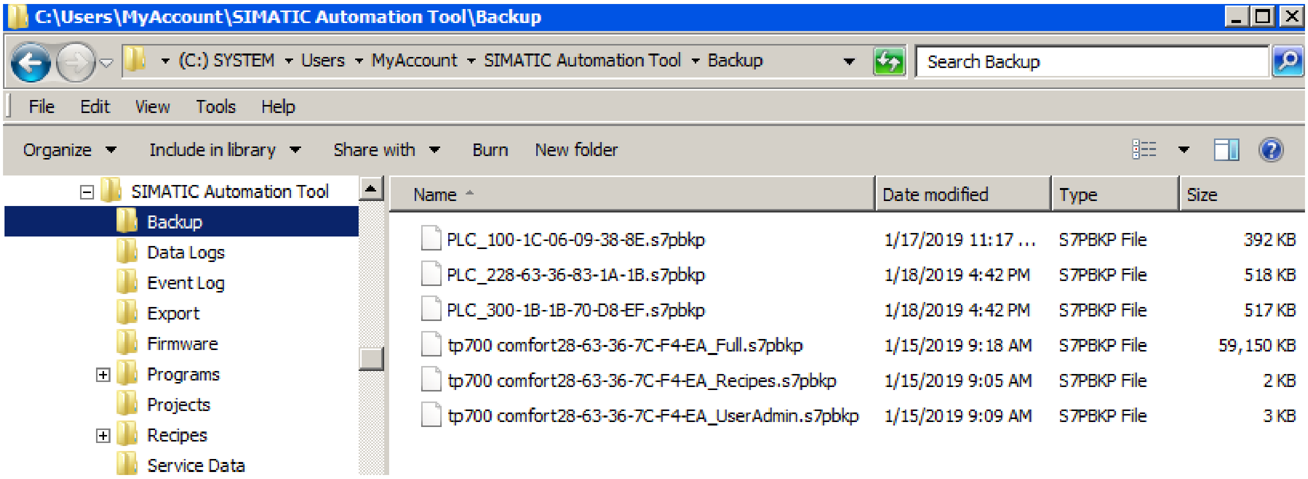
You can use these files in the Restore from Backup operation (Page 59).

You must back up a CPU through an interface on the CPU. You cannot back up a CPU through a CM or CP connection to the CPU.

The SIMATIC Automation Tool provides the following types of backup:

- Full backup of a CPU or an HMI
- HMI recipe backup (Page 58)
- HMI user administration data backup (Page 59)

The file name for a CPU backup combines the project name, backup type, and MAC address. The file name for an HMI backup combines the HMI type, the MAC address, and the type of backup. All backup files have the .s7pbkp file extension. The SIMATIC Automation Tool copies the files to the backup and restore folder: (Page 85)



Name	Date modified	Type	Size
PLC_100-1C-06-09-38-8E.s7pbkp	1/17/2019 11:17 ...	S7PBKP File	392 KB
PLC_228-63-36-83-1A-1B.s7pbkp	1/18/2019 4:42 PM	S7PBKP File	518 KB
PLC_300-1B-1B-70-D8-EF.s7pbkp	1/18/2019 4:42 PM	S7PBKP File	517 KB
tp700 comfort28-63-36-7C-F4-EA_Full.s7pbkp	1/15/2019 9:18 AM	S7PBKP File	59,150 KB
tp700 comfort28-63-36-7C-F4-EA_Recipes.s7pbkp	1/15/2019 9:05 AM	S7PBKP File	2 KB
tp700 comfort28-63-36-7C-F4-EA_UserAdmin.s7pbkp	1/15/2019 9:09 AM	S7PBKP File	3 KB

Performing a full backup

You can start the backup operation from any tab selection.

To create a backup file, follow these steps:

1. Select one or more devices to include in the operation. You can use the "Devices" check box at the top of the Device table to select or deselect all devices. Alternatively, you can use the right-click shortcut menu or the "View > Select" menu command to access the "Select Row(s)" and "Deselect Row(s)" commands.
2. Select the "Backup Device > Full Backup" menu command from the Operations menu. Alternatively, click the Backup and Restore toolbar button and select "Backup Device > Full Backup" from the drop-down menu.

The Event Log (Page 123) shows the results of the operation.

For a successful operation, the SIMATIC Automation Tool creates a backup file for each selected device.

4.15.2 Backing up HMI recipes

You can start the backup operation from any tab selection.

To create an HMI recipe backup file for one or more HMI devices, follow these steps:

1. Select one or more HMI devices to include in the operation. You can use the "Devices" check box at the top of the Device table to select or deselect all devices. Alternatively, you can use the right-click shortcut menu or the "View > Select" menu command to access the "Select Row(s)" and "Deselect Row(s)" commands.
2. Select the "Backup Device > HMI > Recipes" menu command from the Operations menu. Alternatively, click the Backup and Restore toolbar button and select "Backup Device > HMI > Recipes" from the button drop-down menu.

The Event Log (Page 123) shows the results of the operation.

For a successful operation, the SIMATIC Automation Tool creates an HMI recipe backup file for each HMI device. The SIMATIC Automation Tool stores the files in the backup and restore folder (Page 85). The SIMATIC Automation Tool ignores devices that are not HMI devices.

4.15.3 Backing up HMI user administration data

You can start the backup operation from any tab selection.

To create a backup file with HMI user administration data for one or more HMI devices, follow these steps:

1. Select one or more HMI devices to include in the operation. You can use the "Devices" check box at the top of the Device table to select or deselect all devices. Alternatively, you can use the right-click shortcut menu or the "View > Select" menu command to access the "Select Row(s)" and "Deselect Row(s)" commands.
2. Select the "Backup Device > HMI > User Administration Data" menu command from the Operations menu. Alternatively, click the Backup and Restore toolbar button and select "Backup Device > HMI > User Administration Data" from the button drop-down menu.

The Event Log (Page 123) shows the results of the operation.

For a successful operation, the SIMATIC Automation Tool creates a backup file with HMI user administration data for each HMI device. The SIMATIC Automation Tool stores the files in the backup and restore folder (Page 85). The SIMATIC Automation Tool ignores devices that are not HMI devices.

4.15.4 Restoring backup files

Restoring devices from backup files

You use the "Restore Device" command to restore backup files to the corresponding devices. CPU and HMI backup files that you created with the "Backup Device" command have the extension name "s7pbkp". You can restore files from the backup and restore folder (Page 85) or browse to another location.

You must restore CPU backup files through an interface on the CPU. You cannot restore CPU backup files when you use a CM or CP connection to the CPU.

If you have a chain communication topology and the Communications settings (Page 82) enable multi-threading, be aware of the risk of communication disruption with this operation.

Note

Fail-Safe devices

If a fail-safe CPU is protected, you must enter the safety password (Page 308) in the "CPU Password" column to restore a backup file to a fail-safe device.

You must confirm an additional prompt and reselect your device if the program in an F-CPU is a safety program (Page 314).

The SIMATIC Automation Tool places "Restore from backup" requests for fail-safe devices in the F-CPU safety-relevant operation queue. The SIMATIC Automation Tool uses only single-thread sequential processing for the safety-relevant operation queue.

The destination device for a safety program must be a fail-safe CPU.

⚠ **WARNING**

Verify that the device is not actively running a process before restoring a device from a backup file

Restoring a CPU causes the CPU to go to STOP mode, which could affect the operation of an online process or machine. Restoring an HMI disrupts operator actions from operator screens, which could also cause process disruption.

Unexpected operation of a process or machine could result in death or injury to personnel and/or property damage.

Verify that your process can handle a disruption before restoring a device from a backup file.

Note

Transfer Settings for an HMI device (SIMATIC HMI Panel)

To communicate with an HMI, you must set the Transfer Settings on the HMI device to PN/IE or Ethernet. You must set the Communications settings (Page 82) for the HMI Transfer Channel to the same setting.

To restore selected devices from a backup file, follow these steps:

1. Click the "Restore from Backup" tab in the Device table.
2. Select one or more devices to include in the operation. You can use the Devices check box at the top of the Device table to select or deselect all devices. Alternatively, you can use the right-click shortcut menu or the "View > Select" menu command to access the "Select Row(s)" and "Deselect Row(s)" commands.

3. For each device, select a backup file name from the "Backup File" drop-down list. The drop-down list shows the names of the .s7pbkp files that exist in the backup and restore folder (Page 85).
   You can also use the browse ... button and navigate to the folder on your programming device that contains backup files. Select a file to add it to the drop-down list. If the file that you selected has the same name as an existing file, the SIMATIC Automation Tool adds a number to the new file name to make the names unique.
   For valid files, a tooltip displays the path and filename. You also see a green check mark by the file name.
   For invalid files, the tooltip displays the file error, which the Event Log also displays. You see a red X by the file name.
4. Enter passwords (Page 308), if used, in the "CPU Password" and "Password in Backup File" columns. "Restore from backup" is a safety-relevant operation. You must enter the safety password (Page 308) for a protected fail-safe device.
5. Select the "Operations > Backup/Restore > Restore Device" menu command to start the operation. Alternatively, click the Backup/Restore toolbar button and select "Restore Device" from the button drop-down menu.

Backup file validation

Before starting the restore operation, the SIMATIC Automation Tool performs limited data checks on the backup file data:

- The file extension name and header data are validated.
- You cannot restore from a backup file that contains a safety program when the target device is not a fail-safe CPU.

If a backup file is not valid, the SIMATIC Automation Tool displays a red "X" in the "Backup File" field. Additional error information is available in a tooltip when you hover over the cell.

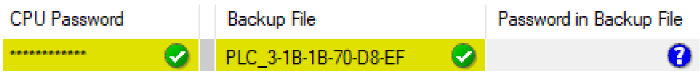
Password handling after the restore operation

If a CPU is protected, then you must supply a CPU password for the restore from backup operation to complete successfully.

After you restore a backup file to a CPU, the new file might include a CPU password. The CPU password you restored might be different from the previous CPU password, if the CPU had a CPU password. You must therefore enter a second CPU password in the "Password in Backup File" column. The second CPU password becomes the CPU password after the restore operation completes.

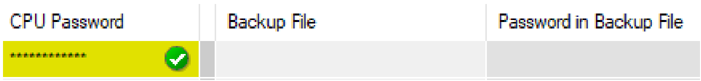
After a successful Restore operation, the SIMATIC Automation Tool automatically copies the second CPU password (the "Password in Backup File" that you entered) to the CPU password field and attempts a connection using the new CPU password. The SIMATIC Automation Tool then clears the "Backup File" and "Password in Backup File" fields.

Before you restore a backup file to a CPU:



CPU Password	Backup File	Password in Backup File
************* ✓	PLC_3-1B-1B-70-D8-EF ✓	?

After you restore a backup file to a CPU:



CPU Password	Backup File	Password in Backup File
************ ✓		

If the CPU password that you entered in the "Password in Backup File" column is incorrect and is not actually a CPU password configured in the restored CPU data, then the Event Log shows a warning after the operation completes. In this case the CPU password shows a red ‘X’ icon to indicate an invalid CPU password.

F-signature validation

A TIA Portal project that contains a safety program has an F-signature. The SIMATIC Automation Tool uses the F-signature to verify the data in a program file, which provides an additional level of security for safety programs. After a Restore from backup operation, the SIMATIC Automation Tool compares the F-Signature in the project file to the F-Signature that is now in the CPU device program.

The Event Log reports a successful comparison as: "Result of CRC comparison, online and offline collective F-signatures match."

The SIMATIC Automation Tool reports an unsuccessful comparison in the Event Log as: "Result of CRC comparison, online and offline collective F-signatures do not match" In the event of an unsuccessful comparison, reset the device to factory defaults (Page 36) and repeat the program update. Do not attempt to go to RUN mode if you are not sure that you have loaded the correct safety program.

Effect of restoring a CPU program on IP address

Before the restore operation, the STEP 7 project in the CPU has a setting in the device configuration for how the IP address is set (Page 17). In short, the setting is one of the following:

- Set IP address in the project (or equivalent)
- IP address is set directly at the device (or equivalent)

The program that you are restoring might have a different setting than the device configuration of the project that is currently in the CPU. In addition, a user might have updated the device IP address (Page 32) with the SIMATIC Automation Tool. In that case, the user-updated device IP address is currently in the CPU.

The IP address of the device after you restore a program and after a subsequent CPU power cycle depends on the setting in the new program and the current IP address. The following table describes various possibilities for the IP address of the device.

Device configuration in backup file	**IP address after restoring from backup**	**IP address after CPU power cycle after restoring backup**
Set IP address in project	IP address is the IP address from the newly-restored program.	IP address is the IP address from the newly-restored program.
IP address is set directly at the device	IP address is the IP address of the device before you restored the program.	IP address is the IP address of the device before you restored the program.

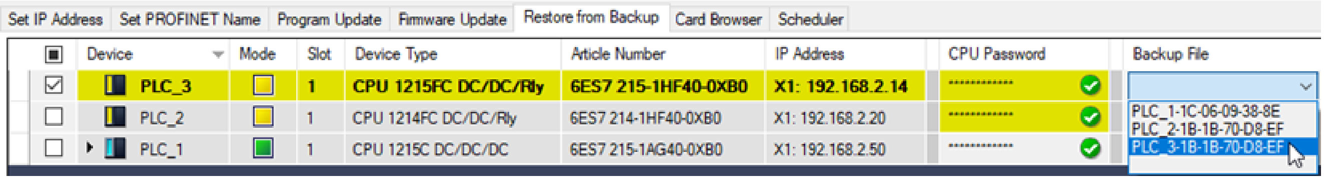
IP address change when device is behind a NAT router

If you are restoring a backup file to a device that is behind a NAT router, note that a change in the IP address makes the device unreachable. The SIMATIC Automation Tool displays the device in italics to indicate that it could not find it on the network. The device is also in blue because it is behind a router.

In such a situation, you must update your NAT router configuration. You must then insert the device (Page 25) into the SIMATIC Automation Tool using the new IP address. After you insert the device at the new IP address, delete the entry in the Device table at the old IP address.

Restore from backup example

This example shows one selected device and the selection of one backup file for the "Backup File" field. For multiple devices, you would select a backup file to restore for each device:



▣	Device	Mode	Slot	Device Type	Article Number	IP Address	CPU Password	Backup File
☑	**PLC_3**	🟨	**1**	**CPU 1215FC DC/DC/Rly**	**6ES7 215-1HF40-0XB0**	**X1: 192.168.2.14**	************* ✓	
☐	PLC_2	🟨	1	CPU 1214FC DC/DC/Rly	6ES7 214-1HF40-0XB0	X1: 192.168.2.20	************* ✓	
☐	PLC_1	🟩	1	CPU 1215C DC/DC/DC	6ES7 215-1AG40-0XB0	X1: 192.168.2.50	************* ✓	

After executing the "Restore Device" operation, the Event Log shows the results of the operation.

Managing project and device files 5

5.1 Creating, saving, and opening .sat project files

Creating a new project file

To create a new project file with an empty Device table, Use the "File > New" menu command.

Saving a project file

Use the "File > Save" or "File > Save as" menu command or click the Save button 💾 to store your project in an encrypted .sat project file. This project file contains your Device table information. The project file does not save device operating mode, selection state, or confirmation of selection state data.

If you have not set a password for your project, the SIMATIC Automation Tool prompts you to set a password (Page 66).

Opening a project file

After you save a SIMATIC Automation Tool project, you can use the "File > Open" menu command or Open button 📂 to restore this project's Device table information. You can use the Refresh command to read the operating mode states.

Guidelines for saving and opening project files

Note the following guidelines for saving and opening project files:

- The Projects settings (Page 84) determine the folder for saving .sat project files.
- You must provide a valid password to save a SIMATIC Automation Tool .sat project file.
- You must enter the correct password to reopen an existing SIMATIC Automation Tool .sat project file.

Project file compatibility with previous versions

You can open a V3.1 or later project file with SIMATIC Automation Tool V4.0 with no loss of data.

SIMATIC Automation Tool V3.1 and later versions support safety-relevant operations that were prohibited in V3.0 and earlier versions. V3.0 and earlier .sat project files do not contain safety data in the project file. When you open a V2.x - V3.0 project file, the SIMATIC Automation Tool notifies you that it must perform a network scan. After the scan completes, the SIMATIC Automation Tool opens the project file and applies the project file data to the devices that the network scan discovered.

Opening V1.x project files is not possible.

You cannot open a V4.0 project with an earlier version of the SIMATIC Automation Tool.

Note

Transferring files from one programming device to another

Create an archive file (Page 66)to restore SIMATIC Automation Tool project and related files from one programming device to another.

5.2 Creating or changing a project password

You must protect your project with a password. Siemens recommends that you protect your SIMATIC Automation Tool project with a strong password. Strong passwords follow these rules:

- Are at least 12 characters in length
- Mix upper and lower case letters, numerals, and special characters
- Are not words that can be found in a dictionary
- Are not names or identifiers that can be derived from personal information

Keep the password secret and change it frequently.

You must enter the project password to open a .sat file.

Setting or changing the project password

Select the **File** > **Project Password** menu command to set or change your project password. You must enter the password twice to avoid keyboard entry errors.

Password validation

When you create a project password, the SIMATIC Automation Tool indicates that it is a weak password if it does not meet the criteria for a strong password.

5.3 Creating and opening project archives

An archive file contains the .sat project file and files you have selected in the following categories:

- Firmware update tab, New Firmware Version
- Firmware update tab, SNMP Profile
- Program update tab, Program Update Folder
- Restore from Backup tab, Backup File

The SIMATIC Automation Tool includes only valid, unique files that you have selected in these categories. It does not include all of the files in the folders defined in the Options Settings (Page 79).

Creating an archive

To create an archive file, follow these steps:

1. Select the File > Create Archive menu command. The SIMATIC Automation Tool opens the Windows File explorer to the default location for Projects (Page 84).
2. Provide a file name and save the file in the default folder or to any other folder on you programming device.
3. Provide a password for the archive file.

The SIMATIC Automation Tool saves the compressed archive file with extension .satz.

Opening an archive

To open an archive file, follow these steps:

1. Select the File > Open Archive menu command.
2. Select an archive file from your programming device.
3. If you agree, click "Yes" to acknowledge the prompt that informs you that the files in the archive will overwrite files of the same name on your programming device. If not, click "No".
4. Enter the password for the archive file.

The SIMATIC Automation Tool fills the Device table with the devices in the archived .sat project file and restores the archived files selected for these categories:

- Firmware update (Page 40)
- SNMP Profiles (Page 40)
- Program update (Page 48)
- Backup File (Page 59)

5.4 Exporting information from the SIMATIC Automation Tool

The SIMATIC Automation Tool provides menu commands to export the following types of information:

- Device information (Page 68) from the Device table
- Device diagnostics (Page 72)
- PC Data (Page 73):
  - Information about the programming device you are using to run the SIMATIC Automation Tool
  - Information about Siemens applications that are installed on your programming device

For device information and device diagnostics, you can choose whether to export the data for all devices or for selected devices. The SIMATIC Automation Tool saves the data in .csv format.

Unlike previous releases of the SIMATIC Automation Tool, you no longer have an import command to fill the Device table from an exported file. Due to the flexibility and customization of the exported device information file, you can use copy and paste to restore exported device information to the Device table.

The menu commands for exporting are as follows:

- **File > Export > Device Information > All Devices:** Saves the entire Device table to a .csv file.
- **File > Export > Device Information > All Selected Devices:** Saves the Device table for the selected devices to a .csv file
- **File > Export > Device Diagnostics > All Devices:** Saves the diagnostic buffer data for all eligible devices to a .csv file. Currently, CPU devices support the export of device diagnostics.
- **File > Export > Device Diagnostics > All Selected Devices:** Saves the diagnostic buffer data for the selected eligible devices to a .csv file.
- **File > Export > PC Data > Local PC:** Saves information about the operating system, processor, memory, installed Siemens applications, and other system information to a .zip file.

5.4.1 Exporting device information

When you export device information, the SIMATIC Automation Tool creates and saves a .csv file that represents the current data in your Device table. The SIMATIC Automation Tool does not rescan the network or refresh network. The export contains the current data in the Device table.

To export the device information, select one of the following menu commands:

- **File > Export > Device Information > All Devices**
- **File > Export > Device Information > All Selected Devices**

Rows in the .csv file correspond to devices, folders of local modules, and folders of distributed I/O if these folders exist. Within each type of row, column headers guide you to the type of data in the row.

The Export settings (Page 88) provide the file path for exporting the Device table.

For security reasons, the SIMATIC Automation Tool does not export CPU passwords.

You can open and edit the .csv file in Microsoft Excel or in a text editor. To copy the edited file contents into the SIMATIC Automation Tool, you must maintain the Device table format. The Device table in the SIMATIC Automation Tool contains communication configuration data. If you put incorrect information in the cells of a Device table, device operations can fail. Correct the device data and try the operation again.

Format of the device information .csv file

The format of the device information .csv file corresponds to the structure of the Device Table. The device information file uses the following structure:

- **Header row**: The first row is a header row that contains the column header names from the Device table:
  - Device
  - Slot
  - Device Type
  - Article Number
  - Serial Number
  - Firmware Version
  - Hardware
  - MAC Address
  - IP Address
  - Subnet
  - Gateway
  - PROFINET Name
  - PROFINET Converted Name
- **Device row**: Each device in the Device table appears in a row of the .csv file with its device data in each column. All SIMATIC Automation Tool Device table columns are in the exported .csv file. If you hide columns in the SIMATIC Automation Tool, the data corresponding to the hidden columns still appears in the .csv file. Hiding columns in the SIMATIC Automation Tool has no effect on the .csv file contents.
- **Local modules folder row:** If a device has local modules, the .csv includes a Local Module row with the following column headers:
  - Device
  - Slot
  - Device Type
  - Article Number
  - Serial Number
  - Firmware Version
  - Hardware
- **Local modules row:** Each local module appears in the .csv file with its module data in each column. Module data is available in the file when the SIMATIC Automation Tool can read the module data and the data is visible in the Device table.
- **Distributed I/O header:** A row with the label "Distributed I/O" indicates that the rows that follow represent distributed I/O modules.

- **Distributed I/O system folder header:** A row for a distributed I/O system contains the name of the distributed I/O system and the following column header names:
  - Device
  - Slot
  - Device Type
  - Article Number
  - Serial Number
  - Firmware Version
  - Hardware
- **Distributed I/O device row header:** Each device in the distributed I/O system appears on a row with the device data in the corresponding columns. Distributed I/O device data is available in the file when the SIMATIC Automation Tool can read the device data and the data is visible in the Device table.
- **Distributed I/O module row:** Each module for each distributed I/O device appears on a row with the device data in the corresponding columns. Module data is available in the file when the SIMATIC Automation Tool can read the module data and the data is visible in the Device table.

A device information export does not include Data Logs or Recipes that are in CPUs.

Note

CPU protection and passwords

For protected CPUs in the Device table, the SIMATIC Automation Tool only exports the local module and distributed I/O information if the Device table includes a CPU password (Page 308) that is at the read access level or higher.

Using Copy and Paste as an alternative to exporting

Note

Copying between the Device table and Microsoft Excel

You can copy and paste selected cells from the Device table to a Microsoft Excel file:

1. Expand all rows in the Device table with the "Edit > Expand > All Devices" menu command.
2. Select Ctrl+A to highlight all rows and columns or drag to select a range of cells.
3. Select Ctrl+C to copy the Device table data to the Windows clipboard.
4. Select Ctrl+V to paste the clipboard data into a Microsoft Excel worksheet.

The Export function allows you to copy all or selected devices in a readable format. Copying from the Device table is not typically necessary but is possible. You can, for example, use copy and paste to modify user-entered device information.

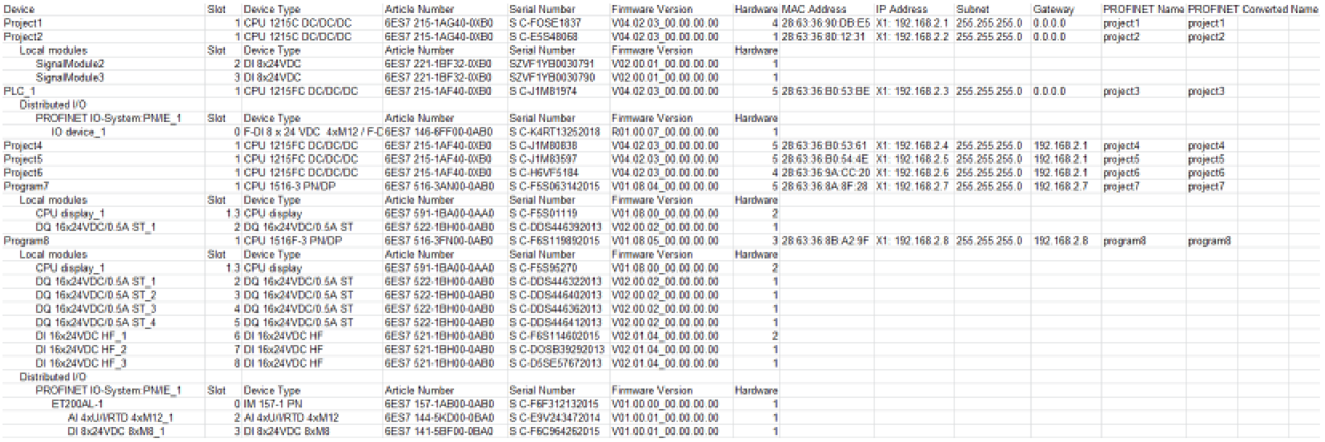
Device information export example

When you open an exported device information file in Microsoft Excel, the columns are fixed width.

To expand the column widths to be readable follow these steps:

1. Click the small triangle in the top left corner between Row 1 and Column A to select the entire worksheet.
2. On the column line of Microsoft Excel, double-click between any two columns to expand the column widths. For example, double-click the column divider between A and B.

The following image shows an export of the device information opened in Microsoft Excel with the column widths expanded:



Device	Slot	Device Type	Article Number	Serial Number	Firmware Version	Hardware	MAC Address	IP Address	Subnet	Gateway	PROFINET Name	PROFINET Converted Name
Project1	1	CPU 1215C DC/DC/DC	6ES7 215-1AG40-0XB0	S C-FOSE1837	V04.02.03_00.00.00.00	4	28:63:36:90:DB:E5	X1: 192.168.2.1	255.255.255.0	0.0.0.0	project1	project1
Project2	1	CPU 1215C DC/DC/DC	6ES7 215-1AG40-0XB0	S C-E5S48068	V04.02.03_00.00.00.00	1	28:63:36:80:12:31	X1: 192.168.2.2	255.255.255.0	0.0.0.0	project2	project2
Local modules	Slot	Device Type	Article Number	Serial Number	Firmware Version	Hardware						
SignalModule2	2	DI 8x24VDC	6ES7 221-1BF32-0XB0	SZVF1YB0030791	V02.00.01_00.00.00.00	1						
SignalModule3	3	DI 8x24VDC	6ES7 221-1BF32-0XB0	SZVF1YB0030790	V02.00.01_00.00.00.00	1						
PLC_1	1	CPU 1215FC DC/DC/DC	6ES7 215-1AF40-0XB0	S C-J1MB1974	V04.02.03_00.00.00.00	5	28:63:36:B0:53:BE	X1: 192.168.2.3	255.255.255.0	0.0.0.0	project3	project3
Distributed I/O												
PROFINET IO-System:PN/IE_1	Slot	Device Type	Article Number	Serial Number	Firmware Version	Hardware						
IO device_1	0	F-DI 8 x 24 VDC 4xM12 / F-C	6ES7 146-6FF00-0AB0	S C-K4RT13252018	R01.00.07_00.00.00.00	1						
Project4	1	CPU 1215FC DC/DC/DC	6ES7 215-1AF40-0XB0	S C-J1MB0838	V04.02.03_00.00.00.00	5	28:63:36:B0:53:61	X1: 192.168.2.4	255.255.255.0	192.168.2.1	project4	project4
Project5	1	CPU 1215FC DC/DC/DC	6ES7 215-1AF40-0XB0	S C-J1M83597	V04.02.03_00.00.00.00	5	28:63:36:B0:54:4E	X1: 192.168.2.5	255.255.255.0	192.168.2.1	project5	project5
Project6	1	CPU 1215FC DC/DC/DC	6ES7 215-1AF40-0XB0	S C-H6VF5184	V04.02.03_00.00.00.00	4	28:63:36:9A:CC:20	X1: 192.168.2.6	255.255.255.0	192.168.2.1	project6	project6
Program7	1	CPU 1516-3 PN/DP	6ES7 516-3AN00-0AB0	S C-F5S063142015	V01.08.04_00.00.00.00	5	28:63:36:8A:8F:28	X1: 192.168.2.7	255.255.255.0	192.168.2.7	project7	project7
Local modules	Slot	Device Type	Article Number	Serial Number	Firmware Version	Hardware						
CPU display_1	1.3	CPU display	6ES7 591-1BA00-0AA0	S C-F5S01119	V01.08.00_00.00.00.00	2						
DQ 16x24VDC/0.5A ST_1	2	DQ 16x24VDC/0.5A ST	6ES7 522-1BH00-0AB0	S C-DDS446392013	V02.00.02_00.00.00.00	1						
Program8	1	CPU 1516F-3 PN/DP	6ES7 516-3FN00-0AB0	S C-F6S119892015	V01.08.05_00.00.00.00	3	28:63:36:8B:A2:9F	X1: 192.168.2.8	255.255.255.0	192.168.2.8	program8	program8
Local modules	Slot	Device Type	Article Number	Serial Number	Firmware Version	Hardware						
CPU display_1	1.3	CPU display	6ES7 591-1BA00-0AA0	S C-F5S95270	V01.08.00_00.00.00.00	2						
DQ 16x24VDC/0.5A ST_1	2	DQ 16x24VDC/0.5A ST	6ES7 522-1BH00-0AB0	S C-DDS446322013	V02.00.02_00.00.00.00	1						
DQ 16x24VDC/0.5A ST_2	3	DQ 16x24VDC/0.5A ST	6ES7 522-1BH00-0AB0	S C-DDS446402013	V02.00.02_00.00.00.00	1						
DQ 16x24VDC/0.5A ST_3	4	DQ 16x24VDC/0.5A ST	6ES7 522-1BH00-0AB0	S C-DDS446362013	V02.00.02_00.00.00.00	1						
DQ 16x24VDC/0.5A ST_4	5	DQ 16x24VDC/0.5A ST	6ES7 522-1BH00-0AB0	S C-DDS446412013	V02.00.02_00.00.00.00	1						
DI 16x24VDC HF_1	6	DI 16x24VDC HF	6ES7 521-1BH00-0AB0	S C-F6S114602015	V02.01.04_00.00.00.00	2						
DI 16x24VDC HF_2	7	DI 16x24VDC HF	6ES7 521-1BH00-0AB0	S C-DOSB39292013	V02.01.04_00.00.00.00	1						
DI 16x24VDC HF_3	8	DI 16x24VDC HF	6ES7 521-1BH00-0AB0	S C-D5SE57672013	V02.01.04_00.00.00.00	1						
Distributed I/O												
PROFINET IO-System:PN/IE_1	Slot	Device Type	Article Number	Serial Number	Firmware Version	Hardware						
ET200AL-1	0	IM 157-1 PN	6ES7 157-1AB00-0AB0	S C-F6F312132015	V01.00.00_00.00.00.00	1						
AI 4xU/I/RTD 4xM12_1	2	AI 4xU/I/RTD 4xM12	6ES7 144-5KD00-0BA0	S C-E9V243472014	V01.00.01_00.00.00.00	1						
DI 8x24VDC 8xM8_1	3	DI 8x24VDC 8xM8	6ES7 141-5BF00-0BA0	S C-F6C964262015	V01.00.01_00.00.00.00	1						

You can edit data in the Microsoft Excel file. You can also zoom to make the content more readable. If you add or delete rows, be sure to maintain the format and hierarchy. When you copy cells to the SIMATIC Automation Tool Device table, follow these steps to display all devices and files:

1. Copy and paste the cell range that you need from Microsoft Excel to the Device table.
2. In the SIMATIC Automation Tool, re-enter passwords for protected CPUs.
3. Select all devices in the Device table.
4. Refresh the Device table using the **"Operations > Scan Network > Refresh Status of All Selected Devices"** menu command. You can also run this command from the toolbar button.

Note

Displaying .csv files in Microsoft Notepad or Microsoft Excel

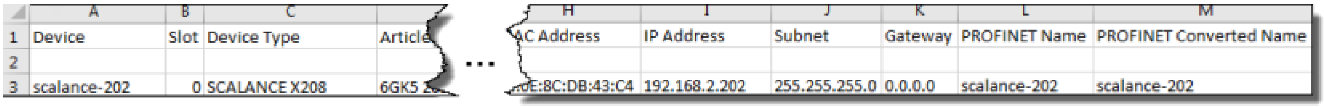
Due to differing character sets, Microsoft Notepad and Microsoft Excel might not correctly display characters in your .csv file. Open the file using the Unicode (UTF-8) character set to see characters in your language that might not be visible otherwise. Regardless of the appearance of header texts, you can export project files and copy content from the exported files to the SIMATIC Automation Tool.

IP address conflicts between the network and exported .csv file

You can open an exported .csv file in Microsoft Excel. You can then copy and paste device data from Microsoft Excel into the SIMATIC Automation Tool Device table. The copied data could include a device with an IP address that is the same IP address of a different device that is currently on your network. The Device table, however, shows the device that is **currently** on your network at that IP address.

Consider the following scenario:

- Currently on your network, you have an S7-1200 CPU named "PLC_1" at IP address X1: 192.168.2.202.
- You copy device data from a previously-exported device information .csv file that includes the following lines:



	A	B	C		…	H	I	J	K	L	M
1	Device	Slot	Device Type	Article		C Address	IP Address	Subnet	Gateway	PROFINET Name	PROFINET Converted Name
2											
3	scalance-202	0	SCALANCE X208	6GK5 2		E:8C:DB:43:C4	192.168.2.202	255.255.255.0	0.0.0.0	scalance-202	scalance-202

- You paste the copied data into the SIMATIC Automation Tool Device table.
- The Event Log reports that the operation completed successfully for device SCALANCE-202.
- The Device table shows "PLC_1" at IP address X1: 192.168.2.202, not "SCALANCE-202". The Device table includes all fields corresponding to PLC_1.

Result: Copying and pasting from a .csv file does not override device data for a device currently on the network when an IP address conflict exists.

5.4.2 Exporting device diagnostics

When you export device diagnostics, the SIMATIC Automation Tool creates and saves a .csv file that contains CPU diagnostic buffers.

To export CPU device diagnostics, select one of the following menu commands:

- **File > Export > Device Information > All Devices**
- **File > Export > Device Information > All Selected Devices**

Format of the device diagnostics .csv file

The diagnostic buffers for the exported devices are in a single .csv file. Header rows separate the end of a diagnostic buffer of one device from the beginning of the diagnostic buffer for another device.

Each device begins with a header row with the following column headers:

- Device
- Date
- Time
- Event Type

- Event
- Event Description

The diagnostic buffer events occupy the next rows in the file and contain all of the events in the current diagnostic buffer for the device.

The next device begins with a new header row. Diagnostic buffer events for that device follow in successive rows. The file continues through all devices or all selected devices, depending on your selection.

Note

Exporting device diagnostics from a large Device table can take a long time. The SIMATIC Automation Tool displays a progress bar during the export. You can cancel an export operation that is in progress.

5.4.3 Exporting PC data

The SIMATIC Automation Tool provides the capability to export data about your PC and files that can be helpful in troubleshooting problems. When you export PC data, the SIMATIC Automation Tool creates and saves a .zip file with detailed information about the programming device on which you run the SIMATIC Automation Tool. You can provide this information to Siemens upon request in the event of problems.

To export PC data, select the "File > Export > PC Data > Local PC" menu command. The SIMATIC Automation Tool saves the PC data .zip file in the Export folder (Page 88).

Settings, menus, and toolbars 6

6.1 Main menu

The SIMATIC Automation Tool provides the following menus for device operations.

- File (Page 75)
- Edit (Page 76)
- View menu (Page 77)
- Operations (Page 77)
- Options (Page 79)
- Tools (Page 92)
- Help (Page 92)

If you press the Alt key, the underlined letter indicates the Alt key you can use to activate a menu or sub-menu command.

Additionally, you can activate some of the menu commands with shortcut key combinations (Page 314).

6.1.1 File menu

Tool icon	Menu command	Description
📄	**New**	Create a new SIMATIC Automation Tool project.
📂	**Open**	Display an "Open" dialog where you can browse to a folder, select an .sat project file, and provide a password to open a protected project file. The "Open" dialog displays the projects folder (Page 84), but you can browse to any location for a project.
💾	**Save**	Save (Page 65) the Device table data in an .sat file. If there is no filename, then this operation uses the "Save As" command. The projects folder (Page 84) is the default folder for saving projects.
	Save As	Save the Device table data in an .sat file. You can browse to a folder, provide an .sat project filename, and assign a password to protect the project file.
	Project Password	Set or change the password for the project (Page 66)
	Create Archive	Create an archive file (Page 66) (requires an advanced license)
	Open Archive	Open an archive file (Page 66) (requires an advanced license)
	Export (Page 68)	
	• **Device Information** – **All Selected Devices**	Save the Device table for the selected devices, including local modules and distributed I/O, to a .csv file.

Tool icon	Menu command	Description
	- **Device Information** – **All Devices**	Save the Device table for all devices, including local modules and distributed I/O, to a .csv file.
	- **Device Diagnostics** – **All Selected Devices**	Save the diagnostic buffer data for the selected devices to a .csv file.
	- **Device Diagnostics** – **All Devices**	Save the diagnostic buffer data for all devices to a .csv file.
	- **PC Data** – **Local PC**	Save PC data for the local programming device to a .zip file.
	Exit	Close the application. If the project has changed since the last save operation, the SIMATIC Automation Tool gives you an opportunity to save the project.

6.1.2 Edit menu

Tool icon	Menu command	Description
✂	**Cut**	Cut the selected data and copy this data to the clipboard. Clipboard entries are compatible with Microsoft Excel, so data can be shared between the two applications. Read-only cells are not deleted.
⎘	**Copy**	Copy the selected data to the clipboard.
📋	**Paste**	Paste the data contained in the clipboard to selected field(s) in the SIMATIC Automation Tool. You cannot paste into read-only cells.
	Insert	
	- **Device**	Insert a device (Page 25) row at the selected row and push the following device rows downward. You can use this command to quickly add a device to the Device table.
	- **Multiple Devices**	Insert multiple devices (Page 25) by IP addresses or MAC addresses.
	Delete	
	- **All Selected Devices**	Delete contents of all selected devices.
	- **All Devices**	Delete contents for all device rows.

6.1.3 View menu

Tool icon	Menu command	Description
	Select	
	• **Select Row(s)**	Select the Device table rows that have focus.
	• **Deselect Row(s)**	Deselect the Device table rows that have focus.
	• **Select All Devices**	Select all devices.
	• **Deselect All Devices**	Deselect all devices.
	Expand	
	• **All Selected Devices**	Expand all selected devices.
	• **All Devices**	Expand all rows for devices and modules.
	Collapse	
	• **All Selected Devices**	Collapse all selected devices.
	• **All Devices**	Collapse all rows for devices and modules.
	Columns	
	• **Restore Defaults**	Restore the Device table columns to defaults
	• **<Individual columns>**	Turn columns on and off from the Device table display
	Sort by	
	• **<Individual columns>**	Select one column for sorting. The SIMATIC Automation Tool then sorts the Device table on the selected column.
	• **Ascending**	Sort in ascending order on the selected column.
	• **Descending**	Sort in descending order on the selected column.
	Size Column to Fit	When you have selected a cell in the Device table, size that column to fit the widest entry in the column with no additional space.
	Size All Columns to Fit	Sizes all the columns in the Device table to the width of the widest entry for each column.
	Refresh	
	• **All Selected Devices**	Refresh all selected devices.
	• **All Devices**	Refresh all devices.

6.1.4 Operations menu

Tool icon	Menu command	Description
	Scan Network	Scan device network. (Page 21)
	RUN	Put selected CPUs in RUN mode. (Page 30)
	STOP	Put selected CPUs in STOP mode. (Page 30)

Tool icon	Menu command	Description	
⬇	**Update**		
	• **Set IP Address**	Update the CPU with the IP Address information for the selected device(s). (Page 32)	
	• **Set PROFINET Name**	Update the CPU with the PROFINET Name for the selected device(s). (Page 33)	
	• **Program Update**	Update the CPU program or HMI operating system and runtime software for the selected device(s). (Page 48)	
	• **Firmware Update**	Update the firmware for the selected device(s). (Page 40)	
	• **Two Step Firmware Update**	Update the firmware for the selected device(s) in two steps. (Page 47)	
		• Download Firmware	Download firmware update file to selected device(s).
		• Activate Firmware	Activate the downloaded firmware on the selected device(s).
💡	**Identify**	Flash the LEDs on devices or flash HMI screens. (Page 27) Use this feature to identify the physical location of a device.	
↔	**Reset**		
	• **Reset Communication Parameters**	Reset DCP communication parameters (Page 35) for devices that support the DCP reset command.	
	• **Memory Reset**	Perform a memory reset on selected devices. (Page 38)	
	• **Reset to Factory Defaults**	Reset selected devices to factory defaults. (Page 36)	
	• **Format Memory Card**	Format memory card in selected devices. (Page 39)	
ℹ	**Diagnostics**		
	• **Show Diagnostics**	Show diagnostic buffer for a selected CPU. (Page 28)	
	• **Read Service Data**	Read Service Data for selected devices. (Page 29)	
🕒	**Set time**	Set time in selected CPUs to your programming device time. (Page 31)	
➜	**Backup Device**		
	• **Full Backup**	Back up all data for all selected devices. (Page 57) The SIMATIC Automation Tool saves a backup file for each selected device.	
	• **HMI**		
		• **Recipes**	Back up HMI recipes. (Page 58)
		• **User Administration**	Back up HMI user administration data. (Page 59)
	Restore Device	Restore data from backup file(s) to the corresponding device(s). (Page 57)	

Tool icon	Menu command	Description
🗎▾	**Data Logs**	**Note:** Data log operations apply only to CPUs with SIMATIC memory cards.
	• **Read Data Logs**	Read selected Data Log files to your programming device. (Page 56)
	• **Delete Data Logs**	Delete selected Data Log files. (Page 56)

6.1.5 Options menu

6.1.5.1 Options menu

The Options menu contains the following menu commands:

Tool icon	Menu command	Description
	Settings	Open the Settings dialog where you can set default settings for the following categories: • General (Page 81) • Communications (Page 82) • Projects (Page 84) • Firmware Update (Page 84) • Program Update (Page 85) • Service Data (Page 85) • Backup / Restore (Page 85) • Recipes (Page 86) • Data Logs (Page 86) • Event Log (Page 87) • Export (Page 88) • SNMP Profiles (Page 88) • Scheduler (Page 91) For the device operations that include file operations, the settings define the default path name for the operation.
	Start Automation License Manager	Start the Automation License Manager with which you can license the SIMATIC Automation Tool.

6.1.5.2 SIMATIC Automation Tool pathnames

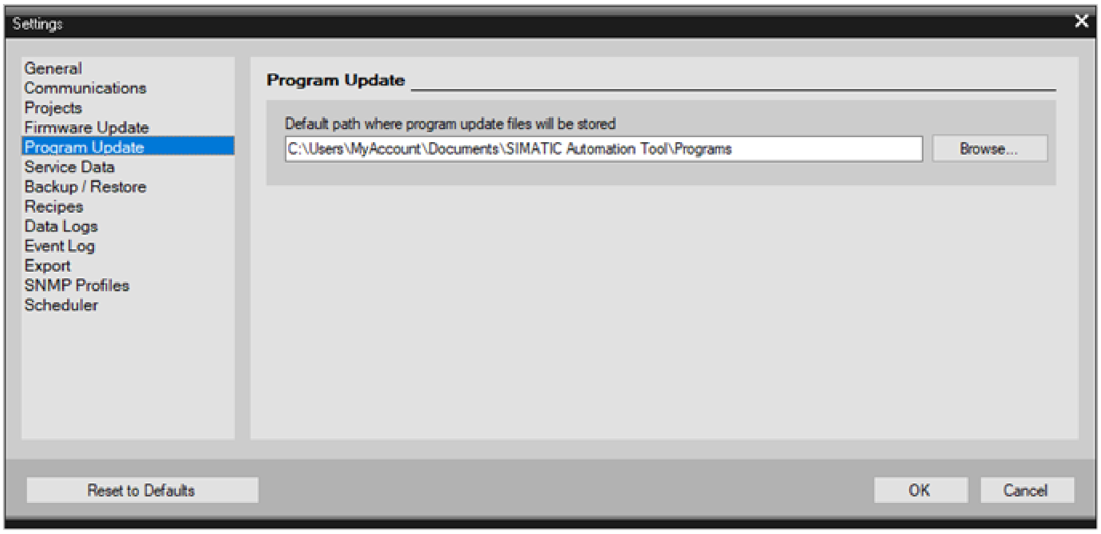
The pathname examples for the "Options > Settings" dialog show pathnames of folders in C:\Users\MyAccount\Documents\SIMATIC Automation Tool\, where "MyAccount" represents your user ID.

You use the "Options > Settings" menu command to provide default paths for the following file locations:

- Projects (Page 84)
- Firmware Update (Page 84)
- Program Update (Page 85)
- Service Data (Page 85)
- Backup/Restore (Page 85)
- Recipes (Page 86)
- Data Logs (Page 86)
- Event Log (Page 87)
- SNMP Profiles (Page 88)

Difference in displayed path and browsing to a folder

The Settings dialog displays paths, for example, as shown in the Program Update settings:



Program Update

Default path where program update files will be stored
C:\Users\MyAccount\Documents\SIMATIC Automation Tool\Programs

If you use the browse function from the Device table to locate a file such as a firmware update file or program update file, you do not see your user ID as a folder under "Users". You see the folder "My Documents" if you use Windows 7. You see the folder "Documents" if you use Windows 10. Expand this folder to find the "SIMATIC Automation Tool" folder.

Navigation from Windows 7 or Windows 10 also shows a different navigation path than the pathname in the Settings dialog.

Windows 7

From Windows Explorer, the Documents folder under the Libraries folder is also equivalent to the "My Documents" folder and the "MyAccount" folder.

Windows 10

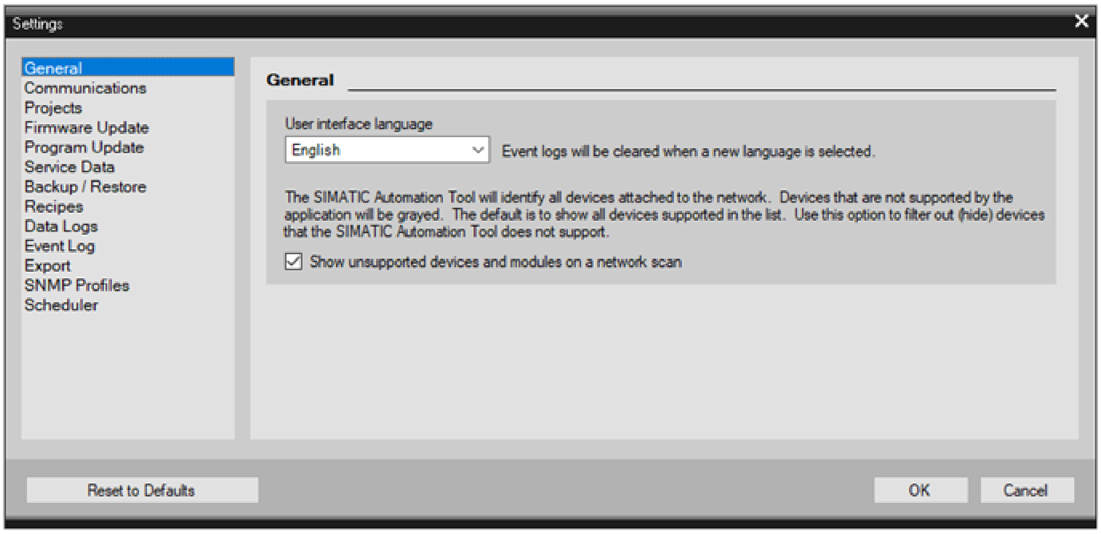
From File Explorer, the Documents folder under Quick Access is also equivalent to the "Documents" folder and the "MyAccount" folder.

6.1.5.3 General settings

You can select the user interface language: English, German, French, Spanish, Italian, or Chinese

Select the check box to show unsupported devices on a network scan (Page 21). The SIMATIC Automation Tool displays unsupported devices as disabled by using gray text in the Device table.

If you deselect the check box, the SIMATIC Automation Tool filters out unsupported devices from the Device table.



User interface language

English

Event logs will be cleared when a new language is selected.

The SIMATIC Automation Tool will identify all devices attached to the network. Devices that are not supported by the application will be grayed. The default is to show all devices supported in the list. Use this option to filter out (hide) devices that the SIMATIC Automation Tool does not support.

☑ Show unsupported devices and modules on a network scan

Note

Changing the user interface language change clears the Event Log

When you change the user interface language, the SIMATIC Automation Tool clears the Event Log.

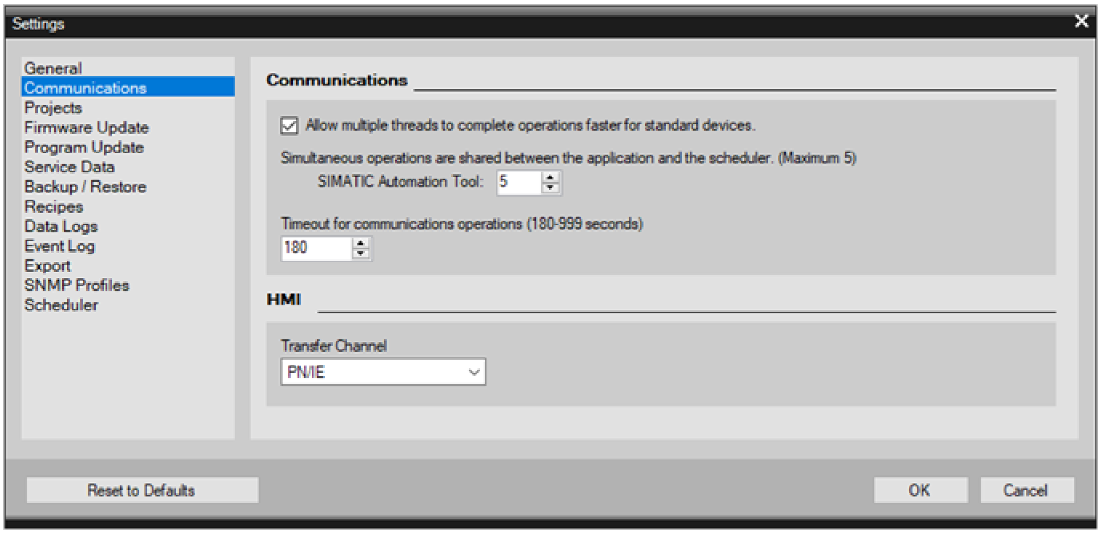
Note

Scan the network to update folder names

If you change the user interface language, folder names in the Device table remain in the previous user interface language. Scan the network to update folder names to the new user interface language.

6.1.5.4 Communications settings

You use the Communications options to set options related to multi-threading and HMI communication.



☑ Allow multiple threads to complete operations faster for standard devices.

Simultaneous operations are shared between the application and the scheduler. (Maximum 5)

SIMATIC Automation Tool: 5

Timeout for communications operations (180-999 seconds)

180

Transfer Channel

PN/IE

Using multiple threads for simultaneous operations

You can enhance SIMATIC Automation Tool performance by allowing operations on multiple devices to occur simultaneously on multiple threads. With a Basic license, you can specify the number of simultaneous operations for the SIMATIC Automation Tool up to a maximum of five simultaneous operations.

The multiple threads work in parallel to perform the simultaneous operations.

Considerations for assigning threads with the SIMATIC Automation Tool

Maximize the number of threads to allow the most simultaneous operations.

Reduce the number of threads if you send an operation command to multiple devices, but one device does not complete the operation. In this case, the Event Log shows that one device did not complete the operation. Other devices, however, execute the operation as expected. In this situation, reduce the number of simultaneous operations (threads). Close and restart the SIMATIC Automation Tool and try the group operation again.

Because threads execute in parallel, there is no guarantee of the order of the device operations. Communication speeds can be different for each device. The length of time to complete an operation can also vary for each device.

You can queue as many jobs of the same type as you want. For example, you can place 100 CPUs in STOP mode by selecting all 100 CPUs and clicking the STOP button. The SIMATIC Automation Tool displays a dialog with a progress bar until all 100 jobs are complete. The SIMATIC Automation Tool blocks other operations until all the STOP operations finish.

Multi-threading and changing IP addresses

If you are re-mapping many IP addresses in a multi-threaded operation, first set the IP addresses to unique temporary addresses. In a second operation, set them to the re-mapped addresses. This technique avoids creating duplicate IP addresses.

Avoid using a multi-threaded Program Update operation to re-map IP addresses for multiple devices. Re-map the addresses to temporary addresses first. Then update the programs with program update files that have the re-mapped IP addresses.

These techniques avoid creating duplicate IP addresses from a multi-threaded Set IP address operation (Page 32) or a multi-threaded Program Update operation (Page 48).

Disable threads if you use a chain network topology

If your network has a chain topology, disable this option to prevent one CPU from disrupting the communication to other devices. A chain topology, for example, would have chain connections from the programming device to CPU 1 to CPU 2 to CPU 3 to others.

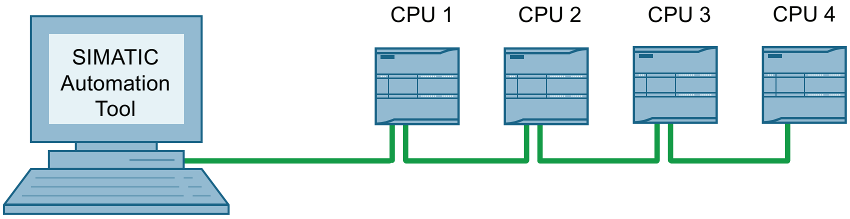


Figure 6-1 Example: Chain topology

With multi-threading, a thread that causes CPU 1 to restart disrupts an operation that is in progress for CPU 2 or any other CPUs in the chain. Note that you might also have a chain topology with CM or CP modules.

Threading strategies if you have an Advanced license

If you have an Advanced license (Page 101), the scheduler application (Page 107) can also perform operations on multiple threads. In this case, you can specify a number of simultaneous operations each for the SIMATIC Automation Tool and the scheduler up to a total of ten simultaneous operations.

If you have an Advanced license, consider advanced strategies for thread allocation (Page 112).

Timeout for communications operations

If you send an operation command to a device and the connection has a very slow data transfer rate, you might get a communication timeout error. If you have this problem, increase the timeout for communications operations.

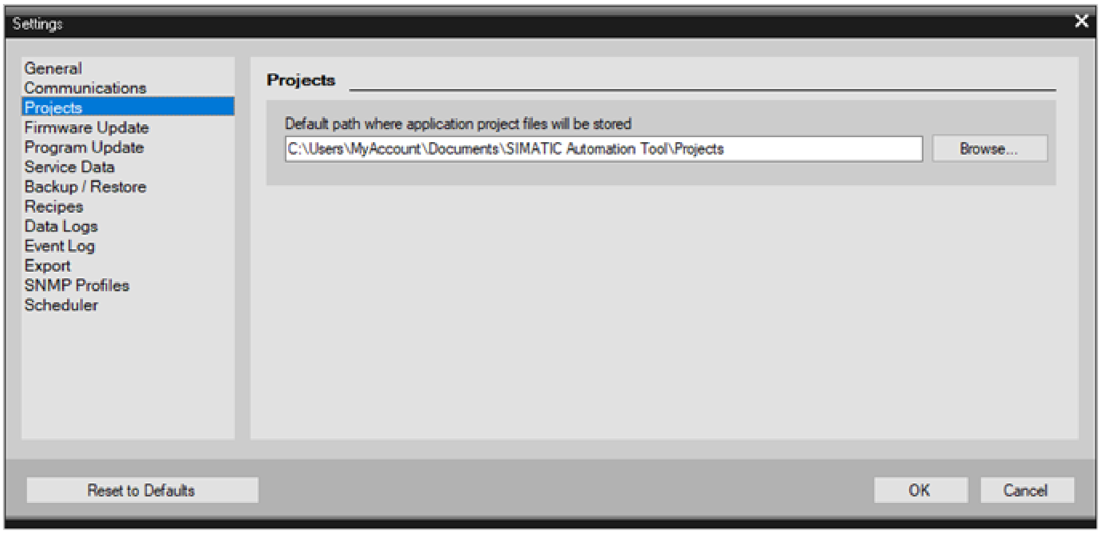
Setting the HMI transfer channel

The SIMATIC Automation Tool can communicate with HMI devices over PN/IE or Ethernet. Select the same setting in the SIMATIC Automation Tool Communications Settings as you select in the File Transfer Settings of your HMI device. Ethernet provides faster communication speeds.

6.1.5.5 Projects settings

You can accept the default path to save SIMATIC Automation Tool project data (Page 65) or assign a new path.

Your path might have a different drive letter and "MyAccount" represents the login name of the current user (Page 79).



Default path where application project files will be stored

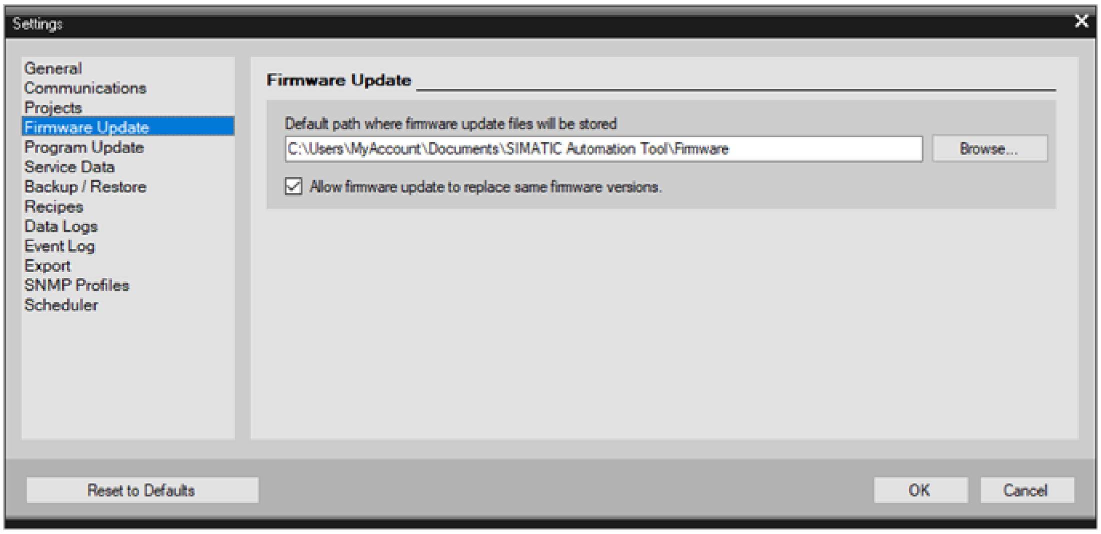
C:\Users\MyAccount\Documents\SIMATIC Automation Tool\Projects

6.1.5.6 Firmware Update settings

You can accept the default path to firmware update files (Page 40) or assign a different path.

Your path might have a different drive letter and "MyAccount" represents the login name of the current user (Page 79).

Click the check box to allow or disallow a firmware update with the same firmware version. Disallowing the replacement of an identical firmware version saves processing time by preventing unnecessary operations.



Default path where firmware update files will be stored

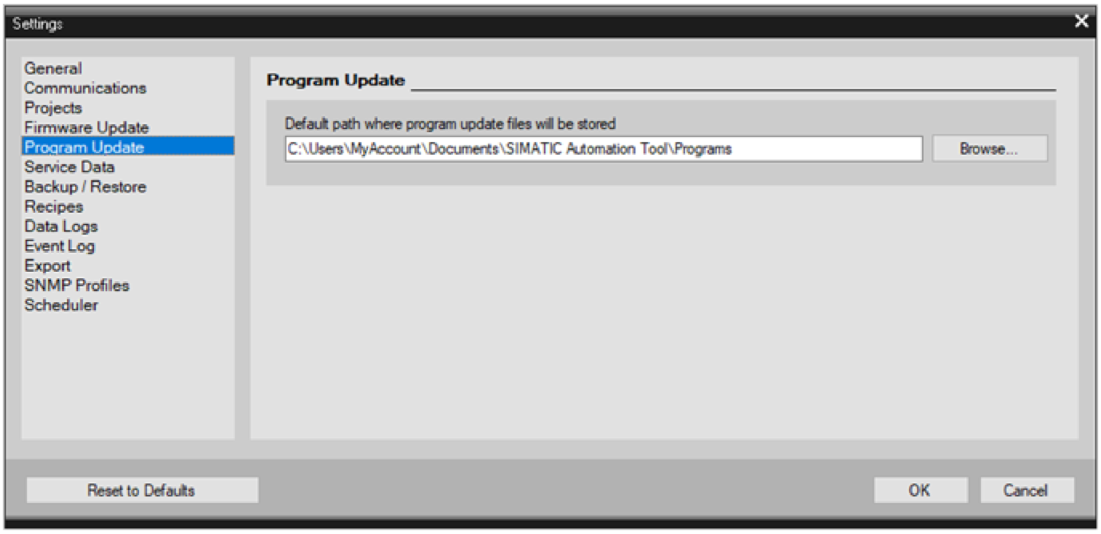
C:\Users\MyAccount\Documents\SIMATIC Automation Tool\Firmware

☑ Allow firmware update to replace same firmware versions.

6.1.5.7 Program Update settings

You can accept the default path to program files (Page 48) or assign a different path.

Your path might have a different drive letter and "MyAccount" represents the login name of the current user (Page 79).



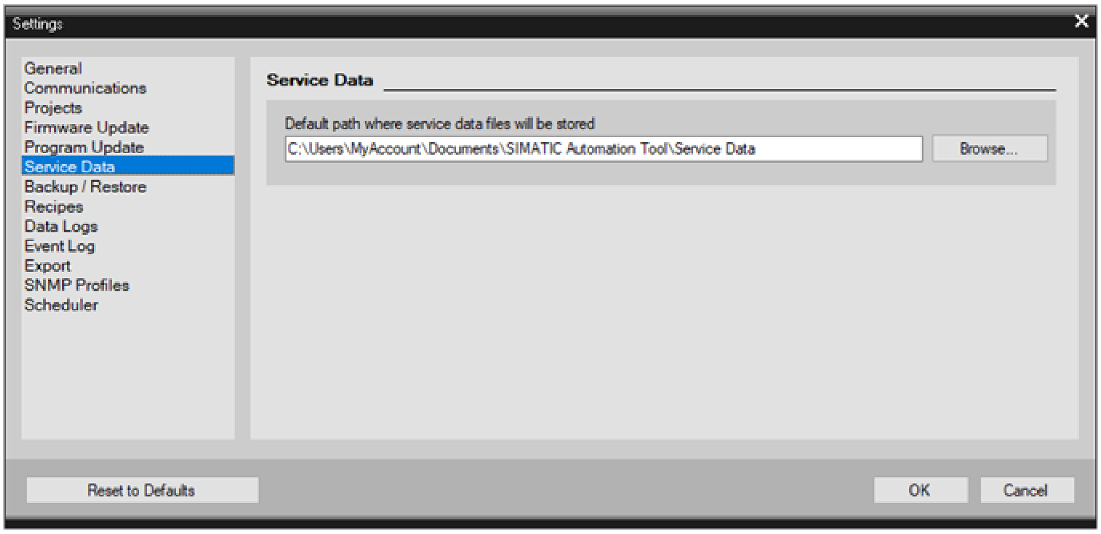
Default path where program update files will be stored

C:\Users\MyAccount\Documents\SIMATIC Automation Tool\Programs

6.1.5.8 Service Data settings

You can accept the default path to service data files (Page 29) or assign a different path.

Your path might have a different drive letter and "MyAccount" represents the login name of the current user (Page 79).



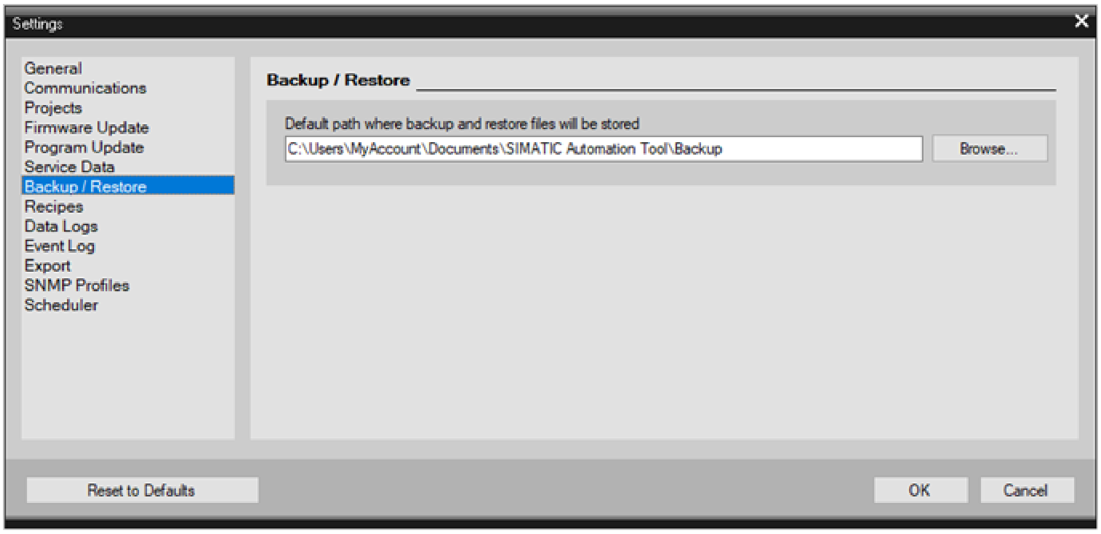
Default path where service data files will be stored

C:\Users\MyAccount\Documents\SIMATIC Automation Tool\Service Data

6.1.5.9 Backup/Restore settings

You can accept the default path to Backup and Restore files (Page 57) or assign a different path.

Your path might have a different drive letter and "MyAccount" represents the login name of the current user (Page 79).



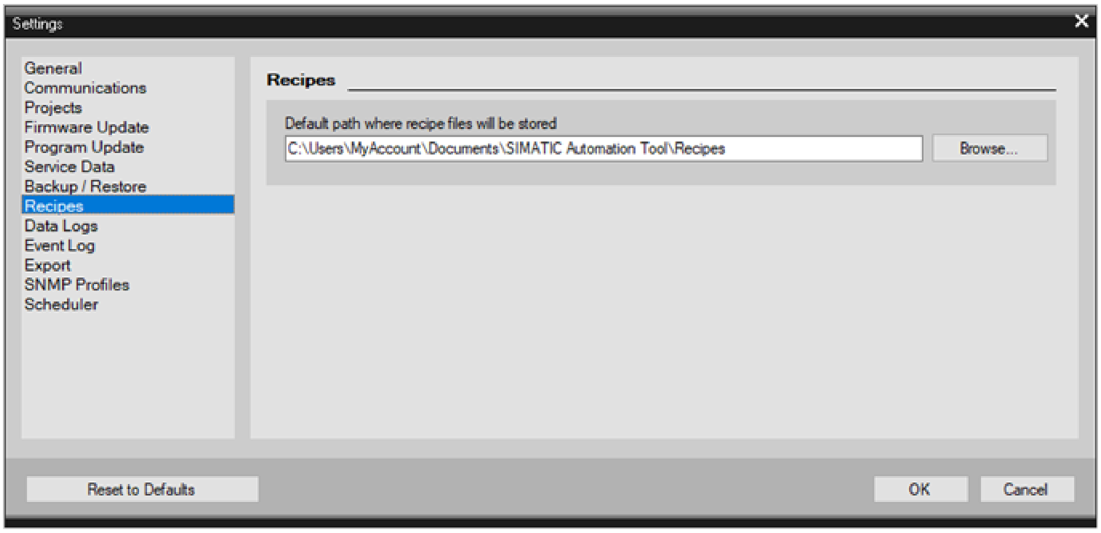
Default path where backup and restore files will be stored

C:\Users\MyAccount\Documents\SIMATIC Automation Tool\Backup

6.1.5.10 Recipes settings

You can accept the default path to recipe files (Page 103) or assign a different path.

Your path might have a different drive letter and "MyAccount" represents the login name of the current user (Page 79).



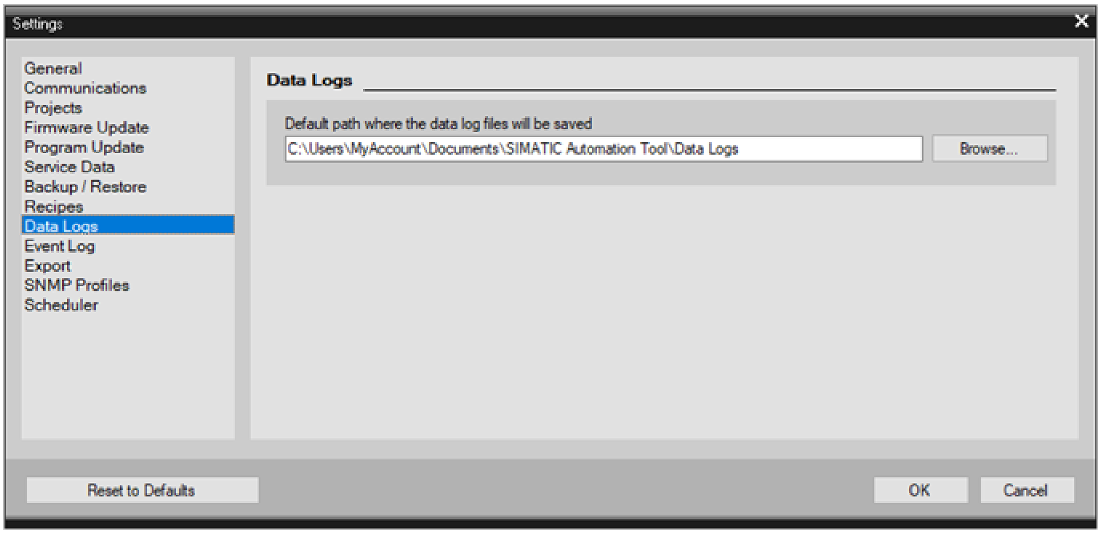
Default path where recipe files will be stored

C:\Users\MyAccount\Documents\SIMATIC Automation Tool\Recipes

6.1.5.11 Data Logs settings

You can accept the default path to Data Log files (Page 56) or assign a different path.

Your path might have a different drive letter and "MyAccount" represents the login name of the current user (Page 79).



Settings

Data Logs

Default path where the data log files will be saved: C:\Users\MyAccount\Documents\SIMATIC Automation Tool\Data Logs

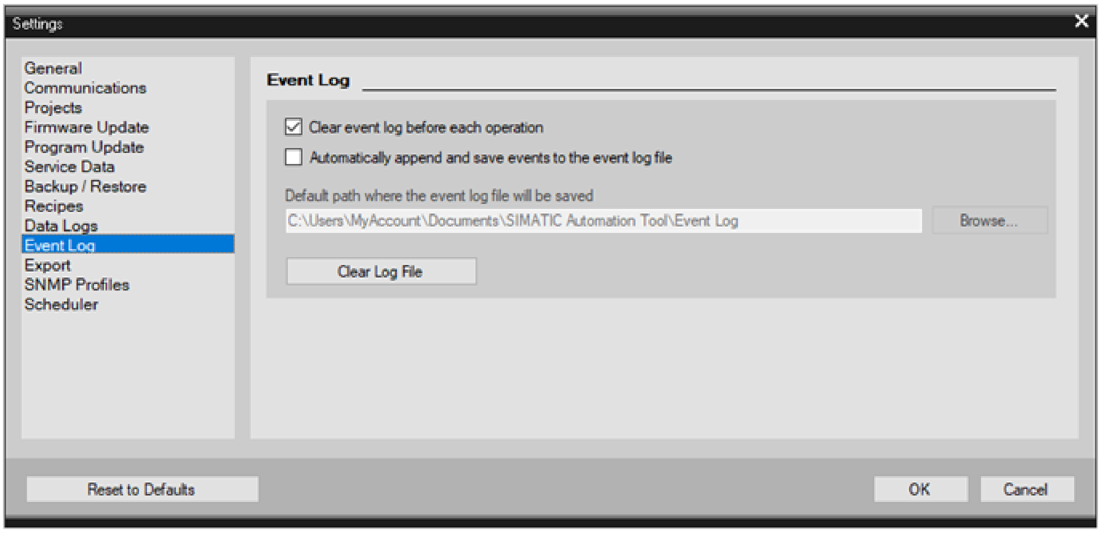
6.1.5.12 Event Log settings

By default, the SIMATIC Automation Tool clears the Event Log (Page 123) at the start of each device operation. You can deselect "Clear event log before each operation" if you want to disable the default.

When you select the "Automatically append and save events to the event log file" check box, then you can accept the default path or assign a different path.

Your path might have a different drive letter and "MyAccount" represents the login name of the current user (Page 79). The SIMATIC Automation Tool then saves each message in the Event Log window to the file "EventLogFile.csv". When you close and re-open the SIMATIC Automation Tool, logging automatically resumes in the Event Log file.

You can clear the content of the Event Log file by clicking the "Clear Log" button. This clears the contents of the file but does not delete it.



Settings

Event Log

☑ Clear event log before each operation

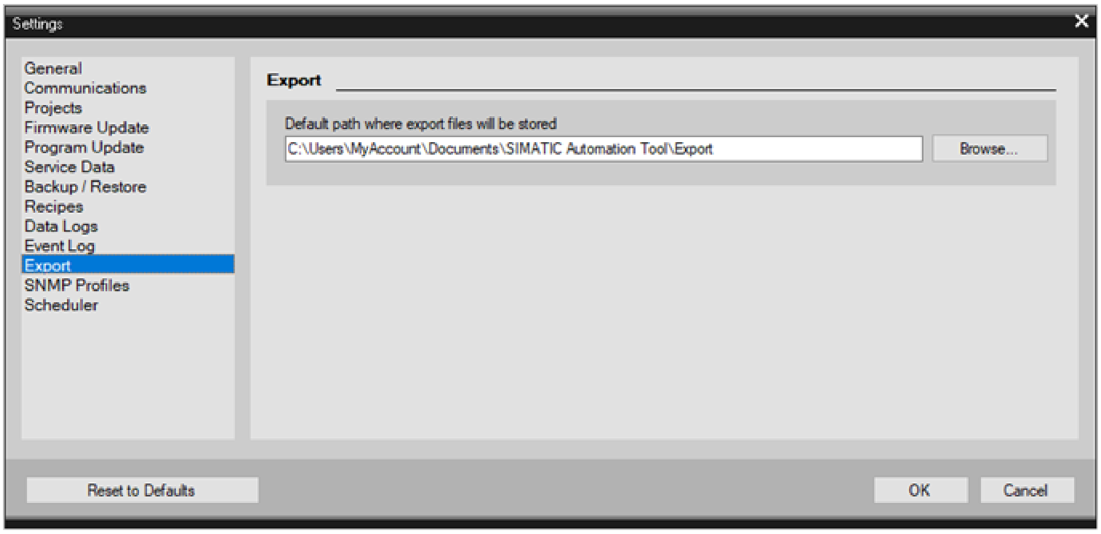
☐ Automatically append and save events to the event log file

Default path where the event log file will be saved: C:\Users\MyAccount\Documents\SIMATIC Automation Tool\Event Log

6.1.5.13 Export settings

You can accept the default path for storage of export files (Page 68) or assign a different path.

Your path might have a different drive letter and "MyAccount" represents the login name of the current user (Page 79).



Export

Default path where export files will be stored: C:\Users\MyAccount\Documents\SIMATIC Automation Tool\Export

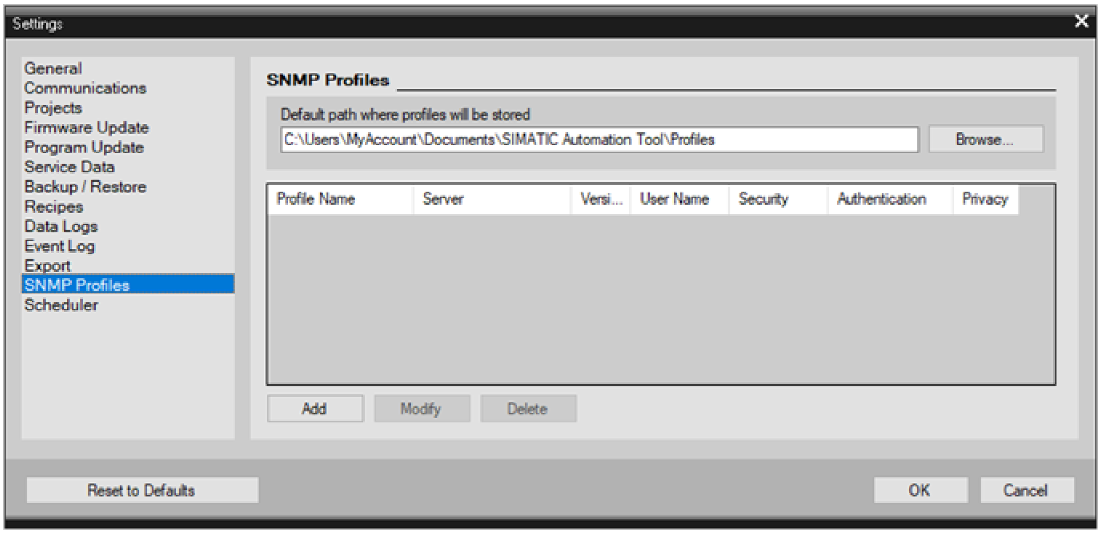
6.1.5.14 SNMP Profiles

To use the SIMATIC Automation Tool to communicate with SCALANCE devices on your network, you must configure the SNMP profiles that you use for the devices. Refer to your SCALANCE device documentation for information about SNMP configuration. If you have many SCALANCE devices on your network, the SIMATIC Automation Tool makes it easy for you to save the SNMP profiles that your use for multiple devices.

You need to provide a TFTP (Trivial File Transfer Protocol) server to install firmware (Page 40) in SCALANCE devices.

You can accept the default path for storage of the SNMP profiles file or assign a different path.

Your path might have a different drive letter and "MyAccount" represents the login name of the current user (Page 79).



The SIMATIC Automation Tool supports the following SNMP profiles:

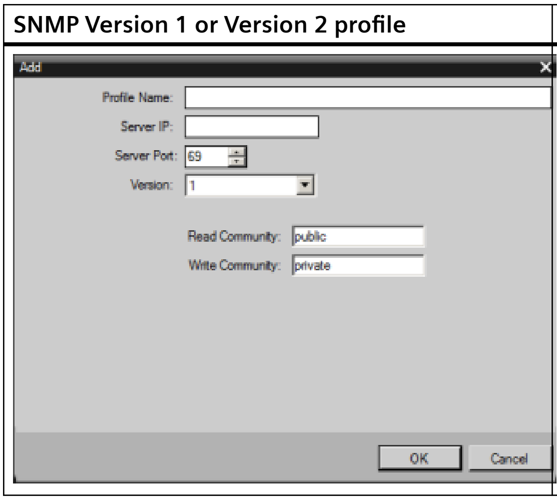
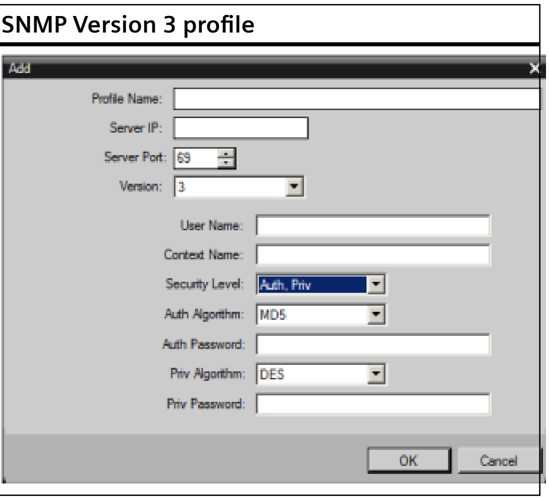
- Version 1: Defined in RFC (Request For Comments) for SNMPv1 by the Internet Engineering Task Force (IETF) and one of two historic SNMP standards
- Version 2: Defined in RFC (Request For Comments) for SNMPv2 by the Internet Engineering Task Force (IETF) and the second of two historic SNMP standards
- Version 3: Defined in RFC (Request For Comments) for SNMPv3 by the Internet Engineering Task Force (IETF) and is the version that the IETF recommends for Internet management

For more information on SNMP protocols, refer to the SNMP Research International, Inc. English-only Web page (http://www.snmp.com/protocol/).

Adding an SNMP profile

To add an SNMP profile, follow these steps:

1. Click the "Add" button from the SNMP Profiles Settings to add an SNMP profile.
2. Enter a name for your profile.
3. Enter the IP address of the TFTP server.
4. Enter or select the server port number.
5. Select the version number from the drop-down list selections.

SNMP Version 1 or Version 2 profile	SNMP Version 3 profile
 Add Profile Name: Server IP: Server Port: 69 Version: 1 Read Community: public Write Community: private OK Cancel	 Add Profile Name: Server IP: Server Port: 69 Version: 3 User Name: Context Name: Security Level: Auth, Priv Auth Algorithm: MD5 Auth Password: Priv Algorithm: DES Priv Password: OK Cancel

For a Version 1 or Version 2 profile, complete the following steps:

1. Enter the read community or accept the default.
2. Enter the write community or accept the default.

For a Version 3 profile, complete these steps:

1. Enter a user name.
2. Enter a context name.
3. Select a security level from the drop-down list:
   - NoAuth, NoPriv: Communication with no authentication and no privacy
   - Auth, NoPriv: Communication with authentication and no privacy
   - Auth, Priv: Communication with authentication and privacy

For a Version 3 profile that includes authentication, configure the following authentication fields:

- Authentication algorithm: Select one from the drop-down list.
- Authentication password

For a Version 3 profile that includes privacy, configure the following privacy fields:

- Privacy algorithm: Select one from the drop-down list.
- Privacy password

After configuring all of the profile information, click OK on the Add dialog to save the profile. The SIMATIC Automation Tool verifies your selections and informs you of any missing or incomplete fields. Click Cancel if you do not want to save the profile.

Note

You can add up to 25 SNMP profiles.

Modifying an SNMP profile

To modify an existing profile, follow these steps:

1. Select the SNMP profile that you want to modify.
2. Click the Modify button.
3. On the Modify dialog, change the fields that you want to modify.

After you complete the modifications, click OK on the Modify dialog to save the profile. The SIMATIC Automation Tool verifies your selections and informs you of any missing or incomplete fields. Click Cancel if you do not want to save the modified profile.

Deleting an SNMP profile

To delete an SNMP profile, follow these steps:

1. Select the SNMP profile that you want to delete.
2. Click the Delete button.
3. Confirm that you want to delete the profile. The SIMATIC Automation Tool informs you whether the profile is in use.

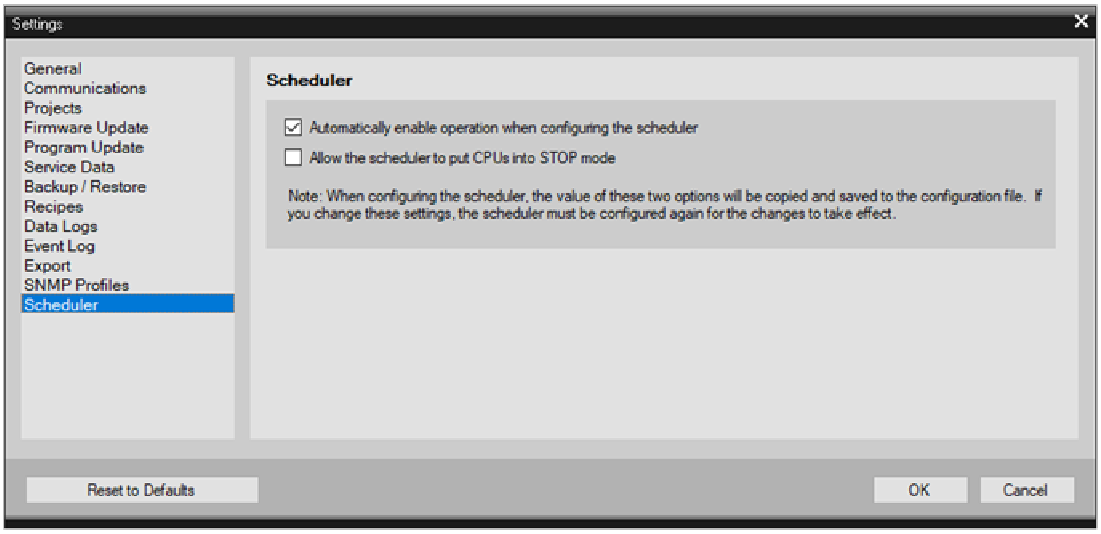
Accepting all additions, modifications, and deletions

Click OK on the SNMP Profiles Settings dialog to accept all of the changes you made. The SIMATIC Automation Tool saves the profiles to the file in the SNMP profiles folder.

Click Cancel to discard all of your changes.

6.1.5.15 Scheduler settings

You use the Scheduler settings to set options for the Scheduler application.



Settings

Scheduler

☑ Automatically enable operation when configuring the scheduler

□ Allow the scheduler to put CPUs into STOP mode

Note: When configuring the scheduler, the value of these two options will be copied and saved to the configuration file. If you change these settings, the scheduler must be configured again for the changes to take effect.

If you enable the operations, then when you launch the Scheduler application all operations default to the enabled state.

If schedule operations for a CPU require the CPU to be in STOP mode, the second setting allows the Scheduler application to change the operating mode to STOP.

These scheduler settings are included in the scheduler configuration file.

6.1.6 Tools menu

Tools menu

The Tools menu is also accessible from the ⚙ toolbar icon.

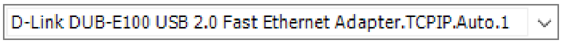
Tool icon	Menu command	Description
⚙	**Check for Firmware Updates**	Check for firmware updates (Page 40) for a single selected device. The SIMATIC Automation Tool displays the Web page with firmware update files for the selected device.
⚙	**Preload Firmware Update Files**	Look for firmware update files (Page 40) in the firmware update folder (Page 84) The SIMATIC Automation Tool preloads firmware update files in the "New Firmware Version" column for all devices. The tool only loads an update file if the file is a newer version than the version in the device. The SIMATIC Automation Tool uploads the newest corresponding file that it finds.
↓	**Configure Scheduler**	Configure the scheduler application (Page 106). (Requires an Advanced license)

6.1.7 Help menu

Tool icon	Menu command	Description
	View User Guide	Open the SIMATIC Automation Tool user guide.
	View Device Catalog	View read-only table of devices in Microsoft Excel format. To view the file, you must have Microsoft Excel on your device.
	About SIMATIC Automation Tool	Displays the About dialog that contains: • Product name • Version • License information • Check for updates button, which allows you to find SIMATIC Automation Tool updates. • License selector to switch to a different type of license

6.2 Toolbar icons

Tool icon	Description
	New: Create a new SIMATIC Automation Tool project file with the ".sat" file name extension.
	Open: Display an "Open" dialog that can browse to a folder, select a project file, and provide a password to open the encrypted project file.
💾	Save the opened project data to a file. If no filename and password are assigned, then the "Save As" dialog is displayed.
✂	Cut the selected data and copy the data to the clipboard. Clipboard data are compatible with Microsoft Excel so data can be shared between the two applications.
	Copy the selected data to the clipboard.
📋	Paste the data contained in the clipboard to the selected field(s).
?	Scan the selected network (Page 21), with the following options: • Scan the selected network interface for accessible CPUs and modules. • Refresh the status of all devices in the Device table.
	RUN: Put selected CPUs in RUN mode. (Page 30)
	STOP: Put selected CPUs in STOP mode. (Page 30)
	Update device(s) with data from the SIMATIC Automation Tool from one of the following choices: • Set IP Address (Page 32) • Set PROFINET Name (Page 33) • Program Update (Page 48) • Firmware Update (Page 40) You must select the corresponding Device table tabs to enter the data to update.
💡	Identify devices (Page 27) by flashing device LEDs or HMI screens on selected devices. Use this feature to identify the physical location of a device.
	Reset functions for selected devices: • Memory Reset (Page 38) • Reset to Factory Defaults (Page 36) • Format Memory Card (Page 39)
ℹ	Access diagnostic information: • Show Diagnostics (Page 28) • Read Service Data from selected devices (Page 29)
	Set time: Set the system time in selected CPUs to current programming device time (Page 31)
	Backup and restore functions: (Page 57) • Create Backup file(s) for selected CPUs and HMI devices. • Restore selected device(s) from backup file(s)
	File operations: • Read or delete Data Logs (Page 56) Delete operations require that the CPU be in STOP mode.
	Check for firmware updates or preload firmware update files (Page 40)

Tool icon	Description
⤓	Download the scheduler configuration file (Page 106)
 D-Link DUB-E100 USB 2.0 Fast Ethernet Adapter.TCPIP.Auto.1	Network interface drop-down list: (Page 312) Select the PROFINET network interface that is connected to the device network.

6.3 SIMATIC Automation Tool program failures

In rare cases, the SIMATIC Automation Tool software can encounter an unexpected error. If such an error occurs, the SIMATIC Automation Tool displays a dialog. Siemens requests that you click the "Send Report" button on this dialog to send the service data for the error to Siemens. The SIMATIC Automation Tool uses the default email client on your programming device to open an email with the contents of the service data in the message body. The recipient is helpline.sii@siemens.com. You can add any additional information or additional recipients to the email. You must then click "Send" to send the email.

If you choose, you can also click the "Save Project" button on the error dialog to save the SIMATIC Automation Tool project that encountered the error to your programming device.

Sending error information if you do not have an email client

If your programming device does not have an email client, you can copy the service data file to a device from which you can send email. To save, copy, and send the service data to Siemens, follow these steps:

1. Copy the "SATServiceData.txt" file from the service data folder (Page 85) to another device.
2. Send that file to helpline.sii@siemens.com.

Supported devices 7

7.1 Device catalog

Device Catalog

You can display the device catalog from the "Help > View Device Catalog" menu command of the SIMATIC Automation Tool. The device catalog shows the supported Siemens devices and supported SIMATIC Automation Tool operations.

The device catalog file is a Microsoft Excel file. You must have at least Version 2010 of Microsoft Excel to view the device catalog properly and to use functions such as sorting and filtering. The device catalog file is read-only. You cannot modify the file to enable features for a device.

The device catalog has columns that display a check mark in a cell for a TRUE condition and a blank for a FALSE condition. If the "SAT Support" column contains a check mark for a device, the device is supported by the SIMATIC Automation Tool. If the SAT Support cell is blank, the SIMATIC Automation Tool does not support the device. The SAT Support cell overrides check marks that indicate support for other functions, for example, Failsafe.

Unsupported devices

The SIMATIC Automation Tool handles unsupported devices as follows:

- Displays the device in a Device table row with partial information
- Displays the unsupported device question mark icon in the row
- Supports only the following DCP operations when the device is connected to the same subnet as the programming device that is running the SIMATIC Automation Tool:
  - Identify
  - Update IP address
  - Update PROFINET name
  - Reset communication parameters

Unsupported firmware

You might have a device on your network that has a newer firmware version than the latest firmware version that the SIMATIC Automation Tool supports. You can only perform device operations for the latest firmware version listed in the device catalog for the device.

7.2 Fail-Safe CPU support

The following table shows all fail-safe CPUs and firmware versions that the SIMATIC Automation Tool supports:

Article number	Type descriptor	Firmware version	First SIMATIC Automation Tool version to support
6ES7 212-1AF40-0XB0	CPU 1212FC DC/DC/DC	V4.2	V3.1
6ES7 212-1AF40-0XB0	CPU 1212FC DC/DC/DC	V4.3	V3.1 SP2
6ES7 212-1AF40-0XB0	CPU 1212FC DC/DC/DC	V4.4	V3.1 SP4
6ES7 212-1HF40-0XB0	CPU 1212FC DC/DC/Rly	V4.2	V3.1
6ES7 212-1HF40-0XB0	CPU 1212FC DC/DC/Rly	V4.3	V3.1 SP2
6ES7 212-1HF40-0XB0	CPU 1212FC DC/DC/Rly	V4.4	V3.1 SP4
6ES7 214-1AF40-0XB0	CPU 1214FC DC/DC/DC	V4.1	V3.1
6ES7 214-1AF40-0XB0	CPU 1214FC DC/DC/DC	V4.2	V3.1
6ES7 214-1AF40-0XB0	CPU 1214FC DC/DC/DC	V4.3	V3.1 SP2
6ES7 214-1AF40-0XB0	CPU 1214FC DC/DC/DC	V4.4	V3.1 SP4
6ES7 214-1HF40-0XB0	CPU 1214FC DC/DC/Rly	V4.1	V3.1
6ES7 214-1HF40-0XB0	CPU 1214FC DC/DC/Rly	V4.2	V3.1
6ES7 214-1HF40-0XB0	CPU 1214FC DC/DC/Rly	V4.3	V3.1 SP2
6ES7 214-1HF40-0XB0	CPU 1214FC DC/DC/Rly	V4.4	V3.1 SP4
6ES7 215-1AF40-0XB0	CPU 1215FC DC/DC/DC	V4.1	V3.1
6ES7 215-1AF40-0XB0	CPU 1215FC DC/DC/DC	V4.2	V3.1
6ES7 215-1AF40-0XB0	CPU 1215FC DC/DC/DC	V4.3	V3.1 SP2
6ES7 215-1AF40-0XB0	CPU 1215FC DC/DC/DC	V4.4	V3.1 SP4
6ES7 215-1HF40-0XB0	CPU 1215FC DC/DC/Rly	V4.1	V3.1
6ES7 215-1HF40-0XB0	CPU 1215FC DC/DC/Rly	V4.2	V3.1
6ES7 215-1HF40-0XB0	CPU 1215FC DC/DC/Rly	V4.3	V3.1 SP2
6ES7 215-1HF40-0XB0	CPU 1215FC DC/DC/Rly	V4.4	V3.1 SP4
6ES7 510-1SJ00-0AB0	CPU 1510SP F-1 PN	V1.7	V3.1
6ES7 510-1SJ00-0AB0	CPU 1510SP F-1 PN	V1.8	V3.1
6ES7 510-1SJ01-0AB0	CPU 1510SP F-1 PN	V1.8	V3.1
6ES7 510-1SJ01-0AB0	CPU 1510SP F-1 PN	V2.0	V3.1
6ES7 510-1SJ01-0AB0	CPU 1510SP F-1 PN	V2.1	V3.1
6ES7 510-1SJ01-0AB0	CPU 1510SP F-1 PN	V2.5	V3.1 SP1
6ES7 510-1SJ01-0AB0	CPU 1510SP F-1 PN	V2.6	V3.1 SP2
6ES7 510-1SJ01-0AB0	CPU 1510SP F-1 PN	V2.8	V3.1 SP4
6ES7 511-1FK00-0AB0	CPU 1511F-1 PN	V1.7	V3.1
6ES7 511-1FK00-0AB0	CPU 1511F-1 PN	V1.8	V3.1
6ES7 511-1FK01-0AB0	CPU 1511F-1 PN	V1.8	V3.1
6ES7 511-1FK01-0AB0	CPU 1511F-1 PN	V2.0	V3.1
6ES7 511-1FK01-0AB0	CPU 1511F-1 PN	V2.1	V3.1
6ES7 511-1FK01-0AB0	CPU 1511F-1 PN	V2.5	V3.1 SP1
6ES7 511-1FK01-0AB0	CPU 1511F-1 PN	V2.6	V3.1 SP2

Article number	Type descriptor	Firmware version	First SIMATIC Automation Tool version to support
6ES7 511-1FK01-0AB0	CPU 1511F-1 PN	V2.8	V3.1 SP4
6ES7 511-1FK02-0AB0	CPU 1511F-1 PN	V2.5	V3.1 SP1
6ES7 511-1FK02-0AB0	CPU 1511F-1 PN	V2.6	V3.1 SP2
6ES7 511-1FK02-0AB0	CPU 1511F-1 PN	V2.8	V3.1 SP4
6ES7 511-1UK01-0AB0	CPU 1511TF-1 PN	V2.1	V3.1
6ES7 511-1UK01-0AB0	CPU 1511TF-1 PN	V2.5	V3.1 SP1
6ES7 511-1UK01-0AB0	CPU 1511TF-1 PN	V2.6	V3.1 SP2
6ES7 511-1UK01-0AB0	CPU 1511TF-1 PN	V2.8	V3.1 SP4
6ES7 512-1SK00-0AB0	CPU 1512SP F-1 PN	V1.7	V3.1
6ES7 512-1SK00-0AB0	CPU 1512SP F-1 PN	V1.8	V3.1
6ES7 512-1SK01-0AB0	CPU 1512SP F-1 PN	V1.8	V3.1
6ES7 512-1SK01-0AB0	CPU 1512SP F-1 PN	V2.0	V3.1
6ES7 512-1SK01-0AB0	CPU 1512SP F-1 PN	V2.1	V3.1
6ES7 512-1SK01-0AB0	CPU 1512SP F-1 PN	V2.5	V3.1 SP1
6ES7 512-1SK01-0AB0	CPU 1512SP F-1 PN	V2.6	V3.1 SP2
6ES7 512-1SK01-0AB0	CPU 1512SP F-1 PN	V2.8	V3.1 SP4
6ES7 513-1FL00-0AB0	CPU 1513F-1 PN	V1.7	V3.1
6ES7 513-1FL00-0AB0	CPU 1513F-1 PN	V1.8	V3.1
6ES7 513-1FL01-0AB0	CPU 1513F-1 PN	V1.8	V3.1
6ES7 513-1FL01-0AB0	CPU 1513F-1 PN	V2.0	V3.1
6ES7 513-1FL01-0AB0	CPU 1513F-1 PN	V2.1	V3.1
6ES7 513-1FL01-0AB0	CPU 1513F-1 PN	V2.5	V3.1 SP1
6ES7 513-1FL01-0AB0	CPU 1513F-1 PN	V2.6	V3.1 SP2
6ES7 513-1FL01-0AB0	CPU 1513F-1 PN	V2.8	V3.1 SP4
6ES7 513-1FL02-0AB0	CPU 1513F-1 PN	V2.5	V3.1 SP1
6ES7 513-1FL02-0AB0	CPU 1513F-1 PN	V2.6	V3.1 SP2
6ES7 513-1FL02-0AB0	CPU 1513F-1 PN	V2.8	V3.1 SP4
6ES7 513-2GL00-0AB0	CPU 1513pro F-2 PN	V2.8	V3.1 SP4
6ES7 515-2FM00-0AB0	CPU 1515F-2 PN	V1.6	V3.1
6ES7 515-2FM00-0AB0	CPU 1515F-2 PN	V1.7	V3.1
6ES7 515-2FM00-0AB0	CPU 1515F-2 PN	V1.8	V3.1
6ES7 515-2FM01-0AB0	CPU 1515F-2 PN	V1.8	V3.1
6ES7 515-2FM01-0AB0	CPU 1515F-2 PN	V2.0	V3.1
6ES7 515-2FM01-0AB0	CPU 1515F-2 PN	V2.1	V3.1
6ES7 515-2FM01-0AB0	CPU 1515F-2 PN	V2.5	V3.1 SP1
6ES7 515-2FM01-0AB0	CPU 1515F-2 PN	V2.6	V3.1 SP2
6ES7 515-2FM01-0AB0	CPU 1515F-2 PN	V2.8	V3.1 SP4
6ES7 515-2FM02-0AB0	CPU 1515F-2 PN	V2.8	V3.1 SP4
6ES7 515-2UM01-0AB0	CPU 1515TF-2 PN	V2.1	V3.1
6ES7 515-2UM01-0AB0	CPU 1515TF-2 PN	V2.5	V3.1 SP1
6ES7 515-2UM01-0AB0	CPU 1515TF-2 PN	V2.6	V3.1 SP2

Article number	Type descriptor	Firmware version	First SIMATIC Automation Tool version to support
6ES7 515-2UM01-0AB0	CPU 1515TF-2 PN	V2.8	V3.1 SP4
6ES7 516-2GN00-0AB0	CPU 1516pro F-2 PN	V2.0	V3.1
6ES7 516-2GN00-0AB0	CPU 1516pro F-2 PN	V2.1	V3.1
6ES7 516-2GN00-0AB0	CPU 1516pro F-2 PN	V2.5	V3.1 SP1
6ES7 516-2GN00-0AB0	CPU 1516pro F-2 PN	V2.6	V3.1 SP2
6ES7 516-2GN00-0AB0	CPU 1516pro F-2 PN	V2.8	V3.1 SP4
6ES7 516-3FN00-0AB0	CPU 1516F-3 PN/DP	V1.5	V3.1
6ES7 516-3FN00-0AB0	CPU 1516F-3 PN/DP	V1.6	V3.1
6ES7 516-3FN00-0AB0	CPU 1516F-3 PN/DP	V1.7	V3.1
6ES7 516-3FN00-0AB0	CPU 1516F-3 PN/DP	V1.8	V3.1
6ES7 516-3FN01-0AB0	CPU 1516F-3 PN/DP	V1.8	V3.1
6ES7 516-3FN01-0AB0	CPU 1516F-3 PN/DP	V2.0	V3.1
6ES7 516-3FN01-0AB0	CPU 1516F-3 PN/DP	V2.1	V3.1
6ES7 516-3FN01-0AB0	CPU 1516F-3 PN/DP	V2.5	V3.1 SP1
6ES7 516-3FN01-0AB0	CPU 1516F-3 PN/DP	V2.6	V3.1 SP2
6ES7 516-3FN01-0AB0	CPU 1516F-3 PN/DP	V2.8	V3.1 SP4
6ES7 516-3FN02-0AB0	CPU 1516F-3 PN/DP	V2.8	V3.1 SP4
6ES7 516-3UN00-0AB0	CPU 1516TF-3 PN/DP	V2.5	V3.1 SP1
6ES7 516-3UN00-0AB0	CPU 1516TF-3 PN/DP	V2.6	V3.1 SP2
6ES7 516-3UN00-0AB0	CPU 1516TF-3 PN/DP	V2.8	V3.1 SP4
6ES7 517-3FP00-0AB0	CPU 1517F-3 PN/DP	V1.6	V3.1
6ES7 517-3FP00-0AB0	CPU 1517F-3 PN/DP	V1.7	V3.1
6ES7 517-3FP00-0AB0	CPU 1517F-3 PN/DP	V1.8	V3.1
6ES7 517-3FP00-0AB0	CPU 1517F-3 PN/DP	V2.0	V3.1
6ES7 517-3FP00-0AB0	CPU 1517F-3 PN/DP	V2.1	V3.1
6ES7 517-3FP00-0AB0	CPU 1517F-3 PN/DP	V2.5	V3.1 SP1
6ES7 517-3FP00-0AB0	CPU 1517F-3 PN/DP	V2.6	V3.1 SP2
6ES7 517-3FP00-0AB0	CPU 1517F-3 PN/DP	V2.8	V3.1 SP4
6ES7 517-3UP00-0AB0	CPU 1517TF-3 PN/DP	V2.0	V3.1
6ES7 517-3UP00-0AB0	CPU 1517TF-3 PN/DP	V2.1	V3.1
6ES7 517-3UP00-0AB0	CPU 1517TF-3 PN/DP	V2.5	V3.1 SP1
6ES7 517-3UP00-0AB0	CPU 1517TF-3 PN/DP	V2.6	V3.1 SP2
6ES7 517-3UP00-0AB0	CPU 1517TF-3 PN/DP	V2.8	V3.1 SP4
6ES7 518-4FP00-0AB0	CPU 1518F-4 PN/DP	V1.5	V3.1
6ES7 518-4FP00-0AB0	CPU 1518F-4 PN/DP	V1.6	V3.1
6ES7 518-4FP00-0AB0	CPU 1518F-4 PN/DP	V1.7	V3.1
6ES7 518-4FP00-0AB0	CPU 1518F-4 PN/DP	V1.8	V3.1
6ES7 518-4FP00-0AB0	CPU 1518F-4 PN/DP	V2.0	V3.1
6ES7 518-4FP00-0AB0	CPU 1518F-4 PN/DP	V2.1	V3.1
6ES7 518-4FP00-0AB0	CPU 1518F-4 PN/DP	V2.5	V3.1 SP1
6ES7 518-4FP00-0AB0	CPU 1518F-4 PN/DP	V2.6	V3.1 SP2

Article number	Type descriptor	Firmware version	First SIMATIC Automation Tool version to support
6ES7 518-4FP00-0AB0	CPU 1518F-4 PN/DP	V2.8	V3.1 SP4
6ES7 518-4FP00-3AB0	CPU 1518F-4 PN/DP ODK	V2.0	V3.1 SP1
6ES7 518-4FP00-3AB0	CPU 1518F-4 PN/DP ODK	V2.1	V3.1 SP1
6ES7 518-4FP00-3AB0	CPU 1518F-4 PN/DP ODK	V2.5	V3.1 SP1
6ES7 518-4FP00-3AB0	CPU 1518F-4 PN/DP ODK	V2.6	V3.1 SP2
6ES7 518-4FP00-3AB0	CPU 1518F-4 PN/DP ODK	V2.8	V3.1 SP4
6ES7 518-4FX00-1AB0	CPU 1518F-4 PN/DP MFP	V2.5	V3.1 SP1
6ES7 518-4FX00-1AB0	CPU 1518F-4 PN/DP MFP	V2.6	V3.1 SP2
6ES7 518-4FX00-1AB0	CPU 1518F-4 PN/DP MFP	V2.8	V3.1 SP4
6ES7-615-4DF10-0AB0	CPU 1504D TF	V2.8	V4.0 SP1
6ES7-615-7DF10-0AB0	CPU 1507D TF	V2.8	V4.0 SP1
6ES7 672-5SC01-0YA0	CPU 1505SP F	V2.0	V3.1 SP2
6ES7 672-5SC01-0YA0	CPU 1505SP F	V2.1	V3.1 SP2
6ES7 672-5SC11-0YA0	CPU 1505SP F	V2.5	V3.1 SP2
6ES7 672-5SC11-0YA0	CPU 1505SP F	V2.6	V3.1 SP2
6ES7 672-5SC11-0YA0	CPU 1505SP F	V2.7	V3.1 SP4
6ES7 672-5SC11-0YA0	CPU 1505SP F	V20.8	V4.0 SP1
6ES7 672-5WC11-0YA0	CPU 1505SP TF	V2.5	V3.1 SP2
6ES7 672-5WC11-0YA0	CPU 1505SP TF	V2.6	V3.1 SP2
6ES7 672-5WC11-0YA0	CPU 1505SP TF	V2.7	V3.1 SP4
6ES7 672-5WC11-0YA0	CPU 1505SP TF	V20.8	V4.0 SP1
6ES7 672-7FC01-0YA0	CPU 1507S F	V2.0	V3.1 SP2
6ES7 672-7FC01-0YA0	CPU 1507S F	V2.1	V3.1 SP2
6ES7 672-7FC01-0YA0	CPU 1507S F	V2.5	V3.1 SP2
6ES7 672-7FC01-0YA0	CPU 1507S F	V2.6	V3.1 SP2
6ES7 672-7FC01-0YA0	CPU 1507S F	V2.7	V3.1 SP4
6ES7 672-7FC01-0YA0	CPU 1507S F	V20.8	V4.0 SP1
6ES7 672-8FC01-0YA0	CPU 1508S F	V2.6	V3.1 SP2
6ES7 672-8FC01-0YA0	CPU 1508S F	V2.7	V3.1 SP4
6ES7 672-8FC01-0YA0	CPU 1508S F	V20.8	V4.0 SP1

8 Understanding the different licenses

The SIMATIC Automation Tool supports the following types of licenses:

- Trial 21-day license
- Basic
- Advanced

You can use the included trial 21-day trial license at no cost. After 21 days, to use many of the SIMATIC Automation Tool features you must purchase a Basic or Advanced license. Refer to the Installation notes for this product for information on purchasing a license.

The SIMATIC Automation Tool features that are available for each license type are as follows:

Feature	No license or expired trial license	Basic license	Advanced license or 21-day trial license
Scan Network (Page 21) to fill the Device table with the accessible devices on the network.	✓	✓	✓
Insert a device (Page 25)	✓	✓	✓
Change CPU operating mode (Page 30)	✓	✓	✓
Show diagnostics (Page 28)	✓	✓	✓
Read service data (Page 29)	✓	✓	✓
Export PC data (Page 73)	✓	✓	✓
Export device information (Page 68)	✓	✓	✓
Export device diagnostics (Page 72)	✓	✓	✓
Set IP address (Page 32)	✓	✓	✓
Set PROFINET name (Page 33)	✓	✓	✓
Identify device (Page 27) on the network by flashing LEDs	✓	✓	✓
Support up to five concurrent device operations (threads) (Page 82)	✓	✓	✓
Update program (Page 48)	HMI only	✓	✓
Update firmware (Page 40)	HMI only	✓	✓
Reset communication parameters (Page 35)	HMI only	✓	✓
Create full backup (Page 57)	HMI only	✓	✓
Back up HMI Recipes (Page 58)	HMI only	✓	✓
Back up HMI user administration data (Page 59)	HMI only	✓	✓
Restore device from backup (Page 59)	HMI only	✓	✓
Reset memory (Page 38)		✓	✓
Reset to factory defaults (Page 36)		✓	✓
Format memory card (Page 39)		✓	✓

Feature	No license or expired trial license	Basic license	Advanced license or 21-day trial license
Set device time (Page 31)		✓	✓
Read and delete Data logs (Page 56)		✓	✓
Support up to ten concurrent device operations (threads) (Page 82)			✓
Schedule device operations (Page 106)			✓
Use card browser to access files on a SIMATIC memory card (Page 103)			✓
Communicate to devices behind NAT Routers (Page 21)			✓
Insert multiple devices (Page 25)			✓
Create and open archives (Page 66)			✓
Communicate to devices through PROFINET CM (Communications Module) or CP (Communications Processor) (Page 21)			✓

The SIMATIC Automation Tool User Guide describes each of these features.

Note

Recognition of previous license

If you have an "Unlimited product license to V3.1 and all service packs" license for a previous installation of the SIMATIC Automation Tool, this license functions as a Basic license. The SIMATIC Automation Tool enables all Basic license features for this license.

Advanced features

9

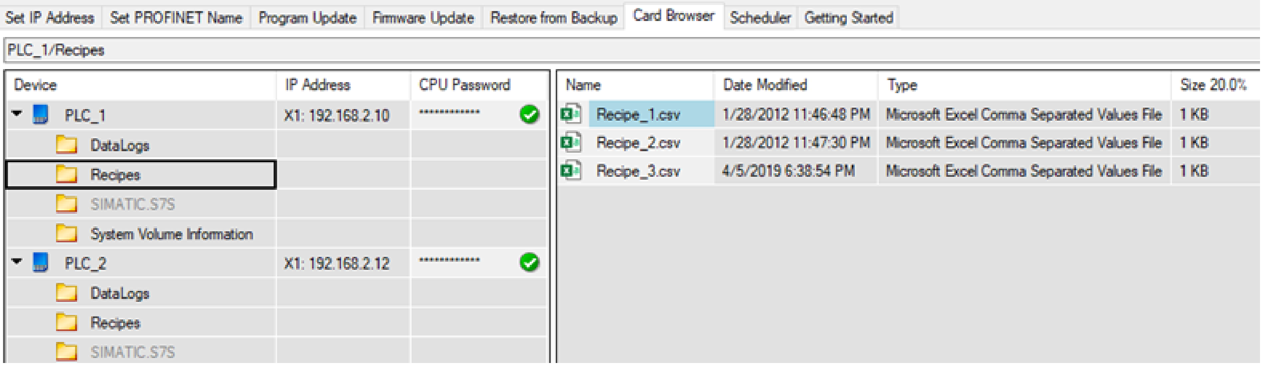
9.1 Using the Card Browser to work with files on SIMATIC memory cards in CPUs

The SIMATIC Automation Tool provides a Card Browser for users who have the Advanced license. With the Card Browser, you can work with files and folders on the SIMATIC memory cards in CPUs. The Card Browser provides the following functions:

- Copy and paste files and folders
- Delete files and folders, except for protected folders
- Download files from the programming device to a memory card in a CPU

When you select the Card Browser tab, you see all the CPUs from the last network scan. CPUs without a memory card appear in light gray.

The Card Browser functions much like Windows File Explorer. It contains a navigation pane on the left and a file list pane on the right. The SIMATIC memory cards are in the navigation pane. You can navigate and expand the folders on the memory cards. When you click a memory card or folder, the file list pane shows the folders and files for your selection.



PLC_1/Recipes

Device	IP Address	CPU Password
PLC_1	X1: 192.168.2.10	************* ✓
DataLogs		
Recipes		
SIMATIC.S7S		
System Volume Information		
PLC_2	X1: 192.168.2.12	************* ✓
DataLogs		
Recipes		
SIMATIC.S7S		

Name	Date Modified	Type	Size 20.0%
Recipe_1.csv	1/28/2012 11:46:48 PM	Microsoft Excel Comma Separated Values File	1 KB
Recipe_2.csv	1/28/2012 11:47:30 PM	Microsoft Excel Comma Separated Values File	1 KB
Recipe_3.csv	4/5/2019 6:38:54 PM	Microsoft Excel Comma Separated Values File	1 KB

The memory cards can have the following folders for specific uses:

- DataLogs
- Recipes
- UserFiles
- ODK1500S

You can create these folders if they do not exist, but you cannot delete, cut, or rename these folders. You can copy them to another location, but you cannot move them to another location. The SIMATIC Automation Tool prevents you from performing an invalid action. You cannot write to a SIMATIC memory Card that is physically write-protected.

To create folders at the root level, right-click in the gray area of the root level in the file list pane and select New > Folder.

SIMATIC memory cards might also contain other custom files and folders. Using the file list pane, you can delete and rename these other folders and their contents.

For data logs, you can copy a data log file to your programming device, but you cannot delete, rename, or change any files in a DataLogs folder. You perform Data Log file operations (Page 56) from the Device table.

You can copy and paste multiple files in a single operation.

The panes of the Card Browser support the following display features:

- Resizing of the panes and columns in either pane
- Navigation pane sort on Device or IP Address
- File list sort by any of the columns
- Ability to show or hide some columns
- Typical Windows mouse and keyboard operations

Password requirements for recipe operations

If a CPU is protected (Page 308), observe these password requirements:

- You must enter a CPU password with at least Read access to copy recipe files to your programming device.
- To delete, add, or replace recipe files, you must enter a CPU password with the Full access (read and write) access level.

If you do not enter a password, or if the password does not provide a sufficient access level, the operation for that CPU fails. The SIMATIC Automation Tool enters an error message in the Event Log.

Deleting files or folders from a SIMATIC memory card

You delete files from a memory card just like you do in the Windows File Explorer. You have an opportunity to confirm or cancel a delete operation to avoid unwanted deletions.

Note that you cannot delete required files and folders from a memory card.

Transferring files from a SIMATIC memory card to your programming device

To transfer files or a folder from a CPU memory card to your programming device, follow these steps:

1. Copy a file or folder from the Card Browser.
2. Paste the file on your programming device.

If you are overwriting an existing file, confirm the operation. You can copy and paste multiple files.

Transferring files from your programming device to a SIMATIC memory card

To transfer files or a folder from your programming device to a CPU memory card, use one of the following methods:

1. Copy a file or folder from your programming device.
2. Paste the file in a folder in the file list pane of Card Browser.

If you are overwriting an existing file, confirm the operation. You can copy and paste multiple files.

Transferring files or folders from one SIMATIC memory card to another

You can copy files from the file list pane for one memory card and paste the selection to the file list pane of another memory card.

The SIMATIC Automation Tool also provides "Copy Special" and "Paste Special" commands in the navigation pane to simplify copying all of the recipe files or user files from one memory card to another.

To transfer recipe files, user files, or both from one memory card to another, follow these steps:

1. Right-click the memory card in the navigation pane from which you want to copy recipe files or user files.
2. Select "Copy Special" from the shortcut menu.
3. From the dialog, select the file types (recipe files, user files, or both) that you want to copy to the clipboard.
4. Right-click the memory card in the navigation pane where you want to paste the recipe files or user files.
5. Select "Paste Special" from the shortcut menu.
6. Make your selections on the "Paste Special" dialog and click OK. The dialog displays what you selected from the "Copy special" dialog and gives you options for the paste operation.

Restrictions

You cannot delete the following folders from the root level of a SIMATIC memor card:

- DataLogs
- Recipes
- UserFIles
- ODK1500S

Operation results

For each operation, the Event Log below the Device table shows the results of the operation.

9.2 Scheduling device operations

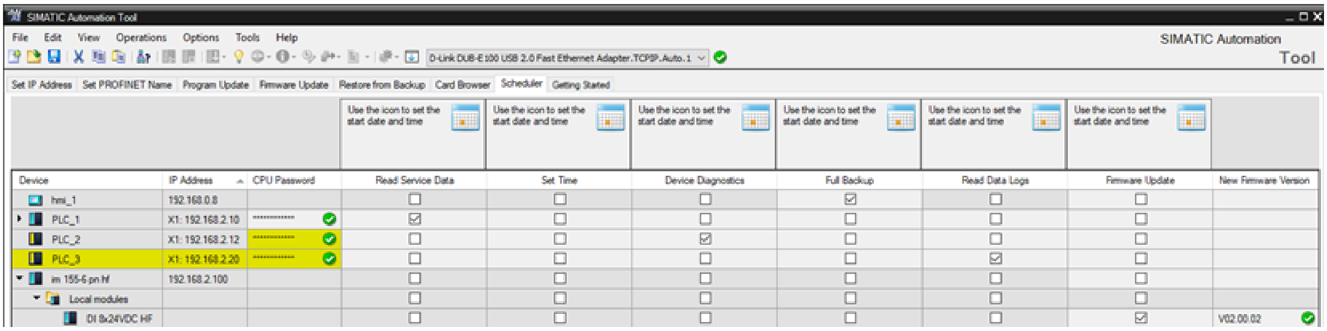
Using the SIMATIC Automation Tool's Scheduler, you can schedule device operations at specific dates and times. You can also schedule recurring operations.

The Scheduler consists of two parts:

- A schedule configurator that enables you to define the schedule for various operations
- A scheduler application that performs the operations based on the configured schedule

Configuring schedules for device operations

To configure a schedule, click the Scheduler tab in the SIMATIC Automation Tool. The SIMATIC Automation Tool displays the devices in your Device table with check boxes for operations that you can schedule:



			Use the icon to set the start date and time	Use the icon to set the start date and time	Use the icon to set the start date and time	Use the icon to set the start date and time	Use the icon to set the start date and time	Use the icon to set the start date and time	
Device	IP Address	CPU Password	Read Service Data	Set Time	Device Diagnostics	Full Backup	Read Data Logs	Firmware Update	New Firmware Version
hmi_1	192.168.0.8		☐	☐	☐	☑	☐	☐	
PLC_1	X1: 192.168.2.10	************ ✓	☑	☐	☐	☐	☐	☐	
PLC_2	X1: 192.168.2.12	************ ✓	☐	☐	☑	☐	☐	☐	
PLC_3	X1: 192.168.2.20	************ ✓	☐	☐	☐	☐	☑	☐	
im 155-6 pn hf	192.168.2.100		☐	☐	☐	☐	☐	☐	
Local modules			☐	☐	☐	☐	☐	☐	
DI 8x24VDC HF			☐	☐	☐	☐	☐	☑	V02.00.02 ✓

Select the check boxes for the operations that you want to run for the devices in the Device table. Depending on the device type, you can only select operations that are applicable for each device. Note that you can select multiple cells and use the space bar to toggle the selection state for the cell range.

For each type of operation that you select, click the calendar symbol to configure the following schedule settings:

- Date
- Time
- Frequency

The drop-down list for Frequency shows you the valid frequency choices:

- Run Once
- Every Day
- Every Week
- Every 2 Weeks
- Every Month
- Every Year

The combination of date, time, and frequency must be valid. For example, you cannot schedule a yearly operation for February 29th or a monthly operation for the 30th or 31st of a month.

For a firmware update operation, you must select a valid firmware update file. If the device is a SCALANCE device, you must also select a valid SNMP profile in addition to a valid firmware update file.

When you have finished configuring your scheduled operations, click the ↓ button. The SIMATIC Automation Tool validates that you have selected at least one operation for one device at a valid frequency setting. The SIMATIC Automation Tool then prompts you to set a scheduler configuration file password.

You must enter this password when you first launch the scheduler application (Page 107) and under either of the following conditions:

- You downloaded a new scheduler configuration file
- Another user has logged in to Windows since your last login

Note

Downloading a schedule while scheduled operations are running

If scheduled operations are running when you the click the ↓ button, the scheduler application performs these actions:

- Cancels queued device operations that have not yet started
- Logs an error event for each canceled operation
- Allows device operations that are currently running to finish
- Logs success or failure message for running operations
- Terminates future scheduled operation

If you change device information for a device with a scheduled operation, download the scheduler configuration file. The scheduler application then has the current information for the device.

9.3 Executing scheduled operations

The SIMATIC Automation Tool includes a separate scheduler application that executes the operations you scheduled (Page 106) according to their scheduled times. If you have downloaded a schedule configuration file, the scheduler application runs by default when you log in to Windows.

When does the scheduler application run?

If you have downloaded a new scheduler configuration file since the last time you logged in to Windows, you must open the scheduler application and enter the scheduler configuration file password. The scheduler does not run any operations if you have not entered the scheduler configuration password.

If other users have logged in to Windows since your last login, you must also open the scheduler application and enter the scheduler configuration file password. The scheduler associates the password entry with a specific Windows user.

If other users log in to Windows, the scheduler application does not run more than once. Only one instance of the scheduler application runs at a time. The scheduler application is associated with a specific Windows user. This behavior is intentional and does not result in an error or Scheduler Event Log message.

The scheduler application runs regardless of whether the SIMATIC Automation Tool is running. After you log on to Windows, you do not need to be present for the scheduler application to run. The scheduler application does not run if you log off or if the programming device is hibernating or powered off.

The programming device must be connected to the PROFINET network for the scheduler application to run scheduled operations. When the scheduler application is running an operation on multiple devices, the devices can complete the operation in any order. Different devices run at different processing speeds. Network communication speeds also vary. For these reasons, the scheduler application establishes no precedence order for multiple devices running an operation.

Managing scheduled operations

The scheduler application has a user interface where you can control and monitor scheduled operations. Scheduled operations run regardless of whether you open the Scheduler user interface. You can open the Scheduler user interface in one of two ways:

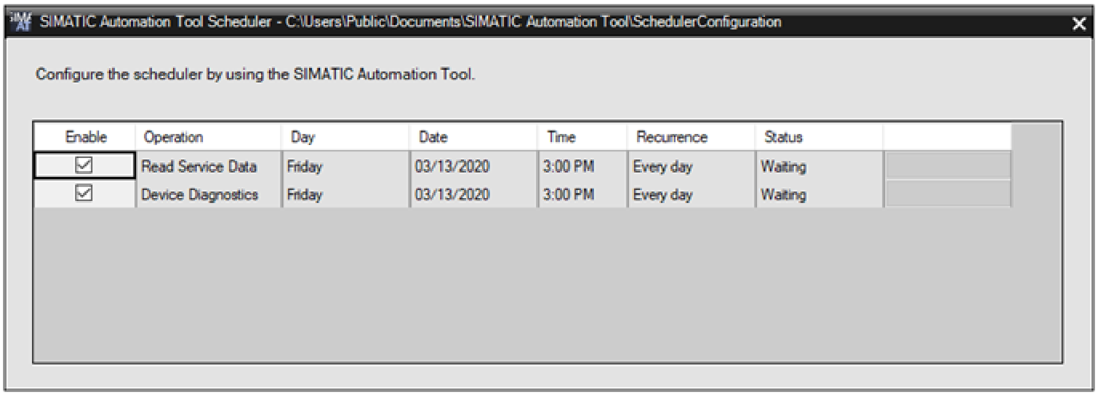
- Select the Scheduler tab in the SIMATIC Automation Tool. Click the toolbar button to download the scheduler configuration file and open the Scheduler user interface.
- Double-click the SIMATIC Automation Tool Scheduler from the notification area of the Windows Taskbar, or use the shortcut menu to open it. If you have not yet downloaded a scheduler configuration file, the scheduler application is empty.
  If the notification area no longer includes the SIMATIC Automation Tool Scheduler, download a valid scheduler configuration file (Page 106) from the SIMATIC Automation Tool.

The scheduler application has its own settings and its own Scheduler Event Log. Both are available from the shortcut menu when you left-click or right-click the SIMATIC Automation Tool Scheduler icon in the notification area.

Enabling and monitoring scheduled operations

When you open the SIMATIC Automation Tool Scheduler, enter the scheduler configuration file password if prompted. Remember that this is the scheduler configuration file password and not the project password.

The scheduler application then displays a table of the operations that you scheduled. You can enable or disable any of the operations. The enabled operations run according to their schedule.


SIMATIC Automation Tool Scheduler - C:\Users\Public\Documents\SIMATIC Automation Tool\SchedulerConfiguration

Configure the scheduler by using the SIMATIC Automation Tool.

Enable	Operation	Day	Date	Time	Recurrence	Status
☑	Read Service Data	Friday	03/13/2020	3:00 PM	Every day	Waiting
☑	Device Diagnostics	Friday	03/13/2020	3:00 PM	Every day	Waiting

When an operation is running, the Status column displays "Running".

Note

Canceling running operations

When an operation is running, the scheduler application displays a Cancel button for the operation. Click the Cancel button to cancel the running operation.

The scheduler application cancels operations that are queued but not running. The scheduler displays "Canceling" in the Status column while it is canceling a queued operation. It displays "Completed" when it finishes canceling the operation.

The scheduler application does not cancel operations that are actively running. These operations must run to completion because they have already started. The scheduler Event Log includes the result of the operation.

The state of each operation, enabled or disabled, remains in effect until you change it. In the Scheduler settings of the Options menu (Page 91), you can select whether to enable all operations by default.

Enabling or disabling all operations

You can left click or right click the SIMATIC Automation Tool Scheduler icon in the notification area of the Windows Taskbar to display a shortcut menu. Select "Enable all operations" if you want to enable all operations.

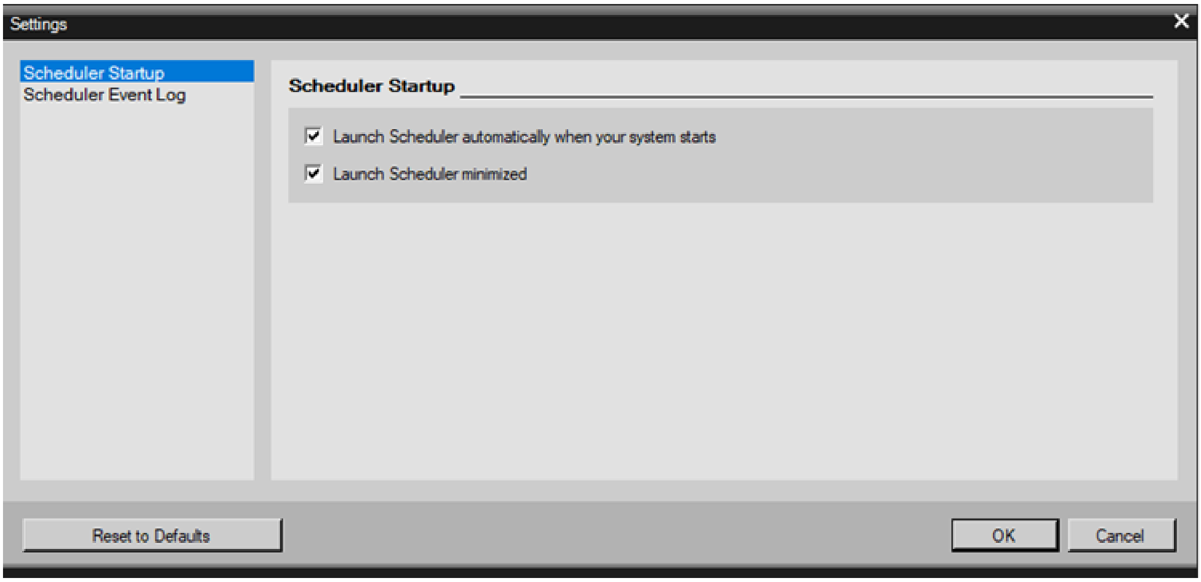
Select "Disable all operations" if you want to disable all operations.

You do not need to open the scheduler application to enable or disable all operations.

Setting Scheduler startup and Event Log options

From the shortcut menu, select "Settings" to configure settings for startup behavior and Event Log options.

Scheduler Startup


Settings

Scheduler Startup
Scheduler Event Log

Scheduler Startup

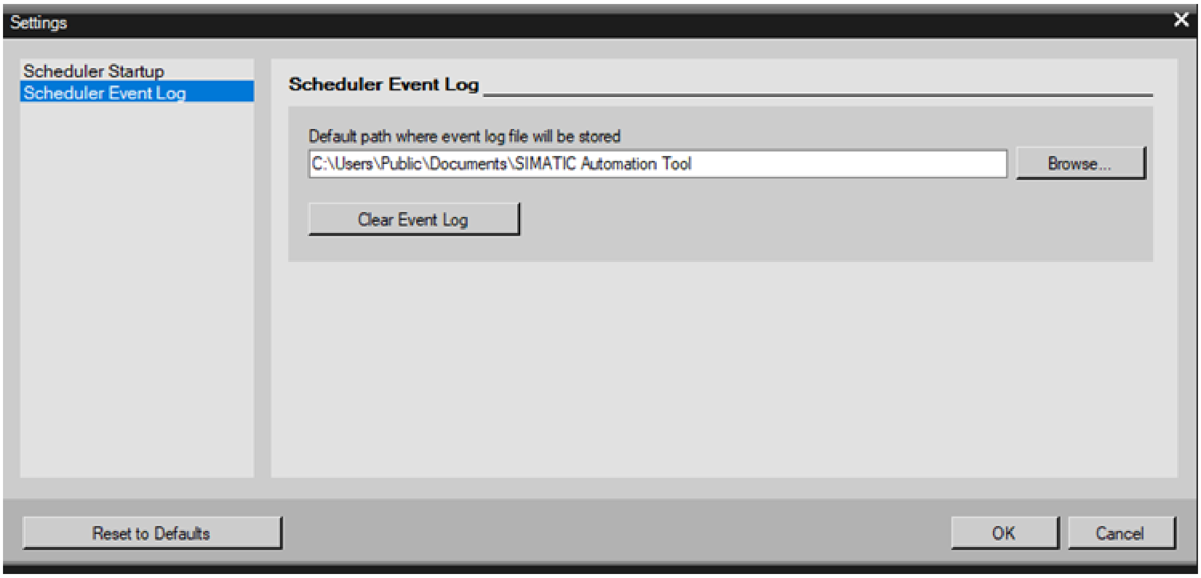
☑ Launch Scheduler automatically when your system starts

☑ Launch Scheduler minimized

If you do not automatically launch the scheduler when you log in to Windows, then you must launch the scheduler application as described previously.

In minimized operation, the Scheduler operation runs operations in the background.

Scheduler Event Log


Settings

Scheduler Startup
Scheduler Event Log

Scheduler Event Log

Default path where event log file will be stored

C:\Users\Public\Documents\SIMATIC Automation Tool

Viewing the Scheduler Event Log

To view the Scheduler Event Log, select "View Event Log" from the scheduler application shortcut menu. The Event Log opens in the application you have associated with .csv files. The Scheduler Event Log displays the following information for each executed operation:

- Success or error
- Date and time of operation

- Type of operation
- Name of device
- MAC address of device
- IP address of device
- Reason for error if operation failed

Note

Considerations when viewing the Scheduler Event Log

Close the scheduler Event Log after you view it. If you leave the file open on your programming device, the scheduler cannot write new event entries to this file. If the scheduler cannot access the file because the file is open, the scheduler will create a new file with the new event entries.

If you have left the scheduler Event Log open and select "View Event Log", you might not see the latest event entries. In this case, browse to the folder on your programming device that currently contains the Scheduler Event Log files. Open the file that corresponds to the events you want to view.

You can find the folder path in for the Scheduler Event Log in the Scheduler Event Log Settings. If you have changed the path in the Scheduler Event Log Settings, old Scheduler Event Log files might be on your programming device in their former location.

Effect of changes to system time on scheduled operations

The system time on your programming device can change due to daylight saving time in some regions or manual changes to the system time. The scheduler application has the following behaviors in the event of a system time change:

- Operations that are already queued run to completion. The system time change does not affect the order of operations in the queue. The scheduler application adds new queued operations to the end of the queue.
- If a system time change sets the time back in time, note the following:
  - Operations that already completed do not run again. A "Run Once" operation, for example, that has already run does not run again. Similarly, an "Every Week" operation that has already run for the week does not run again. The same principle applies for other schedule frequencies.
  - Operations with a schedule time that were missed have a chance to run again. When the schedule time occurs, the scheduler application queues the missed operation to run. Recurring operations can run two or more times depending on the time change.
- If a system time change sets the time forward, operations can be missed one or more times. Operations with a schedule time that were missed do not execute.

In all cases, the following conditions are true:

- Invalid operations remain invalid.
- Disabled operations remain disabled.

- Completed operations remain completed.
- The scheduler application executes scheduled operations based on the new system time.

Exiting the scheduler application

If you select "Exit" from the SIMATIC Automation Tool Scheduler shortcut menu in the notification area of the Windows Taskbar, the scheduler application stops running. Operations that are currently running run to completion. Queued and future scheduled operations do not run. Take care when you exit the scheduler application because operations can be in progress.

Note

Exiting the scheduler application

Do not use the Windows Task Manager to end the scheduler application. Forcing the scheduler to terminate unexpectedly while device operations are in progress could leave devices in an indeterminate state. You might have to power cycle devices to restore communication with the SIMATIC Automation Tool.

Use only the SIMATIC Automation Tool Scheduler shortcut menu to exit the scheduler application.

If you click the X in the top right corner of the SIMATIC Automation Tool Scheduler window, the scheduler application user interface closes, but the scheduler application continues running in the background.

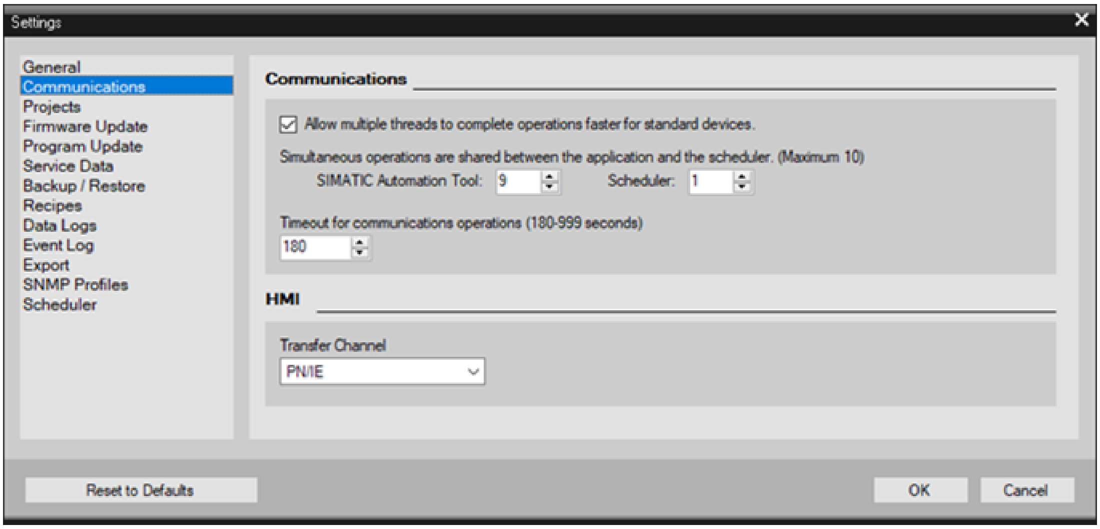
Updating the scheduler configuration file

If you change device information for a device with a scheduled operation, download the scheduler configuration file (Page 106). The scheduler then has the current information for the device.

9.4 Advanced multi-threading strategies

You select whether to allow multiple threads in the Communications settings (Page 82) dialog.

If you have an Advanced license (Page 101), you can distribute the threads between the SIMATIC Automation Tool and the Scheduler (Page 107).



☑ Allow multiple threads to complete operations faster for standard devices.

Simultaneous operations are shared between the application and the scheduler. (Maximum 10)

SIMATIC Automation Tool: 9 Scheduler: 1

Timeout for communications operations (180-999 seconds)

180

Transfer Channel

PN/IE

Consider these recommendations for sharing simultaneous operations between the SIMATIC Automation Tool and the Scheduler:

- If you do not use the Scheduler application, allocate the maximum number of simultaneous operations to the SIMATIC Automation Tool.
- If you use the Scheduler application for many frequent operations on many devices, allocate the maximum number of simultaneous operations to the Scheduler.
- If you plan to use the SIMATIC Automation Tool and the Scheduler application about equally, then allocate an equal number of simultaneous operations to each.

You can, of course, set other proportions of simultaneous operations as appropriate.

9.5 Working with devices connected to CMs or CPs

The SIMATIC Automation Tool can communicate with CPUs through a CM (Communications Module) or CP (Communications Processor) connection. To enable this type of communication, follow these steps:

1. From the TIA Portal, open the Device Configuration for the CM or CP in the STEP 7 project.
2. In the "Communication types" setting of the Properties, select "Enable online functions".
3. In the "SNMP" setting of the properties, select "Enable SNMP".
4. Download the STEP 7 project to the CPU.

You can connect your PROFINET network to the Ethernet interface of the CM or CP. The SIMATIC Automation Tool will find the connected CPU on a network scan (Page 21).

Device table representation of a CPU connected through a CM or CP

When communicating to a CPU through the Ethernet interface of a CM or CP, the MAC address and IP address in the Device table for the CPU are the MAC address and IP address of the CM or CP. The serial number and all other data in the Device table row are the data from the CPU. The Device table row then looks like the CPU is directly attached to the SIMATIC Automation Tool. The only difference is that the MAC address and IP address belong to the CM or CP.

Limitations

The SIMATIC Automation Tool prohibits the following device operations to CPUs connected through a CM or CP:

- Program Update (Page 48)
- Firmware Update (Page 40)
- Full Backup (Page 57)
- Restore Device (Page 59)
- Reset to Factory Defaults (Page 36)
- Format Memory Card (Page 39)

Also, you cannot schedule (Page 106) a firmware update or backup for a CPU connected through a CM or CP.

9.6 About devices behind routers

A network scan cannot find devices behind routers. To insert a device behind a router, insert it by entering the device's IP address (Page 25).

Open port requirements for NAT routers

To use CPUs and HMIs behind NAT routers you must open the following ports, based on the device.

Device type	Ports to open
CPU	102 161
HMI with Ethernet transfer channel (Page 82)	5001 161
HMI with PN/IE transfer channel (Page 82)	102 161

Limitations of devices behind IP and NAT routers

The following DCP (Discovery and Configuration Protocol) operations are not supported for devices behind routers:

- Set IP Address (including Subnet and Gateway) (Page 32)
- Set PROFINET Name (Page 33)
- Identify device (Page 27)
- Reset Communication Parameters (Page 35)

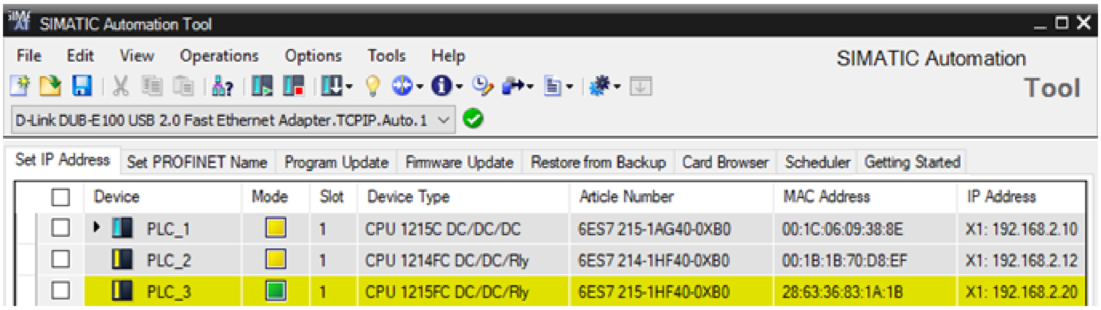
If a device behind the router connects to a network through a CM (Communications Module) or CP (Communication Processor), then additional restrictions (Page 113) are in effect.

The program update (Page 48) and restore (Page 59) operations have additional limitations for devices behind NAT routers.

Additional features 10

10.1 Device table power features

After you scan the network (Page 21), the Device table shows the connected devices. The Device table displays devices behind routers in blue text. For each device, the SIMATIC Automation Tool displays columns with data about the device. Tabs in the Device table support a variety of device operations (Page 21) and provide data entry fields:



☐	Device	Mode	Slot	Device Type	Article Number	MAC Address	IP Address
☐	PLC_1	🟨	1	CPU 1215C DC/DC/DC	6ES7 215-1AG40-0XB0	00:1C:06:09:38:8E	X1: 192.168.2.10
☐	PLC_2	🟨	1	CPU 1214FC DC/DC/Rly	6ES7 214-1HF40-0XB0	00:1B:1B:70:D8:EF	X1: 192.168.2.12
☐	PLC_3	🟩	1	CPU 1215FC DC/DC/Rly	6ES7 215-1HF40-0XB0	28:63:36:83:1A:1B	X1: 192.168.2.20

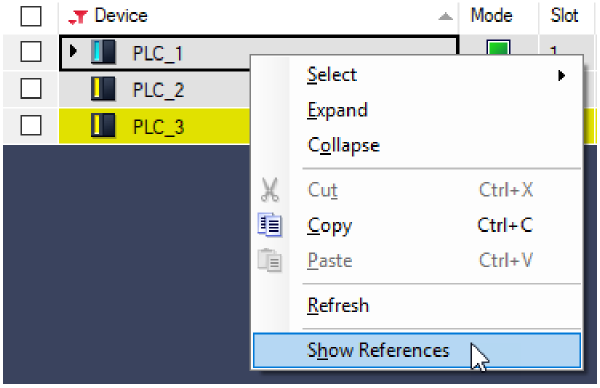
Working with the Device table

The following tips can help you use the Device table:

- Select one or more devices for operations to perform. The SIMATIC Automation Tool displays selected device rows in bold text.
- You can scan the network to fill the Device table with devices on your network. You can also insert devices (Page 25).
- Click the arrow to the left of a device to expand it. You then see more information about the device such as local I/O, distributed I/O, and Data Logs.
- Select the Device check box at the top of the Device table to select or deselect all devices. Alternatively, you can use the "View > Select" menu command to choose either "Select Row(s)" or "Deselect Row(s)". You can also right-click a device row to access the shortcut menu.
- Click a column header to sort the rows by that column's data.
- Right-click a column header to show/hide columns, resize columns, or choose a column for sorting.
- Click the cell at the left of a row's check box to select an entire row. Drag the cursor up or down from the selection to select multiple rows.
- Click the cell at the left of the Device check box to select all the rows in the Device table.
- You can also export (Page 68) a Device table to a .csv file for viewing and editing in Microsoft Excel or another tool.

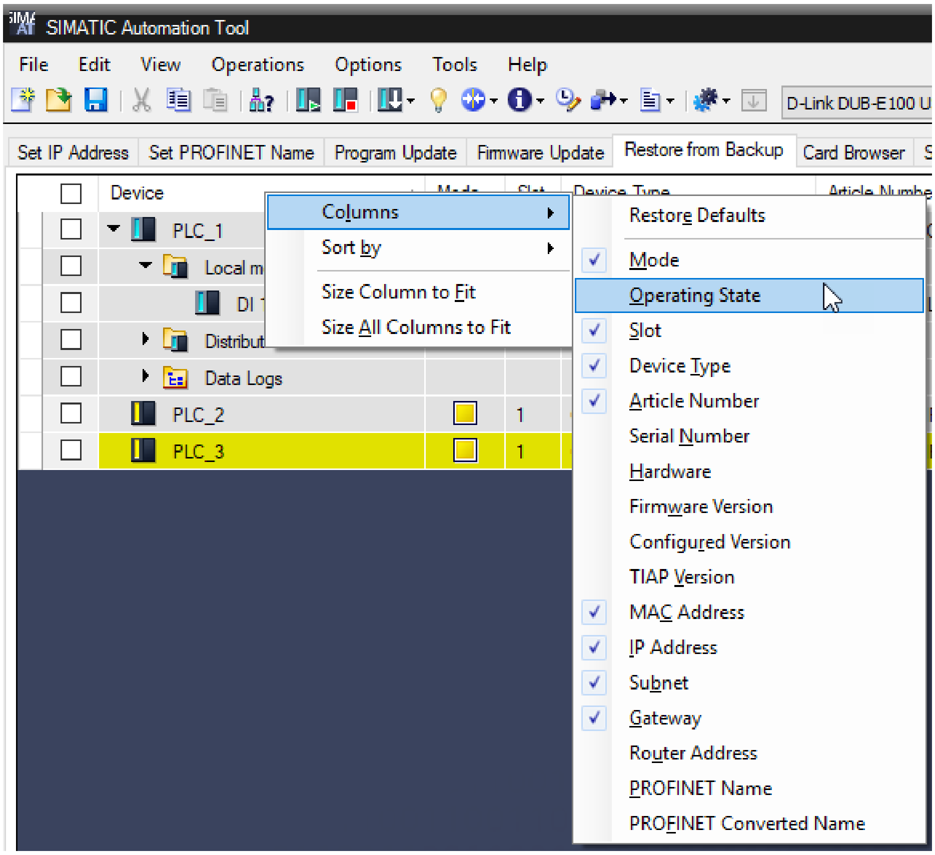
Shortcut menu for table cells

When you right-click a device row or cell in the Device table, the shortcut menu includes menu selections that are specific for your selection.



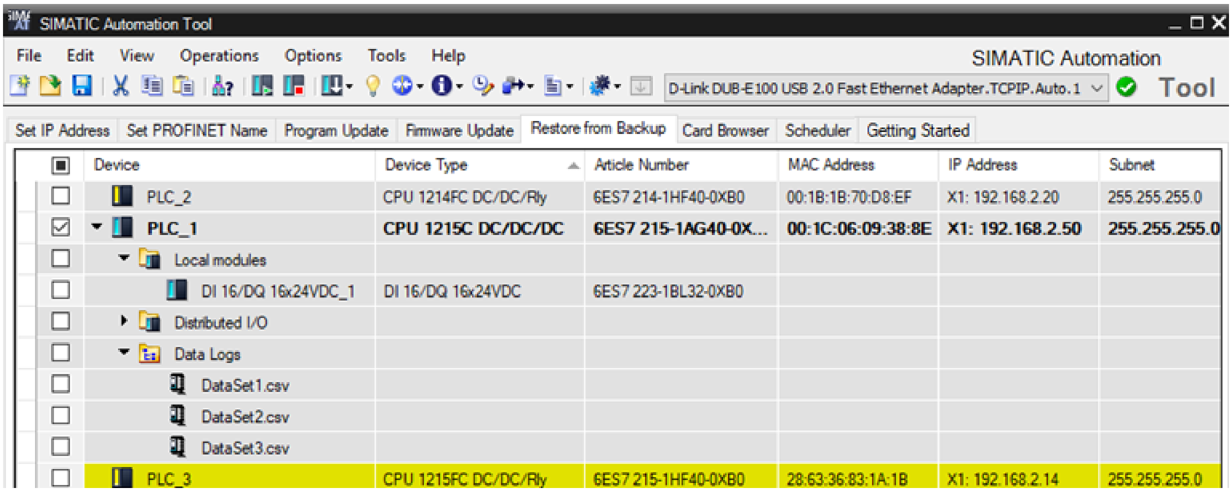
Shortcut menu for column headers

For each tab of the Device table, the SIMATIC Automation Tools displays a set of columns by default. You can right-click the header row to choose which columns to show or hide.



Expanding and collapsing device rows

Click the expand icon ▸ to expand a Device table row and see local modules, distributed I/O devices, and Data Logs. Click the collapse icon to ▾ collapse a Device table row. Use the right-click shortcut menu or Edit menu to expand or collapse all levels:



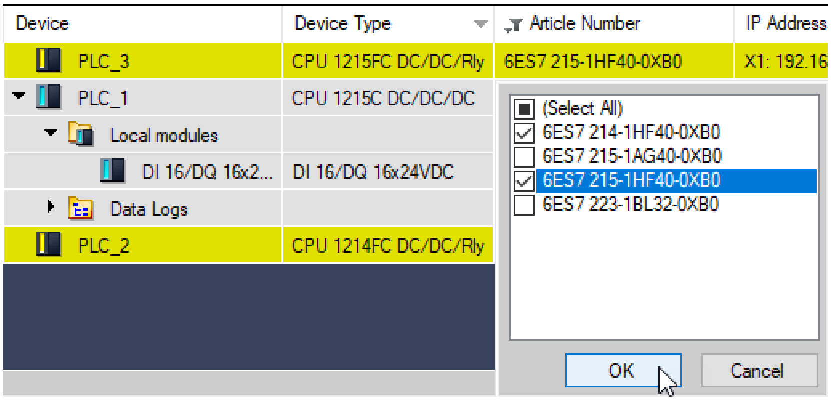
D-Link DUB-E100 USB 2.0 Fast Ethernet Adapter.TCPIP.Auto.1

■	Device	Device Type	Article Number	MAC Address	IP Address	Subnet
☐	PLC_2	CPU 1214FC DC/DC/Rly	6ES7 214-1HF40-0XB0	00:1B:1B:70:D8:EF	X1: 192.168.2.20	255.255.255.0
☑	▾ PLC_1	CPU 1215C DC/DC/DC	6ES7 215-1AG40-0X...	00:1C:06:09:38:8E	X1: 192.168.2.50	255.255.255.0
☐	▾ Local modules					
☐	DI 16/DQ 16x24VDC_1	DI 16/DQ 16x24VDC	6ES7 223-1BL32-0XB0			
☐	▸ Distributed I/O					
☐	▾ Data Logs					
☐	DataSet1.csv					
☐	DataSet2.csv					
☐	DataSet3.csv					
☐	PLC_3	CPU 1215FC DC/DC/Rly	6ES7 215-1HF40-0XB0	28:63:36:83:1A:1B	X1: 192.168.2.14	255.255.255.0

Filtering the displayed rows

You can filter the Device, Device Type, and Article Number columns. Click the filter icon ▽ next to one of these three column headers to open the filter window.

For example, you can select article numbers 6ES7 214-1HF40-0XB0 and 6ES7 215-1HF40-0XB0. When you click the OK button, the Device table only displays rows that have these article number values:



Device	Device Type	Article Number	IP Address
PLC_3	CPU 1215FC DC/DC/Rly	6ES7 215-1HF40-0XB0	X1: 192.16
▾ PLC_1	CPU 1215C DC/DC/DC		
▾ Local modules			
DI 16/DQ 16x2...	DI 16/DQ 16x24VDC		
▸ Data Logs			
PLC_2	CPU 1214FC DC/DC/Rly		

■ (Select All)
☑ 6ES7 214-1HF40-0XB0
☐ 6ES7 215-1AG40-0XB0
☑ 6ES7 215-1HF40-0XB0
☐ 6ES7 223-1BL32-0XB0

If you filter by the Device Type or Article Number, the Device table shows all devices at the root level. With devices filtered, the Device table does not show a tree structure with modules beneath CPUs. Without such a tree structure representation, the SIMATIC Automation Tool disables all safety-relevant operations. You cannot perform safety-relevant operations if you have filtered the Device table on the Device Type or Article Number.

You can use the General settings (Page 81) to enable/disable the display of unsupported devices.

Filtering unsupported devices

The SIMATIC Automation Tool displays unsupported devices in gray. You can perform only the following operations on unsupported devices:

- Set IP address
- Set PROFINET name
- Identify device
- Reset communication parameters

10.2 Copying and pasting to multiple Device table cells

The SIMATIC Automation Tool makes it easy for you to copy the contents of a cell into many cells in the same column. For example, if you want to update the firmware (Page 40) in many devices of the same type, you can select the New Firmware Version file once and use a drag and drop technique to copy the selection to the New Firmware Version cell for the other devices.

This rapid copy and paste function works as one editing operation within a column. You do not have to copy repeatedly into multiple cells. Just like in Microsoft Excel, you can multi-select cells for pasting by dragging the mouse pointer through a selection of cells. The SIMATIC Automation Tool recognizes which cells are valid for the paste operation and which cells are not. The SIMATIC Automation Tool only pastes the copied content into cells that are valid.

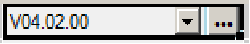
The rapid copy and paste function is available for any of the user-entry columns in the Device table.

How to use the rapid copy and paste feature

Consider a case where you have many CPUs of the same type (for example, many S7-1200 1215C DC/DC/DC CPUs). You want to update all of these CPUs to firmware version V04.02.00, which you have available in your firmware update folder (Page 84). You have other types of CPUs in your Device table as well. You do not want to copy the New Firmware Version manually to all of the S7-1200 1215C DC/DC/DC CPUs and skip the other devices.

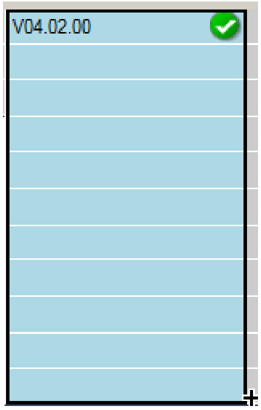
To copy a cell selection to multiple cells for this example, follow these steps:

1. Select the New Firmware Version (Page 40) for one of your S7-1200 1215C DC/DC/DC CPUs, perhaps the first one in your list.
2. When the SIMATIC Automation Tool shows that your selection is valid, click the New Firmware Version cell. The SIMATIC Automation Tool displays a focus rectangle for the cell. This is the source cell for your copy operation:



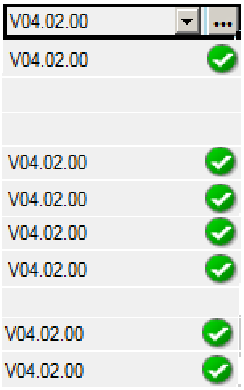
V04.02.00

3. Drag the bottom right corner of the focus rectangle through the New Firmware Version column. Like Microsoft Excel, the SIMATIC Automation Tool shows you which cells you are including in your new focus rectangle. You can select the entire column if you choose, or just a few cells:



V04.02.00 ✓

4. Release the left mouse button to copy your selection into all of the cells in the new focus rectangle for which the original selection is valid. In this example, the SIMATIC Automation Tool copies the New Firmware Version V04.02.00 cell into cells that correspond to S7-1200 1215C DC/DC/DC CPUs. The SIMATIC Automation Tool only copies to CPUs that are of the same CPU type and for which the firmware update is valid:



V04.02.00
V04.02.00 ✓

V04.02.00 ✓
V04.02.00 ✓
V04.02.00 ✓
V04.02.00 ✓

V04.02.00 ✓
V04.02.00 ✓

You can use the rapid copy and paste technique for any user-editable column in the Device table. The technique can save time for many operations where you want to copy one selection to many places, for example:

- Device selection
- Backup files
- Comments
- Firmware updates
- Gateway
- Passwords (CPU and Program file)
- Program updates
- Recipes
- Subnet

Copy and paste with incremented values

For new IP addresses and new PROFINET names, the copy and paste function increments the copied text when pasting into the destination cells:

- For new IP addresses, each address increments by one (+ 1) in the pasted cells.
- For new PROFINET names, the SIMATIC Automation Tool copies the new PROFINET name and adds a number increment to each name in the pasted cells.

Note

Hidden cells

If you have collapsed one or more expandable cell rows, the paste operation pastes into the hidden cells. When you expand the collapsed rows, you can see the pasted content.

10.3 Refreshing a device's data

To refresh Device table data, choose one of these methods:

- Select the "View > Refresh" menu command and then choose one of the following options from the Refresh menu command:
  - All Selected Devices F5
  - All Devices
- Right-click a device row and select Refresh from the shortcut menu.
- Press the F5 key to refresh "All Selected Devices"

The SIMATIC Automation Tool refreshes the device data that it reads from the devices and retains all user-entered data fields. If you refresh devices that are no longer present on the network, the SIMATIC Automation Tool displays the device row data in italics.

A refresh differs from a network scan (Page 21). A network scan scans the entire network. A refresh operation is only for the devices that you select.

Note

Refreshing selected devices

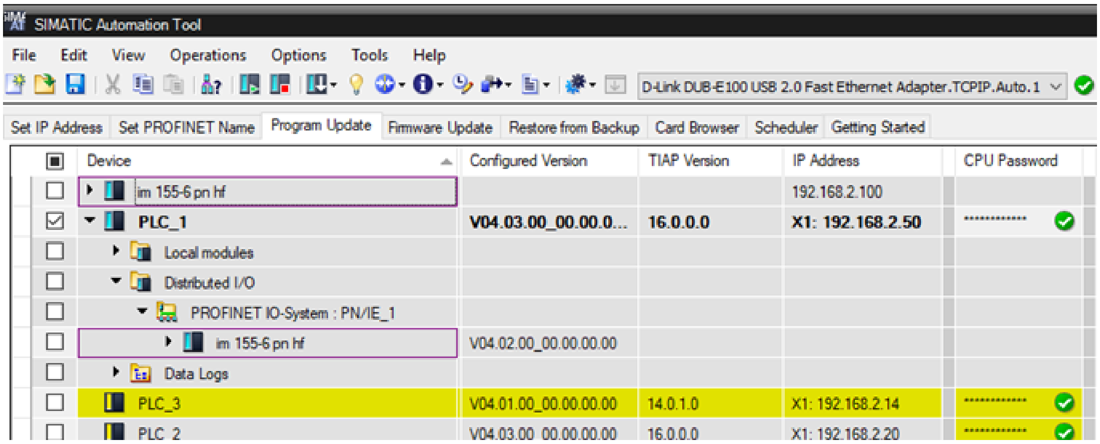
You can only refresh top level selected devices. If a device is connected beneath another device in the Device table, you cannot refresh that device alone.

For example, if you have an interface module (IM) that is distributed I/O to a CPU, you cannot refresh only the interface module. If you select this interface module and refresh it, the SIMATIC Automation Tool refreshes the CPU and all modules connected through it.

10.4 Showing device references

For any top-level device in the Device table, you can right-click the device and select Show References from the shortcut menu (Page 117).

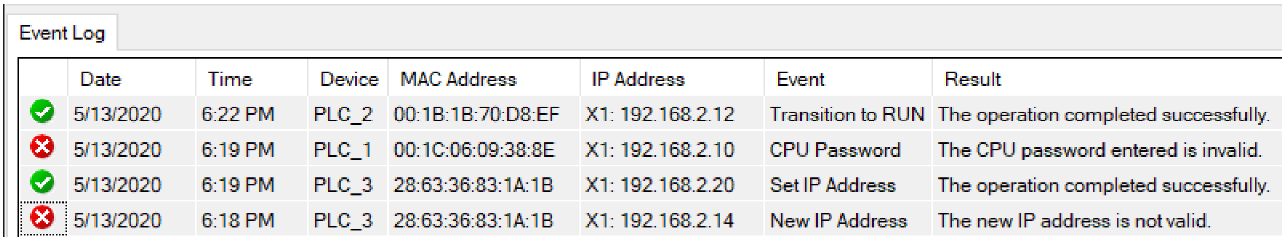
When you select Show References, the SIMATIC Automation Tool shows all references to the device that you selected based on the device name. If necessary, the SIMATIC Automation Tool expands device rows as necessary to show device references:



▣	Device	Configured Version	TIAP Version	IP Address	CPU Password
□	im 155-6 pn hf			192.168.2.100	
☑	PLC_1	V04.03.00_00.00.0...	16.0.0.0	X1: 192.168.2.50	************ ✓
□	Local modules				
□	Distributed I/O				
□	PROFINET IO-System : PN/IE_1				
□	im 155-6 pn hf	V04.02.00_00.00.00.00			
□	Data Logs				
□	PLC_3	V04.01.00_00.00.00.00	14.0.1.0	X1: 192.168.2.14	************ ✓
□	PLC_2	V04.03.00_00.00.00.00	16.0.0.0	X1: 192.168.2.20	************ ✓

10.5 Understanding the Event Log

The SIMATIC Automation tool displays the Event Log in the window area below the Device table. When you scan the network, or select devices and perform operations, the messages in the Event Log (Page 125) show results of operations. Each message is for a specific event for a specific device



	Date	Time	Device	MAC Address	IP Address	Event	Result
✓	5/13/2020	6:22 PM	PLC_2	00:1B:1B:70:D8:EF	X1: 192.168.2.12	Transition to RUN	The operation completed successfully.
✖	5/13/2020	6:19 PM	PLC_1	00:1C:06:09:38:8E	X1: 192.168.2.10	CPU Password	The CPU password entered is invalid.
✓	5/13/2020	6:19 PM	PLC_3	28:63:36:83:1A:1B	X1: 192.168.2.20	Set IP Address	The operation completed successfully.
✖	5/13/2020	6:18 PM	PLC_3	28:63:36:83:1A:1B	X1: 192.168.2.14	New IP Address	The new IP address is not valid.

Refer to the Event Log settings (Page 87) for configuration settings for the Event Log.

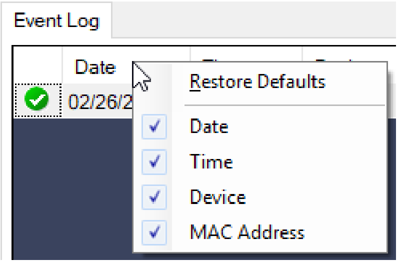
Types of Event Log messages

The icons in the Event Log have the following meanings:

✓	Operation is successful
✖	Operation has failed. The Result column describes the reason for failure. If you save the Event Log, these entries begin with "FAILED:".
⚠	Operation is successful but includes a warning message. The Result column describes the warning information. If you save the Event Log, these entries begin with "WARNING:".
ℹ	Information only

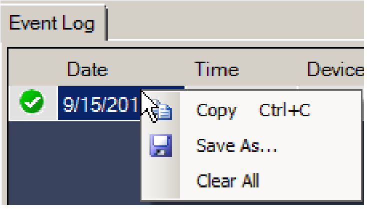
Showing and hiding columns

Right-click an Event Log column header to show or hide columns:



Copying, saving, and clearing Event Log messages

Right-click an event row to copy, save, or clear Event Log messages:



Note

Event Log and user interface language change

When you change the SIMATIC Automation Tool user interface language, the SIMATIC Automation Tool clears the Event Log. Information about previous events is no longer available.

Troubleshooting Event Log messages 11

The following table provides additional information for Event Log messages. If you need more information about a message, copy the message from the Event Log and search the online help for the message text.

Event Log message	Probable cause and corrective action
The CPU password entered is not valid to perform this operation.	You entered an invalid CPU password (Page 308) for the operation. Enter a valid CPU password with sufficient privileges for the operation.
The CPU failed to enter RUN mode. Possible reasons: TIA Portal project downloaded might not be valid for this CPU type or a program has never been downloaded to the CPU. See the diagnostics buffer for more details.	The CPU could not go to RUN mode. The CPU might not have a valid user program. Check the CPU diagnostics for more details.
The CPU failed to enter STOP mode.	The CPU could not go to STOP mode. Check the CPU diagnostics for more details.
The device is not accepting the new configuration.	The device rejected a DCP command to set the IP address (Page 32) or PROFINET name (Page 33). Check network communication connections. Verify the IP address or the PROFINET name that you entered.
Failed to connect to the device. Verify the MAC address is correct.	The SIMATIC Automation Tool could not identify (Page 27) the device on the network with that MAC address. Verify the MAC address you entered and the MAC address of the device.
The connection to the device was lost prematurely.	The SIMATIC Automation Tool is no longer connected to the device. Verify network communication connections.
The operation failed and returned error: ID#	Internal error within the SIMATIC Automation Tool. Contact your Siemens representative.
Failed to disconnect	The SIMATIC Automation Tool was unable to disconnect from the device. Reattempt the operation.
The specified firmware update file is not compatible with the device.	The firmware update file (Page 40) that you selected is not compatible with this device. Select a firmware update version that corresponds to the specific device.
The specified firmware update file is not compatible with the device because the hardware requires a firmware version that must be different.	The device hardware requires that the firmware update file (Page 40) differ from the existing version. Select a firmware update version that is different from the existing version.
The specified firmware update file is not compatible with the device because the hardware requires a firmware version that must be the same.	The device hardware requires that the firmware update file (Page 40) be the same as the existing version. Select a firmware update version that is the same version.
The specified firmware update file is not compatible with the device because the hardware requires a firmware version that is newer.	The device hardware requires that the firmware update file (Page 40) must have a newer version than the version in the device. Select a newer firmware update version.

The specified firmware update file is not compatible with the device because the hardware requires a firmware version that is older.	The device hardware requires that the firmware update file (Page 40) must have an older version than the version in the device. Select an older firmware update version.
The module ID is not valid.	The API FirmwareUpdate method (Page 208) received a value for the hardwareID parameter that does not correspond to a module. When calling the FirmwareUpdate update method, provide the correct ID for the module.
The firmware update file was not accepted by the module.	Select a firmware update (Page 40) file that is compatible with the module you are updating.
The module is not reachable. Download a valid hardware configuration to the CPU or connect directly to the device.	The SIMATIC Automation Tool cannot communicate with the device. Download a valid device configuration from the TIA Portal to the device. Alternatively use the SIMATIC Automation Tool to update the CPU program (Page 48).
Firmware update of this module is not supported by this tool.	The SIMATIC Automation Tool does not support firmware update (Page 40) for this type of module. You cannot update the firmware for this module with the SIMATIC Automation Tool.
The firmware update file has the same firmware version as the device.	The firmware update (Page 40) file has the same firmware version as the device. No firmware update performed and no further action required.
Format memory card is not supported on this device.	You cannot format the memory card for this device.
The gateway address is not valid.	The gateway address is not valid with the subnet mask and IP address. Verify that your IP address, subnet mask, and gateway (Page 32) are correct for your device and network.
An internal error has occurred.	Internal error within the SIMATIC Automation Tool. Contact your Siemens representative.
Target server address is invalid.	The SIMATIC Automation Tool could not connect to the CPU or HMI device. Verify the MAC address or IP address of the device.
The project cannot be opened with this version of the SIMATIC Automation Tool.	The project is from an older release of the SIMATIC Automation Tool. The SIMATIC Automation Tool cannot open a project from this release. See the topic "Saving and opening .sat project files (Page 65)".
Invalid signature detected. Repair the installation.	The SIMATIC Automation Tool installation is incomplete or corrupted. Reinstall the SIMATIC Automation Tool.
The IP address is not valid.	The IP address does not follow requirements for IP addresses. Enter a valid IP address (Page 32).
The MAC address is not valid.	The MAC address does not follow requirements for MAC addresses. Enter a valid MAC address.
This operation is not permitted because another configuration tool such as the TIA Portal is connected.	Only one software tool can connect to a device. Close the other connection to the device, for example, by going offline in the TIA Portal. Then reattempt the operation from the SIMATIC Automation Tool.

An already active ES instance does not support MultiES.	This device does not support more than one online connection from the SIMATIC Automation Tool, the TIA Portal, or another tool at the same time. Go offline in the TIA Portal or other tool and then reattempt the SIMATIC Automation Tool operation.
The maximum number of CPU connections has been exceeded.	This device supports a limited number of online connections from the SIMATIC Automation Tool, the TIA Portal, or another tool at the same time. The limit has been exceeded. Go offline in the TIA Portal or other tool and then reattempt the SIMATIC Automation Tool operation.
Connections to this CPU from multiple tools is not supported.	This device supports a limited number of online connections from the SIMATIC Automation Tool, the TIA Portal, or another tool at the same time. Connections from multiple tools is not supported. Go offline in the TIA Portal or other tool and then reattempt the SIMATIC Automation Tool operation.
Client wants to make write operation on object variable or link which cannot be changed in CPU RUN Mode.	The CPU cannot perform this operation in RUN mode. Change the CPU to STOP mode (Page 30).
The operation completed successfully.	Information only
The operation was canceled by the user.	Information only
The device does not support the requested operation.	The Event column shows the operation that did not succeed for the device in the Device column. Check the Device catalog (Page 95) for the operations that the device supports for the current firmware version.
Parameter is out of range.	An application called the API method SetOperatingState (Page 257) with an invalid value. You must call this method with either "Stop" or "Run" for the operating mode state.
The PROFINET name is not valid.	The PROFINET name (Page 33) is not valid. Enter a valid PROFINET name that is unique from all other devices.
Could not establish a connection to the device	The SIMATIC Automation Tool could not connect to the device. Check your network connections. Verify that another software tool, such as the TIA Portal, does not have an online connection to the device. If so, go offline from the other tool. Verify that the device is on the subnet (Page 32) you specified. Check your routing if applicable. Set your network interface (Page 11) to the "Auto" selection if you haven't already.
The operation has timed out.	The CPU did not complete the operation. Reattempt the operation. If the operation continues to time out, check your device and network communications.
Connection is de-legitimated because CPU password timeout.	The CPU password (Page 308) has timed out due to inactivity. Enter the CPU password and try the operation again.

Failed to set IP address	The SIMATIC Automation Tool could not set the IP address (Page 32). Check that the IP address is valid and that it is unique. Verify that the device configuration in the STEP 7 project permits setting the IP address directly at the device.
Failed to set PROFINET name	The SIMATIC Automation Tool could not set the PROFINET name (Page 33). Check that the PROFINET name is valid and that it is unique. Verify that the device configuration in the STEP 7 project permits setting the IP address directly at the device.
The subnet mask is not valid.	The subnet mask (Page 32) does not follow requirements for a valid subnet mask. Enter a valid subnet mask.
The device is not supported or could not be initialized.	The SIMATIC Automation Tool does not support this device, or could not initialize the device. Check the Device catalog (Page 95) for supported devices and versions.
The device did not accept the firmware update file.	The device rejected the firmware update (Page 40). Verify that the firmware update file is valid for your device. If updating from the old format, ensure that you select the header .upd file for the "New Firmware Version" field. Some modules require external power sources. If the module has dual processors, verify that power is applied to both processors. If the module has a power LED indicator, verify that the LED is green. Test that the device is functioning properly.
Failed to zip contents of folder	The SIMATIC Automation Tool could not zip the contents of the folder. Check for sufficient file system space for the zip file. Also, check for file folder permissions. Either condition could be the source of the problem.
Error writing to file	The SIMATIC Automation Tool could not save the file to the programming device. Check your file space and file permissions.
Error creating file	The SIMATIC Automation Tool could not create the file on the programming device. Check your file space and file permissions.
Error deleting file	The SIMATIC Automation Tool could not delete the file from the programming device. Check your file permissions.
Error deleting folder	The SIMATIC Automation Tool could not delete the folder from the programming device. Check the folder permissions on your programming device. Be sure the folder is not open.
Error creating folder	The SIMATIC Automation Tool could not create a temporary folder for saving service data. Check file space on your programming device.
You must purchase a license to use the API.	You do not have a valid license to use the API (Page 149). You can use the 21-day free trial license for 21 days. Otherwise, purchase a license for the SIMATIC Automation Tool or purchase the Software Developers Kit (SDK) to use the API.

Invalid timeout value. Valid values are 180-999 seconds.	In the Communications options (Page 82), the timeout value for communications operations is out of range. Set the value to a number from 180 to 999 seconds.
No data in device to backup	The SIMATIC Automation Tool cannot create a backup (Page 57) if the device does not have a user program. Download your project from STEP 7 to the device, or update the program (Page 48) from the SIMATIC Automation Tool. If you have a previous backup file for this device, you can restore (Page 59) it from the SIMATIC Automation Tool.
Error reading from stream	The SIMATIC Automation Tool could not read from a previously-saved project file (Page 65). Verify that the project file is from a compatible version of the SIMATIC Automation Tool. Open a valid project if one is available.
Error writing to stream	The SIMATIC Automation Tool could not write and save a project file (Page 65) for the current project. Check file space on your programming device.
The HMI project file is invalid or incomplete.	You are attempting to perform an HMI program update (Page 48). The program update file is corrupt. You cannot restore this file.
The HMI project is not compatible with the device.	You are attempting perform a program update (Page 48) to an HMI device that is not compatible with the HMI device. You can only perform a program update that corresponds to your HMI device hardware.
A file operation could not be completed because of write protection.	The location for the file operation is write-protected. Choose a folder location on your programming device that is not write-protected.
Failed to change the IP address of this device	The SIMATIC Automation Tool could not set the IP address (Page 32). Enter a valid IP address. Check your network communication connections.
Failed to change the PROFINET name of this device	The SIMATIC Automation Tool could not set the PROFINET name (Page 33). Enter a valid PROFINET name. Check your network communication connections.
Request aborted	Information only
The device does not yet support device addon transfer.	The program update (Page 48) file includes addons. The HMI device you are trying to update does not support addons. You can only update this HMI device with a program that does not have addons.
Attribute not found	An HMI operation is attempting to read or write to a location that does not exist. Verify that the program update file (Page 48) or backup file (Page 57) is valid for the HMI.
Buffer too small for all data requested	The HMI device reported that an internal buffer was too small. Contact customer support for your HMI device.
HMI certificate has expired.	The HMI certificate is no longer valid. Update the certificate in the HMI device.

General failure on checking certificate signature	Internal error in the HMI device. Contact customer support for your HMI device.
File certificate is invalid.	Internal error in the HMI device. Contact customer support for your HMI device.
Certificate not yet valid	Internal error in the HMI device. Contact customer support for your HMI device.
Certificate has been revoked.	Internal error in the HMI device. Contact customer support for your HMI device.
Established client connection required for action	Internal error in the HMI device. Contact customer support for your HMI device.
A file read operation failed.	Internal error in the HMI device. Contact customer support for your HMI device.
A file write operation failed.	Internal error in the HMI device. Contact customer support for your HMI device.
Firmware of type not installed	Internal error in the HMI device. Contact customer support for your HMI device.
Firmware of type not supported	Internal error in the HMI device. Contact customer support for your HMI device.
The firmware file is not valid for this device.	You selected a firmware update file (Page 40) that is not valid for the device. Choose a firmware update file that corresponds to your device and version.
This addon is not compatible with this device/firmware	The program update file (Page 48) includes addons. The addon in the project update file is not compatible with the HMI device that you are trying to update. You can only update a program that has no addons or has compatible addons.
Invalid arguments passed to method	Internal error in the HMI device. Contact customer support for your HMI device.
Invalid file path	The file path does not exist or is invalid. Provide a valid file path.
License control failed.	Internal error in the HMI device. Contact customer support for your HMI device.
A logical volume is missing.	Internal error in the HMI device. Contact customer support for your HMI device.
A logical volume is out of space.	Internal error in the HMI device. Contact customer support for your HMI device.
Object not found	Internal error in the HMI device. Contact customer support for your HMI device.
Not enough resources to run requested operation	The requested operation requires more system resources than are available. Close some applications on your programming device and try again.
Insufficient space in file system	The file system does not have enough space for the requested operation. Delete files to create some space on the file system or choose a different file system that has more space.
Remote transfer disabled	Internal error in the HMI device. Contact customer support for your HMI device.
Rescue backup not possible, for example due to wrong device state or wrong firmware file	Internal error in the HMI device. Contact customer support for your HMI device.

Rescue restore not possible, for example due to wrong device state	Internal error in the HMI device. Contact customer support for your HMI device.
Runtime installation is broken.	Internal error in the HMI device. Contact customer support for your HMI device.
Runtime is not installed.	Internal error in the HMI device. Contact customer support for your HMI device.
General problem with security library handling	Internal error in the HMI device. Contact customer support for your HMI device.
Another service is already active.	Internal error in the HMI device. Contact customer support for your HMI device.
General failure on checking file signature	Internal error in the HMI device. Contact customer support for your HMI device.
File signature is invalid.	Internal error in the HMI device. Contact customer support for your HMI device.
File is not signed.	Internal error in the HMI device. Contact customer support for your HMI device.
A store read operation failed.	Internal error in the HMI device. Contact customer support for your HMI device.
A store write operation failed.	Internal error in the HMI device. Contact customer support for your HMI device.
Type conversion failed	Internal error in the HMI device. Contact customer support for your HMI device.
An unexpected operating system error occurred.	Internal error in the HMI device. Contact customer support for your HMI device.
An addon with the given name is not installed.	Internal error in the HMI device. Contact customer support for your HMI device.
An application with the given id is not installed.	Internal error in the HMI device. Contact customer support for your HMI device.
An application addon with the given id is not installed.	Internal error in the HMI device. Contact customer support for your HMI device.
An application addon references an application that is not installed.	Internal error in the HMI device. Contact customer support for your HMI device.
The recipe filename is not valid.	Check the file name. Provide a valid file name.
Not enough storage space to perform this operation	The SIMATIC memory card in the CPU does not have enough space to store the file. Delete extraneous content from the memory card or use a memory card with a higher capacity.
A more recent incompatible firmware image exists on the target device. If you update the operating system, then recipe data, user data, and some system settings could be permanently deleted.	The firmware on the device is more recent than the firmware update (Page 40) you have chosen. If you proceed with this firmware update, you risk deleting recipe data, user data, and some system settings. Verify that you want to update to this firmware before proceeding.
An older incompatible firmware image exists on the target device. If you update the operating system update recipe data user data and some system settings could be deleted permanently.	Internal error in the HMI device. Contact customer support for your HMI device.
A more recent incompatible Runtime exists on the target device.	Internal error in the HMI device. Contact customer support for your HMI device.

An older incompatible Runtime exists on the target device.	Internal error in the HMI device. Contact customer support for your HMI device.
No Runtime is installed on the target device.	Internal error in the HMI device. Contact customer support for your HMI device.
Not enough memory on the target device.	Internal error in the HMI device. Contact customer support for your HMI device.
Incorrect display orientation (Portrait configured Landscape on the HMI device).	Internal error in the HMI device. Contact customer support for your HMI device.
Incorrect display orientation (Landscape configured Portrait on the HMI device).	Internal error in the HMI device. Contact customer support for your HMI device.
General error. Compile the project again and repeat the download.	Internal error in the HMI device. Contact customer support for your HMI device.
General error. Repeat the download.	Internal error in the HMI device. Contact customer support for your HMI device.
No Runtime is installed on the target device.	Internal error in the HMI device. Contact customer support for your HMI device.
The configured device does not match the target device.	Internal error in the HMI device. Contact customer support for your HMI device.
A non-compatible Runtime is installed on the target device.	Internal error in the HMI device. Contact customer support for your HMI device.
An incorrect runtime version is installed. The installed runtime cannot process the project. Install the runtime manually on the target device.	Internal error in the HMI device. Contact customer support for your HMI device.
An error occurred and the device identity could not be determined.	The SIMATIC Automation Tool could not determine the identify of this CPU.
The device identity was changed. Scan the network again.	The CPU device identity has changed from when the SIMATIC Automation Tool connected to the CPU. Rescan the network (Page 21) to establish the actual device identity for the SIMATIC Automation Tool.
The connected interface was not found.	The selected Network Interface no longer exists. Check the connections of your network interface and test that it is working. When you know that the network interface is valid, select this network interface (Page 312) from the SIMATIC Automation Tool.
Device is not selected.	No device is selected. You must select a device from the Device table.
Safety-relevant operations require the safety password.	You must provide the safety password (Page 308) to perform the safety-relevant operation.
Internal application error	An internal error occurred when setting a CPU password (Page 258). Set a valid CPU password and check whether the device is protected.
The CPU password entered is invalid.	Enter a valid CPU password (Page 308).
The IP address of the device is a duplicate on the network.	The SIMATIC Automation Tool found a duplicate IP address on the network, perhaps from an external change. Sort your Device table (Page 117) by the IP address to find the duplicate. Correct one of the addresses to eliminate the duplicate.

The IP address entered is a duplicate on the network.	You entered an IP address that is a duplicate of another IP address. Enter an IP address that is unique on the network. Sort your Device table (Page 117) by the IP address to find the duplicate.
Duplicate PROFINET name	Enter a PROFINET name that is unique on the network.
Safety-relevant operations must be confirmed.	To perform any of the following operations for an F-CPU, you must confirm the operation: - Reset to factory defaults (Page 36) - Restore (Page 59) - Program update (Page 48) - Format memory card (Page 39) If you reached this message from an application using the API, you must set the SelectedConfirmed property (Page 231) for the CPU.
There is no SD card present or the SD card is already empty.	Formatting a memory card (Page 39) requires the presence of a SIMATIC memory card. Insert a SIMATIC memory card in the CPU. If the card is present, but has no file contents, you can ignore this message.
The program file is invalid or cannot be opened by this version of the SIMATIC Automation Tool.	The SIMATIC Automation Tool cannot access the program update file (Page 48) at the path specified in the Options dialog. Verify the path name in the Program Update settings of the Options dialog.
Result of CRC comparison online and offline collective F-signatures do not match.	The program update (Page 48) failed. The program update file does not match the program in the CPU after the program update operation. Try these steps: 1. Reset the CPU to factory defaults 2. Format the memory card if you are using one. 3. Reattempt the program update. If the error persists, you cannot use this program update file.
The program file cannot be downloaded to an older device.	The device version of the target device is older than the version in the program file. Upgrade the device to a newer version or choose a program update file (Page 48) that is compatible for the older device.
Projects IP suite is not reachable	Verify the IP address, subnet mask, and gateway (Page 32). Correct any values that are incorrect for your network.
IP suite is not reachable.	Verify the IP address, subnet mask, and gateway (Page 32). Correct any values that are incorrect for your network.
The backup file contained an IP suite that is not reachable via PROFINET communications.	After the restore (Page 59) operation, the SIMATIC Automation Tool cannot access the device. The IP suite (Page 32) in the restored project (IP address, subnet mask, gateway) does not correspond to the actual device and subnet. Correct and set the IP suite values for the device. Set the Network Interface Card to the "Auto" selection if you haven't already.

The IP address in the new program file is not unique to the network.	Resolve the conflict in IP addresses. Set one of the devices to a different IP address.
A safety program cannot be loaded to a standard CPU.	You cannot download a safety program to a standard CPU. You can download only a standard program to a standard CPU.
The diversity check on the CPU password entered failed.	The SIMATIC Automation Tool cannot connect to the CPU. Device identification data such as the MAC address or serial number is different between the SIMATIC Automation Tool Device table and the actual device. Scan the network (Page 21).
The backup file is not valid for this device.	The backup file (Page 57) you selected is not valid for this CPU. Select a backup file that corresponds to your CPU device type.
The backup file extension is not valid for this device.	The file extension of the file is invalid. Select a backup file (Page 57) with the .s7pbkp file extension.
The operation failed.	One of the following operations failed: - Reset to factory defaults (Page 36) - Memory reset (Page 38) - Format memory card (Page 39) As applicable, set the device to STOP mode or ensure that a SIMATIC memory card is in the CPU. Reattempt the operation.
The firmware file is not valid for this device.	Select a firmware update file (Page 40) that corresponds to the device.
The program file password could not be validated.	The "Password in Program File" that you entered for a program update file (Page 48) does not correspond to any of the passwords in the program file. To update a program, you must enter a valid program file password.
The program file contains an IP address that already exists on the network.	The SIMATIC Automation Tool does not allow you to restore (Page 59) a program file that causes a duplicate IP address. Change the IP address of the device that is the same or restore a different program file.
Missing program file password	You must enter a valid password (Page 308) for the program.
Invalid program file password	You must enter a valid password (Page 308) for the program.
The program file is not valid for this device.	Select a program update file (Page 48) that corresponds to the device.
The program file password entered is invalid or not sufficient to complete the operation.	Enter a password (Page 308) for the program file that has a sufficient access level for the operation.
The requested device was not found on the network.	The SIMATIC Automation Tool could not find the device on the network. Check network interface, network connection, and IP address. Close S7-PLCSIM or other simulation software if running.
You cannot insert a device that already exists.	You attempted to insert a device (Page 25) that is already in the Device table. You can only insert a device that is not yet in the Device table.
The IP address cannot be changed because it is already on the network.	You attempted to set an IP address (Page 32) to an IP address that is already in use. Set the device IP address to a unique address.

The PROFINET name cannot be changed because it is already on the network	You attempted to set the PROFINET name (Page 33) to a PROFINET name that is already in use. Select a unique PROFINET name.
Could not establish a connection to the device	The SIMATIC Automation Tool could not establish a connection to the device. Check network interface, network connection, and IP address.
The device could not be fully initialized.	The SIMATIC Automation Tool could not initialize the device. Ensure that the device is on the same network as your programming device.
The SIMATIC Automation Tool does not support CPU with newer programs.	You are attempting to use a program that has a newer version than The SIMATIC Automation Tool supports. Check the Device catalog (Page 95) to determine the version support for your device.
IP suite is not valid.	The IP address, gateway, and subnet (Page 32) combination is not valid. Enter a valid IP suite.
The selected backup file is not compatible with this device	The backup file (Page 57) that you chose is not compatible for your device. Check that the backup file is for the correct device, and that it is for the correct version of the device.
The IP address changed or there is a duplicate IP address on the network.	The IP address for this device has been changed since the last network scan, or another device has the same IP address. Scan the network (Page 21) and resolve any IP address conflicts.
No devices were found on the scan.	The SIMATIC Automation Tool did not find any devices on the network scan. Check network interface and network connections. Check the subnet for the devices. Close S7-PLCSIM or other simulation software if running.
Result of CRC comparison of online and offline collective F-signatures match.	Information only: Your operation with an F-CPU was successful. You do not need to perform any additional action.
The device could not be inserted.	The SIMATIC Automation Tool could not insert the device you entered into the Device table (Page 25). Check that the IP address is unique and that the data you entered corresponds to a device on your network.
You cannot insert a device that has an IP address that is already on the network.	You attempted to insert a device (Page 25) with an IP address that belongs to another device. You must enter a unique IP address when you enter a device.
The new gateway address is not valid.	You entered an invalid gateway address (Page 32). Enter a gateway address that is valid for your network.
The new IP address is not valid.	You entered an invalid new IP address (Page 32). Enter a valid IP address that.
The new PROFINET name is not valid.	You entered an invalid new PROFINET name (Page 33). Enter a valid PROFINET name.
The new subnet mask is not valid.	You entered an invalid subnet mask (Page 32). Enter a subnet mask that is valid for your device and network.

Could not read the F-signatures	F-CPUs include a fail-safe signature. The SIMATIC Automation Tool could not read the fail-safe signature from the device. Your operation cannot proceed with this error. Format the memory card (Page 39) if you are using one. Reset the device to factory settings (Page 36). Then repeat the operation.
Invalid or not specified network interface	Select a valid network interface (Page 312) from the dropdown list for your communication setup.
The CPU password entered is not sufficient to complete the operation.	You need a higher access level to perform the operation. Enter a CPU password with a sufficient access level for the operation you want to perform (Page 308).
There are no passwords in the program file.	The program file contains no passwords. This message is informational and requires no action.
The project contains a file format that is no longer supported.	The project is from an older release of the SIMATIC Automation Tool. The SIMATIC Automation Tool cannot open a project from this release. See the topic "Saving and opening .sat project files (Page 65)".
Project open has been canceled.	You started to open a project (Page 65) and canceled the action. No further action is necessary.
A program file password is needed	You must enter the password for the program file.
Could not open, read, or process the backup file.	The SIMATIC Automation Tool could not open, read, or process the backup file (Page 57). Use a backup file that corresponds to your device and version,
Incompatible program file for this device.	The program file is not compatible for your device. Use a program file that corresponds to your device and version.
The SIMATIC Automation Tool does not support the project file.	The SIMATIC Automation Tool does not support the project file in the device. Check the device configuration in the STEP 7 project and check the device support in the Device catalog.
The program's hardware configuration is not valid for the attached device.	The hardware configuration of the STEP 7 program is not valid for the actual device on the network. Check the device configuration in the STEP 7 project and check the device support in the Device catalog.
The file system cannot be loaded. Try power cycling the CPU.	The SIMATIC Automation Tool could not load the files from the CPU. CPU files are possibly corrupt. Power cycle the CPU. If the problem persists, reset the CPU to factory defaults.
SNMP: An invalid password was entered.	You entered an invalid password in the SNMP Version 3 profile (Page 88). Enter a valid password for the device.
SNMP: The value does not exist	An SNMP value is invalid. Verify your SNMP profile (Page 88) settings.
SNMP: The value cannot be changed	In an SNMP profile (Page 88), you attempted to change a value to an invalid value. Enter a valid value.

SNMP: The value is read only and cannot be changed.	You attempted to change a read-only value in an SNMP profile (Page 88). You cannot change this value.
Invalid authentication algorithm	The authentication algorithm in the SNMP Version 3 profile is invalid. Select a valid authentication algorithm from the dropdown list on the SNMP profile (Page 88) "Add" dialog.
Invalid authentication password	The authentication password in the SNMP Version 3 profile is invalid. Enter a valid authentication password on the SNMP profile (Page 88) "Add" dialog.
Invalid context name	The context name in the SNMP Version 3 profile is invalid. Enter a valid context name on the SNMP profile (Page 88) "Add" dialog.
Invalid privacy algorithm	The privacy algorithm in the SNMP Version 3 profile is invalid. Enter a valid privacy algorithm on the SNMP profile (Page 88) "Add" dialog.
Invalid privacy password	The privacy password in the SNMP Version 3 profile is invalid. Enter a valid privacy password on the SNMP profile (Page 88) "Add" dialog.
Invalid profile name	The profile name in the SNMP Version 1, 2, or 3 profile is invalid. Enter a valid profile name on the SNMP profile (Page 88) "Add" dialog.
Invalid read community	The read community in the SNMP Version 1 or 2 profile is invalid. Enter a valid read community on the SNMP profile (Page 88) "Add" dialog.
Invalid security level	The security level in the SNMP Version 3 profile is invalid. Enter a valid security level on the SNMP profile (Page 88) "Add" dialog.
Invalid server IP address	The server IP address in the SNMP Version 1, 2, or 3 profile is invalid. Enter a valid server IP address on the SNMP profile (Page 88) "Add" dialog.
Invalid server port	The server port in the SNMP Version 1, 2, or 3 profile is invalid. Enter a valid server port on the SNMP profile (Page 88) "Add" dialog.
Invalid SNMP version	The SNMP version is invalid. Select 1, 2, or 3 for the SNMP version number on the SNMP profile (Page 88) "Add" dialog.
Invalid user name	The user name in the SNMP Version 3 profile is invalid. Enter a valid user name on the SNMP profile (Page 88) "Add" dialog.
Invalid write community	The write community in the SNMP Version 1 or 2 profile is invalid. Enter a valid write community on the SNMP profile (Page 88) "Add" dialog.
The profile name already exists.	The profile name in the SNMP Version 1, 2, or 3 profile belongs to another profile. Enter a unique profile name on the SNMP profile (Page 88) "Add" dialog.
This method is obsolete. Use the newer method with the same name.	Not applicable.

Invalid profile	The SNMP profile (Page 88) is invalid for the device. Verify the configuration on the "Add" dialog and make necessary changes.
Failed to initiate firmware transfer	The firmware update operation (Page 40) failed. Check all network connections. Check that the firmware update file is valid for the device. Reattempt the firmware update.
An SNMP error has occurred	An error occurred in SNMP communications to the device. Verify your SNMP profile (Page 88) settings. If the problem persists, contact customer support for the device.
Invalid transfer channel	The selection for the HMI transfer channel is invalid. Set either PN_IE or Ethernet for the HMITransferChannel type for the API SetTransferChannel method (Page 277).
The device returned an invalid MAC address	The SIMATIC Automation Tool could not determine the MAC address for the IP address that you provided. This error can only occur when you insert a device (Page 25) by IP address. Verify that you are inserting a device that is available on your network. Verify that the IP address is correct for the device. Alternatively, insert the device by its MAC address.
Failed to update duplicate IPs and PROFINET names	The SIMATIC Automation Tool has an internal error. Rescan the network (Page 21). If the problem persists, contact your Siemens representative.
The PROFINET name entered is a duplicate on the network.	Another device uses this PROFINET name. Enter a unique PROFINET name.
Error occurred while performing a backup	The CPU did not provide valid backup (Page 57) data to the SIMATIC Automation Tool. Power cycle the CPU and try again. If the problem persists, contact your Siemens representative.
The Failsafe Control object on the CPU does not have the correct type.	The current program in the F-CPU does not exist as a safety program or is corrupt. Download the safety program from the TIA Portal or update the program (Page 48) from the SIMATIC Automation Tool.
The specified IP address is invalid or already used by the NIC	You entered an invalid address or one that another device uses. Enter a valid, unique IP address.
The firmware update failed due to integrity checks.	The .upd file for the firmware update (Page 40) is corrupt. Use a valid .upd file for the firmware update.
Invalid backup type	The HMI backup (Page 57) type is invalid. Valid backup types for an HMI are full, recipe, or user administration data. Provide a valid backup type.
One or more device diagnostic buffers could not be accessed	When exporting device diagnostics (Page 72), the SIMATIC Automation Tool could not access one or more diagnostic buffers. Check that the devices in the Device table are connected to the communications network. Check the Event Log for additional messages about inaccessible devices. If necessary, correct network connections and scan the network (Page 21).
Automation License Manager error - connection failed, check network access to ALM server	The Automation License Manager had an error as described. Follow the advice in the message.

Automation License Manager error - internal error, task has been aborted, contact Siemens support	The Automation License Manager had an error as described. Follow the advice in the message.
Automation License Manager error - internal error, Wrong function arguments, contact Siemens support	The Automation License Manager had an error as described. Follow the advice in the message.
Automation License Manager error - internal error, bad result, contact Siemens support	The Automation License Manager had an error as described. Follow the advice in the message.
Automation License Manager error - Check user access privileges to folder	The Automation License Manager had an error as described. Follow the advice in the message.
Automation License Manager error - internal error, Batch API, contact Siemens support	The Automation License Manager had an error as described. Follow the advice in the message.
Automation License Manager error - internal error, Batch API Output file couldn't be created, contact Siemens support	The Automation License Manager had an error as described. Follow the advice in the message.
Automation License Manager error - internal error, Batch API output file already exists, contact Siemens support	The Automation License Manager had an error as described. Follow the advice in the message.
Automation License Manager error - internal error, Batch API wrong argument, contact Siemens support	The Automation License Manager had an error as described. Follow the advice in the message.
Automation License Manager error - internal error, Batch API wrong argument number, contact Siemens support	The Automation License Manager had an error as described. Follow the advice in the message.
Automation License Manager error - internal error, Batch API wrong input file, contact Siemens support	The Automation License Manager had an error as described. Follow the advice in the message.
Automation License Manager error - internal error, Batch API wrong input xml stream, contact Siemens support	The Automation License Manager had an error as described. Follow the advice in the message.
Automation License Manager error - internal error, Batch API wrong output parameter, contact Siemens support	The Automation License Manager had an error as described. Follow the advice in the message.
Automation License Manager error - connection failed, check network access to ALM server	The Automation License Manager had an error as described. Follow the advice in the message.
Automation License Manager error - internal error, cryptography failed, contact Siemens support	The Automation License Manager had an error as described. Follow the advice in the message.
Automation License Manager error - internal error, global cleanup failed, attempt ALM restart	The Automation License Manager had an error as described. Follow the advice in the message.
Automation License Manager error - Initialization Error, attempt ALM restart	The Automation License Manager had an error as described. Follow the advice in the message.
Automation License Manager error - internal error, session cleanup failed, attempt ALM restart	The Automation License Manager had an error as described. Follow the advice in the message.
Automation License Manager error - internal error, session initialization failed, attempt ALM restart	The Automation License Manager had an error as described. Follow the advice in the message.
Automation License Manager error - internal error, function not found, contact Siemens support	The Automation License Manager had an error as described. Follow the advice in the message.
Automation License Manager error - internal error, bad pointer, contact Siemens support	The Automation License Manager had an error as described. Follow the advice in the message.
Automation License Manager error - No connection available, check the network connection	The Automation License Manager had an error as described. Follow the advice in the message.

Automation License Manager error - internal error, open session failed, contact Siemens support	The Automation License Manager had an error as described. Follow the advice in the message.
Automation License Manager error - the timeout time was reached, restart the system or contact Siemens support	The Automation License Manager had an error as described. Follow the advice in the message.
Automation License Manager error - internal error, not enough memory, restart the system or contact Siemens support	The Automation License Manager had an error as described. Follow the advice in the message.
Automation License Manager error - Failure in receiving network data, check the network connection	The Automation License Manager had an error as described. Follow the advice in the message.
Automation License Manager error - internal error, resources missing, contact Siemens support	The Automation License Manager had an error as described. Follow the advice in the message.
Automation License Manager error - internal error, result mismatch, contact Siemens support	The Automation License Manager had an error as described. Follow the advice in the message.
Automation License Manager error - Failed sending network data, check the network connection	The Automation License Manager had an error as described. Follow the advice in the message.
Automation License Manager error - service not running, restart the ALM service	The Automation License Manager had an error as described. Follow the advice in the message.
Automation License Manager error - internal error, session ID missing, contact Siemens support	The Automation License Manager had an error as described. Follow the advice in the message.
Automation License Manager error - internal error, session ID doesn't exist, contact Siemens support	The Automation License Manager had an error as described. Follow the advice in the message.
Automation License Manager error - internal error, task is already running, contact Siemens support	The Automation License Manager had an error as described. Follow the advice in the message.
Automation License Manager error - internal error, unknown error, contact Siemens support	The Automation License Manager had an error as described. Follow the advice in the message.
Automation License Manager error - internal error, User specified an unknown option, contact Siemens support	The Automation License Manager had an error as described. Follow the advice in the message.
The maximum number of PC connections has been exceeded	The Automation License Manager had an error as described. Follow the advice in the message.
The filename is not valid	The Automation License Manager had an error as described. Follow the advice in the message.
The running service request has been aborted by the client	The Automation License Manager had an error as described. Follow the advice in the message.
SNMP version is invalid	The SNMP version (Page 88) is invalid. Select 1, 2, or 3 for the SNMP version number.
You must power cycle the CPU before performing this operation.	This operation requires a power cycle of the CPU. Power cycle the CPU and try the operation again.
Duplicate folder names with a different folder path is not allowed.	When you create an archive file, you cannot use a duplicate folder name in a different folder path. Use a unique folder name or use the same path.
Duplicate file names with a different folder path is not allowed.	When you create an archive file, you cannot use a duplicate file name in a different folder path. Use a unique file name or use the same path.
Feature not supported under current licensing model. Upgrade your license to perform this operation.	The SIMATIC Automation Tool supports a Basic and Advanced license. Purchase the license (Page 101) you need for your operations.

This feature is not supported for third party API users.	The SIMATIC Automation Tool user interface requires this feature but it is not available for developers using the API to create custom applications.
The advanced parameter is not supported for third party API users.	A parameter in an API method call is reservered for the SIMATIC Automation Tool. The parameter is not for use in application development using the API.
The CPU now contains a protection level that is weaker than what previously existed.	After a program update or restore from backup, the CPU password (Page 308) is at an access level that is at a lower protection level than before the operation. For example, a CPU with full access (write) protection might have only read access protection after the operation.
Could not read protection level to check access level.	The CPU password (Page 308) is not sufficient to check the access level to the CPU. Enter a password with at least read access.
Changing the IP address is not allowed in NAT router configurations.	You cannot change the IP address for a device in a network with a NAT router configuration.
IP address and Router IP address cannot be the same	The IP address of the device and the IP address of the router must be unique.
The operation is not allowed through a CM / CP	You can only perform this operation though the CPU Ethernet interface. You cannot perform this operation through a CM/CP interface.
Requested item does not exist on memory card.	The requested file does not exist on the SIMATIC memory card of the device.
DCP operations are not possible to devices behind routers.	Devices behind routers do not support DCP operations (Page 114). You can only perform operations through the IP address.
The operation is not supported through a CM/CP communication module.	The SIMATIC Automation Tool cannot perform the named operation through a CM or CP. Check the specific device operation (Page 113) for restrictions.
No firmware to activate.	The API called the FirmwareActivate method without first calling the FirmwareUpdate method. From the API, you must call FirmwareUpdate without activation prior to calling FirmwareActivate. From the SIMATIC Automation Tool, you must select "Download Firmware" before you select "Activate Firmware" (Page 47).
Failed to activate downloaded firmware.	The device could not activate the downloaded firmware update. Download a valid firmware update file to the device before you activate it. (Page 47)
The operation is not supported by the device or not supported through a CM/CP.	The SIMATIC Automation Tool could not perform the named operation for one of the following reasons: • The device does not support the operation. • You cannot perform the operation on a device through a CM or CP. Check the Device Catalog (Page 95) for device support and the specific device operation (Page 21) for any CM or CP restrictions.

Error reading memory card	When refreshing the CPU, the SIMATIC Automation Tool could not read the memory card. Verify that a memory card is properly inserted in the CPU.
Cannot rename file because the name already exists.	The filename already exists. Choose a unique filename.
File cannot be replaced because it is write protected.	You cannot replace a write-protected file. Save to another filename.
Folder cannot be replaced because it is write protected.	You cannot replace a write-protected folder. Save to another folder.
Memory card is not present.	The CPU does not have a memory card. The CPU must have a memory card for the selected operation.
Operation requires the CPU to be in STOP mode	You cannot perform this operation when the CPU is in RUN mode. Place the CPU in STOP mode (Page 30).
Error writing file on memory card.	The SIMATIC Automation Tool could not write the file to the memory card. Check for adequate space and write privileges.
Error creating folder on memory card.	The SIMATIC Automation Tool could not create the folder on the memory card. Check for adequate space and write privileges.
Error reading service data	The SIMATIC Automation Tool could not read the service data from the device. Check the Device Catalog (Page 95) for support for reading service data. Check the connection to the device.
No files in list to download	Internal error from API. This error does not occur when you use the SIMATIC Automation Tool.
The CPU password entered is not sufficient for SIMATIC Automation Tool operations.	After a refresh operation, the CPU password (Page 308)does not have sufficient privileges for SIMATIC Automation Tool operations. Enter a CPU password that provides an access level for the operation you want to perform.
An unknown communication error occurred or the operation is not supported by the HMI.	The API could not communicate with the HMI or the HMI did not support the attempted operation. Check the connection to the device. Check the operations that the device supports.
Project file was not opened because the Cancel button was clicked.	Information only: You canceled the browser for opening a project file (Page 65).
Project file <filename> was opened successfully.	Information only: The SIMATIC Automation Tool opened the named project file (Page 65).
Project file <filename> failed to open. <exception error>	The SIMATIC Automation Tool could not open the named project file (Page 65). Verify that the file is a valid .sat project and that is compatible with your version.
The password entered is incorrect.	You entered an invalid project file (Page 65) password. You must enter the correct password for this project file.
Project file was not saved because the Cancel button was selected.	Information only: You canceled the browser for saving a project file (Page 65).

Export of PC Data failed.	The export PC data operation (Page 73) could not create the export file. Check for adequate disk space and permissions for the Export folder (Page 88).
Project file <filename> failed to save. <exception error>	The SIMATIC Automation Tool could not save the project file (Page 65). Check folder permissions and disk space on your programming device.
Export was not performed because the Cancel button was clicked.	Information only: You started an export (Page 67) and canceled the export.
Insert device was not performed because the Cancel button was clicked.	Information only: You started to insert a device (Page 25) and canceled the device insert.
A valid product license was not found; some product features are disabled.	You do not have a valid product license. You can only perform tasks that the SIMATIC Automation Tool supports for an unlicensed version (Page 101). To access all features, purchase a license as described in the Installation Notes.
A valid product license was found. All product features are enabled.	Information only: With a valid product license, you can perform all SIMATIC Automation Tool tasks (Page 101).
Windows Administrative privileges are required to access some PC Export data. Restart the SIMATIC Automation Tool using "Run as Administrator" to export this data.	Run the export PC data operation (Page 73) with Windows Administrative privileges to access some of the data. Restart the SIMATIC Automation Tool using "Run as Administrator" to export this data.
The export acquired all of the requested data. The log file contains a detailed list of the exported data.	Information only: The export PC data operation (Page 73) was successful. The log file in the export .zip file lists the exported data.
The export could not acquire all of the data. The log file contains details about the data that the export successfully acquired and the reasons why it could not obtain the other data.	The export PC data operation (Page 73) did not export all of the data. Check the log file for the reason. The log file in the export .zip file contains the following information: • Information about the data that the export operation successfully acquired and exported • Reasons why the export operation could not acquire the other data
Project file <filename> was saved successfully.	Information only: The SIMATIC Automation Tool successfully saved the named project file (Page 65).
The operation was canceled because the Cancel button was selected.	Information only
Changes to the project file were not saved because the No button was selected.	Information only
The project password was changed.	Information only
Project password was not changed because the Cancel button was selected.	Information only
Caps Lock is on	Information only
The project is already opened.	Information only
Archive file <filename> was opened successfully.	Information only
Archive file <filename> failed to open. <exception error>	The SIMATIC Automation Tool could not open this archive file and returned the listed error.
Archive file was not opened because the operation was canceled.	Information only

Archive file <filename> was created successfully.	Information only
Archive file <filename> was not created due to errors. <exception error>	The SIMATIC Automation Tool could not create this archive file and returned the listed error.
The archive file was not created because there are data input errors within the device table.	The SIMATIC Automation Tool could not create the archive file. Correct all invalid user data fields in the Device table and try again.
Archive file was not created because the Cancel button was selected.	Information only
Failed to delete the Scheduler configuration file.	The SIMATIC Automation Tool could not delete the scheduler configuration file. Perhaps it is in use. Wait until no scheduled operations are running and try again.
The Scheduler configuration file was successfully created.	Information only
At least one operation must be configured before the configuration file can be created.	You must schedule at least one operation for one device to create a Scheduler configuration file.
Scheduler file was not created because the Cancel button was selected.	Information only
An error occurred creating the Scheduler configuration file. The operation has been canceled.	The SIMATIC Automation Tool could not create the scheduler configuration file. Perhaps it is in use. Wait until no scheduled operations are running and try again.
The date or time is not valid because it is in the past.	You can only schedule operations based on a time in the future.
No devices were selected for Device Diagnostics. The operation will not be scheduled.	You must select one or more devices for the scheduled operation.
No devices were selected for Firmware Update. The operation will not be scheduled.	You must select one or more devices for the scheduled operation.
No devices were selected for Full Backup. The operation will not be scheduled.	You must select one or more devices for the scheduled operation.
No devices were selected for Read Data Logs. The operation will not be scheduled.	You must select one or more devices for the scheduled operation.
No devices were selected for Read Service Data. The operation will not be scheduled.	You must select one or more devices for the scheduled operation.
No devices were selected for Set Time. The operation will not be scheduled.	You must select one or more devices for the scheduled operation.
The CPU password entered is not sufficient to read Device Diagnostics. The operation will not be scheduled.	You must enter a CPU password (Page 308) with read access or higher to schedule an operation to read device diagnostics.
The CPU password entered is not sufficient to perform a Firmware Update. The operation will not be scheduled.	You must enter a CPU password that is sufficient for a firmware update operation (Page 40). TheCPU password access level depends on your device model and firmware version.
The CPU password entered is not sufficient to perform a Full Backup. The operation will not be scheduled.	You must enter a CPU with at least read access to schedule this operation.
The CPU password entered is not sufficient to Read Data Logs. The operation will not be scheduled.	You must enter a CPU with at least read access to schedule this operation.
The CPU password entered is not sufficient to Read Service Data. The operation will not be scheduled.	You must enter a CPU with at least read access to schedule this operation.

The CPU password entered is not sufficient to Set Time. The operation will not be scheduled.	You must enter a CPU with at least read access to schedule this operation.
The firmware version file is not valid for the device. The operation will not be scheduled.	You cannot use this firmware version for this device. Select a firmware update files that is compatible with the device to schedule this operation.
The SNMP profile name is either not valid or not specified for device: <device name>	The SNMP profile name you selected In the scheduler tab (Page 106) is invalid for this device. The red X indicates an invalid profile.
Enter the password in the Scheduler application to open the configuration file.	Enter the password in the Scheduler application to open the configuration file.
Use the Scheduler application to open the configuration file.	Use the Scheduler application to open the configuration file.
The start date and time is invalid for Device Diagnostics. The operation will not be scheduled.	Enter a start date and time that is in the future. Then schedule the operation.
A start date and time was not specified for Device Diagnostics. The operation will not be scheduled.	Enter a start date and time that is in the future. Then schedule the operation.
The start date and time is invalid for Firmware Update. The operation will not be scheduled.	Enter a start date and time that is in the future. Then schedule the operation.
A start date and time was not specified for Firmware Update. The operation will not be scheduled.	Enter a start date and time that is in the future. Then schedule the operation.
The start date and time is invalid for Full Backup. The operation will not be scheduled.	Enter a start date and time that is in the future. Then schedule the operation.
A start date and time was not specified for Full Backup. The operation will not be scheduled.	Enter a start date and time that is in the future. Then schedule the operation.
The start date and time is invalid for Read Data Logs. The operation will not be scheduled.	Enter a start date and time that is in the future. Then schedule the operation.
A start date and time was not specified for Read Data Logs. The operation will not be scheduled.	Enter a start date and time that is in the future. Then schedule the operation.
The start date and time is invalid for Read Service Data. The operation will not be scheduled.	Enter a start date and time that is in the future. Then schedule the operation.
A start date and time was not specified for Read Service Data. The operation will not be scheduled.	Enter a start date and time that is in the future. Then schedule the operation.
The start date and time is invalid for Set Time. The operation will not be scheduled.	Enter a start date and time that is in the future. Then schedule the operation.
A start date and time was not specified for Set Time. The operation will not be scheduled.	Enter a start date and time that is in the future. Then schedule the operation.
The project file contained within the archive is already opened.	You cannot open this archive file because it is already open.
The recurrence setting is not valid for the start date specified for Device Diagnostics. The operation will not be scheduled.	The frequency setting must be a valid calendar date for every recurrence. For example, the 31st of the month does not occur every month. Enter a valid start date and frequency setting for every scheduled operation. Then save the scheduler configuration file.
The recurrence setting is not valid for the start date specified for Firmware Update. The operation will not be scheduled.	The frequency setting must be a valid calendar date for every recurrence. For example, the 31st of the month does not occur every month. Enter a valid start date and frequency setting for every scheduled operation. Then save the scheduler configuration file.

The recurrence setting is not valid for the start date specified for Full Backup. The operation will not be scheduled.	The frequency setting must be a valid calendar date for every recurrence. For example, the 31st of the month does not occur every month. Enter a valid start date and frequency setting for every scheduled operation. Then save the scheduler configuration file.
The recurrence setting is not valid for the start date specified for Read Data Logs. The operation will not be scheduled.	The frequency setting must be a valid calendar date for every recurrence. For example, the 31st of the month does not occur every month. Enter a valid start date and frequency setting for every scheduled operation. Then save the scheduler configuration file.
The recurrence setting is not valid for the start date specified for Read Service Data. The operation will not be scheduled.	The frequency setting must be a valid calendar date for every recurrence. For example, the 31st of the month does not occur every month. Enter a valid start date and frequency setting for every scheduled operation. Then save the scheduler configuration file.
The recurrence setting is not valid for the start date specified for Set Time. The operation will not be scheduled.	The frequency setting must be a valid calendar date for every recurrence. For example, the 31st of the month does not occur every month. Enter a valid start date and frequency setting for every scheduled operation. Then save the scheduler configuration file.
This device has a duplicate IP address on the network and Device Diagnostics cannot be performed. The operation will not be scheduled.	The SIMATIC Automation Tool cannot schedule operations for devices with duplicate IP addresses. Set unique IP addresses (Page 32) for the devices. Then schedule the operation and save the scheduler configuration file.
This device has a duplicate IP address on the network and Firmware Update cannot be performed. The operation will not be scheduled.	The SIMATIC Automation Tool cannot schedule operations for devices with duplicate IP addresses. Set unique IP addresses (Page 32) for the devices. Then schedule the operation and save the scheduler configuration file.
This device has a duplicate IP address on the network and Full Backup cannot be performed. The operation will not be scheduled.	The SIMATIC Automation Tool cannot schedule operations for devices with duplicate IP addresses. Set unique IP addresses (Page 32) for the devices. Then schedule the operation and save the scheduler configuration file.
This device has a duplicate IP address on the network and Read Data Logs cannot be performed. The operation will not be scheduled.	The SIMATIC Automation Tool cannot schedule operations for devices with duplicate IP addresses. Set unique IP addresses (Page 32) for the devices. Then schedule the operation and save the scheduler configuration file.
This device has a duplicate IP address on the network and Read Service Data cannot be performed. The operation will not be scheduled.	The SIMATIC Automation Tool cannot schedule operations for devices with duplicate IP addresses. Set unique IP addresses (Page 32) for the devices. Then schedule the operation and save the scheduler configuration file.
This device has a duplicate IP address on the network and Set Time cannot be performed. The operation will not be scheduled.	The SIMATIC Automation Tool cannot schedule operations for devices with duplicate IP addresses. Set unique IP addresses (Page 32) for the devices. Then schedule the operation and save the scheduler configuration file.

The Scheduler application did not launch. Close other instances or restart the computer.	The Scheduler application did not launch. Check for a running instance of the Scheduler application and close it if running. If the error persists, reboot your programming device.
New Scheduler configuration was not created because the Cancel button was selected.	Information only
The file <filename> was not replaced because it was skipped by the user.	Information only
The file <filename> was not replaced because it is write protected.	The SIMATIC Automation Tool cannot replace a write-protected file. Select a different filename or change the permissions.
The folder <folder name> was not replaced because it is write protected.	The SIMATIC Automation Tool cannot replace a write-protected folder. Select a different folder name or change the permissions.
Opening a project archive requires an advanced license.	Purchase an advanced license (Page 101) to open a project archive.
Recipes files cannot be pasted because the Recipes folder is missing.	The SIMATIC memory card must have a "Recipes" folder for this paste to succeed. Create the folder.
User files cannot be pasted because the User Files folder is missing.	The SIMATIC memory card must have a "UserFiles" folder for this paste to succeed. Create the folder.

Device-specific messages

Devices can also return device-specific messages. These messages are not in the list above. If a device-specific message is not self-explanatory, contact your device manufacturer.

Messages from exporting PC data

Exporting PC data (Page 73) can generate messages that are not in the list above. The SIMATIC Automation Tool logs these messages as returned by the export process. If you get an Event Log message in the Event class "Export PC Data" try these corrective measures:

- Check for adequate disk space. The exported PC data .zip file might require more space than you have available. If necessary, create additional space on the drive.
- Verify permissions for the Export folder (Page 88).

If you have further unexplained Event Log messages from the "Export PC Data" event class, contact your Siemens representative.

SIMATIC Automation Tool API for .NET framework 12

12.1 Getting started with the API

The SIMATIC Automation Tool Application Programming Interface (API) for PROFINET network and device maintenance allows you to create custom applications using an extensive set of .NET interfaces, classes, and methods (Page 155). You will design your custom application for your specific purposes.

Setting a network interface

The first task your application must perform is to set or select a network interface for communicating to your PROFINET network. Your computer or programming device is likely to have multiple network interfaces.

If you do not know the name of the network interface you wish to use, you can use the `QueryNetworkInterfaceCards` (Page 184) method of the Network class to read a list of all network interfaces. You can present this list to the user of your application to select a specific network interface. Recommend that the users of your application select the network interface with "Auto" in the name. The "Auto" selection allows to the API to find all devices.

You then use the `SetCurrentNetworkInterface` (Page 184) method to set the network interface that your application will use for device communication.

Communicating to devices

As the application designer, you can choose whether to operate on devices found by a network scan or on devices that are added by the Insert Device methods (Page 200). If you only have a few devices on your network and you know their IP or MAC addresses, then use the Insert Device methods for application performance. If you do not know the IP or MAC addresses, then use the `ScanNetworkDevices` (Page 186) method to find all devices connected to your PROFINET network.

Scanning a network

Your application might need to connect to a PROFINET network and scan for all attached devices. In this case, your application can scan the network to fill a collection of all devices on your network. The API provides a `ScanNetworkDevices` (Page 186) method for this purpose. This method sends a broadcast DCP command to the network and fills an `IProfinetDeviceCollection` (Page 191) of the accessible devices.

Inserting devices

If you or your application user knows the IP or MAC addresses of your devices, you might want to design your application so that you or your user can add specific devices. The API provides methods for inserting a device by either IP address or MAC address. If you anticipate your user performing operations on specific devices after adding them, you might choose to develop an application based on the `InsertDeviceByIP` (Page 200) method or `InsertDeviceByMAC` (Page 201) method.

Note

Devices behind routers

If you are communicating to devices that are behind a router, you must use the `InsertDeviceByIP` (Page 200) method. A network scan does not discover devices behind routers.

Working with protected CPUs

If a CPU is protected, the `Protected` (Page 231) property is `true` for the `ICPU` interface. You must first check whether a CPU is protected before calling the `SetPassword` (Page 258) method. If the CPU is NOT protected, do NOT call the `SetPassword` (Page 258) method. If you call the `SetPassword` (Page 258) method for a CPU that is not protected, the API throws a critical error exception. The critical error exception lets you know that you are using the API incorrectly.

For CPUs that are protected, your application must call the `SetPassword` (Page 258) method with a password that provides a sufficient access level (Page 308) for the operation. After you set a valid password, you can refresh the status of the device.

The best practice is to call `SetPassword` (Page 258) on all devices for which you wish to communicate immediately after scanning the network. Set the passwords on standard CPUs to the write access password (Page 308) and for F-CPUs to the safety password (Page 308) to avoid API errors.

Using RefreshStatus

Objects in the `IProfinetDeviceCollection` (Page 191) that represent devices have only partial data from each device after a network scan. To obtain all the data about a device and to use the API properly, you must do the following:

1. Call the `SetPassword` (Page 258) method for each protected CPU. The CPU password must provide sufficient privileges to read all the device data.
2. Call the `RefreshStatus` (Page 219) method for each device in the `IProfinetDeviceCollection` (Page 191).

The `RefreshStatus` (Page 219) method updates all the data that represents the status of the device.

Critical error exceptions

If the API throws a critical error exception, then you are not using the API correctly for the status of the device. Remember to call the `RefreshStatus` (Page 219) method to keep the data about a device up to date before using additional API functions.

Additional requirements

Many of the API methods have specific prerequisites for using the methods. The method descriptions list these prerequisites as applicable.

Safety-relevant operations on F-CPUs (Page 159) require additional safeguards in your application design.

Example: Initializing devices by scanning the network

```
using Siemens.Automation.AutomationTool.API;

#region Getting started with a network scan
//-----------------------------
// Setting a Network Interface
//-----------------------------
Network myNetwork = new Network();
List<String> interfaces = new List<String>();
Result retVal = myNetwork.QueryNetworkInterfaceCards(out interfaces);
if (retVal.Succeeded)
{
   for (Int32 index = 0; index < interfaces.Count; index++)
   {
      String strInterfaceName = interfaces[index];
      // ---------------------------------------------
      // Find the NIC that is connected to your
      // PROFINET network or display a dialog and let
      // the user select a NIC from the list
      // ---------------------------------------------
      if (strInterfaceName == "myNIC.Auto.1")
      {
         retVal = myNetwork.SetCurrentNetworkInterface(strInterfaceN
ame);
      }
   }
}
//--------------------
// Scanning a Network
//--------------------
IProfinetDeviceCollection scannedDevices;
IScanErrorCollection scanResults = myNetwork.ScanNetworkDevices(out
scannedDevices);
if (scanResults.Succeeded)
{
   foreach (IProfinetDevice device in scannedDevices)
   {
      ICPU cpu = device as ICPU;
      if (cpu != null)
      {
         if (cpu.Protected)
         {
            //-------------------------------------------
            // Set the password for all CPUs on your
            // PROFINET network with at least read-write
            // privileges to use the API
            //-------------------------------------------

            retVal = cpu.SetPassword(new EncryptedString("CPUPasswor
d"));
         }
      }
      //-------------------------------------------------------
      // Update all device data that was not available on scan
      //-------------------------------------------------------
      retVal = device.RefreshStatus();
   }
}

//-------------------------------------------------------------
// You are now ready to use the API and call device operations
//-------------------------------------------------------------

#endregion
```

Example: Initializing devices by inserting devices

```
using Siemens.Automation.AutomationTool.API;

#region Getting started with inserting devices
//-----------------------------
// Setting a Network Interface
//-----------------------------
Network myNetwork = new Network();
List<String> interfaces = new List<String>();
Result retVal = myNetwork.QueryNetworkInterfaceCards(out interfaces);
if (retVal.Succeeded)
{
   for (Int32 index = 0; index < interfaces.Count; index++)
   {
      String strInterfaceName = interfaces[index];
      // ---------------------------------------------
      // Find the NIC that is connected to your
      // PROFINET network or display a dialog and let
      // the user select a NIC from the list
      // ---------------------------------------------
      if (strInterfaceName == "myNIC.Auto.1")
      {
         retVal = myNetwork.SetCurrentNetworkInterface(strInterfaceN
ame);
         if (retVal.Succeeded)
         {
            break;
         }
      }
   }
}
//-------------------
// Inserting Devices
//-------------------
IProfinetDeviceCollection insertedDevices = Network.GetEmptyCollecti
on();
retVal = insertedDevices.InsertDeviceByIP(0, 0xC0A80001); // 192.168
.0.1
if (retVal.Succeeded)
{
   foreach (IProfinetDevice device in insertedDevices)
   {
      ICPU cpu = device as ICPU;
      if (cpu != null)
      {
         if (cpu.Protected)
         {
            //-------------------------------------------
            // Set the password for all CPUs on your
            // PROFINET network with at least read-write
            // privileges to use the API
            //-------------------------------------------
```

```
              retVal = cpu.SetPassword(new EncryptedString("CPUPasswor
d"));
            }
         }
         //-----------------------------------------------------------
         // Update all the device data that was not available on scan
         //-----------------------------------------------------------
         retVal = device.RefreshStatus();
      }
   }
//---------------------------------------------------------------
// You are now ready to use the API and call device operations
//---------------------------------------------------------------


/* For simplicity the code examples above do not check for */
/* errors. Checking for and processing errors returned     */
/* from methods is vital to a program's overall quality    */

#endregion
```

12.1.1 Architectural overview

The API provides classes, interfaces, and methods to support communication with a PROFINET network of SIMATIC devices.

Note

Use only documented API classes, interfaces, and methods in your application

The API includes public methods to support the SIMATIC Automation Tool user interface in addition to public methods for your application programming. In your application development, use only the classes, interfaces, and methods that this User Guide describes. Any other public methods that exist in the API that the User Guide does not describe are reserved for exclusive use by the SIMATIC Automation Tool user interface.

Networks

The `.NET` class `Network` (Page 184) represents the PROFINET network as a whole. This class performs functions using a network interface card (NIC) installed on a programming device. The `Network` class is used to search for available network interface cards and to select the network interface that is attached to your PROFINET network.

The Network class includes the constructor and the following property and methods:

- Network constructor (Page 184)
- QueryNetworkInterfaceCards method (Page 184)
- SetCurrentNetworkInterface method (Page 184)
- CurrentNetworkInterface property (Page 185)

- ScanNetworkDevices method (Page 186)
- SetCommunicationsTimeout method (Page 186)
- GetCommunicationsTimeout method (Page 187)
- GetEmptyCollection method (Page 188)

Devices

The individual devices on the PROFINET network are represented by interfaces. Each interface class provides properties and methods appropriate for the represented network device. Each hardware device on the network is best represented by one of the following interfaces:

`IProfinetDevice` (Page 203) – All PROFINET devices directly accessible on the PROFINET network can be represented by this interface, because all devices are derived from this class.

`ICPU` (Page 230) – This represents S7-1X00 CPUs that are directly connected to the network. Specific functionality is supported for CPUs.

`ICPUClassic` (Page 267) – This represents classic type S7-300 and S7-400 CPUs that are directly connected to the network.

`IHMI` (Page 270) – This represents SIMATIC HMIs that are directly connected to the network. Specific functionality is supported for HMIs.

`IBaseDevice` (Page 180) – This interface is used to represent devices not directly connected to the PROFINET network but accessible through another device. For example, a PROFIBUS slave station that is connected to a CPU on the network is represented as an `IBaseDevice`.

`IModule` (Page 229) – This interface is used to represent individual I/O modules that are plugged into a CPU, PROFINET device, or PROFIBUS station.

`IHardware` (Page 179) – This is the base class for all other interfaces. This interface provides access to properties that are common for all hardware items recognized on the network.

`IScalance` (Page 278) – This is the interface that represents SCALANCE devices.

The interfaces are grouped into collections that represent groups of devices. Collections are provided to support iteration, filtering, and searching.

`IProfinetDeviceCollection` (Page 191) – A collection of all devices directly accessible on the network.

`IModuleCollection` (Page 181) – A collection that may represent modules plugged into a CPU or IM.

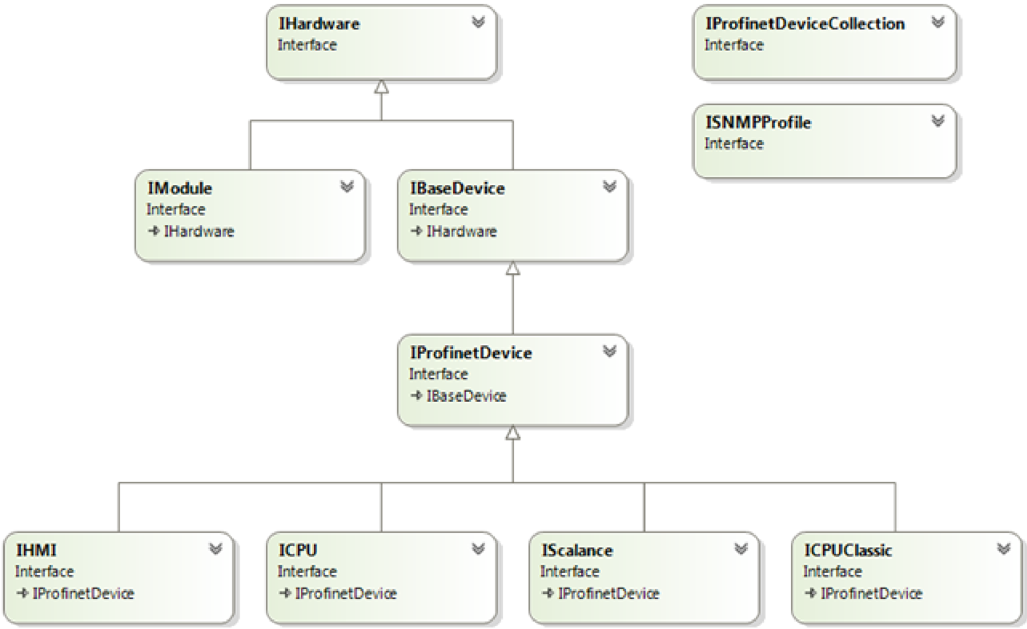
`IHardwareCollection` (Page 181) – This collection represents a CPU and all its modules.

`IScanErrorCollection` (Page 181) – This collection represents the set of errors returned from all devices on a network scan operation.

Device classes, interfaces, and methods:

- IProfinetDeviceCollection class (Page 191)
- IProfinetDevice interface (Page 203)
- ICPU interface (Page 230)
- IHMI interface (Page 270)

The following class diagram shows the inheritance relationship between these interface classes:



Note

See the example (Page 306) PROFINET network and the SIMATIC Automation Tool API classes that are used to represent each network component.

12.1.2 Referencing the API in a user interface application

Siemens delivers the API with several DLLs, executables, and source files:

- `SIMATICAutomationToolAPI.dll`
- `DeviceManagerClient.dll` (HMI)
- `hmitr.dm.client.proxy.dll` (HMI)
- `hmitr.dm.client.stub.exe` (HMI)
- `hmitr.ipc.dll` (HMI)
- `SimaticAutomationToolHealthCheck.dll`
- `AsModels` folder and subfolders (Offline object models)

When using the API, these files and the `AsModels` folder must be in the same folder where you develop your custom application.

Programming environment

Siemens developed the API with Microsoft Visual Studio 2017 using the .NET Framework 4.6.2. You can use this API with applications that you create with either of these Microsoft Visual Studio versions:

- Microsoft Visual Studio 2017
- Microsoft Visual Studio 2015 SP2 Update 3

To create a project that can work with the SIMATIC Automation Tool API, follow these steps:

1. In Visual Studio, create a new project that is a Visual C# Windows Forms Application.
2. From the Solutions Platforms drop-down list, select "Configuration Manager".
3. In the Configuration Manager, click the Platform drop-down and create a new platform.
4. From the New Project Platform dialog, select "x64" for the new platform. Click OK and close the Configuration Manager.
5. In the project properties, set the target framework to .NET Framework 4.6.2.

References in your solution

To include the API in your application, add `SIMATICAutomationToolAPI.dll` as a reference in your Visual Studio solution.

To be able to export PC data from your applications, add `SimaticAutomationToolHealthCheck.dll` as a reference.

Required "using" statements

In any application that references the API classes, you must add the following statement referencing the API namespace:

```
using Siemens.Automation.AutomationTool.API;
```

If your application exports PC Data, you must add the following statement:

```
using Siemens.Automation.AutomationTool.HealthCheck;
```

To compile any of the code samples in this document, you must put the correct `using` statements in the same source file (*.cs) as the example code. For simplicity, the individual code examples in this document do not include the using statement.

12.2 API software license and version compatibility

Software license required

You must have a valid SIMATIC Automation Tool license to use the API (Application Programming Interface).

You can use the API for programming custom applications after you perform these tasks:

- Install the SIMATIC Automation Tool
- Purchase a SIMATIC Automation Tool license or use the free 21-day trial license

You do not have permission to redistribute any part of the SIMATIC Automation Tool including any API .exe or .dll files, unless you purchase the SIMATIC Automation Tool SDK.

To distribute your custom user interface application to a third party, the third party must also have a valid license for the SIMATIC Automation Tool. Alternatively, you can purchase the SIMATIC Automation Tool SDK, which allows you to create and distribute your applications to a third party, without the third party having a license.

Compatibility with previous versions

The V4.0 SP2 API is compatible with API V3.1.x and V4.0 versions. You can run API programs that you previously developed after you update or verify your solution references (Page 157) and rebuild your application with the current API.

12.3 Requirement for S7 communications

S7 communications as you develop your application

To run applications that you develop with the API, you must have S7 communications components installed on your programming device. If you have installed the SIMATIC Automation Tool, you have installed the required S7 communications components. With S7 communications present on a programming device, you can place the files and folders named in Referencing the API in a user interface application (Page 157) in any folder on the programming device. Place your executable for your custom application in the same folder. Run the application from this folder and perform your testing.

S7 communications for end users of your application

Your end users must install the SIMATIC Automation Tool and purchase a SIMATIC Automation Tool license or use the free 21-day trial license. As an alternative, you can purchase SIMATIC Automation Tool SDK and your end users will not have to purchase a SIMATIC Automation Tool license to use your application. Your custom application must be in the same folder with the required files and folders (Page 157) on the end user's programming device.

12.4 Designing a user interface application for fail-safe devices and safety-relevant operations

12.4.1 API support for safety-relevant operations and fail-safe devices

The SIMATIC Automation Tool API supports the following safety-relevant operations:

- Program update
- Restore device from backup file
- Reset to factory defaults
- Format memory card

Note

The *SIMATIC Safety - Configuring and Programming* manual contains a warning identified as "S078". This warning states, "When using tools for the automation or operation (of TIA Portal or Web server) which allow access protection for the F-CPU to be bypassed (e.g. saving or automatic entry of a CPU password for the protection level "Full access incl. fail-safe (no protection)" or Web server password), the safety relevant project data may not be protected against unintentional changes anymore."

This S078 warning does not apply to the SIMATIC Automation Tool API. The API supports communicating with F-CPUs and storing CPU passwords for F-CPUs.

Safety features that the API provides

TÜV SÜD has certified the API for the SIMATIC Automation Tool and SIMATIC Automation Tool SDK.

If your custom application performs a safety-relevant operation to an F-CPU, you should certify your application with TÜV SÜD or an equivalent company. Just because the Siemens API is certified doesn’t mean that your application is automatically certified.

The API provides a set of protections for your application. Your application should also follow the recommended set of user interface guidelines for working with F-CPUs (Page 161).

The SIMATIC Automation Tool API uses diverse and redundant techniques. The API thus helps protect user application program code from performing potentially unsafe operations. The API provides the following features:

- Independent connection and legitimization process for each safety-relevant operation
- Identity checks for safety-relevant operations to fail-safe devices
- Identification of safety programs
- Required use of safety password (Page 308) for any safety-relevant operation to a protected F-CPU
- Use of 32-bit CRC checksums to compare fail-safe device online and offline representations
- Use of hamming codes (Page 171) to indicate TRUE and FALSE states
- Comparison of F-signatures after the program update and restore from backup operations to verify that the operation completed successfully

12.4.2 User interface programming guidelines for safety-relevant operations

⚠ **WARNING**

Protect safety-relevant operations as much as possible

Fail-Safe CPUs together with fail-safe I/O and safety programs provide the capability for a high degree of operational safety.

When you use the SIMATIC Automation Tool API, ensure that safety-relevant operations are as safe as possible. Siemens assumes no liability for user interface applications developed with the SIMATIC Automation Tool API. The software developer assumes all liability.

Failure to follow adequate programming practices can result in death or personal injury when the user operates your user interface application.

Identifying and protecting safety-relevant operations

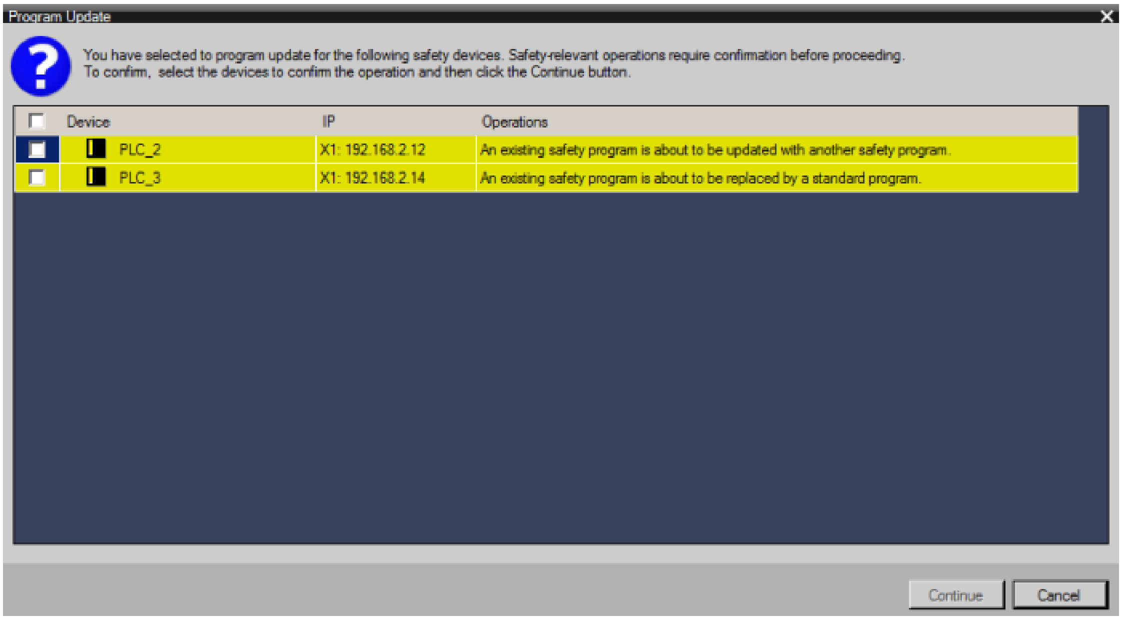
Siemens recommends that you provide a two-step process before performing a safety-relevant operation to an F-CPU.

1. The user should take an explicit action to select an F-CPU and select the operation to be performed before continuing to step 2.
2. Your software should present a dialog or equivalent to the user with one of the Siemens-defined confirmation messages (Page 238) for safety-relevant operations. Your application should prompt the user to confirm that they wish to continue before your code calls the safety-relevant operation to an F-CPU.

As you develop your user interface application using the API, identify whether an operation is one of the following safety-relevant operations for an F-CPU:

- Program update (Page 248)
- Restore device from backup file (Page 253)
- Reset to factory defaults (Page 251)
- Format memory card (Page 243)

For operations that are safety-relevant, provide a confirmation dialog for your users. Use the DetermineConfirmationMessage API method (Page 238) to determine the type of confirmation dialog to display. Providing an additional confirmation dialog protects users from accidentally performing an unintended safety-relevant operation. The following dialog is an example of a confirmation dialog for a program update operation:



Program Update

You have selected to program update for the following safety devices. Safety-relevant operations require confirmation before proceeding.
To confirm, select the devices to confirm the operation and then click the Continue button.

☐	Device	IP	Operations
☐	PLC_2	X1: 192.168.2.12	An existing safety program is about to be updated with another safety program.
☐	PLC_3	X1: 192.168.2.14	An existing safety program is about to be replaced by a standard program.

Recommended programming practices

Use the following programming practices to ensure that you protect safety-relevant operations and minimize the chance of unsafe user action:

- Perform all safety-relevant operations on a single thread.
- Require entry of the safety password (Page 308)for safety-relevant operations. Verify the entered password against the CPU password. Use asterisks to hide passwords from the display when the user enters passwords.
- As described above, provide a confirmation dialog for the user to confirm safety-relevant operations on F-CPUs.
- Check the return codes of all methods. Ensure that your program logic only proceeds upon successful method returns.
- Include appropriate exception handling in your implementation. The API throws exceptions for critical internal faults that it detects when you are not using the API correctly. Be sure that your software handles any exceptions in an appropriate manner.
- For all safety-relevant operations, evaluate whether the operation succeeded. Display a message to the user for a successful operation. Display an error message upon an unsuccessful operation.
- Use hamming codes (Page 171) in your application to implement Boolean states.
- Use yellow coloring (Page 163) in the application to indicate fail-safe devices, safety programs, safety passwords, and other user-entered data.
- Prompt for confirmation for all operating mode changes (RUN/STOP).
- Refresh the user interface after each operation so that the application displays the correct device data.

Program update requirements

For program updates on selected F-CPUs, provide an additional dialog for the user to reselect the fail-safe devices and confirm the following operations:

- An operation to a standard program is about to be initiated using the safety password
- An existing safety program is about to be deleted
- An existing safety program is about to be updated with another safety program
- An existing safety program is about to be replaced by a standard program
- A safety program is about to be loaded for the first time

Following a safety program update, the API automatically verifies that the F-signature of the program updated in the CPU. Check all function return values.

Restore from backup requirements

Before restoring a backup file, evaluate whether the file is a safety program and prompt for user confirmation according to the same requirements for program updates.

Certification

Note

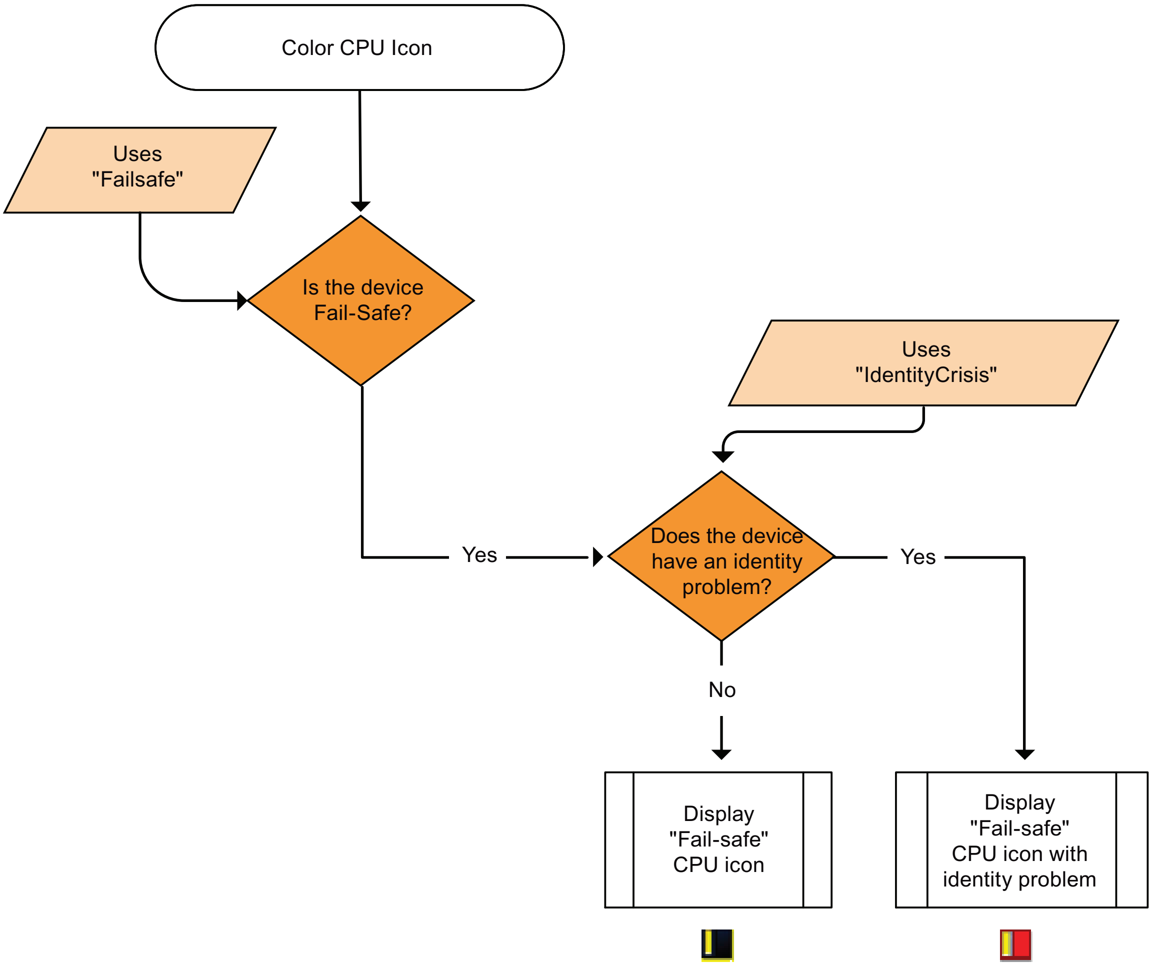
Acquire certification for your user interface application

Siemens strongly suggests using a certified body such as TÜV SÜD to certify the safety of your design and implementation.

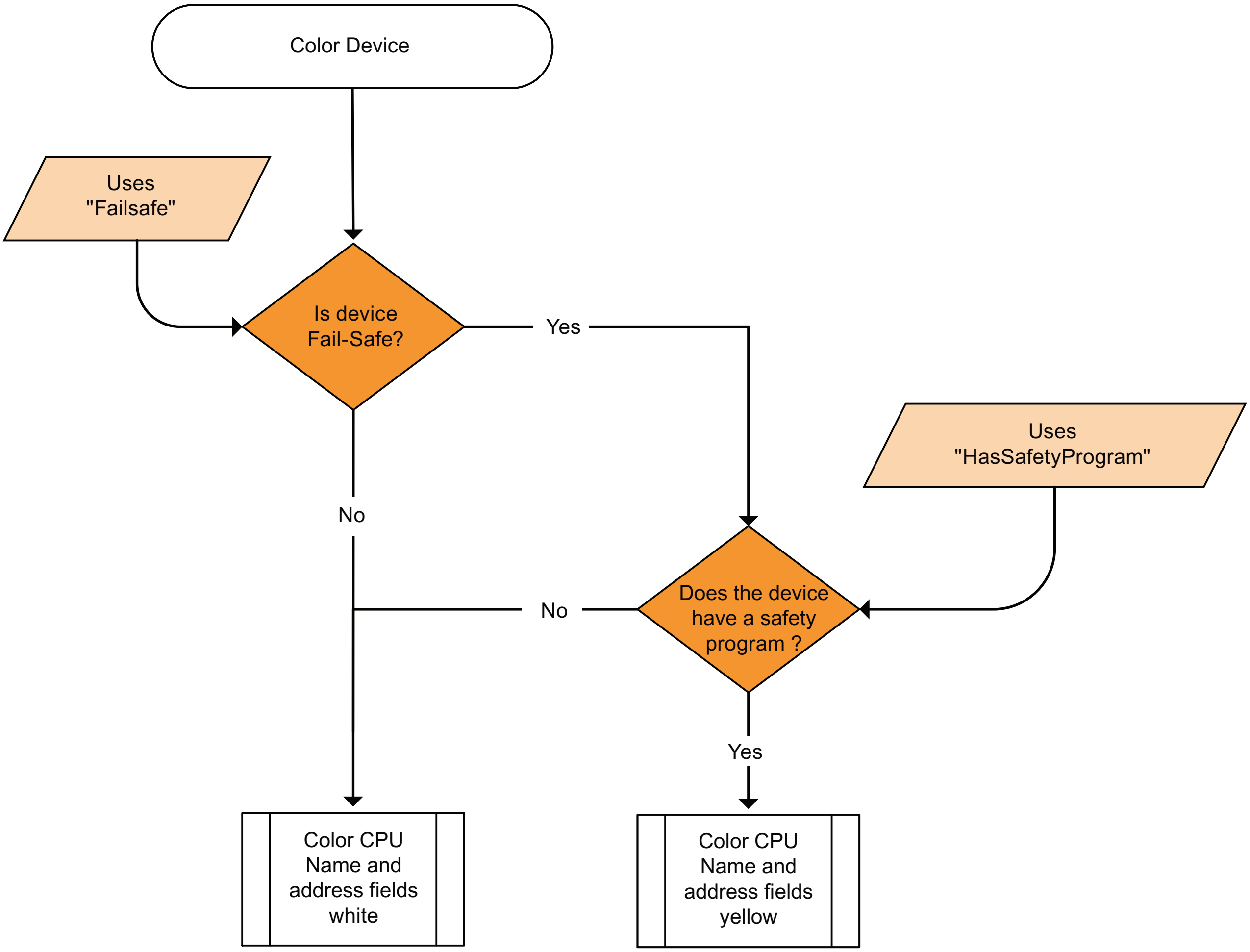
12.4.3 Color coding safety fields in your user interface

If you develop a user interface using the API, Siemens strongly recommends that you use color coding to give the user a visual indication of anything related to fail-safe CPUs and safety programs. The decision trees indicate the logic Siemens recommends in color coding typical safety-relevant fields. Consider adopting an identical or similar approach as you design your application.

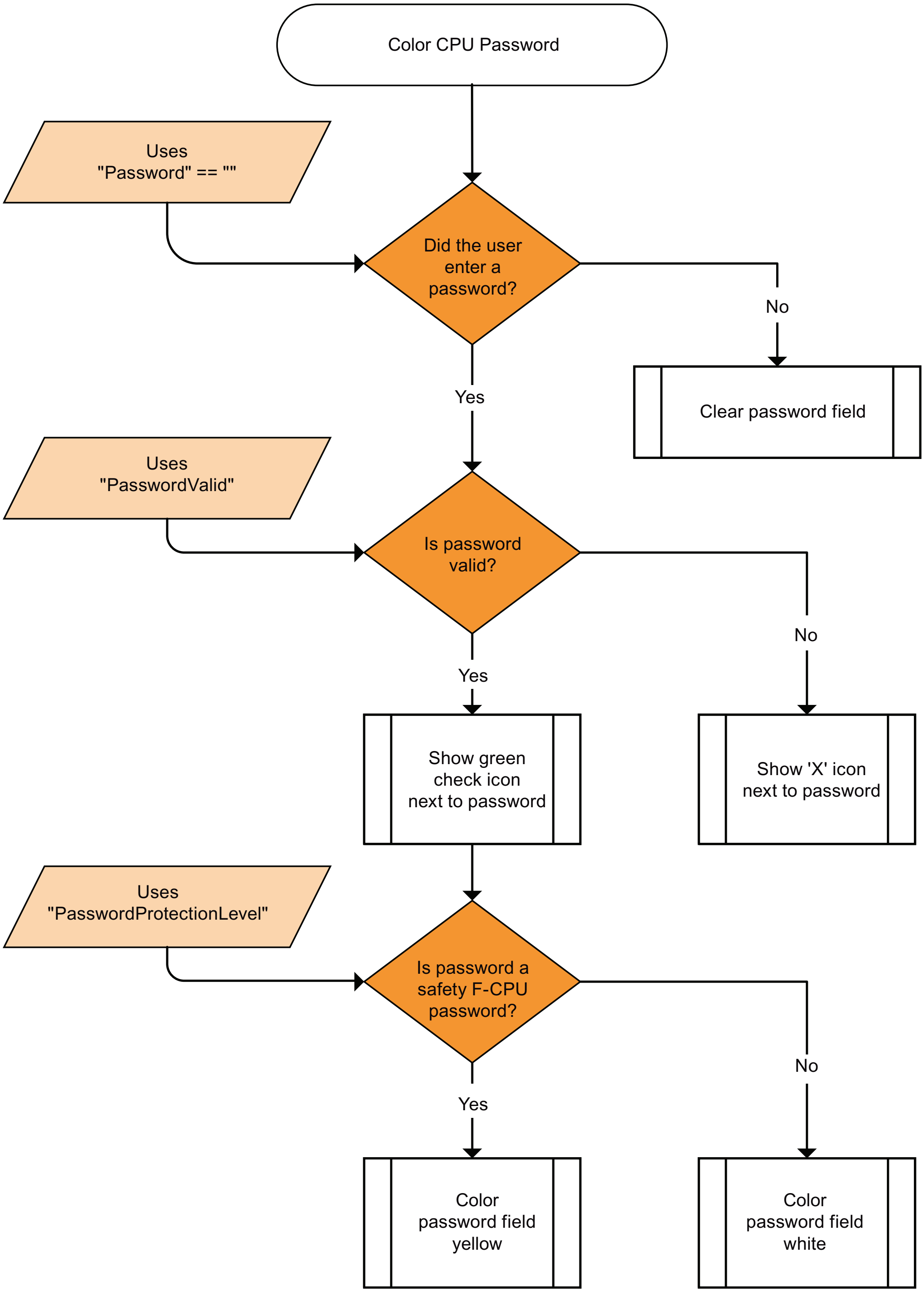
12.4.3.1 Coloring a CPU device icon


Color CPU Icon
Uses "Failsafe"
Is the device Fail-Safe?
Uses "IdentityCrisis"
Yes
Does the device have an identity problem?
Yes
No
Display "Fail-safe" CPU icon
Display "Fail-safe" CPU icon with identity problem

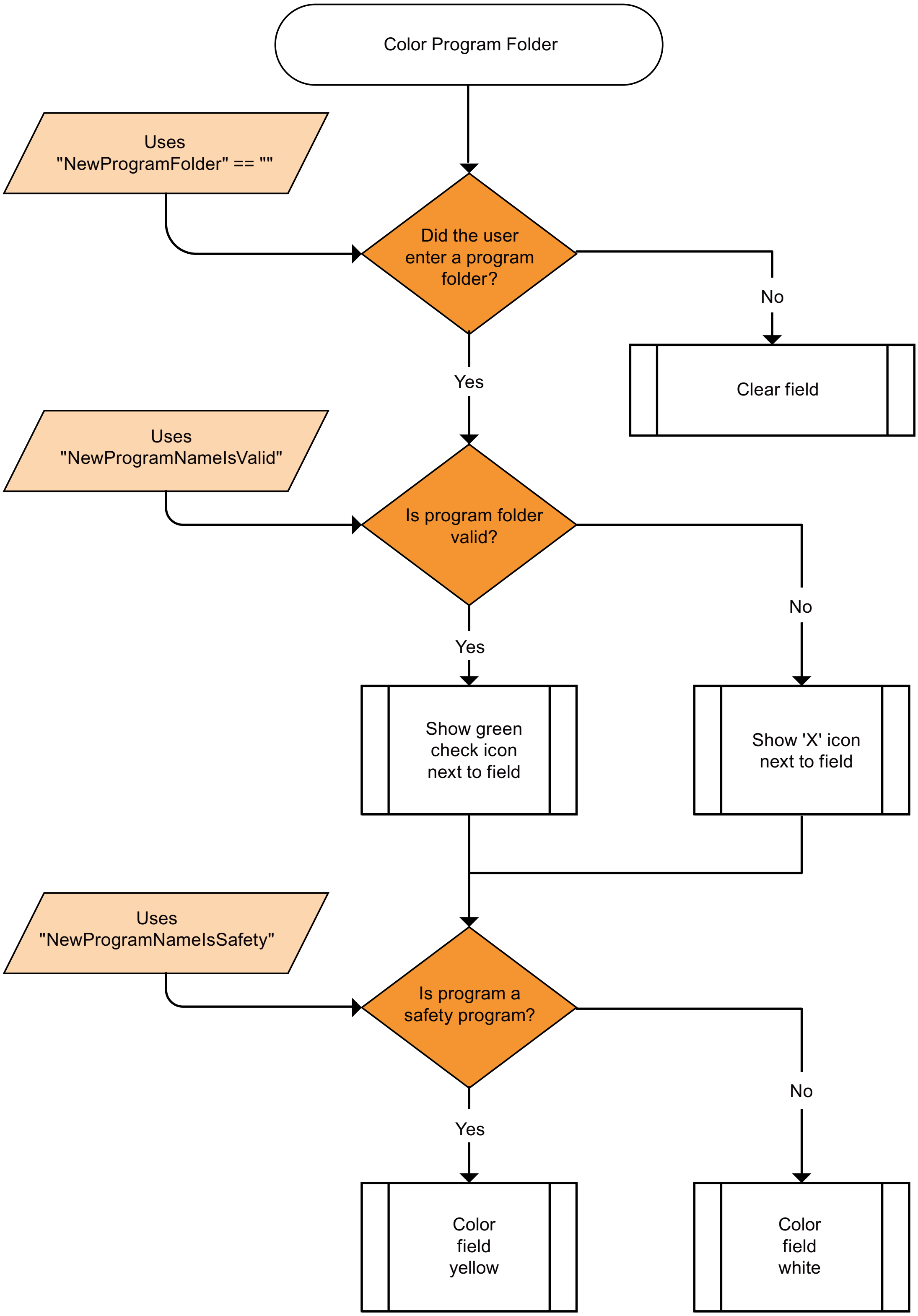
12.4.3.2 Coloring device data


Color Device
Uses "Failsafe"
Is device Fail-Safe?
Yes
No
Uses "HasSafetyProgram"
Does the device have a safety program ?
No
Yes
Color CPU Name and address fields white
Color CPU Name and address fields yellow

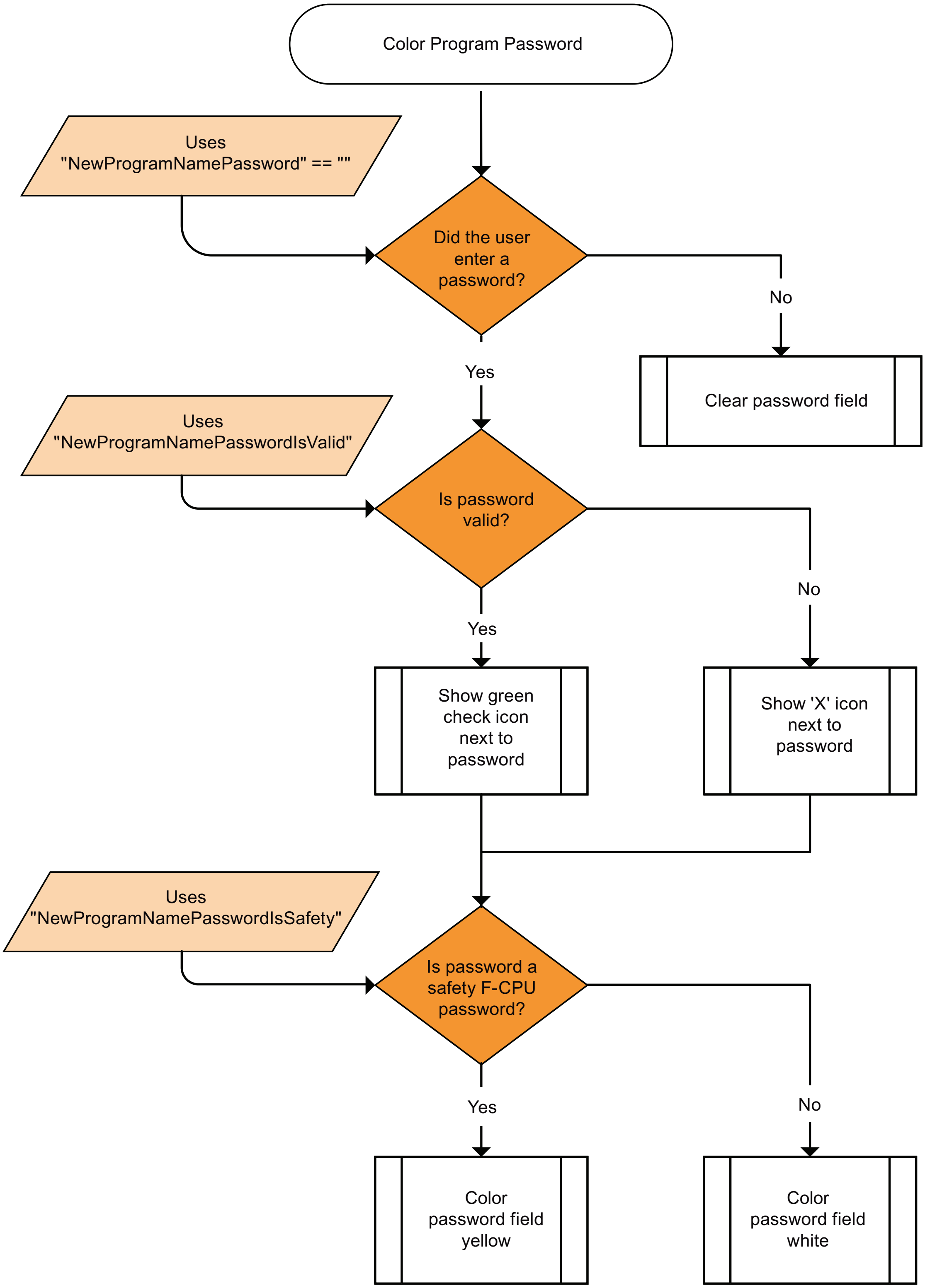
12.4.3.3 Coloring a CPU password


Color CPU Password
Uses "Password" == ""
Did the user enter a password?
No
Clear password field
Yes
Uses "PasswordValid"
Is password valid?
No
Yes
Show green check icon next to password
Show 'X' icon next to password
Uses "PasswordProtectionLevel"
Is password a safety F-CPU password?
No
Yes
Color password field yellow
Color password field white

12.4.3.4 Coloring a program folder


Color Program Folder
Uses "NewProgramFolder" == ""
Did the user enter a program folder?
No
Clear field
Yes
Uses "NewProgramNameIsValid"
Is program folder valid?
No
Yes
Show green check icon next to field
Show 'X' icon next to field
Uses "NewProgramNameIsSafety"
Is program a safety program?
No
Yes
Color field yellow
Color field white

12.4.3.5 Coloring a program password


Color Program Password
Uses "NewProgramNamePassword" == ""
Did the user enter a password?
No
Clear password field
Yes
Uses "NewProgramNamePasswordIsValid"
Is password valid?
No
Yes
Show green check icon next to password
Show 'X' icon next to password
Uses "NewProgramNamePasswordIsSafety"
Is password a safety F-CPU password?
Yes
No
Color password field yellow
Color password field white

12.4.4 Hamming codes

Hamming codes are binary codes. They can detect incidental bit errors. The SIMATIC Automation Tool API uses 32-bit hamming codes with a hamming distance of eight. The API uses hamming codes to represent all Boolean values related to safety-relevant operations. You can program your user interface application to use the provided Boolean value states for safety-relevant operations. Because the API implements these states with hamming codes, you can have high confidence in the data integrity of the safety-relevant Boolean states.

12.5 Common support classes

12.5.1 EncryptedString class

Many API operations on the `ICPU` interface require a legitimized connection to a protected S7 CPU. For these operations, the methods require a password as one of the parameters to the method. The CPU accepts the password in an encrypted format. To accomplish password encryption, the API provides the `EncryptedString` class.

Constructor	Description
`EncryptedString()`	An empty encrypted string
`EncryptedString(string strText)`	An encrypted string

Property name	Return type	Description
`IsEmpty`	`bool`	True, if there is no password
`IsEncrypted`	`bool`	True if there is an encrypted password

Method name	Return type	Description
`ToString()`	`string`	Hexadecimal string representation of the encrypted password
`Clear()`	`void`	Clears the encrypted password
`Copy(EncryptedString password)`	`void`	Copies the encrypted password
`GetHash()`	`byte[]`	Password encrypted hash array representation of the password
`WriteToStream(Stream stream)`	`void`	Serialize password from a stream
`ReadFromStream(Stream stream)`	`void`	Deserializes a password from a stream

This class provides a way to encrypt a plain text password so that you can legitimize a CPU connection. Many of the code examples show a typical usage of this class.

To encrypt a password to use multiple times in your code, instantiate the `EncryptedString`. You can then pass it as a parameter to one or more calls, as follows:

```
EncryptedString pwd = new EncryptedString("password");
myCPU.SetPassword(pwd);
```

The `EncryptedString` object does not store the user-specified plain text password. However, if your application codes passwords as literal strings, you create a security risk.

For example, `new EncryptedString("myPassword")` compiles the plain text "`myPassword`" into the user application. This password can be visible to others using .NET reflection.

12.5.2 Result class

The `Result` class provides information about whether a given API action succeeded. Inspect the `Result` object that API operations return to determine whether the operation succeeded or failed.

Constructor	Description
`Result()`	Creates successful result with no warnings
`Result(ErrorCode nCode)`	Creates specific error with no warnings
`Result(string strDefinedError)`	Device-defined error description
`Result(Result result)`	Creates a result with the result parameter contents

Property name	Return type	Description
`Warnings`	`ErrorCode[]{get;}`	Returns all warnings in an array of error codes
`Error`	`ErrorCode {get;}`	Returns error code
`HasWarnings`	`bool {get;}`	True when warnings exist
`Failed`	`bool {get;}`	True when the result failed
`Succeeded`	`bool {get;}`	True when the result is successful
`OMSCode`	`bool {get;}{set;}`	Gets and sets OMS code

Method Name	Return type	Description
`Clear()`	`void`	Clears error and all warnings
`SetError(ErrorCode error)`	`void`	Sets the current error
`AddWarning(ErrorCode warning)`	`void`	Adds a warning
`ChangeErrorToWarning()`	`void`	Changes current error to a warning

Method Name	Return type	Description
GetErrorDescription(Language language)	string	Gets current error string in specified language
GetWarningDescription(Language language)	string[]	Gets warning string in specified language

Example: Checking whether an operation succeeded

To check whether a given action was successful, check the Succeeded property:

```
ICPU devAsCpu = dev as ICPU;
if (devAsCpu != null)
{
   //-------------------------------------------------------
   // The device is a CPU.
   // You can use the ICPU interface to interact with it.
   //-------------------------------------------------------


    dev.Selected = true;
    Result retVal = dev.RefreshStatus();
    if (retVal.Succeeded)
    {
        //--------------------------------------
        // Continue operations....
        //--------------------------------------
    }
    dev.Selected = false;
}
```

Example: Checking specific error codes

In other cases, it can be helpful to have more information about the failure. To inspect the specific error, use the ErrorCode property (Page 295), as follows:

```
    dev.Selected = true;
    Result retVal = dev.RefreshStatus();
    if (retVal.Succeeded)
    {
        //--------------------------------------
        // Continue operations....
        //--------------------------------------
    }
    else
    {
        //--------------------------------------
        // What happened?
        //--------------------------------------
        switch (retVal.Error)
        {
            case ErrorCode.AccessDenied:
                break;
            case ErrorCode.TooManySessions:
                break;
        }
    }
    dev.Selected = false;
```

If the device returns an error that is a device-specific error, the returned `ErrorCode` is `DeviceDefinedError`. The method `GetErrorDescription` provides the device-specific error string.

Example: Getting the error description in a specific language

The `Result` class also provides a language-specific error description.
The `GetErrorDescription` method uses a `Language` value (Page 302)as a parameter to provide the error description in the specified language.

For example, the following code returns the error description in German:

```
String strError = retVal.GetErrorDescription(Language.German);
```

Example: Checking warnings

The API has a warnings feature for information about issues that have occurred. For instance, the Refresh that is performed on the device at the end of a `ProgramUpdate` can create warnings that are not directly related to the main calling function. You can access these warnings through the `Result` class, as follows:

```
    if (retVal.HasWarnings)
    {
        foreach (ErrorCode warning in retVal.Warnings)
        {
            //---------------------
            // Continue operations
            //---------------------
        }
    }
```

12.5.3 DiagnosticsItem class

A diagnostics item contains diagnostic information for a single event. You can read the diagnostic buffer from a CPU with the GetDiagnosticsBuffer method (Page 246).

Constructor	Description
`DiagnosticsItem()`	Creates default diagnostic item

Property Name	Return type	Description
`TimeStamp`	`DateTime {get;}`	Time stamp of the diagnostic event
`State`	`Byte {get;}`	0=Outgoing event; 1=Incoming event
`Description1`	`String {get;}`	Basic description
`Description2`	`String {get;}`	Detailed description

12.5.4 DataChangedEventArgs class

A data changed event contains information about data that has changed within the API. See the IProfinet interface (Page 203) chapter for details.

Constructor	Description
`DataChangedEventArgs(DataChangedType type)`	Creates event of specific type
`DataChangedType type`	Type of data that has changed

Property Name	Return type	Description
`Type`	`DataChangedType`	Type of event

The `DataChangedEventArgs` class is used with the following event handler:

```
public delegate void DataChangedEventHandler(object sender,
DataChangedEventArgs e);
```

12.5.5 ProgressChangedEventArgs class

A progress changed event contains information about data that has changed within the API. See the IProfinetDevice interface (Page 203) chapter for details.

Constructor	Description
`ProgressChangedEventArgs(ProgressAction action, int index, int count, uint hardwareID)`	Used to create and default a progress changed event args class
`ProgressChangedEventArgs(ProgressAction action, int index, int count, int fileNumber, uint hardwareID)`	Used to create and default a progress changed event args class
`ProgressAction action`	Type of progress that has occurred
`int index`	Index of current item being processed
`int count`	Total items to process
`int fileNumber`	Number of the file, if applicable
`uint hardwareID`	ID of the item being processed

Property Name	Return type	Description
`Action`	`ProgressAction {get;}`	Action type of this event
`Cancel`	`bool {get;set;}`	Set to true to terminate current operation
`Count`	`int {get;}`	Maximum value
`FileNumber`	`int {get;}`	Applicable file number
`ID`	`uint {get;}`	Hardware ID of item
`Index`	`int {get;}`	Current value

The `ProgressChangedEventArgs` class is used with the following event handler:

```
public delegate void ProgressChangedEventHandler(object sender,
ProgressChangedEventArgs e);
```

12.5.6 ExportProgressEventArgs class

An export progress changed event contains information about data that has changed within the API. Please see the section regarding The IProfinetDeviceCollection class.

Constructor	Description
ExportProgressEventArgs(int WorkItem, int MaxEntries)	Used to create and default an export progress changed event args class
int WorkItem	Index of current item being processed
int MaxEntries	Total items to process

Property Name	Return type	Description
`Cancel`	`bool {get;set;}`	Set to true to terminate current operation
`MaxEntries`	`int {get;}`	Maximum value
`WorkItem`	`int {get;}`	Current value

The `ExportProgressEventArgs` class is used with the following event handler:

```
public delegate void ExportProgressChangedEventHandler(object
sender, ExportProgressEventArgs e);
```

12.5.7 FileProgressChangedEventArgs class

A file progress changed event contains information about file data that has changed within the API. See the IProfinetDevice interface (Page 203) chapter for details.

Constructor	Description
`FileProgressChangedEventArgs(Progress Action action, int nIndex, int nCount)`	Used to create and default a file progress changed event args class
`FileProgressChangedEventArgs(Progress Action action)`	Used to create and default a progress changed event args class

Constructor	Description
`FileProgressChangedEventArgs()`	Used to create and default a progress changed event args class
`ProgressAction action`	Type of progress that has occurred
`int nIndex`	Index of current item being processed
`int nCount`	Total items to process

Property Name	Return type	Description
`Action`	`ProgressAction {get;set;}`	Action type of this event
`Cancel`	`bool {get;set;}`	Set to true to terminate current operation
`DoThisForRemainingOperations`	`bool {get;set;}`	Action done for remaining operations
`FileCount`	`uint {get;set;}`	Number of files
`FileIndex`	`int {get;set;}`	Index of current file
`FilePercentComplete`	`int {get;set;}`	Percent of action complete
`LocalFile`	`string {get;set;}`	Local file name
`RemoteFile`	`string {get;set;}`	Remote file name
`Replace`	`bool {get;set;}`	Replace file

Method Name	Return type	Description
`GetRemoteFileNameOnly()`	`string`	Get only the remote file name

The `FileProgressChangedEventArgs` class is used with the following event handler:

```
public delegate void FileProgressChangedEventHandler(object sender,
ProgressChangedEventArgs e);
```

12.5.8 HealthCheckProgressEventArgs class

A health check progress changed event contains information about the health check operation progress. Please see the section regarding The HealthCheck Class.

Constructor	Description
`HealthCheckProgressEventArgs(string name, int workItem, int maxEntries)`	Used to create and default a health check progress event args class
`string name`	Name of the health entry
`int workItem`	Index of current health entry being processed
`int maxEntries`	Total health entries count to process

Property Name	Return Type	Description
`Name`	`string {get;set;}`	Health check item name
`Cancel`	`bool {get;set;}`	Set to true to terminate current operation
`MaxEntries`	`int {get;set}`	Maximum number of health check items
`WorkItem`	`int {get;set;}`	Index of current health check item

The HealthCheckProgressEventArgs class is used with the following event handler:

```
public delegate void HealthCheckProgressUpdateHandler(object sender,
HealthCheckProgressEventArgs e);
```

12.6 Common support interfaces

12.6.1 IRemoteFile interface

IRemoteFile is an interface used to represent files used in datalogs and recipes.

Property name	Return type	Description
`Selected`	`bool {get; set;}`	Selected state
`FileSize`	`ulong {get;}`	Size on the file on the CPU
`Name`	`string {get;set;}`	Filename and extension on the device

12.6.2 IRemoteFolder interface

`IRemoteFolder` represents folders used in data logs and recipes.

Method name	Return type	Description
`SetRemoteFile(string strFile)`	`Result`	Set the remote file name
`string strFile`		Full file name and path of the remote file

Property name	Return type	Description
`Exists`	`bool {get;}`	True if this folder exists on the device
`Files`	`List<IRemoteFile> {get;}`	List of files in this folder
`FileUpdateAllowed`	`bool {get;}`	True if the folder can add or replace a file in the list
`FolderType`	`RemoteFolderType {get;}`	Type of folder (datalog or recipe)
`Name`	`string {get; set;}`	Name of the folder
`NewFile`	`string {get;}`	Full file path of the file to add or replace

Property name	Return type	Description
NewFileName	string {get;}	Name of the file to add or replace
NewFileNameErrorCode	Result {get;}	The error code saved after calling SetRemoteFile Method
NewFileNameIsValid	bool {get;}	True if the file name is valid
Selected	bool {get;set;}	True if the folder is selected
SelectedCount	int {get;}	The number of files selected

12.6.3 IRemoteInterface interface

IRemoteInterface is an interface used to represent distributed I/O on a network.

Property name	Return type	Description
Devices	List<IBaseDevice>{get;}	List of remote interfaces used to represent decentralized I/O
InterfaceType	RemoteInterfaceType{get;}	Type of remote interface such as PROFINET or PROFIBUS
Name	string {get;}	Filename and extension on the device

12.6.4 IHardware interface

IHardware is an interface used to represent the basic common hardware interface for devices and modules.

Method name	Return type	Description
SetFirmwareFile(string strFile)	Result	Sets the firmware file to update on this device or module
ValidateUserData()	Result	Confirms that the user data is valid.

Property Name	Return Type	Description
ArticleNumber	string {get;}	Article number of device or module
Comment	string {get;set;}	Comment for each device and module
Configured	bool {get;}	True if this device or module is configured
ConfiguredVersion	string {get;}	The configured version number for this device
Description	string {get;}	Description of article number of device or module
Failsafe	bool {get;}	True if device or module is failsafe
FirmwareUpdateSupported	bool {get;}	True when this device or module supports firmware update
FirmwareVersion	string {get;}	Firmware version of device or module
HardwareNumber	short {get;}	Hardware revision number of device or module

Property Name	Return Type	Description
`ID`	`uint {get;}`	ID of device or module
`Name`	`string {get;}`	Name of device or module
`NewFirmwareFile`	`string {get;}`	Full file path of the firmware file
`NewFirmwareNameErrorCode`	`Result {get;}`	Last error from SetFirmwareFile method
`NewFirmwareNameIsValid`	`bool {get;}`	True when the firmware file is valid for this device or module
`NewFirmwareVersion`	`string {get;}`	Version of the firmware file which is displayed in the dropdown
`Selected`	`bool {get;set;}`	Used to for external storage of the selected state
`SerialNumber`	`string {get;}`	Serial number of device or module
`Slot`	`uint {get;}`	Slot number of device or module
`SlotName`	`string {get;}`	Name of the slot of device or module
`StationNumber`	`uint {get;}`	Station number of device or module
`SubSlot`	`uint {get;}`	Sub slot number of device or module
`Supported`	`bool {get;}`	True if this device or module is supported

12.6.5 IBaseDevice interface

`IBaseDevice` is an interface used to extend the `IHardware` interface which represents the most basic device type.

Method name	Return type	Description
`GetHardwareFromID(uint hardwareID)`	`IHardware`	Finds a device or module using an ID

Property name	Return type	Description
`HardwareInDisplayOrder`	`IHardware`	Collection of hardware items in the order to be displayed
`HardwareInFirmwareOrder`	`IHardware`	Collection of hardware items in the order of firmware update order
`Modules`	`IModuleCollection`	Collection of modules
`ThreadNumber`	`int`	Current thread number of operation
`Family`	`DeviceFamily`	Family type enum

Events	Return type	Description
`ProgressChanged`	`ProgressChangedEventArgs`	Called to monitor progress
`DataChanged`	`DataChangedEventArgs`	Called when data changes in the API
`FileProgressChanged`	`FileProgressChangedEventArgs`	Called when an SAT initiated file transfer has updated progress to report.

12.6.6 IHardwareCollection interface

`IHardwareCollection` is an interface used to represent a list of `IHardware` interfaces. This interface extends a .NET List class.

Method Name	Return Type	Description
`GetHardwareFromID(uint hardwareID)`	`IHardware`	Finds a device or module using an ID.

Parameters			
`hardwareID`	`uint`	in	Hardware identifier number

12.6.7 IModuleCollection interface

`IModuleCollection` is an interface used to represent array `IModule` interface. This interface extends a .NET List class. This interface is used to represent local and remote modules in the hardware rack.

Property name	Type	Description
None		

12.6.8 IScanErrorCollection class

This interface class encapsulates the logic needed to determine whether a network scan for devices succeeded using the ScanNetworkDevices call. Additionally, this class provides error information for any device on the network for which a scan was not successful by returning a ScanErrorEvent for the device. The IScanErrorCollection object returned by a ScanNetworkDevices call should always be inspected for success or failure. This interface is added to allow the user to get more device specific scan error information that is useful in diagnosing network scan issues.

Constructor	Description
`ScanErrorCollection()`	Creates successful network scan result

Property Name	Return type	Description
`Count`	`Int`	Number of scan error events
`ScanErrorEvent[int ]`	`IScanErrorEvent`	Returns the designated scan event in the collection of scan events Returns null for an invalid element
`Failed`	`bool {get;}`	True when the result failed
`Succeeded`	`bool {get;}`	True when the result is successful `Succeeded` = `Failed`

The following conditions result in a failed scan status:

- No SAT license present or SDK installation
- Invalid network interface
- No devices found on scan
- Operation canceled by user

Method name	Return type	Description
`GetEnumerator()`	`IEnumerator<IScanErrorEvent >`	Gets an enumerator to the scan error event collection

12.6.9 IScanErrorEvent class

This class provides information about a failed scan attempt of a device as a result of a call to the `ScanNetworkDevices` (Page 186) method. The content of the `ScanErrorEvent` provides available information about the device and specifies the error as information, warning, error, or invalid.

Constructor	Description
`ScanErrorEvent()`	Creates a ScanErrorEvent

Property Name	Return type	Description
`Code`	`ErrorCode {get;}`	The scan error
`IP`	`Uint {get;}`	The IP address of the device
`MAC`	`ulong {get;}`	The MAC address of the device
`Name`	`string {get;}`	The name of the device
`TimeStamp`	`DateTime {get;}`	The time stamp associated with the error
`Type`	`ScanErrorType {get};`	The type of the scan on which the error occurred

12.6.10 IHealthCheckErrorEvent interface

Interface for getting data associated with a Health Check error event.

Property Name	Return Type	Description
`TimeStamp`	`DateTime{get;}`	Array of files in this folder listed in ascending order
`Type`	`LogType{get;}`	Array of files in this folder
`Description`	`string{get;}`	Type of folder (datalog or recipe)

12.6.11 IDiagnosticBuffer interface

The IDignosticBuffer interface is an interface for acquiring diagnostic buffers

<table>
<tr><th colspan="2">Method Name</th><th>Return Type</th><th>Description</th></tr>
<tr><td colspan="2">GetDiagnosticsBuffer()</td><td>Result</td><td>Get a diagnostic buffer</td></tr>
<tr><th colspan="4">Parameters</th></tr>
<tr><td>Name</td><td>Data type</td><td>Parameter type</td><td>Description</td></tr>
<tr><td>aDiagnosticItems</td><td>List<DiagnosticsItem></td><td>out</td><td>Collection of diagnostic items</td></tr>
<tr><td>language</td><td>Language</td><td>in</td><td>Language for the diagnostic items</td></tr>
</table>

Property Name	Return Type	Description
DiagBufferAllowed	bool{get;}	Diagnostic buffer acquisition is allowed
ScheduleDeviceDiagnostics	bool{get;set;}	Schedule get diagnostics operation

12.6.12 IResult interface

The `IResult` interface supports the `IResult` class.

Property name	Return type	Description
Warnings	ErrorCode[]{get;}	Returns all warnings in an array of error codes
Error	ErrorCode {get;}	Returns error code
HasWarnings	bool {get;}	True when warnings exist
Failed	bool {get;}	True when the result failed
Succeeded	bool {get;}	True when the result is successful Succeeded = not Failed

Method Name	Return type	Description
GetErrorDescription(Language language)	string	Gets current error string in specified language
GetWarningDescription(Language language)	string[]	Gets warning string in specified language

12.7 Network class

12.7.1 Network constructor

The `.NET` class `Network` performs functions using a network interface card (NIC) installed on the programming device. The `Network` class is used to search for available interface cards and to select the interface card that communicates with the PROFINET network.

To interact with the PROFINET network, your program declares a variable of type `Network`, as follows:

```
Network myNetwork = new Network();
```

You can use this object to find available network interfaces and to select a network interface.

12.7.2 QueryNetworkInterfaceCards method

Return type	Method name
`Result`	`QueryNetworkInterfaceCards`

Parameters			
Name	**Data type**	**Parameter type**	**Description**
`aInterfaces`	`List<string>`	`Out`	A collection of all the network interface cards on the programming device listed by name

To identify the available network interface cards, use the `QueryNetworkInterfaceCards` method. The method outputs a list of strings. Each item in the list represents an available network interface identified by name.

Example: Querying the network interfaces

See the example and description in Getting started with the API (Page 149) for the procedure of querying the network interfaces on your computer or programming device.

12.7.3 SetCurrentNetworkInterface method

Return type	Method name
`Result`	`SetCurrentNetworkInterface`

Parameters			
Name	**Data type**	**Parameter type**	**Description**
`strInterface`	`string`	In	The name of the network interface to use. It is one of the names returned from the `QueryNetworkInterfaceCards` method.

To use one of the network interfaces on your computer or programming devices to access the PROFINET network, you must "set" this interface. `SetCurrentNetworkInterface` enables your application to communicate with a specific PROFINET network and the devices on that network.

Example: Setting the network interface

See the example and description in Getting started with the API (Page 149) for the procedure of setting the network interface.

See also

QueryNetworkInterfaceCards method (Page 184)

12.7.4 CurrentNetworkInterface property

This read-only property queries for the currently-selected network interface. The following example shows how to use this property:

This property returns an empty string if no network interface was selected by a previous call to the `SetCurrentNetworkInterface` (Page 184) method.

Example: Getting the current network interface

```
//-------------------------------------------------------------
// Insert the necessary code from Getting started with the API (Page 149)
// here if you want to compile the example
//-------------------------------------------------------------

#region Getting the current network interface
string currentInterface = myNetwork.CurrentNetworkInterface;

/* For simplicity the code examples above do not check for */
/* errors. Checking for and processing errors returned     */
/* from methods is vital to a program's overall quality    */

#endregion
```

12.7.5 ScanNetworkDevices method

After you set a network interface (Page 184), you can search for the devices on the PROFINET network. The `ScanNetworkDevices` method outputs an `IProfinetDeviceCollection` (Page 191). Each item in the collection represents a device connected directly to the PROFINET network. These devices can include CPUs, HMIs, and other devices.

Return type	Method name
`IScanErrorCollection`	`ScanNetworkDevices`

Parameters			
Name	**Data type**	**Parameter type**	**Description**
`baseDevices`	`IProfinetDeviceCollection`	Out	A collection containing an `IProfinetDevice` element for each accessible device on the PROFINET network

Scanning a network with many devices can take several minutes. The Succeeded property in the returned `IScanErrorCollection` (Page 181) tells whether the scan succeeded or failed.

Note

Requirements for use of `ScanNetworkDevices` method.

You must have a valid and unexpired SAT license or an installation of the SDK to successfully use the ScanNetworkDevices method. If the SDK or a valid unexpired SAT license is not present at runtime, the ScanNetworkDevices method returns an empty collection. ScanNetworkDevices returns no device information to the calling application.

Example: Scanning the network for devices

See the example and descriptions in Getting started with the API (Page 149) for the complete procedure and required information for scanning a network.

12.7.6 SetCommunicationsTimeout method

You can set a time limit on S7-1200 and S7-1500 CPU communication operations that you call using the API. `SetCommunicationsTimeout` allows you to specify a time limit in seconds from 180 to 999 seconds. Any value outside of this range results in a failure of the operation.

Return type	Method name
`Result`	`SetCommunicationsTimeout`

Parameters			
Name	**Data type**	**Parameter type**	**Description**
nTimeout	uint	In	Specified time for operations to time out

Example: Setting the communications timeout

```
//---------------------------------------------------------------
// Insert the necessary code from Getting started with the API (Page 149)
// here if you want to compile the example
//---------------------------------------------------------------

#region Setting the CPU network communications timeout

// Get and set the CPU network communications timeout
uint timeout = Network.GetCommunicationsTimeout();
if (timeout > 180)   // Set timeout to 3 minutes
{
   retVal = Network.SetCommunicationsTimeout(180);
}

/* For simplicity the code examples above do not check for */
/* errors. Checking for and processing errors returned     */
/* from methods is vital to a program's overall quality    */

#endregion
```

12.7.7 GetCommunicationsTimeout method

You use the `GetCommunicationsTimeout` method to retrieve the CPU network communications timeout value in seconds.

Return type	Method name
uint	GetCommunicationsTimeout

Example: Setting the communications timeout

```
//---------------------------------------------------------------
// Insert the necessary code from Getting started with the API (Page 149)
// here if you want to compile the example
//---------------------------------------------------------------

#region Getting the CPU network communications timeout

// Get the CPU network communications timeout
uint timeout = Network.GetCommunicationsTimeout();

/* For simplicity the code examples above do not check for */
/* errors. Checking for and processing errors returned     */
/* from methods is vital to a program's overall quality    */
```

```
#endregion
```

12.7.8 GetEmptyCollection method

The `GetEmptyCollection` method returns an empty `IProfinetDeviceCollection`. You can then insert devices into the empty collection. Inserting devices into an empty collection is an alternative to scanning the network for devices.

Return type	Method name
`IProfinetDeviceCollection`	`GetEmptyCollection`

Example: Querying the network interfaces

See the example and description in Getting started with the API (Page 149) for the procedure of getting an empty collection with the `GetEmptyCollection` method and inserting devices.

12.7.9 ValidateNetworkInterface method

Call the `ValidateNetworkInterface` method to determine whether the network interface is valid. If the method cannot find the network interface, it returns the error “InvalidNetworkInterface”. This error could result if the interface card has been removed or is in a failed state.

<table>
<tr><th colspan="2">Method Name</th><th colspan="2">Return Type</th><th>Description</th></tr>
<tr><td colspan="2"><code>ValidateNetworkInterface()</code></td><td colspan="2"><code>Result</code></td><td>Verify if this network interface is (still) valid on this system.</td></tr>
<tr><td colspan="5"><b>Parameters</b></td></tr>
<tr><td><b>Name</b></td><td><b>Data type</b></td><td><b>Parameter type</b></td><td colspan="2"><b>Description</b></td></tr>
<tr><td><code>nTimeout</code></td><td><code>string</code></td><td>in</td><td colspan="2">Network interface to be validated</td></tr>
</table>

12.8 The HealthCheck class

12.8.1 HealthCheck constructor

The .NET class `HealthCheck` supports creation of the PC data file and is defined in the `SimaticAutomationToolHealthCheck.dll`.

To interact with the PROFINET network, your program instantiates an object of type `HealthCheck`, as follows:

```
HealthCheck myHealthCheck = new HealthCheck();
```

12.8.2 ExportPCData method

The `ExportPCData` method creates a HealthCheck.zip file containing data about the programming device. The returned `HealthCheckResultType` object indicates the outcome of the operation. The HealthCheck.log file in the HealthCheck.zip file contains details about the exported data or about errors that occurred.

Return type	Method name
`HealthCheckResultType`	`ExportPCData`

Parameters			
Name	**Data type**	**Parameter type**	**Description**
`filePath`	`string`	`In`	The complete file path for storing the zip file containing the exported PC data

Example: Exporting PC data

```
using Siemens.Automation.AutomationTool.HealthCheck;

#region Exporting PC data

HealthCheck myHealthCheck = new HealthCheck();
String healthCheckFilePath = @"c:\export\healthcheck.zip";
HealthCheckResultType hcResult =
myHealthCheck.ExportPCData(healthCheckFilePath);
if (hcResult == HealthCheckResultType.Success)
{
  //------------------------------------------------------
  // Continue with operations.
  // The HealthCheck.log file in the HealthCheck.zip file
  // contains detailed information about the exported data.
  //-------------------------------------------------------
}
else
{
  //-------------------------------------------------------------
  // The action failed, had warnings, or was canceled by user.
  //
  // If the operation was not canceled, the HealthCheck.log file
  // in the HealthCheck.zip file contains information
  // about the export operation.
  //-------------------------------------------------------------
}
#endregion
```

Example: Using the ProgressChanged event to monitor the progress of a PC data export

The API provides the `ProgressChanged event` (Page 227) to monitor the progress of methods that can take a long time. `ExportPCData` is a method that can take a long time.

To use the `ProgressChanged` event to monitor the progress of `ExportPCData`, attach an event handler to the event. A change in the progress of the operation then automatically calls the event handler.

The following example shows how to use the `ProgressChanged` event to monitor the progress of the PC data export. The example code defines an event handler and attaches it to the `ProgressChanged` event. When `ExportPCData` finishes, the example code detaches the event handler from the event.

```
using Siemens.Automation.AutomationTool.HealthCheck;

#region Monitoring the progress of exporting PC data

{
HealthCheck myHealthCheck = new HealthCheck();
String healthCheckFilePath = @"c:\export\healthcheck.zip";
// Add enrollment for progress event
myHealthCheck.ProgressChanged += HealthCheck_ProgressChanged;

HealthCheckResultType hcResult =
myHealthCheck.ExportPCData(healthCheckFilePath);

// Remove enrollment for progress event
myHealthCheck.ProgressChanged -= HealthCheck_ProgressChanged;
}

void HealthCheck_ProgressChanged(object
sender,HealthCheckProgressEventArgs e)
{
   String strProgress = String.Format("Processing {0} of {1}",
e.WorkItem, e.MaxEntries);
   // Set to true to cancel to terminate progress
   e.Cancel = false;
}

/* For simplicity the code examples above do not check for */
/* errors. Checking for and processing errors returned     */
/* from methods is vital to a program's overall quality    */

#endregion
```

12.9 IProfinetDeviceCollection class

12.9.1 Iterating items in the collection

12.9.1.1 Iterating items in the collection

The `ScanNetworkDevices` (Page 186) method outputs an object of type `IProfinetDeviceCollection` (Page 190). This class provides the ability to iterate the items in the collection using `foreach` syntax or to access each `IProfinetDevice` (Page 203) using array syntax.

Scanning a network with many devices can take several minutes. The `Succeeded` property in the returned `IScanErrorCollection` tells whether the scan succeeded or failed.

Example: Iterating through each device in the collection

```
//-------------------------------------------------------------
// Insert the necessary code from Getting started with the API (Page 149)
// here if you want to compile the example
//-------------------------------------------------------------

#region Iterating through each PROFINET device
foreach (IProfinetDevice dev in scannedDevices)
{
   //--------------------------------------------
   // The variable "dev" represents the
   // next item in the IProfinetDeviceCollection
   //--------------------------------------------
}
#endregion
```

Example: Iterating through the scanned devices as an array

```
//-------------------------------------------------------------
// Insert the necessary code from Getting started with the API (Page 149)
// here if you want to compile the example
//-------------------------------------------------------------

#region Iterating through the scanned devices as an array

for (int deviceIdx = 0; deviceIdx < scannedDevices.Count; deviceIdx+
+)
{
   //----------------------------------------------------
   // Each item in the collection is an IProfinetDevice.

   // The variable "dev" is the next indexed device.
   //----------------------------------------------------
   IProfinetDevice dev = scannedDevices[deviceIdx];
}
#endregion
```

12.9.1.2 GetEnumerator method

Return type	Method name
`IEnumerator<IProfinetDevice>`	`GetEnumerator`

Parameters			
Name	**Data type**	**Parameter type**	**Description**
None			

This method is used to enumerate all `IProfinetDevices` in the `IProfinetDeviceCollection`.

12.9.1.3 SortByIP method

Method Name	Return Type	Description
`SortByIP()`	`void`	Sorts the device collection in a numeric sequence based on IP address.

12.9.1.4 Count property

Return type	Property name
`int`	`Count`

This property returns the count of the number of `IProfinetDevices` in the `IProfinetDeviceCollection.`

12.9.1.5 [ ] property

Return type	Property name
`IProfinetDevice`	`this[int index]`

This property returns the `IProfinetDevice` at a specific index. See the example below:

```
IProfinetDeviceCollection collection = Network.GetEmptyCollection();
MemoryStream stream = new MemoryStream();

Result result = collection.WriteToStream(stream);
if (retVal.Succeeded)
```

```
{
    //-----------------------------------------------------
    // Collection was serialized successfully
    //-----------------------------------------------------

    IProfinetDevice device = collection[0];
}
```

12.9.2 Filtering items in the collection

12.9.2.1 Collection items

The collection contains an item for each device on the PROFINET network. The collection can contain devices from different multiple product families, for example, S7-1200, S7-1500, and ET 200S).

The collection can also contain different "categories" of devices (for example, CPUs or IO stations). For different categories of devices, specific operations are available. You might find it useful to filter the collection to include only certain devices.

12.9.2.2 FilterByDeviceFamily method

The `FilterByDeviceFamily` method returns a collection that includes only devices of the specified product families.

Return type	Method name
`List<IProfinetDevice>`	`FilterByDeviceFamily`

Parameters			
Name	**Data type**	**Parameter type**	**Description**
`familiesToInclude`	`List<DeviceFamily>`	In	Device family types to return in the list

First, contruct a filter of one or more device families. Pass this filter to the `FilterByDeviceFamily` method. The result is an `IProfinetDeviceCollection` that contains only the devices of the specified product families,

Note

Passing an empty `List<DeviceFamily>` returns an empty collection.

Example: Filtering the scanned devices by CPU family

This example filters the scanned devices for only S7-1200 and S7-1500 devices.

```
//-------------------------------------------------------------
// Insert the necessary code from Getting started with the API (Page 149)
// here if you want to compile the example
//-------------------------------------------------------------
#region Filtering devices by S7-1200 and S7-1500 devices

List<DeviceFamily> families = new List<DeviceFamily>
{ DeviceFamily.CPU1200, DeviceFamily.CPU1500 };
List<IProfinetDevice> onlyPlus =
scannedDevices.FilterByDeviceFamily(families);
#endregion
```

12.9.2.3 FilterOnlyCPUs method

The API supports many operations that only apply to CPUs. The `FilterOnlyCPUs` method provides the ability to filter the collection to include only the CPUs on the scanned network.

Return type	Method name
`List<ICPU>`	`FilterOnlyCPUs`

This method returns an `ICPU` list. The `ICPU` interface (Page 230) provides the properties and methods for CPUs.

Example: Filtering the scanned devices by CPU family

This example filters the scanned devices for only S7-1200 and S7-1500 devices.

```
//-------------------------------------------------------------
// Insert the necessary code from Getting started with the API (Page 149)
// here if you want to compile the example
//-------------------------------------------------------------
#region Filtering only CPUs from the scanned devices

List<ICPU> cpus = scannedDevices.FilterOnlyCpus();
foreach (ICPU cpu in cpus)
{
   //----------------------------------------------------------
   // Iterate through the list that only includes CPU devices
   //----------------------------------------------------------
}
#endregion
```

12.9.3 Finding a specific device in the collection

12.9.3.1 FindDeviceByIP method

Use the `FindDeviceByIP` method to search for a specific device in the collection by its IP address.

Return type	Method name
`IProfinetDevice`	`FindDeviceByIP`

Parameters			
Name	**Data type**	**Parameter type**	**Description**
`ip`	`uint`	In	The IP address to search for

Example: Finding the device at a specific IP address

```
//------------------------------------------------------------
// Insert the necessary code from Getting started with the API (Page 149)
// here if you want to compile the example
//------------------------------------------------------------

#region Finding device an an IP address
IProfinetDevice dev = scannedDevices.FindDeviceByIP(0xC0A80001); //
192.168.0.1
if (dev != null)
{
    // Found it
}
#endregion
```

12.9.3.2 FindDeviceByMAC method

Use the `FindDeviceByMAC` method to search for a specific device in the collection by its MAC address.

Return type	Method name
`IProfinetDevice`	`FindDeviceByMAC`

Parameters			
Name	**Data type**	**Parameter type**	**Description**
`mac`	`ulong`	In	The MAC address to search for

Example: Finding the device with a specific MAC address

```
//---------------------------------------------------------------
// Insert the necessary code from Getting started with the API (Page 149)
// here if you want to compile the example
//---------------------------------------------------------------

#region Finding device with a specific MAC address
IProfinetDevice dev =
scannedDevices.FindDeviceByMAC(0x112233445566);
// MAC address: 11:22:33:44:55:66
if (dev != null)
{
    // Found it
}
#endregion
```

12.9.4 Serialization of the device collection

12.9.4.1 WriteToStream method

Use the `WriteToStream` method to store the contents of the collection to an external destination such as a file.

Return type	Method name
`Result`	`WriteToStream`

Parameters			
Name	**Data type**	**Parameter type**	**Description**
`stream`	`Stream`	In	Destination for serialized output of the collection

Example: Writing a collection to a file

```
//---------------------------------------------------------------
// Insert the necessary code from Getting started with the API (Page 149)
// here if you want to compile the example
//---------------------------------------------------------------

#region Writing the device collection to a file
FileStream f = File.Create("myDataFile.SAT");

retVal = scannedDevices.WriteToStream(f);

f.Close();

/* For simplicity the code examples above do not check for */
/* errors. Checking for and processing errors returned     */
/* from methods is vital to a program's overall quality    */
```

```
#endregion
```

12.9.4.2 ReadFromStream method

The `ReadFromStream` method creates a collection from a previously-created serialization file.

Return type	Method name
`Result`	`ReadFromStream`

Parameters			
Name	**Data type**	**Parameter type**	**Description**
`stream`	`Stream`	In	The source for de-serializing the collection

Example: Creating a collection from a file

```
//----------------------------------------------------------------
// Insert the necessary code from Getting started with the API (Page 149)
// here if you want to compile the example
//----------------------------------------------------------------

#region Reading a device collection from a file
IProfinetDeviceCollection devices = Network.GetEmptyCollection();

FileStream f = File.OpenRead("myDataFile.SAT");

retVal = devices.ReadFromStream(f);

f.Close();

/* For simplicity the code examples above do not check for */
/* errors. Checking for and processing errors returned     */
/* from methods is vital to a program's overall quality    */

#endregion
```

12.9.4.3 ExportDeviceInformation method

The `ExportDeviceInformation` method creates and exports a .csv file containing the current device collection data:

Return type	Method name
`Result`	`ExportDeviceInformation`

Parameters			
Name	**Data type**	**Parameter type**	**Description**
`ExportFilePath`	`string`	In	Destination file path for the generated export file

Example: Exporting device information

```
//---------------------------------------------------------------
// Insert the necessary code from Getting started with the API (Page 149)
// here if you want to compile the example
//---------------------------------------------------------------

#region Exporting device information to a file


String exportFilePath = @"c:\export\DeviceInformation.csv";
retVal = scannedDevices.ExportDeviceInformation(exportFilePath);

/* For simplicity the code examples above do not check for */
/* errors. Checking for and processing errors returned     */
/* from methods is vital to a program's overall quality    */

#endregion
```

12.9.4.4 ExportDeviceDiagnostics method

The `ExportDeviceDiagnostics` method creates and exports a .csv file containing diagnostic data of each CPU in the current device collection. Column headers in the .csv file are in English.

Return type	Method name
`IScanErrorCollection`	`ExportDeviceDiagnostics`

Parameters			
Name	**Data type**	**Parameter type**	**Description**
`strPath`	`string`	In	Destination file path for the generated export file
`language`	`Language`	In	Language for the exported diagnostic buffer entries
`format`	`TimeFormat`	In (optional)	Format for displaying date and time for diagnostic entries

For each CPU in the collection, the returned `IScanErrorCollection` (Page 181) designates whether acquistion of the diagnostic data was successful. If a required password was not provided or if a network error occurred, then the method returns an error for the CPU. An error results in storing a null data entry for the CPU.

Example: Exporting device diagnostics

```
//---------------------------------------------------------------
// Insert the necessary code from Getting started with the API (Page 149)
// here if you want to compile the example
//---------------------------------------------------------------

#region Exporting device diagnostics to a file
```

```
//------------------------------------------
// Select all CPUs in the device list to
// include for diagnostic information
//------------------------------------------
foreach (IProfinetDevice profi in scannedDevices)
{
   if((profi.Family == DeviceFamily.CPU1200)||
      (profi.Family == DeviceFamily.CPU1500)||
      (profi.Family == DeviceFamily.CPU300) ||
      (profi.Family == DeviceFamily.CPU400))

   profi.Selected = true;
}


// With the API, the client application determines the file path and
name

String exportFilePath = @"c:\export\Device\DeviceDiagnostics.csv";

IScanErrorCollection exportResult =
   scannedDevices.ExportDeviceDiagnostics(exportFilePath,
Language.English,TimeFormat.Local);

/* For simplicity the code examples above do not check for */
/* errors. Checking for and processing errors returned     */
/* from methods is vital to a program's overall quality    */

#endregion
```

Example: Using the ProgressChanged event to monitor the progress of an export of device diagnostics

The API provides the `ProgressChanged` (Page 227) event to monitor the progress of methods that can take a long time. ExportDeviceDiagnostics is a method that can take a long time.

To use the ProgressChanged event to monitor the progress of `ExportDeviceDiagnostics`, attach an event handler to the event. A change in the progress of the operation then automatically calls the event handler.

The following example shows how to use the `ProgressChanged` event to monitor the progress of an export of device diagnostics. The example code defines an event handler and attaches it to the `ProgressChanged` event. The code then calls the `ExportDeviceDiagnostics` method, which can take a long time. When `ExportDeviceDiagnostics` finishes, the example code detaches the event handler from the event.

```
//---------------------------------------------------------------
// Insert the necessary code from Getting started with the API (Page 149)
// here if you want to compile the example
//---------------------------------------------------------------

#region Monitoring the progress of exporting device diagnostics

{
// Enroll for progress event
scannedDevices.ProgressChanged += Export_ProgressChanged;
IScanErrorCollection ExportDeviceDiagnosticsErrors =
scannedDevices.ExportDeviceDiagnostics(@"C:\MyDocuments
\DeviceDiagnostics.csv", Language.English, TimeFormat.UTC);

// Remove enrollment for progress event

scannedDevices.ProgressChanged -= Export_ProgressChanged;

}
void Export_ProgressChanged(object sender,ExportProgressEventArgs e)
{
   String strProgress = String.Format("Processing {0} of {1}",
e.WorkItem, e.MaxEntries);
   // Set to true to cancel to terminate progress
   e.Cancel = false;
}

/* For simplicity the code examples above do not check for */
/* errors. Checking for and processing errors returned     */
/* from methods is vital to a program's overall quality    */

#endregion
```

12.9.5 Manually adding items to the collection

The API provides the following methods for inserting a device to the collection:

- InsertDeviceByIP method (Page 200)
- InsertDeviceByMAC method (Page 201)

Depending on the physical topology of the PROFINET network, devices might exist on the network that cannot respond to a DCP command, but that you can add by IP address. Additionally, you might choose to design your application for inserting devices rather than discovering devices by a network scan.

For devices behind an IP address router or NAT router, you can only insert them into the collection. A network scan does not discover devices behind an IP address router or NAT router. See the topic about devices behind routers (Page 114) and open port requirements.

12.9.5.1 InsertDeviceByIP method

The `InsertDeviceByIP` method adds a device to the `IProfinetDeviceCollection` (Page 191). The method inserts the device with a specific IP address at the specified index.

Return type	Method name
`Result`	`InsertDeviceByIP`

Parameters			
Name	**Data type**	**Parameter type**	**Description**
index	int	In	Location in the collection to insert the value
ip	uint	In	The IP address of the device to add to the collection
routerIP	uint	In (optional)	When the device is behind a router, this is the applicable router IP address

Example: Inserting a device by IP address

See the Getting started with the API (Page 149) example for inserting devices into an empty collection.

The method call in the example is as follows:

```
retVal = insertedDevices.InsertDeviceByIP(0, 0xC0A80001); // 192.168
.0.1
```

12.9.5.2 InsertDeviceByMAC method

The `InsertDeviceByMAC` method adds a device to the `IProfinetDeviceCollection` (Page 191). The method inserts the device with a specific MAC address at the specified index.

Return type	Method name
Result	InsertDeviceByMAC

Parameters			
Name	**Data type**	**Parameter type**	**Description**
index	int	In	Location in the collection to insert the value
mac	ulong	In	The MAC Address of the device to add to the collection

Example: Inserting a device by MAC address

```
//------------------------------------------------------------
// Insert the necessary code from Getting started with the API (Page 149)
// here if you want to compile the example
//------------------------------------------------------------

#region Inserting a device by MAC address
UInt64 targetMAC = 0x112233445566; // equivalent to string
11:22:33:44:55:66
retVal = insertedDevices.InsertDeviceByMAC(0, targetMAC);
}
```

```
/* For simplicity the code examples above do not check for */
/* errors. Checking for and processing errors returned     */
/* from methods is vital to a program's overall quality    */

#endregion
```

12.9.6 Copying data from a collection

12.9.6.1 CopyUserData method

Use the `CopyUserData` method to copy the user-entered data from one `IProfinetDeviceCollection` to another. You can use this method to avoid requiring your user to re-enter data.

Return type	Method name
`Result`	`CopyUserData`

Parameters			
Name	**Data type**	**Parameter type**	**Description**
`oldNetwork`	`IProfinetDeviceCollection`	In	Previous list used in the application

Example: Copying user data from one scan to another

```
//-----------------------------------------------------------
// Insert the necessary code from Getting started with the API (Page 149)
// here if you want to compile the example
//-----------------------------------------------------------

#region Copying user data from a former scan to a new scan

IProfinetDeviceCollection rescannedDevices;
IScanErrorCollection scanResult = myNetwork.ScanNetworkDevices(out
rescannedDevices);
retVal = rescannedDevices.CopyUserData(scannedDevices);

/* For simplicity the code examples above do not check for */
/* errors. Checking for and processing errors returned     */
/* from methods is vital to a program's overall quality    */

#endregion
```

12.9.7 Removing devices from the collection

12.9.7.1 Clear method

The `Clear` method clears the contents of the scanned devices.

Return type	Method name
`void`	`Clear`

Parameters			
Name	**Data type**	**Parameter type**	**Description**
None			

12.9.7.2 Remove method

The `Remove` method deletes a specific device from the collection.

Return type	Method name
`void`	`Remove`

Parameters			
Name	**Data type**	**Parameter type**	**Description**
`device`	`IProfinetDevice`	In	Device to remove from the collection

12.10 IProfinetDevice interface

12.10.1 IProfinetDevice properties

Each item in the `IProfinetDeviceCollection` (Page 191) collection is represented by the `IProfinetDevice` interface. This interface provides access to the data and provides operations for all devices that are directly connected to the PROFINET network.

The `IProfinetDevice` interface supports the following properties that provide information about the network device. To ensure the properties return the current information, call the `RefreshStatus` (Page 219) method on the device before reading a property.

Property name	Return type	Description
`ArticleNumber`	`string {get;}`	The order number for the module. This is also known as MLFB or "article number".
`Comment`	`string {get;set;}`	This allows the user to specify a comment for the device and is used in the SIMATIC Automation Tool user interface. The comment is not relevant for API operations.
`Configured`	`bool {get;}`	True when the device has a valid configuration
`DefaultGateway`	`uint {get;}`	The default gateway address of the device, represented as an unsigned integer. The encoded gateway address uses one byte to represent each decimal value in the address. For example, the encoded value 0xC0A80001 is equivalent to the more common string representation of 192.168.0.1
`DefaultGatewaystring`	`string {get;}`	The default gateway address of the device, represented as a string in the form "xx.xx.xx.xx" Example: 192.168.0.1
`Description`	`string {get;}`	A description of the hardware item, based on the article number. This is the same description that the user would see in TIA Portal. Example: CPU 1215 DC/DC/DC
`DeviceFound`	`bool {get;}`	Was the device found on the network scan?
`DuplicateIP`	`bool {get;}`	Does the device have an IP address that is a duplicate?
`DuplicateNewIP`	`bool {get;}`	A proposed New IP address is duplicated
`DuplicateProfinetName`	`bool {get;}`	Does the device have a PROFINET Name that is a duplicate?
`DuplicateNewProfinetName`	`bool {get;}`	A proposed new PROFINET name is duplicated
`Failsafe`	`bool {get;}`	Based on its ArticleNumber, Is this a fail-safe device?

Property name	Return type	Description
Family	DeviceFamily {get;}	What is the family of the device? For more information refer to the description of the DeviceFamily enum (Page 295).
FamilyName	string {get;}	What is the family name of the device?
FirmwareUpdateAllowed	bool {get;}	Are firmware updates allowed for this device?
FirmwareVersion	string {get;}	Current firmware version of the device
ID	uint {get;}	The unique identifier for every device and module in the station. This is used as the unique identifier when executing a FirmwareUpdate.
IdentifyAllowed	bool {get;}	Is Identify currently allowed and permitted on this device?
IdentifySupported	bool {get;}	Does this device support Identify?
Initialized	bool {get;}	True: The device is in the initialized state.
HardwareNumber	short {get;}	Hardware version or "F-Stand" for the device. (Functional State)
IP	uint {get;}	The IP Address of the device, represented as an unsigned integer. The encoded IP Address uses one byte to represent each decimal value in the IP Address. For example, the encoded value 0xC0A80001 is equivalent to the more common string representation of "192.168.0.1" NOTE: The SIMATIC Automation Tool supports only IPv4 addresses. Ipv6 addressing is not supported.
IPString	string {get;}	The IP Address of the device, represented as a string in the form "xx.xx.xx.xx" (i.e. 192.168.0.1)
MAC	ulong {get;}	The unique MAC address assigned to the device. The encoded MAC address uses one byte to encode each of the 6 octets defined for the address. For example, the encoded MAC address 0x112233445566 is equivalent to the more common string representation of 11:22:33:44:55:66.

Property name	Return type	Description
MACstring	string {get;}	The unique MAC address assigned to the device, represented as a string in the form 11:22:33:44:55:66
Modules	IModuleCollection {get;}	A collection of the modules (Page 228) plugged on the station.
Name	string {get;}	Name of the device
NewFirmwareFile	string {get;}	Location of the firmware file to be used in firmware update
NewFirmwareNameErrorCode	Result {get;}	ErrorCode attached to new Firmware name
NewFirmwareNameIsValid	bool {get;}	Is the set Firmware file valid?
NewFirmwareVersion	string {get;}	This property is used in the SIMATIC Automation Tool user interface. It is not relevant for API operations.
NewDefaultGateway	string {get;}	This property is used in the SIMATIC Automation Tool user interface. It is not relevant for API operations.
NewIP	string {get;set;}	This property is used in the SIMATIC Automation Tool user interface. It is not relevant for API operations.
NewProfinetName	string{get;set;}	This property is used in the SIMATIC Automation Tool user interface. It is not relevant for API operations.
NewProgramName	string {get;set;}	This property is used in the SIMATIC Automation Tool user interface. It is not relevant for API operations.
NewRestoreName	string{get;set;}	This property is used in the SIMATIC Automation Tool user interface. It is not relevant for API operations.
NewSubnetMask	string {get;set;}	This property is used in the SIMATIC Automation Tool user interface. It is not relevant for API operations.
NotAccessiblewithDCP	string{get;}	True: DCP access is currently not possible with this device. It is behind a router
ProfinetConvertedName	string {get;}	Converted PROFINET name for the device
ProfinetName	string {get;}	PROFINET name for the device
ResetCommunicationsParametersAllowed	bool {get;}	Is "reset communications parameters" currently allowed and permitted on this device?

Property name	Return type	Description
`ResetCommunicationsParametersSupported`	`bool {get;}`	Does this device support "reset communications parameters"?
`ResetToFactoryAllowed`	`bool {get;}`	Is ResetToFactory allowed on the device?
`ResetToFactorySupported`	`bool {get;}`	Does this device support ResetToFactory?
`RouterIP`	`uint {get;}`	The IP Address of the router, if used, represented as an unsigned integer. The encoded IP Address uses one byte to represent each decimal value in the IP Address. For example, the encoded value 0xC0A80001 is equivalent to the more common string representation of “192.168.0.1” If no router is used, this property is 0. NOTE: SIMATIC Automation Tool supports only IPv4 addresses.
`RouterIPString`	`string {get;}`	The IP Address of the device, represented as a string in the form “xx.xx.xx.xx” (for example, “192.168.0.1”)
`Selected`	`bool {get;set;}`	Marks the device as selected to enable operations to be performed
`SerialNumber`	`string {get;}`	Unique serial number for the device
`SetIPAllowed`	`bool {get;}`	Is "Set IP Address" currently enabled and permitted on this device?
`SetIPSupported`	`bool {get;}`	Does this device support "Set IP address"?
`SetProfinetNameAllowed`	`bool {get;}`	Is "Set PROFINET Name" currently enabled and permitted on this device?
`SetProfinetNameSupported`	`bool {get;}`	Does this device support "Set PROFINET Name"?
`Slot`	`uint {get;}`	Slot number for the hardware item
`SlotName`	`string {get;}`	This property is used in the SIMATIC Automation Tool user interface. It is not relevant for API operations.
`StationNumber`	`uint {get;}`	Station number of the device
`SubSlot`	`uint {get;}`	The subslot of the device. This is relevant for pluggable submodules such as S7-1200 SB modules.

Property name	Return type	Description
Supported	bool {get;}	True when the article number exists in the database and the device is supported by current SIMATIC Automation Tool API.
SubnetMask	uint {get;}	The subnet mask of the device, represented as an unsigned integer. The encoded subnet mask uses one byte to represent each decimal value in the address. For example, the encoded value 0xFFFFFF00 is equivalent to the more common string representation of 255.255.255.0.
SubnetMaskString	string {get;}	The subnet mask of the device, represented as a string in the form "xx.xx.xx.xx" (i.e. 192.168.0.1)

12.10.2 IProfinetDevice methods

12.10.2.1 FirmwareUpdate method

You use the IHardware method SetFirmwareFile (Page 179)and the IProfinetDevice method FirmwareUpdate to update the firmware of the following hardware through a CPU interface:

- CPU
- Local I/O Modules
- Distributed I/O
- Displays, Signal Boards, Signal Modules, Communication Modules, and others

Return type	Method name
Result	FirmwareUpdate

Parameters			
Name	**Data type**	**Parameter type**	**Description**
hardwareID	uint	In	Hardware identifier of the module
bUpdateSameVersion	bool	In	If true, the method will proceed with the update. The update proceeds if the update file is the same version as the current firmware version of the module.
type	FirmwareUpdateType (Page 301)	In	Optional input parameter that specifies the type of firmware update operation to perform

Note

Classic and Plus firmware update files

There are two different types of firmware update files:

- Classic firmware update folders contain several files that make up the firmware update. The header.upd or cpu_hd.upd in this folder is the file that is passed to the FirmwareUpdate method.
- The Plus firmware update file is a single update file. This is the file that is passed to the FirmwareUpdate method.

Restrictions

The API only supports the CPU network interface for a firmware update. You cannot update firmware through a CM or CP interface.

Precautions

A firmware update puts the CPU in STOP mode.

Firmware update of a CPU directly connected to the network interface

To update the firmware of a CPU, the application must perform these steps:

1. Set the Selected (Page 203) property of the CPU.
2. Call the SetFirmwareFile (Page 179) method with the appropriate UPD file for this CPU.
3. Call the FirmwareUpdate method on the CPU with the ID of the CPU.
4. Clear the Selected (Page 203) property of the CPU.

Firmware update of a directly connected CPU's local modules, displays, signal boards, signal modules

To update the firmware of a directly connected CPU's local module, the application must perform these steps:

1. Set the `Selected` (Page 203) property of the local module.
2. Call the `SetFirmwareFile` (Page 179) method with the appropriate UPD file for the local module.
3. Call the `FirmwareUpdate` method on the CPU with the ID of the local module.
4. Clear the `Selected` (Page 203) property of the local module.

Note

Updating firmware for PROFINET devices that are directly connected to the network interface

If a PROFINET device that is not a CPU is directly connected to the network interface, you can follow the same procedures and code examples to update the firmware of those devices.

Accessing distributed I/O devices

You can also update the firmware of devices that are configured as distributed I/O on a PROFINET network. You must configure the distributed I/O devices in the TIA Portal's Device Configuration. You must download the configuration to the CPU for the CPU to forward the firmware update to the appropriate device.

Restrictions

The API only supports firmware update over PROFINET. The API does not support the network options PROFIBUS and AS-i distributed I/O.

Firmware update of ET200 IM (distributed I/O) through a CPU interface

To update the firmware, the application must perform these steps:

1. Set the `Selected` (Page 203) property of the PROFINET device
2. Call the `SetFirmwareFile` (Page 179) method with the appropriate UPD file for this PROFINET device.
3. Call the `FirmwareUpdate` method on the CPU with the ID of the PROFINET device.
4. Clear the `Selected` (Page 203) property of the PROFINET device.

Firmware update of ET200 IM local modules (distributed I/O) through a CPU interface

To update the firmware, the application must perform these steps:

1. Set the `Selected` (Page 203) property of the PROFINET device local module.
2. Call the `SetFirmwareFile` (Page 179) method with the appropriate UPD file for this PROFINET device's local module.
3. Call the `FirmwareUpdate` method on the CPU with the ID of the PROFINET device's local module.
4. Clear the `Selected` (Page 203) property of the PROFINET device local module.

Example: Firmware update

```
//------------------------------------------------------
// Insert the necessary code from Getting started with the API (Page 149)
// here if you want to compile the example
//------------------------------------------------------
#region Firmware Update

IProfinetDevice myDevice =
scannedDevices.FindDeviceByIP(0xC0A80001); // 192.168.0.1
if (myDevice != null)
{
   //------------------------------------------
   // Updating CPU or PROFINET Device Firmware
   //------------------------------------------
   myDevice.Selected = true;
   retVal = myDevice.SetFirmwareFile("C:\\DeviceFirmwareFile.upd");
   retVal = myDevice.FirmwareUpdate(myDevice.ID, false);
   myDevice.Selected = false;

   //----------------------------------------
   // Updating CPU Local Modules or PROFINET
   // Device Local Modules Firmware
   //----------------------------------------
   foreach (IModule module in myDevice.Modules)
   {
      module.Selected = true;
      retVal = module.SetFirmwareFile("C:\
\LocalModuleFirmwareFile.upd");
      retVal = myDevice.FirmwareUpdate(module.ID, false);
      module.Selected = false;
   }

   //----------------------------------------------
   // Updating Distributed I/O configured in a CPU
   //----------------------------------------------
   ICPU myCPU = myDevice as ICPU;
   if (myCPU != null)
   {
      foreach (IRemoteInterface interFace in myCPU.RemoteInterfaces)
      {
         foreach (IBaseDevice remoteDevice in interFace.Devices)
```

```
            {
               //-------------------------------
               // Updating a Distributed I/O IM
               //-------------------------------
               remoteDevice.Selected = true;
               retVal = remoteDevice.SetFirmwareFile("C:\
\RemoteDeviceFirmwareFile.upd");
               retVal = myCPU.FirmwareUpdate(remoteDevice.ID, false);
               remoteDevice.Selected = false;

               //---------------------------------------------
               // Updating a Distributed I/O IM Local Modules
               //---------------------------------------------
               foreach (IModule module in remoteDevice.Modules)
               {
                  module.Selected = true;
                  retVal = module.SetFirmwareFile("C:\
\RemoteModuleFirmwareFile.upd");
                  retVal = myCPU.FirmwareUpdate(module.ID, false);
                  module.Selected = false;
               }
            }
         }
      }
   }

/* For simplicity the code examples above do not check for */
/* errors. Checking for and processing errors returned     */
/* from methods is vital to a program's overall quality    */

#endregion
```

12.10.2.2 FirmwareActivate method

The `FirmwareActivate` method activates the downloaded firmware update file for the specified hardware item (`hardwareID`) on the device. The `hardwareID` can specify either the device itself or a module on the same rack.

Return type	Method name
`Result`	`FirmwareActivate`

Parameters			
Name	**Data type**	**Parameter type**	**Description**
`hardwareID`	`uint32`	In	Hardware identifier of the module

Types of firmware update

The API supports the following types of firmware update:

- Typical firmware update that includes file download and activation
- Two step firmware update without activation. The file download occurs in RUN mode.
- Two step firmware update without activation. The file download occurs in STOP mode.

If a device is not a CPU, it does not have an operating mode (RUN/STOP). The second and third types are the same.

Firmware update with download and activation

The typical firmware update downloads the firmware update file to the CPU or device and immediately activates the new firmware. Before downloading the firmware update file, the `FirmwareUpdate` (Page 208) method puts all CPUs in STOP mode. Refer to the `FirmwareUpdate` (Page 208) topic for a description and example code.

Two step firmware update with download in RUN mode and no activation

You can choose to download the firmware to the device and not immediately activate it. You can choose to activate later at a time of your choosing by calling the method `FirmwareActivate`. Two Step refers to the download step + the activation step. You can choose to download the firmware update files to all of your devices that support this feature, while they are controlling your process and are subsequently in RUN mode. Then after all devices have accepted the firmware update file, you can choose to activate all the devices at the same time. The process of activation puts the CPUs in STOP mode

Two step firmware update with download in STOP mode and no activation

You can choose to download the firmware to the device and not immediately activate it. You can choose to activate later at a time of your choosing by calling the method `FirmwareActivate`. Two step refers to the download step + the activation step. You can choose to download the firmware update files to all of your devices that support this feature by placing them in STOP mode prior to the download. Then after all devices have accepted the firmware update file, you can choose to activate all the devices are the same time

Restrictions

The API only supports the CPU network interface for a firmware update. You cannot update firmware through a CM or CP interface.

Precautions

Activating firmware puts the CPU in STOP mode if it is not already in STOP mode.

Example: Two step firmware activation

```
//-------------------------------------------------------
// Insert the necessary code from Getting started with the API (Page 149)
// here if you want to compile the example
//-------------------------------------------------------
#region Two Step Firmware Update

IProfinetDevice myDevice = scannedDevices.FindDeviceByIP(0xC0A80001)
; // 192.168.0.1
if (myDevice != null)
{
   //------------------------------------------
   // Updating CPU or PROFINET Device Firmware
   //------------------------------------------
   myDevice.Selected = true;
   retVal = myDevice.SetFirmwareFile("C:\\DeviceFirmwareFile.upd");

   // STEP 1
   retVal = myDevice.FirmwareUpdate(myDevice.ID, false, FirmwareUpda
teType.DownloadWithoutActivation);

   // STEP 2 (This can be done later, but is shown here for simplicity)
   retVal = myDevice.FirmwareActivate(myDevice.ID);
   myDevice.Selected = false;

   //-----------------------------------------------
   // Updating CPU Local Modules or PROFINET Device
   // Local Modules Firmware
   //-----------------------------------------------
   foreach (IModule module in myDevice.Modules)
   {
      module.Selected = true;
      retVal = module.SetFirmwareFile("C:\
\LocalModuleFirmwareFile.upd");

      // STEP 1
      retVal = myDevice.FirmwareUpdate(module.ID, false, FirmwareUpd
ateType.DownloadWithoutActivation);

      // STEP 2 (This can be done later but is shown here for simpli
city)
      retVal = myDevice.FirmwareActivate(module.ID);
      module.Selected = false;
   }

   //----------------------------------------------
   // Updating Distributed I/O configured in a CPU
   //----------------------------------------------
   ICPU myCPU = myDevice as ICPU;
   if (myCPU != null)
   {
      foreach (IRemoteInterface interFace in myCPU.RemoteInterfaces)
      {
         foreach (IBaseDevice remoteDevice in interFace.Devices)
         {
            //--------------------------------
            // Updating a Distributed I/O IM
            //--------------------------------
            remoteDevice.Selected = true;
            retVal = remoteDevice.SetFirmwareFile("C:\
\RemoteDeviceFirmwareFile.upd");

            // STEP 1
            retVal = myCPU.FirmwareUpdate(remoteDevice.ID, false, Fi
rmwareUpdateType.DownloadWithoutActivation);

            // STEP 2 (This can be done later but is shown here for
simplicity)
            retVal = myCPU.FirmwareActivate(remoteDevice.ID);
            remoteDevice.Selected = false;

            //----------------------------------------------
            // Updating a Distributed I/O IM Local Modules
            //----------------------------------------------
            foreach (IModule module in remoteDevice.Modules)
            {
               module.Selected = true;
               retVal = module.SetFirmwareFile("C:\
\RemoteModuleFirmwareFile.upd");

               // STEP 1
               retVal = myCPU.FirmwareUpdate(module.ID, false, Firmw
areUpdateType.DownloadWithoutActivation);

               // STEP 2 (This can be done later but is shown here f
or simplicity)
               retVal = myCPU.FirmwareActivate(module.ID);
               module.Selected = false;
            }
         }
      }
   }
}

/* For simplicity the code examples above do not check for */
/* errors. Checking for and processing errors returned     */
/* from methods is vital to a program's overall quality    */

#endregion
```

12.10.2.3 Identify method

The `Identify` method flashes a device LED or HMI screen for a specific network device. The flashing light helps identify the physical location of the device. You cannot use the `Identify` method to identify devices behind routers.

Return type	Method name
`Result`	`Identify`

Example: Identifying a device by flashing

```
//-------------------------------------------------------------
// Insert the necessary code from Getting started with the API (Page 149)
// here if you want to compile the example
//-------------------------------------------------------------

#region Identifying a device by flashing LEDs or screen
IProfinetDevice myDevice = scannedDevices.FindDeviceByIP(0xC0A80001)
as IProfinetDevice;
// 192.168.0.1
if (myDevice != null)
{
   retVal = myDevice.Identify();
}

/* For simplicity the code examples above do not check for */
/* errors. Checking for and processing errors returned     */
/* from methods is vital to a program's overall quality    */

#endregion
```

12.10.2.4 IsFirmwareUpdateAllowed method

The `IsFirmwareUpdateAllowed` method determines whether a firmware update can be performed on the device based on the parent and child device capabilities.

Return type	Method name
bool	IsFirmwareUpdateAllowed

Parameters			
Name	Data type	Parameter type	Description
hardwareID	uint	In	Hardware identifier of the module

12.10.2.5 IsFirmwareUpdate2StepAllowed method

The `IsFirmwareUpdate2StepAllowed` method determines whether a two-step firmware update can be performed on the device based on the parent and child device capabilities.

Return type	Method name
bool	IsFirmwareUpdate2StepAllowed

Parameters			
Name	Data type	Parameter type	Description
hardwareID	uint	In	Hardware identifier of the module

12.10.2.6 IsIPAddressOnNetwork method

The `IsIPAddressOnNetwork` method determines whether the IP address uniquely exists in the `IProfinetDeviceCollection` (Page 191).

Return type	Method name
`bool`	`IsIPAddressOnNetwork`

Parameters			
Name	**Data type**	**Parameter type**	**Description**
`nIP`	`uint`	In	IP address to check

Example: Checking whether an IP address is on the collection

```
//--------------------------------------------------------------
// Insert the necessary code from Getting started with the API (Page 149)
// here if you want to compile the example
//--------------------------------------------------------------

#region Checking whether an IP address is in the collection
//-----------------------------------
// Find a device by its MAC address
//-----------------------------------
IProfinetDevice dev =
scannedDevices.FindDeviceByMAC(0x112233445566);
// MAC Address 11:22:33:44:55:66
//-----------------------------------
// Check whether the IP address for this
// device is in the collection
//-----------------------------------

if (dev != null)

{
   if (dev.IsIPAddressOnNetwork(dev.IP)
   {

      //------------------------------------------------------------
      // A device at the specified IP address is in the collection.
      // Continue operations.
      //------------------------------------------------------------
   }
}
/* For simplicity the code examples above do not check for */
/* errors. Checking for and processing errors returned     */
/* from methods is vital to a program's overall quality    */

#endregion
```

12.10.2.7 IsPROFINETNameOnNetwork method

The `IsProfinetNameOnNetwork` method determines whether a PROFINET name uniquely exists in the `IProfinetDeviceCollection` (Page 191).

Return type	Method name
`bool`	`IsProfinetNameOnNetwork`

Parameters			
Name	**Data type**	**Parameter type**	**Description**
`strName`	`string`	In	PROFINET name to check

Example: Checking whether a PROFINET name is in the collection

```
//---------------------------------------------------------------
// Insert the necessary code from Getting started with the API (Page 149)
// here if you want to compile the example
//---------------------------------------------------------------

#region Checking whether a PROFINET name is in the collection
//---------------------------------
// Find a device by its MAC address
//---------------------------------
IProfinetDevice dev =
scannedDevices.FindDeviceByMAC(0x112233445566);
// MAC Address 11:22:33:44:55:66
//---------------------------------------------------------------

// Check the PROFINET name for the device is in the collection

//---------------------------------------------------------------

if (dev != null)

{

   if (dev.IsProfinetNameOnNetwork(dev.ProfinetName))

   {

      //------------------------------------------------------
      // A device with the Profinet name is in the collection.
      // Continue operations.
      //------------------------------------------------------
    }

}
/* For simplicity the code examples above do not check for */
/* errors. Checking for and processing errors returned     */
/* from methods is vital to a program's overall quality    */

#endregion
```

12.10.2.8 IsUploadServiceDataAllowed method

Use the `IsUploadServiceDataAllowed` method to determine whether you can upload the service data from a device based on the parent and child device capabilities.

Return type	Method name
`bool`	`IsUploadServiceDataAllowed`

Parameters			
Name	**Data type**	**Parameter type**	**Description**
`hardwareID`	`uint`	In	Hardware identifier of the module

12.10.2.9 RefreshStatus method

When the the `ScanNetworkDevice` (Page 186)`s` method creates an `IProfinetDeviceCollection` (Page 191), the scan returns only a minimal amount of information about each device. To get all the available information for the device, call the `RefreshStatus` method. This method makes a connection to the device, queries for various information, and then disconnects from the device.

Return type	Method name
`Result`	`RefreshStatus`

Using RefreshStatus

Objects in the `IProfinetDeviceCollection` (Page 191) that represent devices have only partial data from each device after a network scan. To obtain all the data about a device and to use the API properly, you must do the following:

1. Call `SetPassword` (Page 258) for each protected CPU. The CPU password must provide sufficient privileges to read all the device data.
2. Call the `RefreshStatus` method for each device in the `IProfinetDeviceCollection` (Page 191).

The `RefreshStatus` method updates all the data that represents the status of the device.

Critical error exceptions

If the API throws a critical error exception, then you are not using the API correctly for the status of the device. Remember to call `RefreshStatus` to keep the data about a device up to date before using additional API functions.

Example: Refreshing status following a network scan

Refer to Getting started with the API (Page 149) for an example of the RefreshStatus method in the context of a network scan with protected CPUs.

12.10.2.10 ResetCommunicationParameters method

Use the `ResetCommunicationParameters` method to reset communication parameters of a PROFINET device to its factory settings. This sets the following parameters:

- NameOfStation to "" empty string
- IP suite parameter to 0.0.0.0
- DHCP parameters (if available) to factory values
- All P Dev parameters (PD IR Data, PD Port Data Adjust, PD Interface MRP Data Adjust ...) to factory values
- Parameters adjusted by SMNP, like sysContact, sysName, and sysLocation from MIB-II to factory values

Return type	Method name
`Result`	`ResetCommunicationParameters`

Note

You cannot use this method to reset a CPU, unless you configured it as an I-Device. The `ICPU interface` (Page 230) provides a `ResetToFactoryDefaults` method (Page 251) for resetting CPUs. You also cannot reset communication parameters for devices behind routers.

Example: Resetting communication parameters for a device

```
//---------------------------------------------------------------
// Insert the necessary code from Getting started with the API (Page 149)
// here if you want to compile the example
//---------------------------------------------------------------

#region Resetting communication parameters for a device

    //-------------------------------------------------
    // Search for the device at 192.168.0.1 and reset
    //-------------------------------------------------
    IProfinetDevice dev = scannedDevices.FindDeviceByIP(0xC0A80001);
    // 192.168.0.1
    if (dev != null)
    {
         retVal = dev.ResetCommunicationParameters();
    }

/* For simplicity the code examples above do not check for */
/* errors. Checking for and processing errors returned     */
/* from methods is vital to a program's overall quality    */

#endregion
```

12.10.2.11 SetIP method

Use the `SetIP` method to set or modify the IP address of a device.

For this operation to be successful, the device port configuration in the STEP 7 project must be set to "IP address is set directly on the device". You cannot use the SetIP method to set the IP address of a device behind a router.

Return type	Method name
Result	SetIP

Parameters			
Name	**Data type**	**Parameter type**	**Description**
nIP	uint	In	New encoded IP address
nSubnet	uint	In	New encoded subnet address
nGateway	uint	In	New encoded gateway address

Example: Setting an IP address

```
//---------------------------------------------------------------
// Insert the necessary code from Getting started with the API (Page 149)
// here if you want to compile the example
//---------------------------------------------------------------

#region Setting the IP address for a device
//-------------------------------
// Search for the device at a MAC
// address and set its IP address
//-------------------------------
IProfinetDevice dev =
scannedDevices.FindDeviceByMAC(0x112233445566);
// MAC Address 11:22:33:44:55:66
if (dev != null)
{
    retVal = dev.SetIP(0xC0A80001, 0xFFFFFF00, 0x0);
    // 192.168.0.1
}
/* For simplicity the code examples above do not check for */
/* errors. Checking for and processing errors returned     */
/* from methods is vital to a program's overall quality    */

#endregion
```

Converting addresses from string format to encoded uint format

The SetIP method expects the addresses to be in encoded format as shown above. You can convert the addresses from string format to encoded uint format using the following C# code:

```
string userEnteredAddress = @"192.168.0.1"; // For example

//------------------------------
// Convert string address to uint
//------------------------------
```

```
System.Net.IPAddress ip =
System.Net.IPAddress.Parse(userEnteredAddress);
byte[] bytes = ip.GetAddressBytes();
Array.Reverse(bytes);

uint encodedIp = BitConverter.ToUInt32(bytes, 0); // encoded IP
address available for use
```

12.10.2.12 SetProfinetName method

Use the `SetProfinetName` method to set or modify the PROFINET station name for the device.

For this operation to be successful, the device port configuration in the STEP 7 project must be set to "PROFINET name is set directly on the device". You cannot use the `SetProfinetName` method to set the PROFINET name of a device behind a router.

Return type	Method name
Result	SetProfinetName

Parameters			
Name	**Data type**	**Parameter type**	**Description**
strName	String	In	New name for the PROFINET station

Example: Setting the PROFINET name

```
//---------------------------------------------------------------
// Insert the necessary code from Getting started with the API (Page 149)
// here if you want to compile the example
//---------------------------------------------------------------

#region Setting the PROFINET name of a device
//--------------------------------
// Search for the device at a MAC
// address and set its IP address
//--------------------------------
IProfinetDevice dev =
scannedDevices.FindDeviceByMAC(0x112233445566);
// MAC Address 11:22:33:44:55:66
if (dev != null)
{
    retVal = dev.SetProfinetName("new name");
}
/* For simplicity the code examples above do not check for */
/* errors. Checking for and processing errors returned     */
/* from methods is vital to a program's overall quality    */

#endregion
```

12.10.2.13 UploadServiceData method

The `UploadServiceData` method can upload the service data from a defective CPU.

Return type	Method name
`Result`	`UploadServiceData`

Parameters			
Name	**Data type**	**Parameter type**	**Description**
`strPath`	`string`	In	A fully-qualified path to the folder containing the program card contents
`format`	`TimeFormat`	In (optional)	Format for displaying date and time. Possible values are UTC and Local. If not provided, the format is Local.
`hardwareID`	`uint`	In (optional)	Hardware ID for the device

Example: Uploading service data from a defective CPU

The following example searches the `IProfinetDeviceCollection` for a CPU at a specific IP address. It then checks the current `OperatingState` of the CPU. If the CPU is defective, then the service data is uploaded:

```
//-------------------------------------------------------------
// Insert the necessary code from Getting started with the API (Page 149)
// here if you want to compile the example
//-------------------------------------------------------------

#region Uploading service data


string strDiagFolder = @"c:\Diagnostics";
ICPU myCPU = scannedDevices.FindDeviceByIP(0xC0A80001) as ICPU;
// 192.168.0.1
if (myCPU != null)

{
   myCPU.SetPassword(new EncryptedString("ReadAccessPassword"));
   myCPU.Selected = true;
   if (myCPU.OperatingMode == OperatingState.Defective)
   {
        // Get service data with default local timestamps format
        retVal = myCPU.UploadServiceData(strDiagFolder);

        Get service data with UTC timestamps format
        retVal = myCPU.UploadServiceData(strDiagFolder,
TimeFormat.UTC);
   myCPU.Selected = false;
   }
}
```

```
/* For simplicity the code examples above do not check for */
/* errors. Checking for and processing errors returned     */
/* from methods is vital to a program's overall quality    */

#endregion
```

12.10.2.14 ValidateIPAddressSubnet method

Use the `ValidateIPAddressSubnet` method to validate that a combination of IP address and Subnet mask is compatible.

Return type	Method name
Result	ValidateIPAddressSubnet

Parameters			
Name	**Data type**	**Parameter type**	**Description**
nIP	uint	In	IP Address
nSubnetMask	uint	In	Subnet Mask

Example: Validating the IP address and subnet mask for a device

```
//---------------------------------------------------------------
// Insert the necessary code from Getting started with the API (Page 149)
// here if you want to compile the example
//---------------------------------------------------------------

#region Validating the IP address and subnet


//------------------------------------------------
// Search for the device at a specific MAC

// address and validate its IP address and subnet
//------------------------------------------------
IProfinetDevice dev =
scannedDevices.FindDeviceByMAC(0x112233445566);
// MAC Address 11:22:33:44:55:66

if (dev != null)
{
   retVal = dev.ValidateIPAddressSubnet(dev.IP, dev.SubnetMask);
}

/* For simplicity the code examples above do not check for */
/* errors. Checking for and processing errors returned     */
/* from methods is vital to a program's overall quality    */

#endregion
```

12.10.2.15 ValidatePROFINETName method

The `ValidatePROFINETName` method validates the provided PROFINET name.

Return type	Method name
`Result`	`ValidatePROFINETName`

Parameters			
Name	**Data type**	**Parameter type**	**Description**
`strName`	`string`	In	PROFINET Name to validate

Example: Validating a PROFINET name before setting it

```
//-------------------------------------------------------------
// Insert the necessary code from Getting started with the API (Page 149)
// here if you want to compile the example
//-------------------------------------------------------------

#region Validating the IP address and subnet


//--------------------------------------------
// Search for a device at a MAC address and
// validate that a given PROFINET name is valid
// before assigning it to the device
//--------------------------------------------

IProfinetDevice dev =
scannedDevices.FindDeviceByMAC(0x112233445566);
// MAC Address 11:22:33:44:55:66

if (dev != null)
{
   string name = "Valid Name";
   retVal = dev.ValidatePROFINETName(name);
   if (retVal.Succeeded)
   {
      retVal = dev.SetProfinetName(name);
   }
}

/* For simplicity the code examples above do not check for */
/* errors. Checking for and processing errors returned     */
/* from methods is vital to a program's overall quality    */

#endregion
```

12.10.3 IProfinetDevice events

12.10.3.1 DataChanged event

The `IProfinetDevice` interface supports the `DataChanged` event .

This event allows the program to monitor whether changes have occurred to a given device on the network, due to other operations through the API. For example, if the program keeps a reference to a specific `IProfinetDevice`, it can detect certain changes to the device.

In the following example, the code attaches to the `DataChanged` event for every device on the network:

```
#region Monitoring data changes
void AttachEvents(IProfinetDeviceCollection devices)
{
  foreach (IProfinetDevice dev in devices)
  {
    dev.DataChanged += new DataChangedEventHandler(Dev_DataChanged);
  }
}

void DetachEvents(IProfinetDeviceCollection devices)
{
  foreach (IProfinetDevice dev in devices)
  {
    dev.DataChanged -= new DataChangedEventHandler(Dev_DataChanged);
  }
}
void Dev_DataChanged(object sender, DataChangedEventArgs e)
{
  if (e.Type == DataChangedType.OperatingState)
  {
    // Device operating state has changed. Respond to changes here.
  }
}
#endregion
```

Now, when any actions by the API cause a device to change operating mode, the method `Dev_DataChanged` is called.

Note

The `DataChanged` event does not actively monitor the live network but monitors the properties of the `IProfinetDevice`. The state of this object must change in order to trigger the event.

The DataChangedEventArgs class

The `DataChanged` event handler receives a `DataChangedEventArgs` (Page 175) object. As shown in the example above, this class has a single property, "`Type`", of data type `DataChangedType` (Page 294).

12.10.3.2 ProgressChanged event

The `IProfinetDevice` interface supports the `ProgressChanged` event.

This event allows the program to monitor the progress of methods that take a long time. `FirmwareUpdate` is one example of such a method.

To utilize the event, attach an event handler to the event. The event handler is called when there is a change in the progress of the operation.

The following example shows how you can monitor execution progress. This example shows a method that updates the firmware for a device on the network. This operation can take noticeable time. To monitor the progress of the action, the method defines and attaches an event handler to the `ProgressChanged` event. When the firmware update is complete, the event handler is detached from the event:

```
#region Monitoring progress
void UpdateCpuAtAddress(IProfinetDeviceCollection devices,uint
targetIPAddress, string updateFile)
{
    IProfinetDevice dev = devices.FindDeviceByIP(targetIPAddress);
    if (dev != null)
    {
        dev.ProgressChanged += new

        ProgressChangedEventHandler(Dev_ProgressChanged);
        dev.SetFirmwareFile(updateFile);
        dev.FirmwareUpdate(dev.ID, true);

        dev.ProgressChanged -= new

        ProgressChangedEventHandler(Dev_ProgressChanged);
    }
}

void Dev_ProgressChanged(object sender, ProgressChangedEventArgs e)
{
    IProfinetDevice device = sender as IProfinetDevice;
    double percent = 0;
    if (device != null)
    {
        if (e.Count != 0)
        {
            string sPercent = e.Index.ToString() + " %";
        }
    }
}

#endregion
```

The ProgressChangedEventArgs class

The `ProgressChanged` event handler receives a `ProgressChangedEventArgs` (Page 175) object.

Property Name	Return Type	Description
`Action`	`ProgressAction` (Page 302)	Description of the current action
`Cancel`	`bool`	Was the action canceled?
`Count`	`int`	Total amount of data to transfer
`ID`	`uint`	Hardware ID
`Index`	`int`	Current amount of data transferred

12.11 IModuleCollection class and module properties

12.11.1 Modules property and IModuleCollection class

The `IProfinetDevice` interface provides information about any modules such as signal modules, signal boards, CMs, and CPs that are on the station. The `Modules` property returns a collection of these modules (Page 181).

The following code shows how to access this information, given an IProfinetDevice that already exists:

```
//---------------------------------------------------------------
// Insert the necessary code from Getting started with the API (Page 149)
// here if you want to compile the example
//---------------------------------------------------------------

#region Collecting data from all modules

//-------------------------------------------------------
// The Modules property returns a collection of IModule
//-------------------------------------------------------
IModuleCollection modules = scannedDevices[0].Modules;
foreach (IModule mod in modules)
{
    //---------------------------------------------------
    // Get article number for every module on the station
    //---------------------------------------------------
    string displayArticleNum = mod.ArticleNumber;
}

/* For simplicity the code examples above do not check for */
/* errors. Checking for and processing errors returned     */
/* from methods is vital to a program's overall quality    */

#endregion
```

12.11.2 IModule interface

Each module on the station is represented as an `IModule` interface. This interface provides a subset of the properties available for a device.

The `IModule` interface extends the `IHardware` interface (Page 179). The `IModule` interface provides no methods. All operations on a module must be initiated at the device.
The `IModule` interface supports the following properties:

Property Name	Return Type	Description
`ArticleNumber`	`string`	The order number for the module. This is also known as MLFB or "article number".
`Comment`	`string`	This allows the user to specify a comment for the device and is used in the SIMATIC Automation Tool user interface. The comment is not relevant for API operations.
`Configured`	`bool`	True when the device has a valid configuration
`ConfiguredVersion`	`string`	Configured version description
`Description`	`string`	A description of the hardware item, based on the article number. This is the same description that the user would see in TIA Portal and in the Device Catalog. (for example, "CPU-1215 DC/DC/DC")
`Failsafe`	`bool`	Based on its ArticleNumber, is this a failsafe device?
`FirmwareUpdateAllowed`	`bool`	Is a firmware update possible for this device?
`FirmwareVersion`	`string`	Current firmware version of the device
`HardwareNumber`	`int`	Hardware identification number
`ID`	`uint`	The unique identifier for every device and module in the station. This is used as the unique identifier when executing a FirmwareUpdate.
`Name`	`string`	Name of the device
`NewFirmwareNameErrorCode`	`Result`	`ErrorCode` attached to new Firmware name
`NewFirmwareNameIsValid`	`bool`	True when the firmware file is valid for this device or module
`NewFirmwareNameIsValid`	`bool`	When the firmware file is valid for this device or module.
`NewFirmwareVersion`	`string`	New firmware version
`NewFirmwareFile`	`string`	New firmware path and file name
`Selected`	`bool`	Is the device currently selected? This is the checkbox state in the GUI.
`SerialNumber`	`string`	Unique serial number for the device
`Slot`	`uint`	Slot number for the hardware item
`SlotName`	`string`	This property is used in the SIMATIC Automation Tool user interface. It is not relevant for API operations.
`StationNumber`	`uint`	Station number of the device
`SubSlot`	`uint`	The subslot of the device. This is relevant for pluggable submodules such as S7-1200 SB modules.

Property Name	Return Type	Description
`Supported`	`bool`	Does the API support this device?
`Visible`	`bool`	This property is used in the SIMATIC Automation Tool user interface. It is not relevant for API operations.

12.12 ICPU interface

12.12.1 Identifying CPU devices in an IProfinetDeviceCollection

The `ScanNetworkDevices` method (Page 186) generates an `IProfinetDeviceCollection` (Page 191). This collection contains an item for every accessible device on the network interface. These devices can include CPUs, HMIs, and other devices. The `IProfinetDevice` interface provides properties and methods that apply to all categories of devices.The `ICPU` interface inherits from `IProfinetDevice` and therefore supports all the `IProfinetDevice` properties and methods (Page 203).

To determine whether a given `IProfinetDevice` interface actually represents a CPU device, cast it to an `ICPU`. If this cast is successful, then the network device is a CPU, and you can use the properties/methods on the `ICPU` interface.

Note

You must set the `Selected` (Page 203) flag for any operation on the `ICPU`.

You must set `SelectedConfirmed` (Page 231) if the operation on the `ICPU` is a safety-relevant operation (Page 159).

Example: Determining whether a PROFINET device is a CPU

```
//--------------------------------------------------------------
// Insert the necessary code from Getting started with the API (Page 149)
// here if you want to compile the example
//--------------------------------------------------------------

#region Determining whether a device is a CPU
ICPU myCPU = scannedDevices.FindDeviceByIP(0xC0A80001) as ICPU;
// 192.168.0.1
if (myCPU != null)
{
   //----------------------------------------------------------
   // The device is a CPU.
   // You can use the ICPU interface to interact with it.
   //----------------------------------------------------------
}
#endregion
```

12.12.2 ICPU properties

The ICPU interface extends IProfinetDevice (Page 203) by adding the following properties. To ensure they will return the current information, your code should first call the RefreshStatus (Page 219) method.

Property name	Return type	Description
ConnectedToCPCM	bool	This CPU is connected through a CP or CM
CPUProtectionLevel	ProtectionLevel	The protection level that is set on the CPU, independent of the password
DataLogFolder	IRemoteFolder	Information about any Data Logs found on the SIMATIC Memory Card of the CPU
FileSystemFolder	ICardFolder	This property is used in the SIMATIC Automation Tool user interface. It is not relevant for API operations.
IdentityCrisis	bool	True when the identity of the device cannot be determined
Initialized	bool	True when the device or module has a valid configuration
InterfaceNumber	uint	Interface through which the device is connected
LastRefreshSuccessful	bool	True when the last call to RefreshStatus completed successfully
OperatingMode	OperatingState	Designates the current mode of the CPU. This value is read-only.
Password	EncryptedString	CPU Password used in functions performed on the device
PasswordProtectionLevel	ProtectionLevel	Protection level of a legitimized CPU password
PasswordValid	bool	Is the call to SetPassword() valid?
Protected	bool	Is the CPU currently protected? This means a password is required to access some or all features depending on access level (Page 308).
RecipeFolder	IRemoteFolder	Information about any Recipes found on the SIMATIC Memory Card of the CPU
RemoteInterfaces	List<IRemoteInterface>	A list of any remote I/O interfaces (Page 265) configured for the CPU. The usage of this property is described in a later section of this document.
SDCard	SDCardType	Gets the SDCard type of the CPU, Read, Write, Unknown, or None
SelectedConfirmed	bool	Methods that perform safety-relevant operations must set the SelectedConfirmed Flag to TRUE, when the user reselects one or more devices from a confirmation dialog for the operation and confirms the operation. SelectedConfirmed means that the operation is selected and confirmed.
TIAPVersion	String	TIA Portal version associated with a CPU project

12.12.3 ICPU flags

12.12.3.1 Program Update flags

To successfully perform safety-relevant functions on a device, more information is needed from the device. The following flags ensure that the Program Update function can be performed on a safety device correctly and securely.

Property name	Return type	Description
`HasSafetyProgram`	`bool`	Boolean value set if the device has a safety program (Page 314) present on the device. This is determined when connecting to a CPU.
`NewProgramFolder`	`string`	What is the folder location for the new program? Value is set through the `SetProgramFolder` (Page 260) method.
`NewProgramName`	`string`	What is the name of the new program?
`NewProgramNameErrorCode`	`Result`	Accessible way to find issues that may be present in validating the new program, such as if the program is invalid for the device or if the IP found in the program already exists on the network
`NewProgramNameFSignature`	`uint`	What is the FSignature of the new project? Used in the comparison process to determine if `ProgramUpdate` (Page 248) finished successfully
`NewProgramNameGateway`	`uint`	Gateway of the device in the new program
`NewProgramNameHasSafetyPassword`	`bool`	True when the method `SetProgramFolder` (Page 260) is called with a valid safety program (Page 314) that has a safety password (Page 308).
`NewProgramNameIP`	`uint`	IP address that is stored in the new program
`NewProgramNameIsSafety`	`bool`	True when the method `SetProgramFolder` (Page 260) is called with a valid a safety program
`NewProgramNameIsValid`	`bool`	True when the method `SetProgramFolder` (Page 260) is called with a valid program
`NewProgramNamePassword`	`EncryptedString`	CPU Password used to attempt a connection after `ProgramUpdate` (Page 248) has finished. Value is set through the use of `SetProgramPassword(EncryptedString)` (Page 260).
`NewProgramNamePasswordErrorCode`	`Result`	Stores the error code of the last call to `SetProgramPassword` (Page 260)
`NewProgramNamePasswordIsSafety`	`bool`	True when the method `SetProgramPassword` (Page 260) is called with a valid password and the password for the new program is the safety password (Page 308)

Property name	Return type	Description
`NewProgramNamePasswordIsValid`	`bool`	True when the method `SetProgramPassword` (Page 260) is called with a valid password
`NewProgramNamePasswordLevel`	`ProtectionLevel`	What is the protection level (Page 308) of the CPU password for the new program?
`NewProgramNamePasswordPresent`	`bool`	True when the method `SetProgramFolder` (Page 260) is called and the program is protected
`NewProgramNameSubnetMask`	`uint`	Subnet mask of the device in the new program
`ProgramUpdateSucceeded`	`bool`	True when the `ProgramUpdate` (Page 248) method succeeds. The program update can still return an error.

12.12.3.2 Restore flags

The API includes the following flags so that the Restore (Page 253) method can be performed on a safety device correctly and securely. A backup file does not contain all Program Update information because a backup file differs in content from a program file.

Property name	Return type	Description
`NewRestoreFile`	`string`	What is the file location for the new program? Value is set through the `SetBackupFile` (Page 255) method
`NewRestoreName`	`string`	What is the name of the new program?
`NewRestoreNameErrorCode`	`Result`	Accessible way to find issues that may be present in validating the new program, such as if the program is invalid or incompatible with the device
`NewRestoreNameFSignature`	`uint`	What is the FSignature of the new project? Used in the comparison process to determine if `Restore` (Page 253) finished successfully
`NewRestoreNameIsSafety`	`bool`	True when the method `SetBackupFile` (Page 255) is called and the restore file is a safety program (Page 314)
`NewRestoreNameIsValid`	`bool`	True when the method `SetBackupFile` (Page 255) is called and the restore file is valid
`NewRestoreNamePassword`	`EncryptedString`	CPU Password used to attempt connection after `Restore` (Page 253) has finished. Value is set through the use of `SetBackupFilePassword(EncryptedString)` (Page 256).
`NewRestoreNamePasswordIsSafety`	`bool`	True when the call `SetBackupFilePassword` (Page 256) contains a valid safety password (Page 308)
`NewRestoreNamePasswordIsValid`	`bool`	True when the call SetBackupFilePassword (Page 256) contains a valid password

Property name	Return type	Description
RestoreNamePasswordIsSafety	bool	True when the call SetBackupFilePassword (Page 256) contains a valid safety password (Page 308)
RestoreNamePasswordIsValid	bool	True when the call SetBackupFilePassword (Page 256) contains a valid password.
RestoreSucceeded	bool	Did the Restore (Page 253) operation succeed?

12.12.3.3 ICPU supported and allowed flags

The SIMATIC Automation Tool includes the feature flags in the ICPU interface and IHMI interface. The return type of these flags is bool.

Property name	Return type	Description
BackupAllowed	bool	TRUE if this device currently allows backups
BackupSupported	bool	TRUE if this device supports backups
ChangeModeAllowed	bool	TRUE if this device currently allows cpu run mode change
ChangeModeSupported	bool	TRUE if this device supports cpu run mode change
FormatMCSAllowed	bool	TRUE if this device currently allows MCS Format
FormatMCSupported	bool	TRUE if this device supports MCS Format
MemoryResetAllowed	bool	TRUE if this device currently allows memory reset
MemoryResetSupported	bool	TRUE if this device supports memory reset
PasswordAllowed	bool	TRUE if this device currently allows passwords
PasswordSupported	bool	TRUE if this device supports passwords
ProgramUpdateAllowed	bool	TRUE if this device currently allows program updates
ProgramUpdateSupported	bool	TRUE if this device supports program updates
RemoteDataLogsAllowed	bool	TRUE if this device currently allows data log reads
RemoteDataLogsSupported	bool	TRUE if this device supports data log reads
RemoteRecipesAllowed	bool	TRUE if this device currently allows recipe access
RemoteRecipesSupported	bool	TRUE if this device supports recipe access
RestoreAllowed	bool	TRUE if this device currently allows restore
RestoreSupported	bool	TRUE if this device supports restore
ServiceDataAllowed	bool	TRUE if this device currently allows service data reads
ServiceDataSupported	bool	TRUE if this device supports service data reads
SetTimeAllowed	bool	TRUE if this device currently allows setting system time
SetTimeSupported	bool	TRUE if this device supports setting system time

12.12.4 ICPU methods

12.12.4.1 Backup method (ICPU interface)

Use the Backup method to back up the data in a CPU. The CPU creates the backup file when you call the Backup method. The Backup method stores the resulting backup file at the location you provide in the `strFile` input parameter. This backup file is available for a `Restore` (Page 253) operation. Backup files contain the entire CPU program (hardware configuration + software). You cannot back up and restore a partial program.

Return type	Method name
`Result`	`Backup`

Parameters			
Name	**Data type**	**Parameter type**	**Description**
`strFile`	`string`	In	A fully-qualified path and filename where the backup should be stored

Restrictions

The API only supports the CPU network interface for backing up a CPU. You cannot backup a CPU through a CM or CP interface.

Backing up a CPU

To back up a CPU to a file, the application must perform these steps:

1. Select the CPU by setting the `Selected` (Page 231) property.
2. Call `Backup`.
3. Clear the `Selected` (Page 231) property.

Example: Backing up a CPU

```
//-------------------------------------------------------------
// Insert the necessary code from Getting started with the API (Page 149)
// here if you want to compile the example
//-------------------------------------------------------------

#region Backing up a CPU
ICPU myCPU = scannedDevices.FindDeviceByIP(0xC0A80001) as ICPU;
// 192.168.0.1
if (myCPU != null)
{
   myCPU.Selected = true;

   retVal = myCPU.Backup("C:\\MyBackup.s7pbkp");

   myCPU.Selected = false;
}
```

```
/* For simplicity the code examples above do not check for */
/* errors. Checking for and processing errors returned     */
/* from methods is vital to a program's overall quality    */

#endregion
```

12.12.4.2 DeleteDataLog method

The `DeleteDataLog` method is used to delete a Data Log file from a CPU's memory card.

Return type	Method name
`Result`	`DeleteDataLog`

Parameters			
Name	**Data type**	**Parameter type**	**Description**
`strFileName`	`string`	In	Filename of Data Log file to delete from a CPU memory card

Example: Deleting data logs

```
//---------------------------------------------------------------
// Insert the necessary code from Getting started with the API (Page 149)
// here if you want to compile the example
//---------------------------------------------------------------
#region Deleting data logs

ICPU myCPU = scannedDevices.FindDeviceByIP(0xC0A80001) as ICPU;
// 192.168.0.1                    if (myCPU != null)

{

   //-----------------------------------------------
   // Check that the CPU supports remote data logs
 (Page 234)   //-----------------------------------------------

   if (myCPU.RemoteDataLogsAllowed)
   {
      //-----------------------------------
      // Check that data logs are available
      // on the memory card
      //-----------------------------------
      if (myCPU.DataLogFolder.Exists)
      {
         //------------------------------
         // Search for all data log files
         //--------------------------------
         foreach (IRemoteFile datalog in myCPU.DataLogFolder.Files)
         {
            datalog.Selected = true;
            //-------------------
            // Delete the data log
            //--------------------
            retVal = myCPU.DeleteDataLog(datalog.Name);
            datalog.Selected = false;
         }
      }
   }
}
/* For simplicity the code examples above do not check for */
/* errors. Checking for and processing errors returned     */
/* from methods is vital to a program's overall quality    */
#endregion
```

See also

ICPU properties (Page 231)

12.12.4.3 DeleteRecipe method

Use the `DeleteRecipe` method to delete a recipe file from a CPU’s memory card.

Return type	Method name
`Result`	`DeleteRecipe`

Parameters			
Name	**Data type**	**Parameter type**	**Description**
`strFileName`	`string`	In	Filename of Recipe file to delete from a CPU memory card

Example: Deleting a recipe

```
//--------------------------------------------------------------
// Insert the necessary code from Getting started with the API (Page 149)
// here if you want to compile the example
//--------------------------------------------------------------
#region Deleting recipes
ICPU myCPU = scannedDevices.FindDeviceByIP(0xC0A80001) as ICPU;
// 192.168.0.1
if (myCPU != null)
{
   //--------------------------------------------
   // Check that the CPU supports remote recipes (Page 234)
   //--------------------------------------------
   if (myCPU.RemoteRecipesAllowed)
   {
      //----------------------------------------------------
      // Check that recipes are available on the memory card
      //-------------------------------------------------------
      if (myCPU.RecipeFolder.Exists)
      {
         //-------------------------
         // Search for all recipes
         //-------------------------
         foreach (IRemoteFile recipe in myCPU.RecipeFolder.Files)
         {
            recipe.Selected = true;
            //--------------------
            // Delete the recipe.
            //--------------------
            retVal = myCPU.DeleteRecipe(recipe.Name);
            recipe.Selected = false;
         }
      }
   }
}

#endregion
```

12.12.4.4 DetermineConfirmationMessage

Use the DetermineConfirmationMessage method to determine the safety action to display to the user before calling a safety-relevant operation on an F-CPU. Refer to User interface programming guidelines for safety-relevant operations (Page 161) before authoring any code that operates on an F-CPU.

Return type	Method name
ConfirmationType	DetermineConfirmationMessage

Parameters			
Name	**Data type**	**Parameter type**	**Description**
operation	FailsafeOperation	In	Operation to evaluate

Restrictions

The API contains a series of defensive coding checks. These checks are present to be sure you are using the API correctly for fail-safe CPUs. If you encounter a critical error that throws an exception, you are not using the API correctly. The code examples show a specific calling order of the API methods. Follow this order to use the API properly.

Preconditions for **ResetToFactoryDefaults** and **FormatMemoryCard**

The safety-relevant operations ResetToFactoryDefaults (Page 251) and FormatMemoryCard (Page 243) share the same preconditions before calling DetermineConfirmationMessage:

- Do NOT call DetermineConfirmationMessage for a standard CPU or a critical error will result. The DetermineConfirmationMessage method is ONLY for an F-CPU.
- Call DetermineConfirmationMessage for an F-CPU if any of these conditions are true:
  - The CPU has a safety program.
  - The CPU is protected.

  You have no need for a confirmation message otherwise.

Preconditions for **ProgramUpdate** and **Restore**

The safety-relevant operations ProgramUpdate (Page 248) and Restore (Page 253) share the same preconditions before calling DetermineConfirmationMessage:

- Do NOT call DetermineConfirmationMessage for a standard CPU or a critical error will result. The DetermineConfirmationMessage method is ONLY for an F-CPU.
- Call DetermineConfirmationMessage for an F-CPU if any of these conditions are true:
  - The CPU has a safety program.
  - The CPU is protected.
  - The program to update or the backup program to restore is a safety program.

  You have no need for a confirmation message otherwise.

Checking the return value to determine the text to display

The DetermineConfirmationMessage method accepts an input operation parameter of type FailsafeOperation (Page 301), which is one of the safety-relevant operations. DetermineConfirmationMessage, when called correctly, returns a ConfirmationType (Page 294) enum type that you can use to know what text to show for the confirmation message. The following table shows the text to display for each ConfirmationType (Page 294) enumeration:

Returned ConfirmationType	Confirmation message to display
SafetyPasswordIsBeingUsed	An operation to a standard program is about to be initiated using the safety password.
DeletingExistingSafetyProgram	An existing safety program is about to be deleted.
ReplacingExistingSafetyProgram	An existing safety program is about to be updated with another safety program.
ReplacingExistingSafetyProgramWithNonSafetyProgram	An existing safety program is about to be replaced by a standard program.
LoadingSafetyProgram	A safety program is about to be loaded for the first time.

Display the message to the user in the confirmation message box. If you want to operate on multiple F-CPUs and do not want to present a series of message boxes, you can implement a dialog with a table that displays the confirmation messages for each F-CPU. Refer to User interface programming guidelines for safety-relevant operations (Page 161) to see an example of SIMATIC Automation Tool's confirmation dialog.

Example: DetermineConfirmationMessage

```
#region Determine Confirmation Message

if (myCPU.Failsafe == false)
{
   //------------------------------------------------
   // Do not call this method if it is not necessary
   //------------------------------------------------
   return;
}

//---------------
// Fail-Safe CPU
//---------------
if ((operation == FailsafeOperation.FormatMCOperation) ||
   (operation == FailsafeOperation.ResetToFactoryOperation))
{
   if ((myCPU.HasSafetyProgram == false) && (myCPU.Protected == fals
e))
   {
   //------------------------------------------------
   // Do not call this method if it is not necessary
   //------------------------------------------------
      return;
   }
}
else if (operation == FailsafeOperation.ProgramUpdateOperation)
{
   if ((myCPU.HasSafetyProgram == false) &&
       (myCPU.NewProgramNameIsSafety == false) &&
       (myCPU.Protected == false))
   {
   //------------------------------------------------
   // Do not call this method if it is not necessary
   //------------------------------------------------
      return;
   }
}
else if (operation == FailsafeOperation.RestoreOperation)
{
   if ((myCPU.HasSafetyProgram == false) &&
         (myCPU.NewRestoreNameIsSafety == false) &&
         (myCPU.Protected == false))
   {
      //------------------------------------------------
      // Do not call this method if it is not necessary
      //------------------------------------------------
      return;
   }
}
else
{
   //------------------------------------------------
   // Do not call this method if it is not necessary
   //------------------------------------------------
   return;
}
//------------------------------------
// Obtain safety confirmation message
//------------------------------------
String messageText = "";
ConfirmationType confirmType = myCPU.DetermineConfirmationMessage(op
eration);
switch (confirmType)
{
   case ConfirmationType.SafetyPasswordIsBeingUsed:
      messageText = "An operation to a standard program is about to
be initiated using the safety password";
      break;
   case ConfirmationType.DeletingExistingSafetyProgram:
      messageText = "An existing safety program is about to be delet
ed";
      break;
   case ConfirmationType.ReplacingExistingSafetyProgram:
      messageText = "An existing safety program is about to be updat
ed with another safety program";
      break;
   case ConfirmationType.ReplacingExistingSafetyProgramWithNonSafety
Program:
      messageText = "An existing safety program is about to be repla
ced by a standard program";
      break;
   case ConfirmationType.LoadingSafetyProgram:
      messageText = "A safety program is about to be loaded for the
first time";
      break;
}
   //------------------------------------------------------------
   // Display safety confirmation message and process user input
   //------------------------------------------------------------
   System.Windows.Forms.DialogResult result = System.Windows.Forms.M
essageBox.Show(messageText, "Title", System.Windows.Forms.MessageBox
Buttons.YesNo);
   if (result == System.Windows.Forms.DialogResult.No)
      return;
   //-----------------------------------------------------
   // Set SelectedConfirmed (Page 231) if the user confirmed
   // proceeding with the safety relevant operation.
   //-------------------------------------------------------
myCPU.SelectedConfirmed = true;

/* For simplicity the code examples above do not check for */
/* errors. Checking for and processing errors returned     */
/* from methods is vital to a program's overall quality    */

#endregion
```

See also

CPU password access levels (Page 308)

12.12.4.5 DownloadRecipe method

Use the `DownloadRecipe` method to download a recipe file from the programming device to the CPU memory card.

Return type	Method name
`Result`	`DownloadRecipe`

Parameters			
Name	**Data type**	**Parameter type**	**Description**
`strFile`	`string`	In	The complete path and filename of the recipe file to download from the programming device to the CPU memory card

Example: Downloading recipes

```
//-----------------------------------------------------------------
// Insert the necessary code from Getting started with the API (Page 149)
// here if you want to compile the example
//-----------------------------------------------------------------

#region Downloading recipes
string rcpFile = @"C:\NewRecipe.csv";
ICPU myCPU = scannedDevices.FindDeviceByIP(0xC0A80001) as ICPU;
// 192.168.0.1
if (myCPU != null)
{
   IRemoteFolder recipes = myCPU.RecipeFolder;
   recipes.Selected = true;
   retVal = recipes.SetRemoteFile(rcpFile);
   retVal = myCPU.DownloadRecipe(rcpFile);
   recipes.Selected = false;
}

/* For simplicity the code examples above do not check for */
/* errors. Checking for and processing errors returned     */
/* from methods is vital to a program's overall quality    */
```

```
#endregion
```

Note

If a recipe with the same name already exists on the CPU memory card, then it is replaced.

See also

ICPU supported and allowed flags (Page 234)

12.12.4.6 FormatMemoryCard method

Use the FormatMemoryCard method to format the removable SIMATIC memory card that is plugged into a CPU.

Return type	Method name
`Result`	`FormatMemoryCard`

Restrictions

The API only supports the CPU network interface for formatting a memory card. You cannot format a memory card through a CM or CP interface.

Precautions

Formatting a memory card puts the CPU in STOP mode.

Formatting a memory card on a standard CPU

To format the memory card on a standard CPU to factory defaults, the application must perform these steps:

1. Select the CPU by setting the `Selected` (Page 203) property.
2. If the CPU is protected, call `SetPassword` (Page 258) with a password that provides write access (Page 308).
3. Call `FormatMemoryCard`.
4. Clear the `Selected` (Page 203) property.

Formatting a memory card on an F-CPU

To format the memory card on an F-CPU, the application must perform these steps:

1. Select the CPU by setting the `Selected` (Page 203) property.
2. If the CPU is protected, call `SetPassword` (Page 258) with the safety password (Page 308).
3. Determine the confirmation message (Page 238) to display and obtain user confirmation before formatting the memory card.

4. Set the `SelectedConfirmed` (Page 231) property.
5. Call `FormatMemoryCard`.
6. Clear the `Selected` (Page 203) and `SelectedConfirmed` (Page 231) properties.

Example: Formatting a memory card

```
//---------------------------------------------------------------
// Insert the necessary code from Getting started with the API (Page 149)
// here if you want to compile the example
//---------------------------------------------------------------
#region Format memory card

ICPU myCPU = scannedDevices.FindDeviceByIP(0xC0A80001) as ICPU;
// 192.168.0.1
if (myCPU != null)
{
   myCPU.Selected = true;
   if (myCPU.Failsafe == false)
   {
      //--------------
      // Standard CPU
      //--------------
      if (myCPU.Protected)
      {
         retVal = myCPU.SetPassword(new EncryptedString("WriteAccess
Password"));
      }
      retVal = myCPU.FormatMemoryCard();
      myCPU.Selected = false;
   }
   else
   {
      //---------------
      // Fail-Safe CPU
      //---------------
      if (myCPU.Protected)
      {
         retVal = myCPU.SetPassword(new EncryptedString("SafetyAcces
sPassword"));
      }

      FailsafeOperation operation = FailsafeOperation.FormatMCOperat
ion;

      //---------------------------------------------------
      // Use DetermineConfirmationMessage to obtain the
      // safety confirmation message to show the user.
      //
      // Insert code here for determining, displaying and
      // verifying the confirmation. If not confirmed, then
      // abort and do not call the safety-relevant operation
      //--------------------------------------------------------

      retVal = myCPU.FormatMemoryCard();
      //-------------------------------------------
      // Reset the confirmed flag after the safety
      // operation is complete
      //-------------------------------------------
      myCPU.Selected = false;
      myCPU.SelectedConfirmed = false;
   }
}

/* For simplicity the code examples above do not check for */
/* errors. Checking for and processing errors returned     */
/* from methods is vital to a program's overall quality  */

#endregion
```

12.12.4.7 GetCurrentDateTime method

The `GetCurrentDateTime` method gets the current date and time for the CPU.

Return type	Method name
`Result`	`GetCurrentDateTime`

Parameters			
Name	**Data type**	**Parameter type**	**Description**
`DateTime`	`System.DateTime`	Out	Current date and time returned from the CPU

Example: Getting the date and time from the CPU

```
//-------------------------------------------------------------
// Insert the necessary code from Getting started with the API (Page 149)
// here if you want to compile the example
//-------------------------------------------------------------

#region Getting CPU data and time
ICPU myCPU = scannedDevices.FindDeviceByIP(0xC0A80001) as ICPU;
// 192.168.0.1
if (myCPU != null)
{
   myCPU.SetPassword(new EncryptedString("ReadAccessPassword"));
   myCPU.Selected = true;
   DateTime curTime = new DateTime();
   retVal = myCPU.GetCurrentDateTime(out curTime);
   myCPU.Selected = false;
}
```

```
/* For simplicity the code examples above do not check for */
/* errors. Checking for and processing errors returned     */
/* from methods is vital to a program's overall quality    */

#endregion
```

12.12.4.8 GetDiagnosticsBuffer method

The `GetDiagnosticsBuffer` method reads the current diagnostic entries from the CPU. The `GetDiagnosticsBuffer` method returns a collection of `DiagnosticsItem` (Page 174) objects.

Use the `Language enum` (Page 302) parameter to get diagnostic entries in a specific language:

Return type	Method name
`Result`	`GetDiagnosticsBuffer`

Parameters			
Name	**Data type**	**Parameter type**	**Description**
`DiagnosticsItems`	`List<DiagnosticsItem>`	Out	A collection of Diagnostics Items: Each item in the collection represents an entry in the diagnostics buffer.
`Language`	`Language`	In	Requested language for the diagnostics buffer entries.

Example: Getting CPU diagnostics

```
//-------------------------------------------------------------
// Insert the necessary code from Getting started with the API (Page 149)
// here if you want to compile the example
//-------------------------------------------------------------

#region Getting CPU diagnostics
ICPU myCPU = scannedDevices.FindDeviceByIP(0xC0A80001) as ICPU;
// 192.168.0.1
List<DiagnosticsItem> aLogs = new List<DiagnosticsItem>();
if (myCPU != null)
{

   myCPU.SetPassword(new EncryptedString("ReadAccessPassword"));
   myCPU.Selected = true;
   retVal = myCPU.GetDiagnosticsBuffer(out aLogs, Language.English);
   if (retVal.Succeeded)
   {
       for (int idxLog = 0; idxLog < aLogs.Count; idxLog++)
       {
       //---------------------------
       //Get information time stamp
       //---------------------------
       string descrTimes = aLogs[idxLog].TimeStamp.ToString();
```

```
          //-----------------------------------
          //Get information basic description
          //-----------------------------------
          string descrBasic = aLogs[idxLog].Description1;

          //---------------------------------------
           //Get information detailed description
          //---------------------------------------
          string descrDetail = aLogs[idxLog].Description2;

          //------------------------------------------
          //Get information state (incoming/outgoing)
          //------------------------------------------
          string descrState = aLogs[idxLog].State.ToString();

          }
      }
      myCPU.Selected = false;
   }

   /* For simplicity the code examples above do not check for */
   /* errors. Checking for and processing errors returned     */
   /* from methods is vital to a program's overall quality    */

   #endregion
```

12.12.4.9 MemoryReset method

The `MemoryReset` method resets CPU memory.

Return type	Method name
Result	MemoryReset

Example: Resetting CPU memory

```
//-----------------------------------------------------------
// Insert the necessary code from Getting started with the API (Page 149)
// here if you want to compile the example
//-----------------------------------------------------------

#region Resetting CPU memory
ICPU myCPU = scannedDevices.FindDeviceByIP(0xC0A80001) as ICPU;
// 192.168.0.1
if (myCPU != null)
{

   if (myCPU.Protected)
   {
      myCPU.SetPassword(new EncryptedString("WriteAccessPassword"));
   }
   myCPU.Selected = true;
   retVal = myCPU.MemoryReset();
   myCPU.Selected = false;
}

/* For simplicity the code examples above do not check for */
/* errors. Checking for and processing errors returned     */
/* from methods is vital to a program's overall quality    */

#endregion
```

12.12.4.10 ProgramUpdate method

Use the ProgramUpdate method to download a new CPU program.

Return type	Method name
`Result`	`ProgramUpdate`

Restrictions

The API only supports the CPU network interface for program update. You cannot update a program through a CM or CP interface.

Precautions

Program update puts the CPU in STOP mode.

Creating a Program Update folder

Use the TIA Portal programming software to create your HMI runtime software. After your program is complete, you can use the ProgramUpdate method to download it. Within the TIA Portal, use the Card Reader function located in the TIA Portal Project tree to create a folder. This folder will contain your CPU program that the ProgramUpdate method can use. After you have created the folder, drag and drop the entire contents of the PLC into this new folder. The ProgramUpdate method only downloads entire CPU programs (hardware configuration + software). You cannot download a partial program.

Program update of a standard CPU

To update the program of a standard CPU, the application must perform these steps:

1. Select the CPU by setting the `Selected` (Page 203) property.
2. If the CPU is protected, call `SetPassword` (Page 258) with a password that provides write access (Page 308).
3. Call `SetProgramFolder` (Page 260) to select the folder that contains the new program to be downloaded to the CPU.
4. If the new program has a CPU protection level set, call `SetProgramPassword` (Page 260) to use it after the download. You must supply a write access password (Page 308) when calling `SetProgramPassword` (Page 260).

5. Call `ProgramUpdate`.
6. Clear the `Selected` (Page 203) property.

Program update of an F-CPU

To update the program of an F-CPU, the application must perform these steps:

1. Select the CPU by setting the `Selected` (Page 203) property.
2. If the CPU is protected, call `SetPassword` (Page 258) with the safety password (Page 308).
3. Call `SetProgramFolder` (Page 260) to select the folder that contains the new program to be downloaded to the CPU.
4. If the new program has a CPU protection level set, call `SetProgramPassword` (Page 260) to use it after the download. If the new program is a fail-safe program, you must supply the safety password (Page 308) when calling `SetProgramPassword` (Page 260). If the new program is not a fail-safe program, you must supply a write access password (Page 308) when calling `SetProgramPassword` (Page 260).
5. Determine the confirmation message (Page 238) to display obtain user confirmation before updating the program.
6. Set the `SelectedConfirmed` (Page 231) property.
7. Call `ProgramUpdate`.
8. Clear the `Selected` (Page 203) and `SelectedConfirmed` (Page 231) properties.

Example: Program update

```
//-------------------------------------------------------
// Insert the necessary code from Getting started with the API (Page 149)
// here if you want to compile the example
//-------------------------------------------------------
#region Program update

ICPU myCPU = scannedDevices.FindDeviceByIP(0xC0A80001) as ICPU; // 1
92.168.0.1
if (myCPU != null)
{
   myCPU.Selected = true;
   if (myCPU.Failsafe == false)
   {
      //--------------
      // Standard CPU
      //--------------
      if (myCPU.Protected)
      {
         retVal = myCPU.SetPassword(new EncryptedString("WriteAccess
Password"));
      }

      FailsafeOperation operation = FailsafeOperation.ProgramUpdateO
peration;
```

```
      retVal = myCPU.SetProgramFolder("C:\\MyProgramFolder");
      if (retVal.Failed && retVal.Error ==
ErrorCode.ProgramPasswordNeeded)
      {
         retVal = myCPU.SetProgramPassword(new
EncryptedString("WriteAccessProgramPassword"));
      }
      retVal = myCPU.ProgramUpdate();
      myCPU.Selected = false;
   }
   else
   {
      //---------------
      // Fail-Safe CPU
      //---------------
      if (myCPU.Protected)
      {
         retVal = myCPU.SetPassword(new
EncryptedString("SafetyAccessPassword"));
      }
      retVal = myCPU.SetProgramFolder("C:\\MyProgramFolder");
      if (retVal.Failed && retVal.Error ==
ErrorCode.ProgramPasswordNeeded)
      {
         retVal = myCPU.SetProgramPassword(new
EncryptedString("SafteyAccessProgramPassword"));
      }
      //-------------------------------------------------------
      // Use DetermineConfirmationMessage to obtain the
      // safety confirmation message to show the user.
      //
      // Insert code here for determining, displaying and
      // verifying the confirmation. If not confirmed, then
      // abort and do not call the safety-relevant operation
      //-------------------------------------------------------
      //-------------------------------------
      // Display safety confirmation message
      //-------------------------------------
      retVal = myCPU.ProgramUpdate();
      //-----------------------------------------------------
      // Reset the confirmed flag after the safety operation
      // is complete
      //-----------------------------------------------------
      myCPU.Selected = false;
      myCPU.SelectedConfirmed = false;
   }
}
#endregion
```

12.12.4.11 ResetToFactoryDefaults method

This method resets a CPU to its factory default values.

Return type	Method name
`Result`	`ResetToFactoryDefaults`

Restrictions

The API only supports the CPU network interface for a reset to factory defaults. You cannot reset a CPU to factory defaults through a CM or CP interface.

Precautions

Reset to factory defaults puts the CPU in STOP mode.

Resetting a standard CPU to factory defaults

To reset a standard CPU to factory defaults, the application must perform these steps:

1. Select the CPU by setting the `Selected` (Page 203) property.
2. If the CPU is protected, call `SetPassword` (Page 258) with a password that provides write access (Page 308).
3. Call `ResetToFactoryDefaults`.
4. Clear the `Selected` (Page 203) property.

Resetting an F-CPU to factory defaults

To reset an F-CPU to factory defaults, the application must perform these steps:

1. Select the CPU by setting the `Selected` (Page 203) property.
2. If the CPU is protected, call `SetPassword` (Page 258) with the safety password (Page 308).
3. Determine the confirmation message (Page 238) to display and obtain user confirmation before resetting the F-CPU to factory defaults.
4. Call `ResetToFactoryDefaults`.
5. Clear the `Selected` (Page 203) and `SelectedConfirmed` (Page 231) properties.

Example: Resetting a CPU to factory defaults

```
//---------------------------------------------------------------
// Insert the necessary code from Getting started with the API (Page 149)
// here if you want to compile the example
//---------------------------------------------------------------
#region Reset CPU to factory defaults

ICPU myCPU = scannedDevices.FindDeviceByIP(0xC0A80001) as ICPU; // 1
92.168.0.1
if (myCPU != null)
```

```
{
   myCPU.Selected = true;
   if (myCPU.Failsafe == false)
   {
      //--------------
      // Standard CPU
      //--------------
      if (myCPU.Protected)
      {
         retVal = myCPU.SetPassword(new EncryptedString("WriteAccess
Password"));
      }
      retVal = myCPU.ResetToFactoryDefaults();

      myCPU.Selected = false;
   }
   else
   {
      //---------------
      // Fail-Safe CPU
      //---------------
      if (myCPU.Protected)
      {
         retVal = myCPU.SetPassword(new EncryptedString("SafetyAcces
sPassword"));
      }

      FailsafeOperation operation = FailsafeOperation.ResetToFactory
Operation;

      //----------------------------------------------------
      // Use DetermineConfirmationMessage to obtain the
      // safety confirmation message to show the user.
      //
      // Insert code here for determining, displaying and
      // verifying the confirmation. If not confirmed, then
      // abort and do not call the safety-relevant operation
      //----------------------------------------------------

      retVal = myCPU.ResetToFactoryDefaults();

      //-----------------------------------------------------
      // Reset the confirmed flag after the safety operation
      // is complete
      //-----------------------------------------------------
      myCPU.Selected = false;
      myCPU.SelectedConfirmed = false;
   }
}

/* For simplicity the code examples above do not check for */
/* errors. Checking for and processing errors returned     */
/* from methods is vital to a program's overall quality    */
```

```
#endregion
```

12.12.4.12 Restore method (ICPU interface)

Use the Restore method to restore a backup file to the CPU.

Return type	Method name
`Result`	`Restore`

Restrictions

The API only supports the CPU network interface for restoring a backup file. You cannot restore a backup file through a CM or CP interface.

Precautions

The Restore method puts the CPU in STOP mode.

Creating a backup file

The CPU creates the backup file when you call the `Backup` (Page 235) method. The `Backup` (Page 235) method stores the resulting backup file at the location you specify. This backup file is available for a `Restore` operation. Backup files contain the entire CPU program (hardware configuration + software). You cannot back up and restore a partial program.

Restoring from a backup file of a standard CPU

To restore a standard CPU backup file, the application must perform these steps:

1. Select the CPU by setting the `Selected` (Page 203) property.
2. If the CPU is protected, call `SetPassword` (Page 258) with a password that provides write access (Page 308).
3. Call `SetBackupFile` (Page 255) to select the backup file to be downloaded to the CPU.
4. If the backup file has a CPU protection level set, then call `SetBackupFilePassword` (Page 256) to use it after download. You must supply a write access password (Page 308) when calling `SetBackupFilePassword` (Page 256).
5. Call `Restore`.
6. Clear the `Selected` (Page 203) property.

Restoring from a backup file of an F-CPU

To restore an F-CPU backup file, the application must perform these steps:

1. Select the CPU by setting the `Selected` (Page 203) property.
2. If the CPU is protected, call `SetPassword` (Page 258) with the safety password (Page 308).
3. Call `SetBackupFile` (Page 255) to select the backup file to be downloaded to the CPU.

4. If the backup file has a CPU protection level set, then call `SetBackupFilePassword` (Page 256) to use it after download. You must supply a safety password (Page 308) when calling `SetBackupFilePassword` (Page 256).
5. Determine the confirmation message (Page 238) to display and obtain user confirmation before restoring the backup file.
6. Set the `SelectedConfirmed` (Page 231) property.
7. Call `Restore`.
8. Clear the `Selected` (Page 203) and `SelectedConfirmed` (Page 231) properties.

Example: Restoring from a backup file

```
//---------------------------------------------------------------
// Insert the necessary code from Getting started with the API (Page 149)
// here if you want to compile the example
//---------------------------------------------------------------

#region Restoring a CPU backup

ICPU myCPU = scannedDevices.FindDeviceByIP(0xC0A80001) as ICPU; //
192.168.0.1
if (myCPU != null)
{
   myCPU.Selected = true;

   if (myCPU.Failsafe == false)
   {
      //--------------
      // Standard CPU
      //--------------
      if (myCPU.Protected)
      {
         retVal = myCPU.SetPassword(new EncryptedString("WriteAccess
Password"));
      }
      retVal = myCPU.SetBackupFile("C:\\MyBackup.s7pbkp");

      //-----------------------------------------------
      // Set a backup file password if the CPU program
      // in the backup file is protected
      //-----------------------------------------------
      retVal = myCPU.SetBackupFilePassword(new
EncryptedString("WriteAccessBackupFilePassword"));

      retVal = myCPU.Restore();

      myCPU.Selected = false;
   }
else
   {
      //---------------
      // Fail-Safe CPU
      //---------------
      if (myCPU.Protected)
      {
         retVal = myCPU.SetPassword(new
EncryptedString("SafetyAccessPassword"));
      }

      FailsafeOperation operation =
FailsafeOperation.RestoreOperation;

      retVal = myCPU.SetBackupFile("C:\\MyBackup.s7pbkp");

      //--------------------------------------------------
      // Set a backup file password if the CPU program in
      // the backup file is protected
      //--------------------------------------------------
      retVal = myCPU.SetBackupFilePassword(new
EncryptedString("SafteyAccessBackupFilePassword"));

      //--------------------------------------------------
      // Use DetermineConfirmationMessage to obtain the
      // safety confirmation message to show the user.
      //
      // Insert code here for determining, displaying and
      // verifying the confirmation. If not confirmed, then
      // abort and do not call the safety-relevant operation
      //--------------------------------------------------

      retVal = myCPU.Restore();

      //-----------------------------------------------------
      // Reset the confirmed flag after the safety operation
      // is complete
      //-----------------------------------------------------
      myCPU.Selected = false;
      myCPU.SelectedConfirmed = false;
   }
}

/* For simplicity the code examples above do not check for */
/* errors. Checking for and processing errors returned     */
/* from methods is vital to a program's overall quality */

#endregion
```

12.12.4.13 SetBackupFile method

The `SetBackupFile` method sets the full path and file name for a backup file to be restored with the `Restore` (Page 253) method.

The `SetBackupFile` method sets the following `ICPU` restore flags (Page 233):

- `NewRestoreName`
- `NewRestoreFile`
- `NewRestoreNameIsValid`

- `NewRestoreNameIsSafety`
- `NewRestorenameFSignature`

Return type	Method name
Result	SetBackupFile

Parameters			
Name	**Data type**	**Parameter type**	**Description**
`strFile`	`string`	In	Location for a backup file

Example: Setting the backup file

`SetBackupFile` is one part of the process of restoring a backup file to a CPU.

See the example and description in the `Restore method topic` (Page 253)for the correct and complete sequence of steps for restoring a backup file.

See also

Getting started with the API (Page 149)

12.12.4.14 SetBackupFilePassword method

When you restore a backup file, the application attempts to reconnect to the device. If the backup program that you restored to the CPU is-protected, use the `SetBackupFilePassword` method to set the updated CPU password. With the updated password, the application can regain access to the device.

The method sets the `NewRestoreNamePassword` flag on the `ICPU` object:

Return type	Method name
`Result`	`SetBackupFilePassword`

Parameters			
Name	**Data type**	**Parameter type**	**Description**
`password`	`EncryptedString`	In	Sets the password for the project that is passed to the CPU during a restore

Example: Setting the CPU password for the backup file

`SetBackupFilePassword` is one part of the process of restoring a backup file to a CPU.

See the example and description in the `Restore` method topic (Page 253)for the correct and complete sequence of steps for restoring a backup file.

12.12.4.15 SetCurrentDateTime method

The `SetCurrentDateTime` method sets the current date and time for the CPU. The configured time transformation rules are not affected by this action. Therefore, the specified `DateTime` value is based on UTC time and not the local time.

Return type	Method name
`Result`	`SetCurrentDateTime`

Parameters			
Name	**Data type**	**Parameter type**	**Description**
`time`	`System.DateTime`	In	New value for the CPU current time

Example: Setting CPU date and time

```
//------------------------------------------------------------
// Insert the necessary code from Getting started with the API (Page 149)
// here if you want to compile the example
//------------------------------------------------------------

#region Setting CPU date and time
ICPU myCPU = scannedDevices.FindDeviceByIP(0xC0A80001) as ICPU;
// 192.168.0.1
if (myCPU != null)
{
    myCPU.SetPassword(new EncryptedString("Password"));
    myCPU.Selected = true;
    retVal = myCPU.SetCurrentDateTime(DateTime.UtcNow);
    myCPU.Selected = false;
}

/* For simplicity the code examples above do not check for */
/* errors. Checking for and processing errors returned     */
/* from methods is vital to a program's overall quality    */

#endregion
```

12.12.4.16 SetOperatingState method

The `SetOperatingState` method sets the operating mode of a CPU.

Return type	Method name
`Result`	`SetOperatingState`

Some CPUs do not support this feature. Check the property C`hangeModeAllowed` to ensure that the current CPU supports this feature.

Example: Setting the operating mode

```
//---------------------------------------------------------------
// Insert the necessary code from Getting started with the API (Page 149)
// here if you want to compile the example
//---------------------------------------------------------------

#region Setting the CPU operating mode
ICPU myCPU = scannedDevices.FindDeviceByIP(0xC0A80001) as ICPU;
// 192.168.0.1
if (myCPU != null)
{
   myCPU.Selected = true;
   if (myCPU.ChangeModeAllowed)
   {
      retVal = myCPU.SetOperatingState(OperatingStateREQ.Run);
   }
   myCPU.Selected = false;
}

/* For simplicity the code examples above do not check for */
/* errors. Checking for and processing errors returned     */
/* from methods is vital to a program's overall quality    */

#endregion
```

12.12.4.17 SetPassword method

Use the `SetPassword` method to set the password for a protected CPU.

Return type	Method name
`Result`	`SetPassword`

Parameters			
Name	**Data type**	**Parameter type**	**Description**
`password`	`EncryptedString`	In	CPU password for the device

Working with protected CPUs

If a CPU is protected, the `Protected` (Page 231) property is `true` for the `ICPU` interface. You must first check whether a CPU is protected before calling the `SetPassword` method. If the CPU is NOT protected, do NOT call the `SetPassword` method. If you call the `SetPassword` method for a CPU that is not protected, the API throws a critical error exception. The critical error exception lets you know that you are using the API incorrectly.

For CPUs that are protected, your application must call the `SetPassword` method with a password that provides a sufficient access level (Page 308) for the operation. After you set a valid password, you can refresh the status of the device.

The best practice is to call `SetPassword` on all devices for which you wish to communicate immediately after scanning the network. Set the passwords on standard CPUs to the write access password (Page 308) and for F-CPUs to the safety password (Page 308) to avoid API errors.

Example: SetPassword

```
//---------------------------------------------------------------
// Insert the necessary code from Getting started with the API (Page 149)
// here if you want to compile the example
//---------------------------------------------------------------

#region Setting a password
ICPU myCPU = scannedDevices.FindDeviceByIP(0xC0A80001) as ICPU; // 1
92.168.0.1
if (myCPU != null)
{
   if (myCPU.Failsafe == false)
   {
      //--------------
      // Standard CPU
      //--------------
      if (myCPU.Protected)
      {
         retVal = myCPU.SetPassword(new EncryptedString("WriteAccess
Password"));
      }
   }
   else
   {
      //----------------
      // Fail-Safe CPU
      //----------------
      if (myCPU.Protected)
      {
         retVal = myCPU.SetPassword(new EncryptedString("SafetyAcces
sPassword"));
      }
   }
}

/* For simplicity the code examples above do not check for */
/* errors. Checking for and processing errors returned     */
/* from methods is vital to a program's overall quality    */

#endregion
```

12.12.4.18 SetProgramFolder method

Call `SetProgramFolder` to select the folder that contains the new program for the `ProgramUpdate` (Page 248) method to download to the CPU.

Return type	Method name
`Result`	`SetProgramFolder`

Parameters			
Name	**Data type**	**Parameter type**	**Description**
`strFolder`	`string`	In	Sets the folder location for the downloaded program

The `SetProgramFolder` method sets the following program update flags (Page 232) on the `ICPU` object:

- `NewProgramFolder`
- `NewProgramName`
- `NewProgramNameIP`
- `NewProgramNameSubnetMask`
- `NewProgramNameGateway`
- `NewProgramNameIsValid`

The `SetProgramFolder` method can also set the following program update flags (Page 232) on the `ICPU` object:

- `NewProgramNameIsSafety`
- `NewProgramNameHasSafetyPassword`

Example: Setting the program folder for a program update

`SetProgramFolder` is one part of the process of a CPU program update.

See the example and description in the `ProgramUpdate` method topic (Page 248)for the correct and complete sequence of steps for updating a CPU program.

12.12.4.19 SetProgramPassword method

Return type	Method name
`Result`	`SetProgramPassword`

Parameters			
Name	**Data type**	**Parameter type**	**Description**
`password`	`EncryptedString`	In	Sets the CPU password for the project that is to be passed to CPU during `ProgramUpdate` (Page 248)

The `SetProgramPassword` method sets the following program update flags (Page 232) on the `ICPU` object:

- `NewProgramNamePasswordIsValid`
- `NewProgramNamePasswordIsSafety`
- `NewProgramNamePasswordLevel`

Example: Setting the program password for a program update

`SetProgramPassword` is one part of the process of a CPU program update.

See the example and description in the `ProgramUpdate method topic` (Page 248)for the correct and complete sequence of steps for updating a CPU program.

12.12.4.20 UploadDataLog method

The `UploadDataLog` method uploads a copy of specified Data Log file from a CPU's memory card to your programming device.

Return type	Method name
`Result`	`UploadDataLog`

Parameters			
Name	**Data type**	**Parameter type**	**Description**
`strFileName`	`string`	In	Filename of the Data Log to upload from a CPU's removable SIMATIC memory card
`strDestinationFolder`	`string`	In	Fully-qualified path where the uploaded file Data Log file is stored

Example: Uploading data logs

```
//-------------------------------------------------------------
// Insert the necessary code from Getting started with the API (Page 149)
// here if you want to compile the example
//-------------------------------------------------------------
#region Uploading data logs

ICPU myCPU = scannedDevices.FindDeviceByIP(0xC0A80001) as ICPU;
// 192.168.0.1
```

```
if (myCPU != null)
{
   //------------------------------------------------------
   // Check that the CPU supports remote data logs (Page 234)
   //------------------------------------------------------
   if (myCPU.RemoteDataLogsAllowed)
   {
      //------------------------------------
      // Check that data logs are available
      // on the memory card
      //------------------------------------
      if (myCPU.DataLogFolder.Exists)
      {
         //-------------------------------
         // Search for all data log files
         //-------------------------------
         foreach (IRemoteFile datalog in myCPU.DataLogFolder.Files)
         {
            datalog.Selected = true;
            //--------------------------------
            // Upload a copy of each data log
            //--------------------------------
            retVal = myCPU.UploadDataLog(datalog.Name,
                @"C:\MyDataLogs");
            datalog.Selected = false;
         }
      }
   }
}

/* For simplicity the code examples above do not check for */
/* errors. Checking for and processing errors returned     */
/* from methods is vital to a program's overall quality    */

#endregion
```

12.12.4.21 UploadRecipe method

Use the `UploadRecipe` method to upload a copy of a recipe file from a CPU's memory card to the programming device.

Return type	Method name
Result	UploadRecipe

Parameters			
Name	**Data type**	**Parameter type**	**Description**
strFileName	string	In	Filename of the recipe to upload from the CPU memory card
strDestinationFolder	string	In	Fully-qualified path where the uploaded Recipe file is written

Example: Uploading recipes

```
//---------------------------------------------------------------
// Insert the necessary code from Getting started with the API (Page 149)
// here if you want to compile the example
//---------------------------------------------------------------
#region Uploading recipes

ICPU myCPU = scannedDevices.FindDeviceByIP(0xC0A80001) as ICPU;
// 192.168.0.1
if (myCPU != null)
{
   //--------------------------------------------------------
   // Check that the CPU supports remote recipes (Page 234)
   //--------------------------------------------------------
   if (myCPU.RemoteRecipesAllowed)
   {
      //---------------------------------
      // Check that recipes are available
      // on the memory card.
      //---------------------------------
      if (myCPU.RecipeFolder.Exists)
      {
         //-----------------------------
         // Search for all recipe files
         //-----------------------------
         foreach (IRemoteFile recipe in myCPU.RecipeFolder.Files)
         {
            recipe.Selected = true;
            //-------------------------------
            // Upload a copy of each recipe.
            //-------------------------------
           retVal = myCPU.UploadRecipe(recipe.Name, @"C:\MyRecipes");
            recipe.Selected = false;
         }
      }
   }
}

/* For simplicity the code examples above do not check for */
/* errors. Checking for and processing errors returned     */
/* from methods is vital to a program's overall quality    */

#endregion
```

12.12.5 RemoteInterfaces property

12.12.5.1 Decentralized I/O modules

Each CPU can support multiple decentralized I/O interfaces. Information about the devices attached on these remote interfaces is available through the `RemoteInterfaces` property of the ICPU interface (Page 231).

Example: Inspecting remote interfaces

```
//--------------------------------------------------------------
// Insert the necessary code from Getting started with the API (Page 149)
// here if you want to compile the example
//--------------------------------------------------------------

#region Inspect remote interfaces for a CPU
ICPU myCPU = scannedDevices.FindDeviceByIP(0xC0A80001) as ICPU;
// 192.168.0.1
if (myCPU != null)
{
   myCPU.Selected = true;

       List<IRemoteInterface> decentralNets = myCPU.RemoteInterfaces;
        foreach (IRemoteInterface net in decentralNets)
        {
            //-------------------------------
            // Inspect the remote interface
            //-------------------------------
        }

   myCPU.Selected = false;
}

/* For simplicity the code examples above do not check for */
/* errors. Checking for and processing errors returned     */
/* from methods is vital to a program's overall quality    */

#endregion
```

12.12.5.2 IRemoteInterface properties

The `IRemoteInterface` interface supports the following properties. These properties are read-only.

Property name	Return type	Description
`Devices`	`List<IBaseDevice>`	A list of any decentralized I/O stations connected to this remote interface
`InterfaceType`	`RemoteInterfaceType`	Communications protocol for this remote interface as defined in RemoteInterfaceType enumeration (Page 303)
`Name`	`string`	Configured name for the remote interface

You can use the `Devices` property to traverse a decentralized network. Each device in the decentralized network is represented by an `IBaseDevice` interface. This interface has a subset of the properties available for an `IProfinetDevice` and provides the limited functionality available for these devices in the API.

The following properties are available on the `IBaseDevice` interface:

Property name	Return type	Description
`ArticleNumber`	`string`	The order number for the module. This is also known as MLFB or article number.
`Comment`	`string`	This allows the user to specify a comment for the device. This is used in the SIMATIC Automation Tool user interface. It is not relevant for API operations.
`Configured`	`bool`	Does the device have a valid configuration?
`Description`	`string`	A description of the hardware item based on the article number. This is the same description that the user sees in TIA Portal. Example: "CPU-1215 DC/DC/DC"
`Failsafe`	`FeatureSupport`	Based on its Article number, is this a fail-safe device?
`Family`	`DeviceFamily`	What is the family of the device? For more information refer to the description of the DeviceFamily (Page 295) enum.
`FirmwareUpdateAllowed`	`bool`	Is firmware update possible for this device?
`FirmwareVersion`	`string`	Current firmware version of the device
`HardwareInFirmwareOrder`	`IHardwareCollection`	Hardware collection in firmware order
`HardwareInDisplayOrder`	`IHardwareCollection`	Hardware in displayed order

Property name	Return type	Description
`HardwareNumber`	`short`	Number identifier
`ID`	`uint`	The unique identifier for every device and module in the station. This is used as the unique identifier when executing a FirmwareUpdate.
`Modules`	`IModuleCollection`	A collection of local modules (Page 228) connected on the station.
`Name`	`string`	Name of the device
`NewFirmwareFile`	`string`	File path to the new firmware file
`NewFirmwareVersion`	`string`	This property is used in the SIMATIC Automation Tool user interface. It is not relevant for API operations.
`NewFirmwareNameIsValid`	`bool`	Is the new firmware file valid?
`Selected`	`bool`	Is the device selected?
`SerialNumber`	`string`	Unique serial number for the device
`Slot`	`uint`	Slot number for the hardware item
`SlotName`	`string`	This property is used in the SIMATIC Automation Tool user interface. It is not relevant for API operations.
`StationNumber`	`uint`	Station number of the device
`SubSlot`	`uint`	The subslot of the device. This is relevant for pluggable submodules such as SB-1200.
`Supported`	`bool`	Is the detected network device supported by current SIMATIC Automation Tool API operations?

Using the Devices property of the `IRemoteInterface`, you can inspect all the stations on the decentralized network.

The following example extends the example in the Decentralized I/O modules (Page 264) topic:

This example traverses all remote PROFINET interfaces for a CPU and creates a list of the article numbers for all decentralized stations.

Since the `IBaseDevice` also supports the `Modules` property, you can extend the example further to look at not only the decentralized stations, but also all the local modules on each station.

Example: Examining properties of stations on a remote interface

```
//---------------------------------------------------------------
// Insert the necessary code from Getting started with the API (Page 149)
// here if you want to compile the example
//---------------------------------------------------------------
```

```
#region Inspect properties of stations on remote interfaces for a CPU
ICPU myCPU = scannedDevices.FindDeviceByIP(0xC0A80001) as ICPU;
// 192.168.0.1
if (myCPU != null)
{
   myCPU.Selected = true;

   List<IRemoteInterface> decentralNets = myCPU.RemoteInterfaces;
   List<string> orderNumbers = new List<string>();
   foreach (IRemoteInterface net in decentralNets)
   {
      //------------------------------
      // Inspect the remote interface
      //------------------------------

      if (net.InterfaceType == RemoteInterfaceType.Profinet)
      {
         //------------------------------------
         // Look at each decentralized station
         //------------------------------------
         List<IBaseDevice> stations = net.Devices;

         //--------------------------------------------
         // Create list of the article numbers
         //--------------------------------------------
         foreach (IBaseDevice station in stations)
         {
             orderNumbers.Add(station.ArticleNumber);
         }
      }
   }
   myCPU.Selected = false;
}

/* For simplicity the code examples above do not check for */
/* errors. Checking for and processing errors returned     */
/* from methods is vital to a program's overall quality    */

#endregion
```

12.13 The ICPUClassic interface

12.13.1 Identifying classic CPU devices in an IProfinetDeviceCollection

The `ScanNetworkDevices` method (Page 186) generates an `IProfinetDeviceCollection` (Page 191) . This collection contains an item for every accessible device on the PROFINET network. These devices can include S7-300 and S7-400 (classic) CPUs.

The IProfinetDevice interface (Page 203) provides properties and methods that are applicable to all categories of devices. The ICPUClassic interface provides one method, GetDiagnosticsBuffer (Page 268), that is specific to classic CPU devices.

To determine whether a given IProfinetDevice interface represents a classic CPU device, cast it to an ICPUClassic. If this cast is successful, then the network device is a classic CPU, and you can use the method of the ICPUClassic interface.

Example: Determining whether a PROFINET device is a classic CPU

```
//---------------------------------------------------------------
// Insert the necessary code from Getting started with the API (Page 149)
// here if you want to compile the example
//---------------------------------------------------------------

#region Determining whether a device is a classic CPU
ICPUClassic myCPUClassic = scannedDevices.FindDeviceByIP(0xC0A80001)
as ICPUClassic;
// 192.168.0.1
if (myCPUClassic != null)
{
   //-------------------------------------------------------------
   // The device is a classic CPU.
   // You can use the ICPUClassic interface to interact with it.
   //-------------------------------------------------------------
}
#endregion
```

12.13.2 GetDiagnosticsBuffer method

The GetDiagnosticsBuffer method of the ICPUClassic interface reads the current diagnostic entries from the classic CPU. The GetDiagnosticsBuffer method returns a collection of DiagnosticsItem (Page 174) objects. The following example searches the IProfinetDeviceCollection for a classic CPU at a specific IP address. When found, it reads the diagnostic information from the classic CPU. Use the Language enum (Page 302) parameter to get diagnostic entries in a specific language.

Return type	Method name
Result	GetDiagnosticsBuffer

Parameters			
Name	**Data type**	**Parameter type**	**Description**
DiagnosticsItems	List<DiagnosticsItem>	Out	A collection of Diagnostics Items: Each item in the collection represents an entry in the diagnostic buffer.
Language	Language	In	Requested language for the diagnostics buffer entries.

```
//-------------------------------------------------------
// Insert the necessary code from Getting started with the API (Page 149)
// here if you want to compile the example
//-------------------------------------------------------

#region Getting diagnostics from a classic CPU
ICPUClassic myCPUClassic = scannedDevices.FindDeviceByIP(0xC0A80001)
as ICPUClassic;
// 192.168.0.1
List<DiagnosticsItem> aLogs = new List<DiagnosticsItem>();
if (myCPUClassic != null)
{
   myCPUClassic.Selected = true;
   retVal = myCPUClassic.GetDiagnosticsBuffer(out aLogs,
Language.English);
   if (retVal.Succeeded)
   {
       for (int idxLog = 0; idxLog < aLogs.Count; idxLog++)
       {
       //get information time stamp
       string descrTimes = aLogs[idxLog].TimeStamp.ToString();

       //get information basic description
       string descrBasic = aLogs[idxLog].Description1;

       //get information detailed description
       string descrDetail = aLogs[idxLog].Description2;

       //get information state (incoming/outgoing)
       string descrState = aLogs[idxLog].State.ToString();
       }
   }
   myCPUClassic.Selected = false;
}

/* For simplicity the code examples above do not check for */
/* errors. Checking for and processing errors returned     */
/* from methods is vital to a program's overall quality    */

#endregion
```

12.14 IHMI interface

12.14.1 IHMI interface

The `ScanNetworkDevices` method (Page 186) generates an `IProfinetDeviceCollection` (Page 191). This collection contains an item for every accessible device on the network interface. These devices can include CPUs, HMIs, and other devices. The `IProfinetDevice` interface provides properties and methods that apply to all categories of devices.The `IHMI` interface inherits from `IProfinetDevice` and therefore supports all the `IProfinetDevice` properties and methods (Page 203).

The `IHMI` interface includes properties and methods that are unique to HMI devices.

To determine whether a given `IProfinetDevice` interface represents an HMI device, cast it to an `IHMI`. If this cast is successful, then the network device is an HMI, and you can use the methods on the `IHMI` interface.

Example: Determining whether a PROFINET device is an HMI

```
//-----------------------------------------------------------
// Insert the necessary code from Getting started with the API (Page 149)
// here if you want to compile the example
//-----------------------------------------------------------

#region Determining whether a device is an HMI
IHMI myHMI = scannedDevices.FindDeviceByIP(0xC0A80001) as IHMI;
// 192.168.0.1
if (myHMI != null)
{
   //-----------------------------------------------------
   // The device is an HMI.
   // You can use the IHMI interface to interact with it.
   //-----------------------------------------------------
}
#endregion
```

12.14.2 IHMI properties and flags

12.14.2.1 IHMI properties

The `IHMI` interface consists of the following properties:

Property name	Return Type	Description
DeviceType	string	Type of HMI that the object represents
FirmwareDeviceVersion	string	Firmware version that is present on the HMI

Property name	Return Type	Description
`RuntimeDeviceVersion`	`string`	Runtime version that is present on the HMI
`TransferChannel`	`HMITransferChannel` (Page 301)	Protocol used to communicate with the HMI device. Default: PN_IE

12.14.2.2 Program update flags

You can use these flags with the IHMI interface:

Property Name	Return Type	Description
`NewProgramNameIsValid`	`bool`	True when the method `SetProgramFolder` is a called with a valid program folder. False if program is not valid.
`ProgramUpdateSucceeded`	`bool`	True when program update is successful even though an error may return from internal refresh status
`NewProgramName`	`string`	Name of the new program
`NewProgramFolder`	`string`	Folder location for the new program: Value is set through the `SetProgramFolder` method
`NewProgramNameErrorCode`	`Result`	Codes to find issues that may be present in validating the new program, such as if the program is invalid for the device or if the IP assignment in the program already exists on the network

12.14.2.3 Restore flags

You can use these flags with the IHMI interface:

Property Name	Return Type	Description
`NewRestoreNameIsValid`	`bool`	True when the method SetBackupFolder is a called with a valid backup file. False if backup file is not valid.
`RestoreSucceeded`	`bool`	True when restore is successful even though an error could have been returned from internal refresh status
`NewRestoreName`	`string`	What is the name of the new program?
`NewRestoreFile`	`string`	What is the file location for the new program? Value is set through the SetbackupFile method.
`NewRestoreNameErrorCode`	`Result`	Accessible way to find issues that may be present in validating the new program, such as if the program is invalid or incompatible with the device

12.14.2.4 Feature flags

You can use these flags with the IHMI interface:

Property Name	Return Type	Description
`BackupAllowed`	`bool`	True if the device allows backups
`BackupSupported`	`bool`	True if the device supports backups
`ProgramUpdateAllowed`	`bool`	True if the device allows program updates

Property Name	Return Type	Description
`ProgramUpdateSupported`	`bool`	True if the device supports program updates
`RestoreAllowed`	`bool`	True if the device allows restores
`RestoreSupported`	`bool`	True if the device supports restores
`ScheduleFullBackup`	`bool`	Reserved for SIMATIC Automation Tool

12.14.3 IHMI methods

12.14.3.1 Backup method (IHMI interface)

Use the `Backup` method of the `IHMI` interface to back up the data for an HMI.

Return type	Method name
`Result`	`Backup`

Parameters			
Name	**Data type**	**Parameter type**	**Description**
`strFile`	`string`	In	A fully-qualified path and filename where the backup file is stored
`type`	`BackupType`	In (optional)	When present, specifies what data to back up: • Full backup (default) • Recipes • User administration data

Backing up an HMI

To back up an HMI, the application must perform these steps:

1. Select the HMI by setting the `Selected` (Page 203) property.
2. Call `Backup` with an optional parameter for the type of backup to perform.
3. Clear the `Selected` (Page 203) property.

Example: Backing up an HMI

```
//---------------------------------------------------------------
// Insert the necessary code from Getting started with the API (Page 149)
// here if you want to compile the example
//---------------------------------------------------------------

#region Backing up an HMI
IHMI myHMI = scannedDevices.FindDeviceByIP(0xC0A80001) as IHMI;
// 192.168.0.1
if (myHMI != null)
{
    myHMI.Selected = true;
    // back up only recipes
    retVal = myHMI.Backup("C:\\MyHMIBackup.s7pbkp",
BackupType.Recipes);

    // or do full HMI backup
    retVal = myHMI.Backup("C:\\MyHMIBackup.s7pbkp");
    myHMI.Selected = false;
}

/* For simplicity the code examples above do not check for */
/* errors. Checking for and processing errors returned     */
/* from methods is vital to a program's overall quality    */

#endregion
```

12.14.3.2 FirmwareUpdate method

Use the `FirmwareUpdate` method of the `IHMI` interface to update on the HMI firmware.

Method Name	Return Type	Description
FirmwareUpdate()	Result	Perform a firmware update to the HMI.

Example: Updating HMI firmware

```
//-------------------------------------------------------------
// Insert the necessary code from Getting started with the API (Page 149)
// here if you want to compile the example
//-------------------------------------------------------------

#region Updating HMI firmware
IHMI myHMI = scannedDevices.FindDeviceByIP(0xC0A80001) as IHMI;
// 192.168.0.1
if (myHMI != null)
{
   myHMI.Selected = true;
   myHMI.SetFirmwareFile(@"c:\myFolder\Firmware\Simatic.HMI\
KTP900_V16_00_00_03.fwf");
   retVal = myHMI.FirmwareUpdate();
   myHMI.Selected = false;
}

/* For simplicity the code examples above do not check for */
/* errors. Checking for and processing errors returned     */
/* from methods is vital to a program's overall quality    */

#endregion
```

12.14.3.3 ProgramUpdate method (IHMI interface)

The `ProgramUpdate` method of the `IHMI` interface updates the HMI device's operating system and run-time software.

Return type	Method name
`Result`	`ProgramUpdate`

Creating a Program Update folder

Use the TIA Portal programming software to create your HMI runtime software. Within the TIA Portal, use the Card Reader function located in the TIA Portal Project tree to create a folder. This folder will contain your HMI operating system and runtime software that the ProgramUpdate method can use. After you have created the folder, drag and drop the entire contents of the HMI device into this new folder. The ProgramUpdate method downloads both the HMI operating system and runtime software. You cannot update a partial program.

The new program folder must contain the following files for a successful program update:
`DownloadTask.xml`
`ProjectCharacteristics.rdf`

The folder that you create in the TIA Portal typically has the following format:
`{DeviceName)\Simatic.HMI\RT_Projects\{ProjectName}.{DeviceName}`

Example folder name:
`"C:\Desktop\hmim14000100a\Simatic.HMI\RT_Projects`
`\DasBasicUndMobilePanelen.hmim14000100a[KTP700 Mobile]"`

Program update for an HMI

To update an HMI program, the application must perform these steps:

1. Select the HMI by setting the `Selected` (Page 203) property.
2. Call `SetProgramFolder` (Page 276) to select the folder that contains the new program to be downloaded to the HMI.
3. Call `ProgramUpdate`.
4. Clear the `Selected` (Page 203) property.

Example: Updating an HMI program

```
//---------------------------------------------------------------
// Insert the necessary code from Getting started with the API (Page 149)
// here if you want to compile the example
//---------------------------------------------------------------

#region Updating an HMI program
IHMI myHMI = scannedDevices.FindDeviceByIP(0xC0A80001) as IHMI;
// 192.168.0.1
if (myHMI != null)
{
    myHMI.Selected = true;
    myHMI.SetProgramFolder( @"c:\myFolder\ProgramUpdate\Simatic.HMI
\RT_Projects\Project1");
   //-------------------------------------------
   // The HMI project for program update is the
   // operating system and runtime software
   //-------------------------------------------
    retVal = myHMI.ProgramUpdate();
    myHMI.Selected = false;
}
/* For simplicity the code examples above do not check for */
/* errors. Checking for and processing errors returned     */
/* from methods is vital to a program's overall quality    */

#endregion
```

12.14.3.4 Restore method (IHMI interface)

Use the `Restore` method of the `IHMI` interface to restore HMI device data from a previous backup of the device.

Return type	Method name
`Result`	`Restore`

Restoring from an HMI backup file

To restore an HMI backup file, the application must perform these steps:

1. Select the HMI by setting the `Selected` (Page 203) property.
2. Call `SetBackupFile` (Page 276) to select the backup file to be downloaded to the HMI.
3. Call `Restore`.
4. Clear the `Selected` (Page 203) property.

Example: Restoring an HMI backup file

```
//-------------------------------------------------------------
// Insert the necessary code from Getting started with the API (Page 149)
// here if you want to compile the example
//-------------------------------------------------------------

#region Restoring an HMI backup

IHMI myHMI = scannedDevices.FindDeviceByIP(0xC0A80001) as IHMI; //
192.168.0.1
if (myHMI != null)
{
   myHMI.Selected = true;
   retVal = myHMI.SetBackupFile(@"C:\MyFolder\Backup.s7pbkp");
   retVal = myHMI.Restore();
   myHMI.Selected = false;
}
```

```
/* For simplicity the code examples above do not check for */
/* errors. Checking for and processing errors returned     */
/* from methods is vital to a program's overall quality    */

#endregion
```

12.14.3.5 SetBackupFile method

Use the `SetBackupFile` method of the `IHMI` interface to select the backup file to be restored to an HMI.

Return type	Method name
`Result`	`SetBackupFile`

Parameters			
Name	**Data type**	**Parameter type**	**Description**
`strFile`	`string`	in	Sets the folder location where the backup file source is stored

The method sets the following flags on the IHMI object:

- `NewRestoreName`
- `NewRestoreFile`
- `NewRestoreNameIsValid`

Example: Setting an HMI backup file

See the example in the `Restore` method topic (Page 275) for setting a backup file to restore to an HMI.

12.14.3.6 SetProgramFolder method

Use the `SetProgramFolder` method to set the folder for the program update (Page 274) for an HMI device:

Return type	Method name
`Result`	`SetProgramFolder`

Parameters			
Name	**Data type**	**Parameter type**	**Description**
`strFolder`	`string`	in	Sets the folder location where the program download source is stored

Example: Setting the program file to update

See the example in the `ProgramUpdate` method topic (Page 274) for setting a program file for updating an HMI.

12.14.3.7 SetTransferChannel method

HMI devices can support multiple protocols to exchange data with the API. The protocols are the transfer channels. The `SetTransferChannel` method sets the HMI transfer channel (Page 301):

Return type	Method name
Result	SetTransferChannel

Parameters			
Name	**Data type**	**Parameter type**	**Description**
transferChannel	HMITransferChannel	in	Sets the type of communications (Transfer Channel) for communication with the HMI device

The default transfer channel is `HMITransferChannel.PN_IE`.

Example: Setting the HMI transfer channel

```
//-------------------------------------------------------------
// Insert the necessary code from Getting started with the API (Page 149)
// here if you want to compile the example
//-------------------------------------------------------------

#region Setting the HMI transfer channel
IHMI myHMI = scannedDevices.FindDeviceByIP(0xC0A80001) as IHMI;
// 192.168.0.1
if (myHMI != null)
{
   // assign PN/IE as the protocol for communications to this HMI
device.
   retVal = myHMI.SetTransferChannel(HMITransferChannel.PN_IE);
}

/* For simplicity the code examples above do not check for */
/* errors. Checking for and processing errors returned     */
/* from methods is vital to a program's overall quality    */

#endregion
```

12.15 IScalance interface

12.15.1 IScalance interface

The `IScalance` interface includes properties and methods that are unique to SCALANCE devices.

The `ScanNetworkDevices` method (Page 186) generates an `IProfinetDeviceCollection` (Page 191). This collection contains an item for every accessible device on the network interface. These devices can include CPUs, HMIs, SCALANCE and other devices. The `IProfinetDevice` interface provides properties and methods that apply to all categories of devices.The `IScalance` interface inherits from `IProfinetDevice` and therefore supports all the `IProfinetDevice` properties and methods (Page 203).

To determine whether a device interface represents a SCALANCE device, cast the IProfinetDevice interface (Page 203) to an IScalance interface. If the cast is successful, the network device is a SCALANCE device, and you can use the IScalance methods (Page 279) and properties (Page 278). .

Example: Determining whether a PROFINET device is a SCALANCE device

```
//-------------------------------------------------------------
// Insert the necessary code from Getting started with the API (Page 149)
// here if you want to compile the example
//-------------------------------------------------------------

#region Determining whether a device is a SCALANCE device
IScalance myScalance = scannedDevices.FindDeviceByIP(0xC0A80001) as
IScalance;
// 192.168.0.1
if (myScalance != null)
{
   //----------------------------------------------------------
   // The device is a SCALANCE device.
   // You can use the IScalance interface to interact with it.
   //----------------------------------------------------------
}
#endregion
```

12.15.2 IScalance properties

The IScalance interface has the following properties:

Property name	Return type	Description
`ProfileName`	`string`	Returns the name of the SNMP profile
`ProfileNameisValid`	`bool`	TRUE if a valid profile name exists

12.15.3 IScalance methods

12.15.3.1 SetProfile method

Call the `SetProfile` method to establish the profile configuration for a SCALANCE device.

Return type	Method name
`Result`	`SetProfile`

Parameters			
Name	**Data type**	**Parameter type**	**Description**
`Profile`	`ISNMPProflie`	In	SNMP profile and TFTP (Trivial File Transfer Protocol) configuration for the SCALANCE device's FW update

Example: Setting an SNMP profile

You can see an example of the `SetProfile` method call in any of the following examples:

- Example: SNMP Version 1 configuration (Page 287)
- Example: SNMP Version 2 configuration (Page 288)
- Example: SNMP Version 3 configuration (Page 290)

In the examples, the `SetProfile` call is near the end of the example code.

12.15.3.2 FirmwareUpdate method

Call the `FirmwareUpdate` method to update the firmware of a SCALANCE device.

Return type	Method name
`Result`	`FirmwareUpdate`

Parameters			
Name	**Data type**	**Parameter type**	**Description**
`type`	`FirmwareUpdateType` (Page 301)	In	Optional input parameter that specifies the type of firmware update operation to perform

Example: Updating the firmware of a SCALANCE device

You can see an example of the `FirmwareUpdate` method call in any of the following examples:

- Example: SNMP Version 1 configuration (Page 287)
- Example: SNMP Version 2 configuration (Page 288)
- Example: SNMP Version 3 configuration (Page 290)

The examples show both types of firmware update:

- Downloading and activating the firmware in a single `FirmwareUpdate` method call
- Downloading the firmware with `FirmwareUpdate` as a first step and activating the firmware with `FirmwareActivate` as a second step.

12.15.3.3 FirmwareActivate method

The `FirmwareActivate` method activates the downloaded firmware update file for the specified SCALANCE device at `hardwareID`.

Return type	Method name
`Result`	`FirmwareActivate`

Parameters			
Name	**Data type**	**Parameter type**	**Description**
`hardwareID`	`uint32`	In	Hardware identifier of the SCALANCE device

Example: Activating downloaded firmware on a SCALANCE device

The following examples show how to perform a `FirmwareActivate`, after a `FirmwareUpdate` (Page 279) that downloads the firmware without activating it. The three examples show each of the three SNMP version types.

- Example: SNMP Version 1 configuration (Page 287)
- Example: SNMP Version 2 configuration (Page 288)
- Example: SNMP Version 3 configuration (Page 290)

12.16 ISNMPProfile interface

12.16.1 ISNMPProfile properties

The ISNMPProfile is a collection of properties that you use to perform a SCALANCE device firmware update using SNMP. You must set up a TFTP server to update firmware of a SCALANCE device.

The properties applicable for a given device's firmware update depend on the configured SNMP protocol version on that device. The property descriptions indicate for which SNMP version (V1, V2, V3) a property is applicable. Properties without a version number in the description are applicable to all SNMP versions.

Property name	Return type	Description
ProfileName	string	Name of the SNMP profile
Version	SNMPVersion	The SNMP protocol version used
ServerIP	string	String representation of the IP address
ServerIPArray	byte[]	IP address of the TFTP server
ServerPort	UInt16	TFTP port assignment
ReadCommunity	string	V1/V2 read privilege ID string (PW)
WriteCommunity	string	V1/V2 write privilege ID string (PW)
UserName	string	V3 user name
ContextName	string	V3 management info context ID
SecurityLevel	SNMPSecurityLevel	V3 security level
AuthAlgorithm	SNMPAuthAlgorithm	V3 authentication Algorithm
AuthKey	byte[]	V3 authentication key
PrivAlgorithm	SNMPPrivAlgorithm	V3 privacy algorithm
PrivKey	byte[]	V3 privacy key

12.16.2 ISNMPProfile methods

12.16.2.1 Validate method

The Validate method evaluates the SNMP profile arguments to ensure that there is a consistent set of required firmware update parameters. The returned result identifies any inconsistencies.

Return type	Method name
Result	Validate

Parameters			
Name	Data type	Parameter type	Description
None			

You can see an example of the Validate method call in any of the following examples:

- Example: SNMP Version 1 configuration (Page 287)
- Example: SNMP Version 2 configuration (Page 288)
- Example: SNMP Version 3 configuration (Page 290)

In these examples, the Validate call occurs after setting the profile parameters.

12.16.3 SNMPProfile class

12.16.3.1 SNMPProfile class properties

The `SNMPProfile` class is for storing and editing SNMP profiles.

Property name	Return type	Description
`ProfileName`	`string`	Name of the SNMP Profile
`Version`	`SNMPVersion`	SNMP protocol version used
`ServerIP`	`string`	String representation of the IP address
`ServerIPArray`	`byte[]`	IP address of the TFTP server
`ServerPort`	`UInt16`	TFTP port assignment
`ReadCommunity`	`string`	V1/V2 read privilege ID string
`WriteCommunity`	`string`	V1/V2 write privilege ID string
`UserName`	`string`	V3 user name
`ContextName`	`string`	V3 management info context ID
`SecurityLevel`	`SNMPSecurityLevel`	V3 security level
`AuthAlgorithm`	`SNMPAuthAlgorithm`	V3 authentication Algorithm
`AuthKey`	`byte[]`	V3 authentication key
`PrivAlgorithm`	`SNMPPrivAlgorithm`	V3 privacy algorithm
`PrivKey`	`byte[]`	V3 privacy key

12.16.3.2 SNMPProfile class methods

SetProfileName

The SetProfileName method assigns the input string value to the name of the profile.

Return type	Method name
`Result`	`SetProfileName`

Parameters			
Name	**Data type**	**Parameter type**	**Description**
`strName`	`string`	in	Profile name

You can see an example of the SetProfileName method call in any of the following examples:

- Example: SNMP Version 1 configuration (Page 287)
- Example: SNMP Version 2 configuration (Page 288)
- Example: SNMP Version 3 configuration (Page 290)

In these examples, the SetProfileName call occurs near the beginning of the code example.

SetSNMPVersion

The SetSNMPVersion method sets the SNMP protocol version. The SNMP protocol is either Version 1, 2, or 3.

Return type	Method name
`Result`	`SetSNMPVersion`

Parameters			
Name	**Data type**	**Parameter type**	**Description**
`nVersion`	`SNMPVersion` (Page 305)	in	SNMP version 1,2, 3

You can see an example of the SetSNMPVersion method call in any of the following examples:

- Example: SNMP Version 1 configuration (Page 287)
- Example: SNMP Version 2 configuration (Page 288)
- Example: SNMP Version 3 configuration (Page 290)

In these examples, the SetSNMPVersion call occurs near the beginning of the code example.

SetServerIP method

The SetServerIP method sets the IP address of the TFTP server. This method is applicable to all SNMP profiles.

Return type	Method name
`Result`	`SetServerIP`

Parameters			
Name	**Data type**	**Parameter type**	**Description**
`strIP`	`string`	in	IP assignment

You can see an example of the SetServerIP method call in any of the following examples:

- Example: SNMP Version 1 configuration (Page 287)
- Example: SNMP Version 2 configuration (Page 288)
- Example: SNMP Version 3 configuration (Page 290)

In these examples, the SetServerIP call occurs near the beginning of the code example.

SetServerPort method

The SetServerPort method sets the port assignment of the TFTP server. This method is applicable to all SNMP profiles.

Return type	Method name
`Result`	`SetServerPort`

Parameters			
Name	**Data type**	**Parameter type**	**Description**
`nVersion`	`UInt16`	in	TFTP port assignment

You can see an example of the SetServerPort method call in any of the following examples:

- Example: SNMP Version 1 configuration (Page 287)
- Example: SNMP Version 2 configuration (Page 288)
- Example: SNMP Version 3 configuration (Page 290)

In these examples, the SetServerPort call occurs just after the SetServerIP call.

SetReadCommunity method

The SetReadCommunity method assigns the read community to the input string. The read community string enables a remote device to retrieve read-only information from a device. This method is applicable for an SNMP Version 1 or Version 2 profile.

Return type	Method name
`Result`	`SetReadCommunity`

Parameters			
Name	**Data type**	**Parameter type**	**Description**
`strCommunity`	`string`	in	Read community assignment

You can see an example of the SetReadCommunity method call in the following examples:

- Example: SNMP Version 1 configuration (Page 287)
- Example: SNMP Version 2 configuration (Page 288)

In these examples, the SetReadCommunity call occurs near the middle of the code example.

SetWriteCommunity method

The SetWriteCommunity method assigns the write community to the input string. The write community string permits both read operations and write operations for a device. This method is applicable for an SNMP Version 1 or Version 2 profile.

Return type	Method name
`Result`	`SetWriteCommunity`

Parameters			
Name	**Data type**	**Parameter type**	**Description**
`strCommunity`	`string`	in	Write community assignment

You can see an example of the SetWriteCommunity method call in any of the following examples:

- Example: SNMP Version 1 configuration (Page 287)
- Example: SNMP Version 2 configuration (Page 288)

In these examples, the SetWriteCommunity call occurs near the beginning of the code example.

SetUserName method

The SetUserName method sets the user name for an SNMP Version 3 profile.

Return type	Method name
`Result`	`SetUserName`

Parameters			
Name	**Data type**	**Parameter type**	**Description**
`strName`	`string`	in	SNMP V3 user name

You can see an example of the SetUserName method call near the middle of the Example: SNMP Version 3 configuration (Page 290) code example.

SetContextName method

The SetContextName method sets the context name for an SNMP Version 3 profile.

Return type	Method name
`Result`	`SetContextName`

Parameters			
Name	**Data type**	**Parameter type**	**Description**
`strName`	`string`	in	SNMP V3 management information context

You can see an example of the SetContextName method call near the middle of the Example: SNMP Version 3 configuration (Page 290) code example.

SetSecurityLevel method

The SetSecurityLevel method sets the security level for an SNMP Version 3 profile.

Return type	Method name
`Result`	`SetSecurityLevel`

Parameters			
Name	**Data type**	**Parameter type**	**Description**
level	`SNMPSecurityLevel` (Page 305)	in	SNMP V3 security level

You can see an example of the SetSecurityLevel method call near the middle of the Example: SNMP Version 3 configuration (Page 290) code example.

SetAuthAlgorithm method

The SetAuthAlgorithm method sets the authentication algorithm for an SNMP Version 3 profile. The authentication algorithm is applicable when the security level (Page 285) requires "Authentication". The authentication password is applicable when the security level requires "Privacy" as well as "Authentication".

Return type	Method name
`Result`	`SetAuthAlgorithm`

Parameters			
Name	**Data type**	**Parameter type**	**Description**
algorithm	`SNMPAuthAlgorithm` (Page 304)	in	SNMP V3 authentication algorithm
strPassword	`string`	in	SNMP V3 authentication password

You can see an example of the SetAuthAlgorithm method call near the middle of the Example: SNMP Version 3 configuration (Page 290) code example.

SetPrivAlgorithm method

The SetPrivAlgorithm method sets the privacy algorithm for an SNMP Version 3 profile. The privacy algorithm is applicable when the security level (Page 285) requires "Privacy".

Return type	Method name
`Result`	`SetPrivAlgorithm`

Parameters			
Name	**Data type**	**Parameter type**	**Description**
algorithm	`SNMPPrivAlgorithm` (Page 304)	in	SNMP V3 privacy algorithm
strPassword	`string`	in	SNMP V3 privacy password

You can see an example of the SetPrivAlgorithm method call near the middle of the Example: SNMP Version 3 configuration (Page 290) code example.

12.17 IScalance and ISNMP firmware update code examples

12.17.1 Example: SNMP Version 1 configuration

This example shows how to configure and initiate a firmware update for a SCALANCE device. In this example, the SCALANCE device uses SNMP Version 1.

The example shows both types of firmware update:

- Downloading and activating the firmware in a single `FirmwareUpdate` (Page 279) method call
- Downloading the firmware with `FirmwareUpdate` as a first step and activating the firmware with `FirmwareActivate` (Page 280) as a second step.

Example: SNMP Version 1 firmware update

```
//-------------------------------------------------------------
// Insert the necessary code from Getting started with the API (Page 149)
// here if you want to compile the example
//-------------------------------------------------------------

#region Updating firmware for a SCALANCE device, SNMP Version 1
profile

SNMPProfile profileV1 = new SNMPProfile();

// Create a profile using SNMP version 1
retVal = profileV1.SetSNMPVersion(SNMPVersion.Version1);

// Set name for this profile
retVal = profileV1.SetProfileName("Profile_1");

// Set TFTP Server IP address, required
retVal = profileV1.SetServerIP("192.168.0.1");

// Set TFTP Server port, default is 69, optional
retVal = profileV1.SetServerPort(69);

// Set SCALANCE read community, default is "public", optional
retVal = profileV1.SetReadCommunity("public");

// Set SCALANCE write community, default is "private", optional
retVal = profileV1.SetWriteCommunity("private");

// Verify profile is valid; this assures we did not miss a
parameter
retVal = profileV1.Validate();

// Update firmware for a specific SCALANCE devices

IScalance myScalance = scannedDevices.FindDeviceByIP(0xC0A80001) as
IScalance;
```

```
if (myScalance != null)
{
   myScalance.Selected = true;
   //  Set  SNMP  profile  required  to  communicate  with  this
SCALANCE  device
   retVal  =  myScalance.SetProfile(profileV1);

   //  Set  new  firmware  file  to  update  SCALANCE
   retVal  =  myScalance.SetFirmwareFile(@"C:\Firmware
\FirmwareFile.lad");

   // -------------------------------------------------------
   // Update new firmware on SCALANCE device.
   // This method call downloads and activates the firmware.
   // -------------------------------------------------------
   retVal  =  myScalance.FirmwareUpdate();

   // --------------------------------------------------
   // Alternatively, use two-step operation to update
   // new firmware on SCALANCE device.
   // This operation downloads the firmware in one step
   // and activates the firmware in a second step.
   // --------------------------------------------------
   retVal  =
myScalance.FirmwareUpdate(FirmwareUpdateType.DownloadWithoutActivati
on);

   // --------------------------------------------------------------
   // This step activates the downloaded firmware
   // on the SCALANCE device. Activation can be done later and might
   // be based on user input. For simplicity, this example activates
   // the firmware after the download.
   // --------------------------------------------------------------
   retVal  =  myScalance.FirmwareActivate();
   myScalance.Selected = false;
}

/* For simplicity the code examples above do not check for */
/* errors. Checking for and processing errors returned     */
/* from methods is vital to a program's overall quality    */

#endregion
```

12.17.2 Example: SNMP Version 2 configuration

This example shows how to configure and initiate a firmware update for a SCALANCE device. In this example, the SCALANCE device uses SNMP Version 2:

The example shows both types of firmware update:

- Downloading and activating the firmware in a single `FirmwareUpdate` (Page 279) method call
- Downloading the firmware with `FirmwareUpdate` as a first step and activating the firmware with `FirmwareActivate` (Page 280) as a second step.

Example: SNMP Version 2 firmware update

```
//------------------------------------------------------------
// Insert the necessary code from Getting started with the API (Page 149)
// here if you want to compile the example
//------------------------------------------------------------

#region Updating firmware for a SCALANCE device, SNMP Version 2
profile

SNMPProfile profileV2 = new SNMPProfile();

// Create a profile using SNMP version 2
retVal = profileV2.SetSNMPVersion(SNMPVersion.Version2);

// Set name for this profile
retVal = profileV2.SetProfileName("Profile_2");

// Set TFTP Server IP address, required
retVal = profileV2.SetServerIP("192.168.0.1");

// Set TFTP Server port, default is 69, optional
retVal = profileV2.SetServerPort(69);

// Set SCALANCE read community, default is "public", optional
retVal = profileV2.SetReadCommunity("public");

// Set SCALANCE write community, default is "private", optional
retVal = profileV2.SetWriteCommunity("private");

// Verify profile is valid, this assures we did not miss a parameter
retVal = profileV2.Validate();

//  Update firmware for a specific  SCALANCE  devices

IScalance myScalance = scannedDevices.FindDeviceByIP(0xC0A80001) as
IScalance;
if (myScalance != null)
{
   myScalance.Selected = true;
   //  Set  SNMP  profile  required  to  communicate  with  this
SCALANCE  device
   retVal  =  myScalance.SetProfile(profileV2);

   //  Set  new  firmware  file  to  update  SCALANCE
   retVal  =  myScalance.SetFirmwareFile(@"C:\Firmware
\FirmwareFile.lad");

   // -------------------------------------------------------
   // Update new firmware on SCALANCE device.
   // This method call downloads and activates the firmware.
   // -------------------------------------------------------
   retVal  =  myScalance.FirmwareUpdate();

   // --------------------------------------------------
   // Alternatively, use two-step operation to update
   // new firmware on SCALANCE device.
   // This operation downloads the firmware in one step
   // and activates the firmware in a second step.
   // --------------------------------------------------
   retVal  =
myScalance.FirmwareUpdate(FirmwareUpdateType.DownloadWithoutActivati
on);

   // --------------------------------------------------------------
   // This step activates the downloaded firmware
   // on the SCALANCE device. Activation can be done later and might
   // be based on user input. For simplicity, this example activates
   // the firmware after the download.
   // --------------------------------------------------------------
   retVal  =  myScalance.FirmwareActivate();
   myScalance.Selected = false;
}

/* For simplicity the code examples above do not check for */
/* errors. Checking for and processing errors returned     */
/* from methods is vital to a program's overall quality    */

#endregion
```

12.17.3 Example: SNMP Version 3 configuration

This example shows how to configure and initiate a firmware update for a SCALANCE device. In this example, the SCALANCE device uses SNMP Version 3 with authentication and privacy:

The example shows both types of firmware update:

- Downloading and activating the firmware in a single `FirmwareUpdate` (Page 279) method call
- Downloading the firmware with `FirmwareUpdate` as a first step and activating the firmware with `FirmwareActivate` (Page 280) as a second step.

Example: SNMP Version 3 firmware update with authentication and privacy

```
//----------------------------------------------------------------
// Insert the necessary code from Getting started with the API (Page 149)
// here if you want to compile the example
//----------------------------------------------------------------

#region Updating firmware for a SCALANCE device, SNMP Version 3
profile

SNMPProfile profileV3 = new SNMPProfile();

// Create a profile using SNMP version 3
retVal = profileV3.SetSNMPVersion(SNMPVersion.Version3);

// Set name for this profile
retVal = profileV3.SetProfileName("Profile_3.1");

// Set TFTP Server IP address, required
retVal = profileV3.SetServerIP("192.168.0.1");

// Set TFTP Server port, default is 69, optional
retVal = profileV3.SetServerPort(69);

// Set User name, required
retVal = profileV3.SetUserName("User1");

// Set Context name, optional
retVal = profileV3.SetContextName("Context1");

// Set SCALANCE security level AuthPriv, required
retVal = profileV3.SetSecurityLevel(SNMPSecurityLevel.AuthPriv);

// Set Authentication algorithm to MD5 and password, required
retVal = profileV3.SetAuthAlgorithm(SNMPAuthAlgorithm.MD5,
"Password1");

// Set Privacy algorithm to DES and password, required
retVal = profileV3.SetPrivAlgorithm(SNMPPrivAlgorithm.DES,
"Password2");

// Verify profile is valid, this assures we did not miss a parameter
retVal = profileV3.Validate();


// Update firmware for a specific SCALANCE device

IScalance myScalance = scannedDevices.FindDeviceByIP(0xC0A80001) as
IScalance;
if (myScalance != null)
{
   myScalance.Selected = true;
   // Set SNMP profile required to communicate with this
SCALANCE device
   retVal = myScalance.SetProfile(profileV3);
   //  Set  new  firmware  file  to  update  SCALANCE
   retVal  =  myScalance.SetFirmwareFile(@"C:\Firmware
\FirmwareFile.lad");

   // -------------------------------------------------------
   // Update new firmware on SCALANCE device.
   // This method call downloads and activates the firmware.
   // -------------------------------------------------------
   retVal  =  myScalance.FirmwareUpdate();

   // -------------------------------------------------------
   // Alternatively, use two-step operation to update
   // new firmware on SCALANCE device.
   // This operation downloads the firmware in one step
   // and activates the firmware in a second step.
   // -------------------------------------------------------
   retVal  =
myScalance.FirmwareUpdate(FirmwareUpdateType.DownloadWithoutActivati
on);

   // ---------------------------------------------------------------
   // This step activates the downloaded firmware
   // on the SCALANCE device. Activation can be done later and might
   // be based on user input. For simplicity, this example activates
   // the firmware after the download.
   // --------------------------------------------------------------
   retVal  =  myScalance.FirmwareActivate();
   myScalance.Selected = false;
}

/* For simplicity the code examples above do not check for */
/* errors. Checking for and processing errors returned     */
/* from methods is vital to a program's overall quality    */

#endregion
```

SNMP Version 3 firmware update with no authentication, no privacy

To modify the example to use no authentication and no privacy, remove the example code that sets the security level, authentication algorithm and privilege algorithm. Replace the deleted code with the following code:

```
// Set SCALANCE security level NoAuthNoPriv, required
retVal = profileV3.SetSecurityLevel(SNMPSecurityLevel.NoAuthNoPriv);
```

SNMP Version 3 firmware update with with authentication, no privacy

To modify the example to use authentication but with no privacy, remove the example code that sets the security level, authentication algorithm and privilege algorithm. Replace the deleted code with the following code:

```
// Set SCALANCE security level AuthNoPriv, required
retVal = profileV3.SetSecurityLevel(SNMPSecurityLevel.AuthNoPriv);

// Set Author algorithm to MD5 and password, required
retVal = profileV3.SetAuthAlgorithm(SNMPAuthAlgorithm.MD5,
"Password1");
```

12.18 Exceptions

12.18.1 CriticalInternalErrorException

The API interface includes an exception when a critical condition has been detected.

When your application detects that the CriticalInternalErrorException exception has triggered, shut down the application that is using the API. A critical error has occurred.

```
#region Critical internal error exception
try
{
   //------------------------------------------------------------
   // Insert the necessary code from Getting started with the API (Page 149)
   // here if you want to compile the example
   //------------------------------------------------------------

   ICPU myCPU = scannedDevices.FindDeviceByIP(0xC0A80001) as ICPU;
// 192.168.0.1
   if (myCPU != null)
   {
      myCPU.SetPassword(new EncryptedString("Password"));
      myCPU.Selected = true;
      if (myCPU.Failsafe)
         myCPU.SelectedConfirmed = true;
      retVal = myCPU.ResetToFactoryDefaults();
   }
}
catch (CriticalInternalErrorException e)
{
   // A critical internal error has occurred within the API

   // Shut down the application.
   myCPU.Selected = false;
   myCPU.SelectedConfirmed = false;
}
catch (Exception e)
{
   // An exception has occurred within the API
   myCPU.Selected = false;
   myCPU.SelectedConfirmed = false;

}
```

```
/* For simplicity the code examples above do not check for */
/* errors. Checking for and processing errors returned     */
/* from methods is vital to a program's overall quality    */

#endregion
```

12.19 API enumerations

12.19.1 BackupType

The BackupType enumeration indicates the protocol used to communicate with an HMI device:

Description	Value
Invalid	0
FullBackup	1
Recipes	2
UserAdministration	3

12.19.2 ConfirmationType

This enumeration is used to indicate the status of fail-safe CPUs:

Description	Value
Invalid	0
SafetyPasswordIsBeingUsed	0x2f161717
DeletingExistingSafetyProgram	0x40232122
ReplacingExistingSafetyProgram	0x492a282b
ReplacingExistingSafetyProgramWithNonSafetyProgram	0x4a2c2b2d
LoadingSafetyProgram	0x46292728

12.19.3 DataChangedType

This enumeration defines the possible argument values for the DataChangedEventHandler: (Page 226)

```
Invalid
OperatingState
RackInformation
Folders
File
ProfinetName
IPAddress
```

```
Password
FileSystem,
StopModeTransition
```

12.19.4 DeviceFamily

This enumeration specifies the product family for a hardware item:

```
None
CPU1200
CPU1500
CPU300
CPU400
ET200SP
ET200MP
ET200AL
ET200PRO
ET200ECO
ET200S
ET200M
HMI
SITOPUPS
SCALANCE
SIMOCODE
Unsupported
SIRIUS_ACT
Gateway
NetworkDevice
MicroDrive
SinamicsDrive
SoftStarter
CommunicationsModule
OpticalReader
RFID
```

12.19.5 ErrorCode

This enumeration lists all possible return values for a Result object:

```
OK
AccessDenied
ServiceTimeout
Disconnected
FailedToDisconnect
ServiceNotConnected
TooManySessions
SessionDelegitimated
NotChangableInRun
InvalidFileName
MultiESNotSupported
```

```
ServiceAborted
MultiESLimitExceeded
MultiESIncompatibleOtherESVersion
MultiESConflict
WriteProtected
DiskFull
InvalidVersion
PathNotFound

Failed
CPUFailedToEnterRunMode
MACAddressIsNotValid
IPAddressIsNotValid
SubnetMaskIsNotValid
GatewayIsNotValid
ProfinetNameIsNotValid
NewIPAddressIsNotValid
NewSubnetMaskIsNotValid
NewGatewayIsNotValid
NewProfinetNameIsNotValid
InvalidPointer
SetIPErrorDueProjectSettings
UnsupportedDevice
SetNameErrorDueProjectSettings
OperationNotSupportedByThisDevice
DeviceNotOnNetwork
FirmwareVersionMatch
FirmwareFileNotCompatibleToNew
FirmwareFileNotCompatibleToOld
FirmwareFileNotCompatibleNotSame
FirmwareFileNotCompatibleSame
FirmwareFileNotCompatible
FirmwareModuleNotReachable
FirmwareModuleNotAccepted
FirmwareIDNotFound
WriteBlockFailed
InvalidProjectVersion
DeviceIsNotAcceptingChanges
InvalidSignature
ParmeterOutOfRange
FailedToZipFolderContents
ErrorWritingToFile
ErrorCreatingFile
ErrorCreatingFolder
NoSATLicensePresent
InvalidTimeoutValue
NoDataToBackup
ErrorWritingToStream
ErrorReadingFromStream
InvalidProjectPath
ProjectNotCompatibleWithDevice
FailedToSetProfinetName
```

```
FailedToSetIPAddress
DownloadInvalidRecipe
IdentityFailure
DeviceMismatch
InvalidInterface
DeviceNotSelected
FailsafeAccessRequired
InternalApplicationError
InvalidPassword
DuplicateIPAddress
DuplicateProfinetName
SafetyDeviceMustBeConfirmed
NoSDCardPresent
InvalidProgramFolder
FSignaturesDoesNotMatch
FSignaturesMatch
DeviceDoesNotSupportProject
ProjectsUpdateIPNotReachable
RestoreIPNotReachable
ProjectIPNotUnique
SafetyProjectDownloadedToStandardNotAllowed
PasswordDiversityFailed
InvalidBackupFile
InvalidBackupFileExtension
IncompatibleBackupFile
InvalidFirmwareFile
OperationWasNotSuccessful
CouldNotValidatePassword
IPAddressAlreadyExistsOnNetwork
MissingProgramFilePassword
InvalidProgramFilePassword
OperationCanceledByUser
InvalidProgramForDevice
InvalidProgramFilePasswordLegitimizationLevel
DeviceNotFound
DeviceAlreadyExists
IPAddressAlreadyOnNetwork
ProfinetNameAlreadyOnNetwork
FailedToConnect
DeviceNotInitialized
CPUNewerVersionNotSupported
IPSuitNotValid
IPAddressChanged
ScanNoDevicesFound
DeviceCannotBeInserted
InsertDeviceDuplicateIP
IPNotReachable
CouldNotReadFSignature
InvalidNetworkInterface
InsufficientLegitimizationLevel
NoProgramPassword
ProjectVersionV1NotSupported
```

```
ProjectOpenCanceled
ProgramPasswordNeeded
RestoreError
IncompatibleProgramFile
UnsupportedProgramFile
ProgramFileFamilyMismatch
DuplicateNewIPAddress
SNMPErrorNoAccess
SNMPErrorReadOnly
SNMPErrorNotWritable
SNMPErrorAuthorizationError
SNMPError
InvalidProfileName
InvalidSNMPVersion
InvalidServerIP
InvalidServerPort
InvalidReadCommunity
InvalidWriteCommunity
InvalidUserName
InvalidContextName
InvalidSecurityLevel
InvalidAuthAlgorithm
InvalidAuthPassword
InvalidPrivAlgorithm
InvalidPrivPassword
InvalidProfile
ProfileNameAlreadyExists
ObsoleteMethod
FailedToInitiateFirmwareTransfer
DeviceDefinedError
ErrorDeletingFile
ErrorDeletingFolder
ErrorInvalidMAC
FailedToUpdateDuplicates
DuplicateNewProfinetName
ErrorBackingupData
FailsafeControlObjectIncorrectType
InvalidIPAddressOrUsedByNIC
FirmwareIntegrityFailed
ExportDiagnosticsBufferError
AlmFailedInitialize
AlmFailedCleanup
AlmFailedSessionInitialize
AlmFailedSessionCleanup
AlmSessionIDMissing
AlmSessionIDUnknown
AlmCouldNotConnect
AlmOutOfMemory
AlmOperationTimeout
AlmFunctionNotFound
AlmAborted
AlmBadfunctionArgument
```

```
AlmUnkownOption
AlmSendError
AlmReceiveError
AlmNoConnectionAvailable
AlmOpenSession
AlmResources
AlmService
AlmCryptography
AlmTaskAlreadyRunning
AlmInvalidPointer
AlmResultMismatch
AlmBadResult
AlmBatchWrongArgumentNumber
AlmBatchWrongArgument
AlmBatchWrongInputFile
AlmBatchWrongInputStream
AlmBatchAPILoadFailed
AlmBatchOutputfileExists
AlmBatchWrongOutputParmater
AlmBatchOutputCreationFailure
AlmBatchAccessDenied
AlmUnknownError
PowerCycleRequired
DuplicateFolders
DuplicateFiles
FeatureRequiresValidLicense
APIFeatureNotSupported
CPUProtectionLevelWeaker
CouldNotReadProtectionLevel
ChangingIPNotAllowedNatRouter
IPAndRouterIPCannotBeTheSame
OperationNotAllowedThroughCPCM
FolderOrFileDoesNotExist
DCPOperationNotSupportedBehindNATRouters
OperationNotSupportedThroughCPCM
CPUFailedToEnterStopMode
NoFirmwareToActivate
FailedToActivateFirmware
InvalidOperation
ErrorRefreshingFolder
ErrorCannotRenameFileAlreadyExists
FileWriteProtected
FolderWriteProtected
MemoryCardNotPresent
OperationRequiresStopMode
ErrorWritingFileMemoryCard
ErrorCreatingFolderMemoryCard
ErrorReadingServiceData
NoFilesInListToDownload
InsufficientLegitimizationLevelRefreshStatus
APIFeatureAdvancedNotSupported
```

```
//HMI
UnexpectedOperatingSystemError
ServiceActive
RemoteTransferDisabled
HardwareSoftwareNotComplete
LogicalVolumneMissing
LogicalVolumneOutOfSpace
Abort
FirwareTypeNotSupported
FirwareTypeNotInstalled
StoreReadFailed
StoreWriteFailed
RescueBackupNotPossible
RescueRestoreNotPossible
ConnectionRequired
ObjectNotFound
BufferToSmall
InvalidArguements
AttributeNotFound
InvalidPath
TypeConversionFailed
FileReadFailed
FileWriteFailed
OutOfResources
OutOfSpace
UnknownAddon
IncompatibleAddon
AddonsUnsupported
LicenseFailed
UnknownApp
UnknownAppAddon
UnknownReferenceApp
RuntimeMissing
RuntimeBroken
SignatureRequired
SignatureInvalid
SignatureFailure
CertificateInvalid
CertificateFailure
CertificateNotReady
CertificateExpired
CertificateRevoked
SecurityLib
WrongRuntimeVersion
MajorRuntimeDowngrade
MajorRuntimeUpgrade
MajorImageDowngrade
MajorImageUpgrade
WrongRuntime
NotEnoughMemory
ProjectCharacteristicsMissing
ProjectCharacteristicsInvalid
```

```
PanelOrientationIsPortrait
PanelOrientationIsLandscape
WrongDevicetype
NoRuntimeInstalled
RuntimeCorrupt
InvalidTransferChannel
InvalidBackupType
HMIUnknownError
```

12.19.6 FailsafeOperation

The `FailSafeOperation` enumeration indicates operations that are safety-relevant:

Description	**Value**
`Invalid`	0
`ResetToFactoryOperation`	0x2f161717
`FormatMCOperation`	0x46292728
`ProgramUpdateOperation`	0x43252224
`RestoreOperation`	0x45262427

12.19.7 FirmwareUpdateType

The `FirmwareUpdateType` specifies the type of firmware update to perform: a standard firmware update or a firmware update in two steps. The first step is the firmware update file download. The second step activates the firmware update on the device:

Description	**Value**
`Invalid`	0
`DownloadWithActivation`	1
`DownloadWithoutActivation`	2
`DownloadWithoutActivationInStopMode`	3

12.19.8 HMITransferChannel

The `HMITransferChannel` enumeration indicates the protocol used to communicate with an HMI device:

Description	**Value**
`Invalid`	-1
`PN_IE`	0
`Ethernet`	1

12.19.9 Language

The Language enumeration allows you to assign the language for returned string data. It contains the following values:

```
English
German
French
Spanish
Italian
Chinese
```

12.19.10 OperatingState

This enumeration defines the possible states of the OperatingState property:

```
NotSupported
StopFwUpdate
StopSelfInitialization
Stop
Startup
Run
RunRedundant
Halt
LinkUp
Update
Defective
ErrorSearch
NoPower
CiR
STOPwithoutODIS
RunODIS
```

12.19.11 OperatingStateREQ

This enumeration defines the possible state transitions that can be requested on a call to the SetOperatingState (Page 257) method:

```
Stop
Run
```

12.19.12 ProgressAction

This enumeration defines the possible argument values that can be sent to a ProgressChangedEventHandler (Page 227):

```
Invalid
Connecting
Reconnecting
```

```
Disconnecting
Initializing
Updating
Processing
Downloading
Uploading
Deleting
Reseting
Rebooting
Verifying
Formatting
Refreshing
Finished
UpdatingFirmware
InstallingRuntime
InstallingAddOns
UninstallingAddOns
UpdatingProgram
Renaming
Creating
FileReplaceOrSkip
FileWriteProtected
FolderWriteProtected
```

12.19.13 ProtectionLevel

The `ProtectionLevel` enumeration gives the protection level of a CPU password:

```
Unknown
Failsafe
Full
Read
HMI
NoAccess
NoPassword
```

12.19.14 RemoteInterfaceType

This enumeration defines the possible states that can be returned from a call to the `InterfaceType` property on the IRemoteInterfaces (Page 264) interface:

```
None
Profinet
Profibus
ASi
```

12.19.15 RemoteFolderType

The `RemoteFolderType` enumeration indicates the remote folder type:

Description	Value
`None`	0
`Recipe`	1
`Datalog`	2
`Files`	3

12.19.16 ScanErrorType

The `ScanErrorType` enumeration indicates the type of error returned by a device scan:

Description	Value
`Invalid`	-1
`Success`	0
`Error`	1
`Warning`	2
`Information`	3

12.19.17 SNMPAuthAlgorithm

The `SNMPAuthAlgorithm` enumeration indicates the authorization algorithm for a SCALANCE device that uses an SNMP Version 3 profile:

Description	Value
`NotSupported`	-1
`MD5`	0
`SHA`	1

12.19.18 SNMPPrivAlgorithm

The `SNMPPrivAlgorithm` enumeration indicates the privacy algorithm for a SCALANCE device that uses an SNMP Version 3 profile:

Description	Value
`NotSupported`	-1
`DES`	0
`AES`	1

12.19.19 SNMPSecurityLevel

The `SNMPSecurityLevel` enumeration indicates the security level for a SNMP version 3 profile. A SCALANCE device that uses an SNMP version 3 profile has security level settings:

Description	**Value**
`NotSupported`	-1
`NoAuthNoPriv`	0
`AuthNoPriv`	1
`AuthPriv`	2

12.19.20 SNMPVersion

The `SNMPVersion` enumeration indicates the SNMP version number for SCALANCE devices:

Description	**Value**
`NotSupported`	-1
`Version1`	0
`Version2`	1
`Version3`	2

12.19.21 TimeFormat

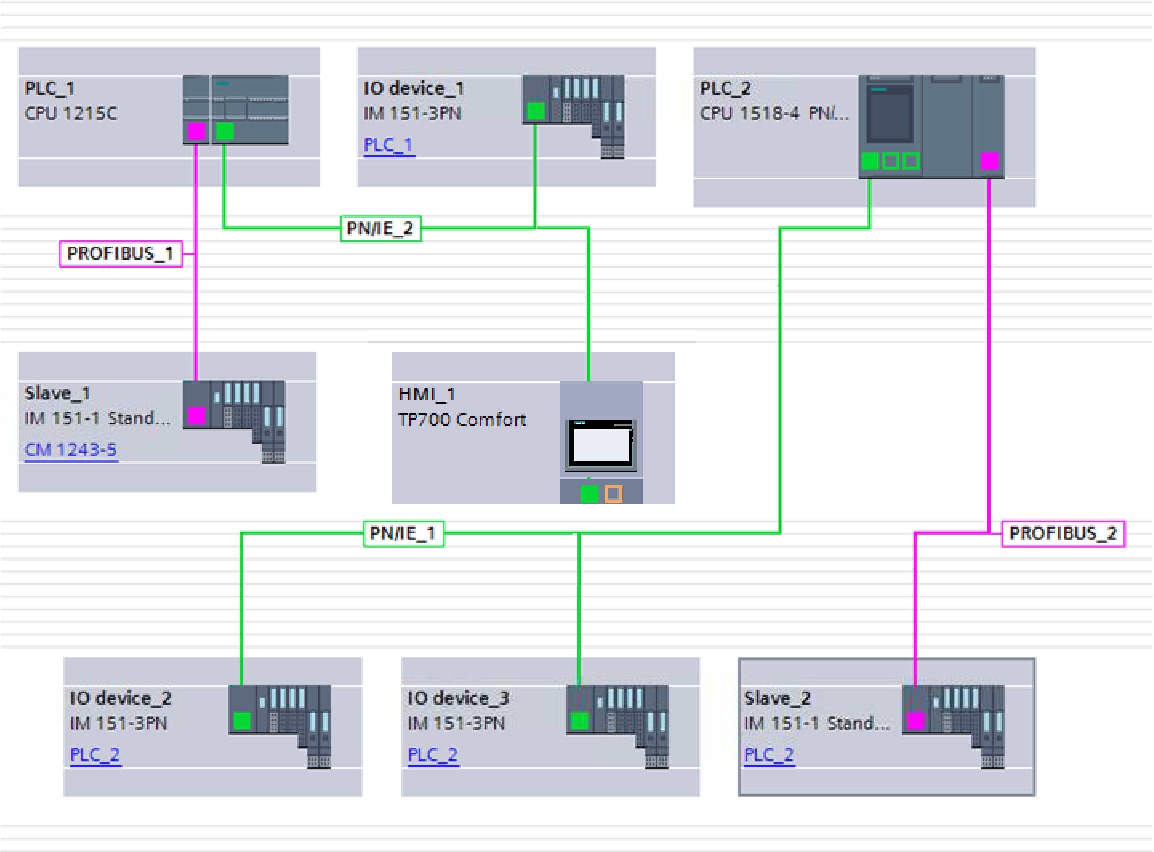
The `TimeFormat` specifies a time format for time strings:

Description	**Value**
`Local`	0
`UTC`	1

UTC indicates Coordinated Universal Time.

12.20 Network example

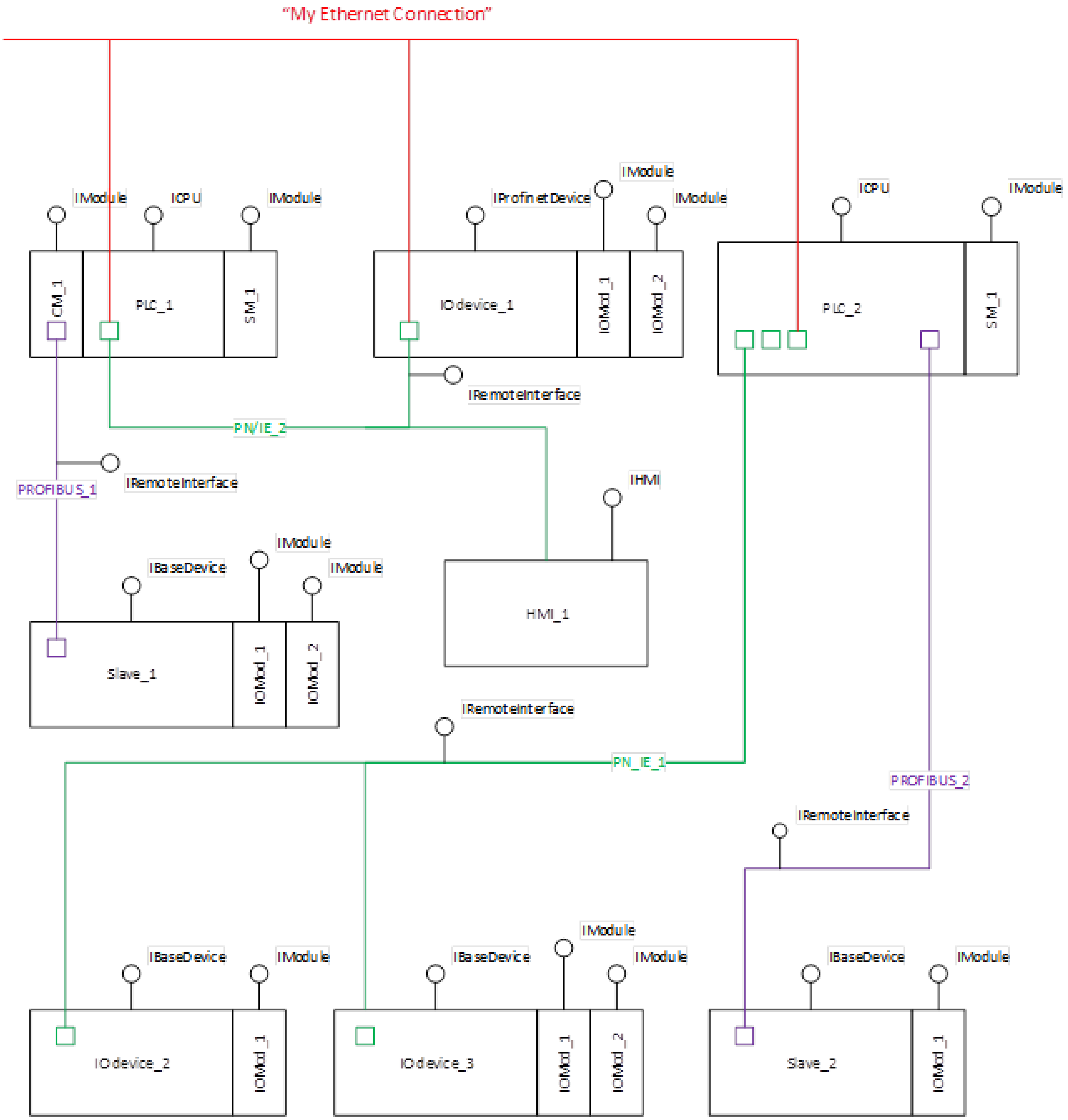
This example shows a TIA Portal network configuration and the API interfaces that represent the networked devices:



Assume that all the devices in the top row (PLC_1, IO device_1, and PLC_2) are connected to an external PROFINET network (not shown). They can be directly accessed by the SIMATIC Automation Tool API. Further, assume that the PROFINET subnet connected to PLC_2 is not connected to the external network.

The SIMATIC Automation Tool API can provide information and operations for all the PLCs and I/O stations in this configuration.

The following diagram shows the same network configuration and the hardware devices on the network:



In the diagram above, the "lollipop" notation shows which SIMATIC Automation Tool API interface class best represents each network component:

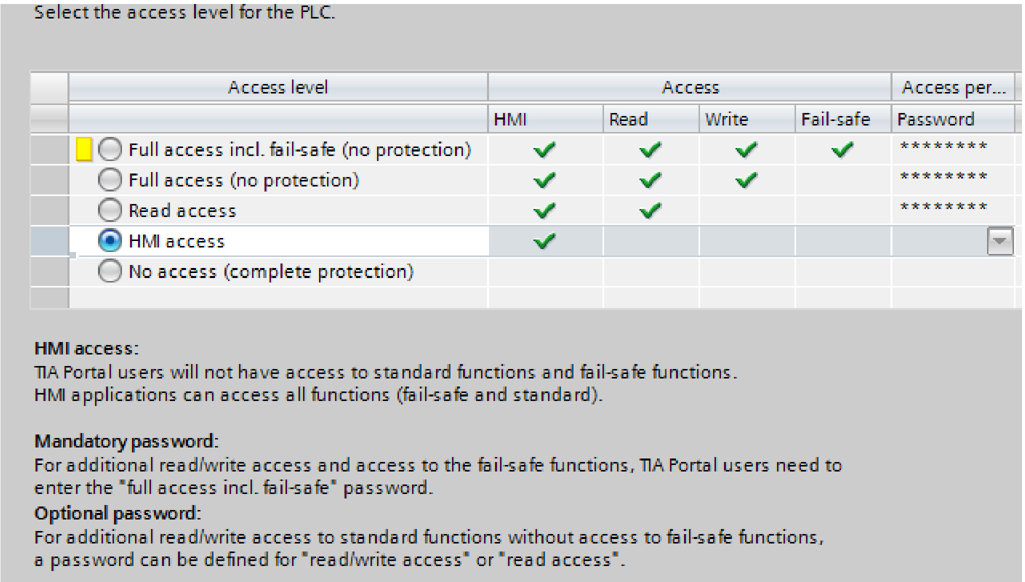
- CPUs directly connected to the external network are represented by the `ICPU` interface (Page 230).
- I/O Stations directly connected to the external network are represented by the `IProfinetDevice` interface (Page 203).
- Subnets originating from a CPU are represented by the `IRemoteInterface` interface (Page 179).

- I/O Stations not directly connected to the external network (but accessible through a CPU) are represented by the `IBaseDevice` interface (Page 180).
- I/O or Communications modules connected to a CPU or IO Station are represented by the `IModule` interface (Page 229).

12.21 CPU password access levels

A standard CPU has four password access levels. A fail-safe CPU has five levels.

In the TIA Portal, you can see the CPU access levels in the Protection & Security section of the CPU device Properties:



Select the access level for the PLC.

Access level	Access				Access per...
	HMI	Read	Write	Fail-safe	Password
○ Full access incl. fail-safe (no protection)	✓	✓	✓	✓	********
○ Full access (no protection)	✓	✓	✓		********
○ Read access	✓	✓			********
● HMI access	✓				
○ No access (complete protection)					

HMI access:
TIA Portal users will not have access to standard functions and fail-safe functions.
HMI applications can access all functions (fail-safe and standard).

Mandatory password:
For additional read/write access and access to the fail-safe functions, TIA Portal users need to enter the "full access incl. fail-safe" password.
Optional password:
For additional read/write access to standard functions without access to fail-safe functions, a password can be defined for "read/write access" or "read access".

Device operations that require read access require any access level that provides the "Read" access.

Device operations that require write access require any access level that provides the "Write" access.

For additional information on access levels and passwords, refer to the *STEP 7 Information System* (online help for the TIA Portal).

Password requirement for safety-relevant operations on F-CPUs

You can perform some operations on a fail-safe CPU using read access or write access. Safety-relevant operations require the safety password, which corresponds to "Full access incl. fail-safe (no protection)" access level.
The safety-relevant operations are:

- Program Update (Page 48)
- Restore from Backup (Page 59)

- Reset to Factory Defaults (Page 36)
- Format Memory Card (Page 39)

If a fail-safe CPU does not use password protection, safety-relevant operations do not require the safety password to initiate the operation.

Throughout this user guide, the term "safety password" is the password with the "Full access incl. fail-safe (no protection)" access level.

Reference information 13

13.1 SIMATIC Automation Tool process interaction

The SIMATIC Automation Tool enforces several requirements to support safe operation of your process.

Canceling operations

From the Device table, you can perform a device operation on one or many devices. When the operation is in progress, you can cancel the operation. The SIMATIC Automation Tool cancels the operation as quickly as possible on devices that have not yet completed the operation.

CPU operating mode

Any time you use the SIMATIC Automation Tool to change the operating mode of a CPU (Page 30), you must confirm a prompt. Confirming the prompt means that you acknowledge the change and accept any associated risks.

The SIMATIC Automation Tool never automatically sets the CPU operating mode to RUN mode. Changes to RUN mode occur only when you explicitly set the CPU operating mode (Page 30) to RUN mode and confirm the prompt.

Some device operations require the CPU to be in STOP mode. The SIMATIC Automation Tool prompts you to permit the change to STOP mode if the CPU is in RUN mode. Confirming the prompt means that you acknowledge the change to STOP mode and accept any associated risks.

Scheduled operations

The SIMATIC Automation Tool ensures that a scheduled operation runs at the date and time specified. It prevents an operation from running at an unscheduled time.

If you use the Scheduler application to disable a scheduled operation, the SIMATIC Automation Tool will not enable the disabled operation. You must enable the operation through the SIMATIC Automation Tool and the Scheduler application.

Refer to Executing scheduled operations (Page 107) for specific information on canceling running operations in the Scheduler application.

13.2 CPU passwords

If a CPU is protected, the SIMATIC Automation Tool displays a lock symbol beside the CPU device name. You must enter a password in the "CPU Password" column for the CPU. The CPU password must provide the required access level for your operation.

Using passwords

The SIMATIC Automation Tool handles passwords in the following ways:

- If a CPU is protected, the SIMATIC Automation Tool enforces the CPU password requirements according to the operation and the access level. For protected operations, you must enter a valid password in the "CPU Password" cell.
- If a CPU password entry is valid, you can hover your mouse cursor over the password field to display a tool tip that shows the access level.
- The SIMATIC Automation Tool displays a device's password cell in yellow when you enter a safety password (Page 308).

Password icons

The SIMATIC Automation Tool provides three password status icons:

✔ Password is valid

✖ Password is invalid

? The SIMATIC Automation Tool cannot yet validate the device password in a backup file

When you enter a password, the SIMATIC Automation Tool validates it. When you initiate a device operation, the access level of the password (Page 308) must be sufficient for that operation. The Event Log displays a message if you have an insufficient password for an operation.

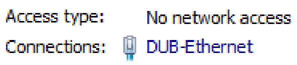
13.3 Identifying your network interface

After you connect your programming device to a network, you can use Windows to see the name of the network interface.

In the following example, the device network connects to the programming device through an Ethernet adapter. The network names that you see on your programming device depend on your network hardware.

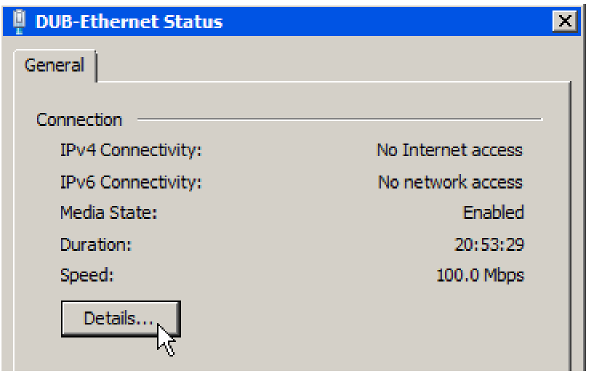
Use the Windows Control Panel or Settings to identify the name of the device:

1. Open the Windows Control Panel or Settings.
2. Open the Network and Sharing center from the Network & Internet settings.
3. View your available networks and click the "Connections:" network that is connected to your devices. The network might be named "Unidentified network" and the "Connections:" name might be a numbered Ethernet label such as "Ethernet 3":



Access type: No network access
Connections: DUB-Ethernet

4. Click the Details button in the Status dialog:



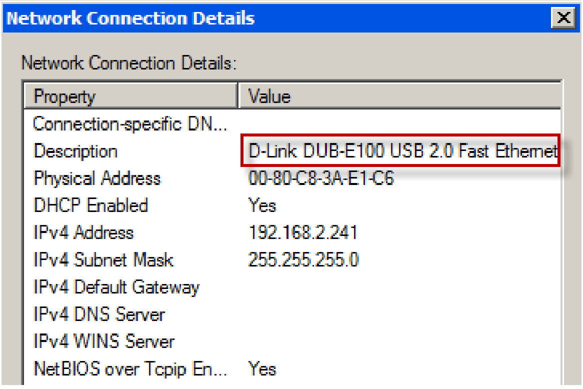
DUB-Ethernet Status

General

Connection

IPv4 Connectivity:	No Internet access
IPv6 Connectivity:	No network access
Media State:	Enabled
Duration:	20:53:29
Speed:	100.0 Mbps

5. View the description of the network interface:



Network Connection Details

Network Connection Details:

Property	Value
Connection-specific DN...	
Description	D-Link DUB-E100 USB 2.0 Fast Ethernet
Physical Address	00-80-C8-3A-E1-C6
DHCP Enabled	Yes
IPv4 Address	192.168.2.241
IPv4 Subnet Mask	255.255.255.0
IPv4 Default Gateway	
IPv4 DNS Server	
IPv4 WINS Server	
NetBIOS over Tcpip En...	Yes

6. In the SIMATIC Automation Tool, choose the ".Auto" network interface (Page 11) corresponding to this network connection.

Note

Communication problems with the SIMATIC Automation Tool

The SIMATIC Automation Tool requires communication to your device network. If S7-PLCSIM is running, close S7-PLCSIM before using the SIMATIC Automation Tool.

You might also send a device operation to multiple devices, but one device does not complete the operation. You see a communication error in the Event Log. Other devices, however, process the device operation as you expect. If you have this problem, follow these steps:

1. Reduce the number of simultaneous operations that you allow in the Communications settings (Page 82).
2. Close and restart the SIMATIC Automation Tool.
3. Try the operation again to one or more devices.

If you execute a device operation and the connection to the device has a very slow data transfer rate, you might get a communication timeout error. If you have this problem, increase the timeout for communications operations in the Communications settings (Page 82).

13.4 Keyboard shortcuts

The SIMATIC Automation Tool supports the following shortcut keys for navigating in the Device table and for some of the menu commands (Page 75).

CTRL+PgUp	Switches between tabs, from right to left
CTRL+PgDn	Switches between tabs, from left to right
CTRL+A	Selects the entire table
CTRL+C	Copies the selection to the clipboard
CTRL+N	Creates new SIMATIC Automation Tool project
CTRL+O	Displays the project open dialog to open a new project file
CTRL+S	Displays the Save As dialog
CTRL+V	Pastes the contents of the clipboard at the insertion point and replaces any selection
CTRL+X	Cuts the selected cells
CTRL+Z	Performs an undo of the last edit or delete action
ARROW KEYS	Moves one cell up, down, left, or right
SHIFT+ARROW KEYS	Extends the selection
DELETE	Removes the contents of the active cell
ENTER	Completes cell editing and validates data
ESC	Cancels cell editing and restores the cell to original value
HOME	Moves to the beginning of a row
CTRL+HOME	Moves to the beginning of the table
END	Moves to the end of a row
CTRL+END	Moves to the end of the table
PAGE DOWN	Moves one screen down in the table
PAGE UP	Moves one screen up in the table
SPACEBAR	Selects or clears the device selection check box
TAB	Moves one cell to the right
F5	Refreshes all selected devices
Alt+F4	Exits

13.5 Safety program definition

A safety program is a program you create in STEP 7 that includes safety blocks. If you have installed STEP 7 Safety Advanced, then whenever you add a fail-safe CPU to your STEP 7 project, the Program Blocks folder automatically includes the safety blocks. When you download this program to a fail-safe CPU, it is a safety program.

13.6 Security information

Siemens provides products and solutions with industrial security functions that support the secure operation of plants, systems, machines and networks.

In order to protect plants, systems, machines and networks against cyber threats, it is necessary to implement – and continuously maintain – a holistic, state-of-the-art industrial security concept. Siemens' products and solutions constitute one element of such a concept.

Customers are responsible for preventing unauthorized access to their plants, systems, machines and networks. Such systems, machines and components should only be connected to an enterprise network or the internet if and to the extent such a connection is necessary and only when appropriate security measures (e.g. firewalls and/or network segmentation) are in place.

For additional information on industrial security measures that may be implemented, please visit (https://www.siemens.com/industrialsecurity).

Siemens' products and solutions undergo continuous development to make them more secure. Siemens strongly recommends that product updates are applied as soon as they are available and that the latest product versions are used. Use of product versions that are no longer supported, and failure to apply the latest updates may increase customers' exposure to cyber threats.

To stay informed about product updates, subscribe to the Siemens Industrial Security RSS Feed visit (https://www.siemens.com/industrialsecurity).

13.7 Service and support

Siemens offers technical expertise at the Siemens Automation Web site (https://www.siemens.com/automation/) and the Siemens Industry Online Support Web site (http://support.industry.siemens.com). Contact your Siemens distributor or sales office for assistance in answering technical questions, for training, or for ordering Siemens products. Your sales representatives are technically trained and have knowledge about your operations, process, industry, and Siemens products.

Document source language

The English version of the *SIMATIC Automation Tool User Guide* is the authoritative (original) language for SIMATIC Automation Tool information.

Index