rexroth
A Bosch Company

Application Manual

ctrlX I/O Engineering

Field Bus Configuration for ctrlX OS 02VRS

R911421586, Edition 03

Disclaimer

The data specified above only serve to describe the product. As our products are constantly being further developed, no statements concerning a certain condition or suitability for a certain application can be derived from our information. The information given does not release the user from the obligation of own judgment and verification. It must be remembered that our products are subject to a natural process of wear and aging.

Table of contents

1 About this documentation

Editions of this documentation

Edition	Release date	Note
01	2023 – 10	Edition for version WRK-V-0202 Revised contents: • ➦ Tab – “Variables” • ➦ Configure Tab – “Slave to slave” • ➦ Configure Tab – “CoE”
02	2024 – 04	Edition for Version WRK-V-0204 New contents: • ➦ Use EPLAN data in I/O Engineering • ➦ Command “Create new project via EPLAN import...” • ➦ Chapter Command “Import from EPLAN...” on page 162 Revised contents: • ➦ Tab – “Variables” • ➦ Configure Tab – “Distributed Clocks”
03	2024 – 07	Edition for Version IOE-V-0206 Revised contents: • ➦ Configure Tab – “Distributed Clocks”

1.1 Licensing notes

Open Source Components

A list of the open source components is available in ctrlX WORKSand in the ctrlX OS interface in the side navigation in the menu item “About” below the button “Open Source Components”, see web documentations:

➦ Window – About

➦ Window – Open Source Components

2 Important directions on use

2.1 Intended use

2.1.1 Introduction

Rexroth products are developed and manufactured to the state-of-the-art.

The products are tested prior to delivery to ensure operational safety and reliability.

⚠ WARNING	**Personal injury and damage to property due to incorrect use of products!** The products may only be used as intended. Failure to use the products as intended may cause situations resulting in property damage and personal injury.

NOTICE	**Damages resulting from unintended use**

Rexroth As the manufacturer does not assume any warranty, liability or compensatory claims for damages resulting from unintended use of the products. The user alone shall bear the risks of an unintended use of the products.

Before using Rexroth products, make sure that all the prerequisites for an intended use of the products are met:

- Personnel that in any way, shape or form uses Rexroth products must first read and understand the relevant safety instructions and be familiar with their intended use
- Leave hardware products in their original state, i.e., do not make any structural modifications. It its not permitted to decompile software products or alter source codes
- Do not install damaged or defective products or commission them
- It has to be ensured that the products have been installed as described in the relevant documentation

2.1.2 Areas of use and application

Products of the ctrlX series are suitable for Motion/Logic applications.

NOTICE	Products of the ctrlX series may only be used with the accessories, mounting parts, and other components specified in this documentation. Components that are not expressly mentioned must neither be attached nor connected. The same applies to cables and lines. Only to be operated with the hardware component configurations and combinations expressly specified and with the software and firmware specified in the corresponding documentations and functional descriptions.

Products of the ctrlX series are suitable for single-axis as well as for multi-axis drive and control tasks. Device types with different equipment and interfaces are available for using the system in specific applications.

Typical areas of application:

- Building automation
- IoT and Security Gateway or Device
- Handling & Robotic

Controls of the ctrlX CORE series may only be operated under the mounting and installation conditions, in the position of normal use and under the ambient conditions (temperature, degree of protection, humidity, EMC, etc.) specified in the related documentations.

2.2 Unintended use

"Unintended use" refers to using the ctrlX products outside of the above-mentioned areas of application or under operating conditions and technical data other than described and specified in the documentation.

ctrlX products must not be used if they are exposed to following conditions:

- Operating conditions that do not meet the specified ambient conditions. Operation under water, under extreme temperature fluctuations or under extreme maximum temperatures is prohibited
- Applications that have not been expressly authorized by Rexroth

3 Safety instructions

The Safety instructions contained in the available application documentation feature specific signal words (DANGER, WARNING, CAUTION or NOTICE) and, where required, a safety alert symbol (in accordance with ANSI Z535.6-2006).

The signal word is meant to draw the reader's attention to the safety instruction and identifies the hazard severity.

The safety alert symbol (a triangle with an exclamation point), which precedes the signal words DANGER, WARNING and CAUTION, is used to alert the reader to personal injury hazards.

The Safety instructions in this documentation are designed as follows:

⚠ DANGER	In case of non-compliance with this safety instruction, death or serious injury **will** occur.
⚠ WARNING	In case of non-compliance with this safety instruction, death or serious injury **could** occur.
⚠ CAUTION	In case of non-compliance with this safety instruction, minor or moderate injury could occur.
NOTICE	In case of non-compliance with this safety instruction, property damage could occur.

4 Introduction

ctrlX I/O Engineering is an application program for configuring field buses and field bus devices on ctrlX devices. The configuration allows maximum flexibility in the selection of programming tools.

ctrlX I/O Engineering is intended for installation on an Engineering PC and is part of the ctrlX WORKS installation package, see: ➦ Installation

The installation and use of ctrlX I/O Engineering is free of charge and does not require a license.

Note to the source software

ctrlX I/O Engineering is based on the configurators of the CODESYS development system of the CODESYS GmbH.

Additionally, ctrlX CORE-specific extensions have been added.

Parts of the following documentation refer to CODESYS products, CODESYS descriptions and documentations.

Starting ctrlX I/O Engineering

ctrlX I/O Engineering can be started via the following calls:

- **Via the Windows Start menu of the Engineering PC**
- **Via the web interface of the ctrlX device**
- **About the icon in ctrlX WORKS**

The type of call has a decisive impact on the behavior of the programming system in the following areas:

- Project handling
- Establishing a connection to the target device
- Project synchronization

The differences in detail:

- **Call via the Windows start menu**: Tile ctrlX I/O Engineering)
  Upon the start via the start menu, ctrlX I/O Engineering is started in stand-alone mode.
  In this mode, there is no connection to a ctrlX device after the start.
  You can create a project based on device-specific ctrlX project templates and transfer it to the target device at a later time.
  - Free project creation using device-specific ctrlX project templates
  - Offline project planning possible
- **Call via the web interface of the ctrlX device**: See ➦Web documentation
  Upon start via the web interface of the ctrlX device, the ctrlX device and ctrlX I/O Engineering are synchronized. For ctrlX devices in factory state, the appropriate basic project is automatically created in ctrlX I/O Engineering. For already configured ctrlX devices, a synchronization of the project files on the ctrlX device and the Engineering PC is performed automatically.
  More advantages:
  - Option to “read out” the current field bus configuration from the ctrlX device
  - Transfer configuration changes directly to the ctrlX device
  - Display and switching option of the bus operating state
  - Display of diagnostics, messages and warnings

Language setting of the of the user interface

In the *“Options → International settings ”* dialog, you can set the national language in which the interface of the development system should appear upon the next start. You can set the national language of the help separately.

When calling I/O Engineering via the command line, you can specify the language of the interface via a parameter.

Also refer to

- ➦Chapter Dialog 'Options' – 'International Settings' on page 182

Copyrights and trademarks

5 Installation

ctrlX I/O Engineering is part of the ctrlX WORKS installation package, which is available in the ctrlX Store, see:

➦ctrlX WORKS download in ctrlX Store

The installation and use of ctrlX I/O Engineering is free of charge and royalty free. Registration is only required for the download.

For further information on the installation, please refer to ctrlX WORKS Application Manual, see: ➦Web documentation

6 Logging in and logging off to/from the control

Each access to the control requires that the user is logged in to the control.

Login via context menu

The login command is contained in the context menu of the control node in the device tree.

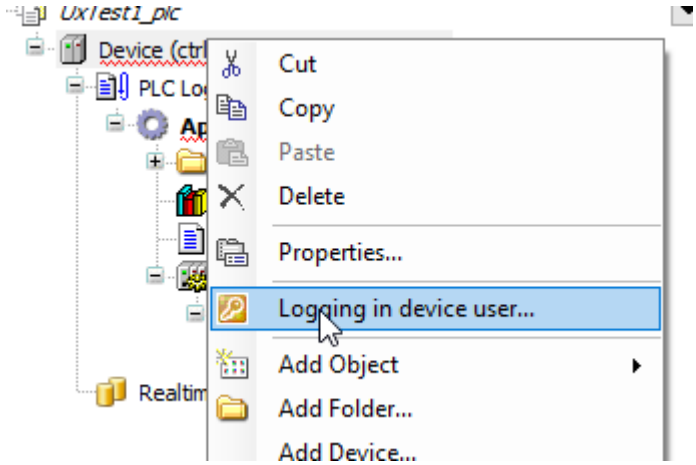


Fig. 1: Command - Logging in device user...

The command opens the “Login - [Control IP address]” dialog by entering the login data for the control (user name & password).

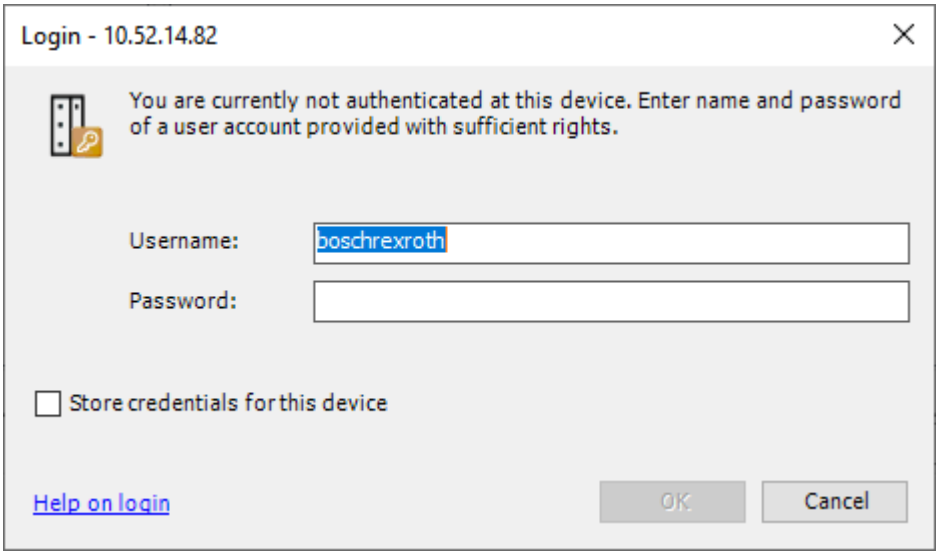


Fig. 2: Dialog - Login - [Control IP address]

The Engineering system “remembers” the entered login data until the next restart. Consequently, the login dialog is not displayed each time the user wants to log in.

When enabling the checkbox “Save login data for this device”, the login data in the Windows Credential Manager and are available after a restart of the Engineering system.

Confirm the settings with “OK”.

The Engineering system connects to the control.

The logged in device user is displayed in the status bar.

Login via status bar

The status bar indicates whether or not a device user is logged in to the ctrlX device.

If a device user is logged in, the user name is displayed in the status bar.

In the following example, the "boschrexroth" device user is logged in to the ctrlX device.

Fig. 3: Status bar - with logged in user "boschrexroth"

If no user is logged in, double-click on the “Device user” field to open the login dialog (see above).

Fig. 4: Status bar - without logged in user

Automatic logoff by function call

Each time a command requiring a login is executed, it is verified if a device user is logged in. If not, a login dialog, see above.

Logoff via context menu

If a device user is logged in, the user can be logged off using the “Log out current device user” command in the context menu of the control node, see ➥ Chapter Command “Log out current device user” on page 139.

NOTICE	Please note that logging off might also cause a logoff from the control!

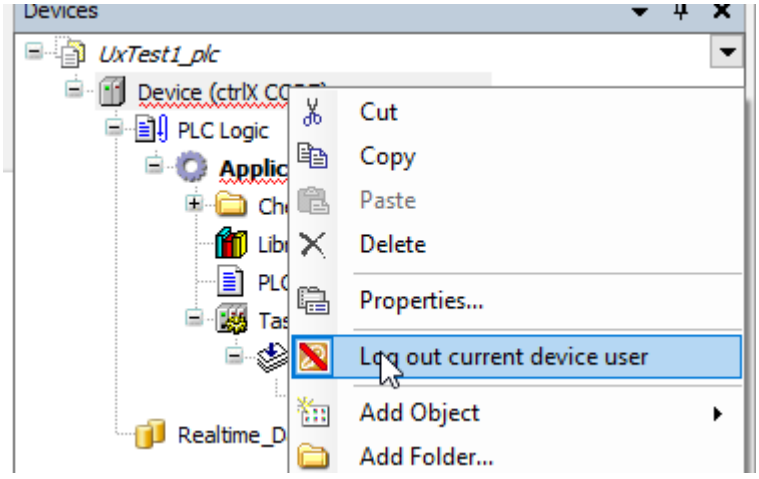


Fig. 5: Command - Log out current device user

Logoff via status bar

If a device user is logged in to the control, a prompt is displayed when double-clicking on the “Device user” field, asking the user if the device user should be logged off.

Logoff by exiting

When exiting the Engineering system, a logged in device user is automatically logged off from the control.

Upon a new login, the login data has to be entered again unless the user has enabled th option to save the login data, see ➥ Chapter 21.4.3 Dialog “Login - [Control IP address]” on page 164.

Further information:

- ➦ Chapter Command “Logging in device user...” on page 138
- ➦ Chapter Command “Log out current device user” on page 139
- ➦ Chapter 21.4.3 Dialog “Login - [Control IP address]” on page 164

7 Configuring the I/O Engineering

You can customize the behavior, appearance, menu composition and window arrangement of ctrlX I/O Engineering System. In the “Tools” menu, there are dialogs for customizing the interface and for setting the I/O Engineering options.

Also refer to:

- ➦ Dialog 'Customize' - 'Command Icons'
- ➦ Dialog 'Customize' - 'Keyboard'
- ➦ Dialog 'Customize' - 'Menu'
- ➦ Dialog 'Customize' - 'Toolbars'

7.1 Setting I/O Engineering Options

You can configure the behavior and appearance of the ctrlX I/O Engineering System in the different tabs of the “Options” dialog. The dialog opens after selecting the *“Tools → Options”* command. Here you can configure the default settings for different editors and functionalities. These settings apply throughout I/O Engineering.

The settings are stored in your current user profile on your local system. For use on other systems, option settings, either user-specific or machine-specific (computer), can be exported to an XML file.

I/O Engineering checks if an older version is already installed when the development system is started for the first time. If this is the case, then the “Import Assistant” dialog opens for transferring the I/O Engineering options set with the older version.

Also refer to

- ➦ Chapter Command 'Options' on page 157
- ➦ Chapter Command 'Import and export options’ on page 157
- ➦ Chapter 21.4.1 'Import Wizard' dialog on page 162
- ➦ Chapter 7.2.1 Customizing Menus on page 16
- ➦ Chapter 7.2.4 Customizing Keyboard Shortcuts on page 19
- ➦ Chapter 7.2.2 Customizing Toolbars on page 18

7.2 Customizing the user interface

In I/O Engineering, you can customize the user interface by changing the window layout as well as the appearance of menus and commands according to your requirements.

7.2.1 Customizing Menus

You can customize the menu commands of the I/O Engineering user interface. In a configuration dialog, you can add or remove menus.

Removing menus and commands

1. Choose the command *“Tools → Customize”*.
   ➡ The “Customize” dialog box opens. The “Menu” tab is visible.
2. Select a menu in the menu tree or a command in a menu.

3. Click “Delete”.
   ➡ The menu or command is deleted from the menu tree.
4. Click “OK”.
   ➡ The dialog box closes and the menu is customized.

Adding menus

1. Choose the command *“Tools → Customize”*.
   ➡ The “Customize” dialog box opens. The “Menu” tab is visible.
2. Scroll to the end of the menu tree.
3. Select the blank symbol (⬜).
4. Click “Add Popup Menu”.
   ➡ The “Add Popup Menu” dialog box opens.
5. Type a name for the new menu in the “Default text” field.
   If localization is unnecessary, then skip to step 9.
6. Click “Add Language”.
   ➡ A drop-down list opens with available languages.
7. Choose the required language.
   ➡ The language is added to the list of languages.
8. Click into the “Text” field and type the language-specific text.
9. Click “OK”.
   ➡ The new menu is added at the bottom of the menu tree.
10. Change the menu order by clicking “Move up” and “Move down”. Click “OK” to close the “Customize” dialog box.
    ℹ The new menu is displayed only when it contains a command.

Adding commands

1. Choose the command *“Tools → Customize”*.
   ➡ The “Customize” dialog box opens. The “Menu” tab is visible.
2. Expand the branch of the menu where the new command should be added.
3. Select the blank symbol (⬜).
4. Click “Add Command”.
   ➡ The “Add Command” opens dialog box.
   The dialog box lists all commands grouped by category.
5. Select the command to be added. Click “OK”.
   ➡ The new command is added to the menu tree.
6. Change the menu order by clicking “Move up” and “Move down”. Click “Add separator” to add a border between commands. Click OK to close the “Customize” dialog box.
   ➡ The new command is now available in the menu.

See also

- ➥ Chapter Dialog 'Customize' - 'Menu' on page 188
- ➥ Chapter 7.2.2 Customizing Toolbars on page 18

7.2.2 Customizing Toolbars

You can customize the toolbars of the I/O Engineering user interface. In a configuration dialog, you can add or remove toolbars.

Removing toolbars and commands

1. Choose the command *“Tools → Customize”*.
   ➧ The “Customize” dialog box opens.
2. Choose the “Toolbars” tab.
3. Select a toolbar or a command from a toolbar tree.
4. Click “Delete”.
   ➧ The toolbar or command is deleted.
5. Click “OK”.
   ➧ The dialog box closes and the toolbar or command is removed.

Adding toolbars

1. Choose the command *“Tools → Customize”*.
   ➧ The “Customize” dialog box opens.
2. Choose the “Toolbars” tab.
3. Select the blank toolbar.
4. Click “Add Toolbar”.
   ➧ The cursor blinks in the new toolbar.
5. Type a name.
6. Change the toolbar order by clicking “Move up” and “Move down”. Click “OK” to close the “Customize” dialog box.
   ℹ I/O Engineering displays the new toolbar only when it contains a command.

Adding commands

1. Choose the command *“Tools → Customize”*.
   ➧ The “Customize” dialog box opens.
2. Choose the “Toolbars” tab.
3. Expand the tree of the toolbar where the new command should be added.
4. Select the blank symbol (▢).
5. Click “Add Command”.
   ➧ The “Add Command” dialog box opens.
   The dialog box lists all commands grouped by category.
6. Select the command to be added. Click “OK”.
   ➧ The new command is added to the toolbar tree.
7. Change the toolbar order by clicking “Move up” and “Move down”. Click “Add separator” to add a border between commands. Click “OK” to close the “Customize” dialog box.
   ➧ The new command is available in the toolbar.

See also

- ➦Chapter Dialog 'Customize' - 'Toolbars' on page 189
- ➦Chapter 7.2.1 Customizing Menus on page 16

7.2.3 Customize Command Icon

I/O Engineering provides the capability of assigning customized icons to commands.

1. Select the command *"Tools → Customize"*.
   ➡ The "Customize" dialog box opens.
2. Click the "Command icons" tab.
3. Select the category "Help" from the list on the left.
   ➡ All commands in this category are listed on the right.
4. Select the command "Information".
5. Click "Assign".
   ➡ A dialog box opens for selecting the icon file (*.ico).
6. Select an icon file.
7. Click the "Open" button.
   ➡ The icon is assigned to the selected command.
8. Click "OK".

See also

- Chapter Dialog 'Customize' - 'Command Icons' on page 189

7.2.4 Customizing Keyboard Shortcuts

I/O Engineering provides the capability of executing commands directly via keyboard shortcuts. You can customize or extend predefined keyboard shortcuts.

1. Choose the command *"Tools → Customize"*.
   ➡ The "Customize" dialog box opens.
2. Choose the "Keyboard" tab.
3. Select the category "Help" from the list on the left.
   ➡ All commands in this category are listed on the right.
4. Select the command "Search".
5. Click into the field "Press Shortcut Keys".
6. Press [Ctrl]+[Shift]+[S].
   ➡ I/O Engineering adds the key combination to the field.
7. Click "Assign".
   ➡ The keyboard shortcut is assigned to the command.
8. Click "OK".
   ➡ You can call the "Search" command by pressing [Ctrl]+[Shift]+[S].

See also

- Chapter Dialog 'Customize' - 'Keyboard' on page 190

7.2.5 Changing the Window Layout

In I/O Engineering, you can easily customize the layout of different views to your individual needs.

1. Drag the view by the caption bar or by the tab.
   ➡ Arrows are shown to mark possible destinations. Example: 

2. Drag the view to one of the arrows.
   ➡ The destination is displayed as a blue-shaded area.
3. Release the left mouse button.
   ➡ The window is inserted into the selected destination.

The window can also be placed outside of the I/O Engineering programming interface.

See also

- Chapter 7.2.6 Resizing Windows on page 20
- Chapter 7.2.7 Auto-Hiding Windows on page 20
- Chapter 7.2.8 Switching Between Windows on page 20

7.2.6 Resizing Windows

1. Move the mouse pointer over the border between two windows or views.
   ➡ The cursor becomes a left-right arrow.
2. Drag the border to another position.

You can resize detached views by moving the frame lines.

See also

- Chapter 7.2.5 Changing the Window Layout on page 19
- Chapter 7.2.7 Auto-Hiding Windows on page 20
- Chapter 7.2.8 Switching Between Windows on page 20

7.2.7 Auto-Hiding Windows

Hiding windows

When you hide a view, it is minimized to a tab in the frame of the user interface. When you move the pointer over the tab, the window is shown automatically.

1. Click into the window to be hidden.
2. Click *“Window → Auto Hide”*.
   Or click the pin symbol ( ) in the upper right corner of the view.
   ➡ The window is hidden and only visible by a small tab on the edge of the main window.
3. Move the mouse pointer over the tab.
   ➡ The window is shown as long as the mouse pointer hovers over the tab.

Showing windows

1. Click the tab of the hidden window.
2. Clear the check box *“Window → Auto Hide”*.
   Or click the pin symbol ( ) in the upper right corner of the view.
   ➡ The window is permanently shown.

See also

- Chapter 7.2.5 Changing the Window Layout on page 19
- Chapter 7.2.6 Resizing Windows on page 20
- Chapter 7.2.8 Switching Between Windows on page 20

7.2.8 Switching Between Windows

It is possible to switch directly between the currently opened views and the editor windows.

1. Press the keystroke combination [Ctrl]+[Tab]. Continue pressing the [Ctrl] key.
   ➡ An overview opens with all active views and editors.
2. Continue pressing the [Ctrl] key and select a window using the arrow keys.
3. Release the [Ctrl] key.
   ➡ The selected view or editor is activated.

See also

- Chapter 7.2.5 Changing the Window Layout on page 19
- Chapter 7.2.6 Resizing Windows on page 20
- Chapter 7.2.7 Auto-Hiding Windows on page 20

8 Creating and configuring a project

8.1 What is a project?

A project contains objects and properties that are required a control program or a control configuration.

Templates

To create projects, templates are provided, containing pre-defined objects, depending on the target device. These templates can be selected when creating a new project.

A project is stored as file in the file system. Optionally, the project can be bundled together with other project-relevant files and information as file via a project archive.

Project properties

Apart from the objects, take the following properties into account:

- In den “Project settings” and “Project information”, the basic configuration and information about the project is contained.
- Each project contains information which I/O Engineering version has been used to create the project. When opening the project in a different version, I/O Engineering refers to possible or necessary updates regarding the required device description files.

Working with projects

Take the following into account when working with projects:

- Projects can be compared, exported, imported and a corresponding documentation can be generated.
- A project can be protected against modifications and against reading out of data.
- Use a user management to specifically access the projects as well as individual objects in the project.

8.2 Creating a new project

I/O Engineering provides a wizard to create projects.

Start the wizard to create a project:

1. Open I/O Engineering and start the wizard to create a project via the menu *“File → New Project...”*
   ➡ The “New Project” dialog opens.

2. In the dialog, select the project properties. The following sections are available in the dialog:

“Categories”	• “Projects”: Category to configure a target device.
“Templates”	Select a template for implementation. The templates depend on the category and provide preconfigured elements.
“Name”	Project name
“Location”	Directory in the file system in which the project is stored

Select the category, the template and the file system directory in which the project is stored.

Assign a name to the project.

Confirm the dialog with “OK”

➡ Depending on the selected template, the dialog for more project settings is opened.

8.2.1 Creating a project on the basis of a template for ctrlX CORE Creating devices

I/O Engineering provides released project templates for ctrlX CORE target devices.

The following example guides through the different steps to create a new project for the “ctrlX CORE” target system.

Steps to create a project for the target system “ctrlX CORE”**:**

1. Open I/O Engineering and start the wizard to create a project via the menu *“File → New Project...”*
   ➡ The “New Project” dialog opens
2. In the dialog, select the “ctrlX CORE” project template.
3. Select the project name an the storage location and confirm the dialog with “OK”.
   ➡ I/O Engineering automatically creates the project with the device node in the device tree.

8.2.2 Creating an empty project

To create an empty project, use the “Empty project” project template..

Steps to create an “empty” project:

1. Open I/O Engineering and start the wizard to create a project via the menu *“File → New Project...”*
2. Select the “Empty project” template.
   Confirm the dialog with “OK”.
   ➡ **I/O Engineering** automatically creates an empty project.
   Add and configure your elements via the context menu of the project.

Related documentation

- ➥Chapter 14.2 Device tree and device editor on page 45
- ➥Chapter 'Project settings' dialog - 'Security' on page 178

8.3 Configuring project

8.3.1 Retrieving and editing project information

You can use the “Project information” object to retrieve information about your project and the relevant file, and edit certain information.

The object contains information on

- File attributes
- Meta information, such as manufacturer, title, or author
- Properties with key
- Statistics
- Licensing

I/O Engineering saves the project information as an object within the project. When transferring a project to another system, the “Project information” object is also transferred. A project archive is not required.

By using property keys, the project information is externally accessed via function blocks. For a library project, additional information about licensing can be queried.

Editing meta information

1. Select *“Project → Project information”*.
   ➡ The “Project information” dialog opens.
2. Select the “Summary” tab.
3. Specify your data in the input fields (example: `0.0.0.1` in the “Version” input field).
   ➡ I/O Engineering creates a property with a key for each given value and manages them on the “Properties” tab. For a library project, I/O Engineering continues to use the properties and sorts the library repository for the properties.

   When selecting the option for I/O Engineering to create a function block for these properties, access to properties is executed programmatically.

8.3.2 Selecting project settings

Configure the settings affecting the behavior of I/O Engineering and the behavior of some editors in the“Project settings” object. The settings are valid across the project and are immediately applied to active editors. You can also access the dialogs of the object with the *“Project → Project settings”* command.

I/O Engineering directly saves the project settings as object in the project. When transferring a project to another system, the “Project settings” object is also transferred. A project archive is not required.

Also refer to

- Chapter Command 'Project settings' on page 141
- Chapter 'Project settings' dialog - 'Users and groups' on page 175
- Chapter Dialog 'Project Settings' - 'Page Setup' on page 177
- Chapter 'Project settings' dialog - 'Security' on page 178

8.4 Retrieving and editing project information

You can use the “Project information” object to retrieve information about your project and the relevant file, and edit certain information.

The object contains information on

- File attributes
- Meta information, such as manufacturer, title, or author
- Properties with key
- Statistics
- Licensing

I/O Engineering saves the project information as an object within the project. When transferring a project to another system, the “Project information” object is also transferred. A project archive is not required.

By using property keys, the project information is externally accessed via function blocks. For a library project, additional information about licensing can be queried.

Editing meta information

1. Select *“Project → Project information”*.
   ➡ The “Project information” dialog opens.
2. Select the “Summary” tab.
3. Specify your data in the input fields (example: `0.0.0.1` in the “Version” input field).
   ➡ I/O Engineering creates a property with a key for each given value and manages them on the “Properties” tab. For a library project, I/O Engineering continues to use the properties and sorts the library repository for the properties.

   When selecting the option for I/O Engineering to create a function block for these properties, access to properties is executed programmatically.

9 Exporting and importing projects

9.1 Basic information

Export and import functions are available for exchanging data from I/O Engineering projects with other programs.

An exchange of I/O Engineering projects between other I/O Engineering development systems is done by a copy of the project file (*.project) or the project archive (*.projectarchive).

Also refer to

- ➦ Exporting and Importing Projects

9.2 Exporting and Importing Projects

I/O Engineering provides commands for exporting and importing objects from and to a file. There are two ways to do this:

- Export to and import from a I/O Engineering XML file (*.export)
  This format is fully compatible with the I/O Engineering project format. The objects are stored in a machine-parsable XML format.
- Export to and import from an XML file in PLCopen format (*.xml)
  You can use this format for exchanging information between programs, for example program editors or documentation tools. PLCopen XML defined a subset of elements that I/O Engineering recognizes. Therefore, 100% compatibility cannot be guaranteed.

Exporting Projects

Prerequisite: A project is opened in I/O Engineering.

1. Click *“Project → Export”* or *“Project → Export PLCopenXML”*.
2. Select the export objects in the “Export” dialog or “Export PLCopenXML” dialog.
3. Click on “OK”.
4. Specify the file name and location and click on“Save”.

Importing Projects

Prerequisite: A project is opened in I/O Engineering.

1. Click *“Project → Import”* or *“Project → Import PLCopenXML”*.
2. Select the import objects in the “Import” dialog or “Import PLCopenXML” dialog.
   ➡ A dialog opens and displays a tree structure of objects that can be inserted in this position.
3. Select the object in the object tree under which the objects to be imported are to be inserted.
4. Select the objects and click “OK”.
   ➡ The objects are added to the existing object tree.

Also refer to

- Chapter Command 'Export' on page 148
- Chapter 'Import...' command on page 148
- Chapter Command 'Import PLCopenXML' on page 149
- Chapter Command 'Export PLCopenXML' on page 149
- Chapter Dialog 'Options' - 'PLCopenXML' on page 184

10 Comparing Projects

10.1 Compare project – Basic information

You can compare the currently open project with another project – a reference project. The differences in contents, properties, or access rights are detected and shown in a comparison view.

With the command *“Project → Compare”* opens the “Project Compare” dialog where you can configure and start the comparison. Subsequently, the result is displayed in the compare view “Compare view - Differences”, where the objects are lined up side by side in a tree structure. Objects that indicate differences from the respective reference object are identified by colors and symbols. This is how to detect whether or not the contents, properties, or access rights are different.

In case of differences in content, you can additionally open the detailed compare view “Project comparison - <Object name> Differences” to zoom into the object. In the detail compare view, the source code of the object is compared to that of the reference object. Identified differences are marked. Previously opened views are not closed. Besides the project compare view, any number of compare views can be open and read at the object level.

To resolve the determined differences, decide whether the state from the reference project should be applied to the current project. Applying from the current project to the reference project is not possible. Enable the transfer in the active compare view, for example, different lines or blocks of code using

either the , or the command. These positions are highlighted in yellow. Make sure that any other open compare views are inactive (write-protected, read-only). therefore, you can activate differences to be accepted in exactly one comparison view only. If supported by the editor, you can add a third compare view. This third view shows the result of the resolution actions for the differences.

When exiting the active compare view, if you confirm that the differences that are activated for acceptance are actually accepted into the current project, then the current project is modified.

In order to exit the project comparison completely, close the project comparison view.

Also refer to

- Chapter 'Compare' command on page 143

10.2 Open detailed compare view

The following description refers to a project comparison in the context of PLC programming, but is also applicable to the comparison of device configurations or fieldbus configurations.

Prerequisite: For example, a user changed the code in a POU of the current project. You have performed the project comparison by clicking on *"Project → Compare"*. The project compare view shows this POU highlighted in red in the opposing project trees.

1. Double-click the line of the aligned POU versions.
   - The compare view switches to the detailed compare view of the POU. The modified code lines are highlighted in gray and written in red.
2. Click on .
   - Code lines with changes (red) are extended by two lines: an line with insert (left, green) and a line with delete (right, blue).
3. Click again.
   - The code line is marked again as modified.
4. Move the mouse pointer to the code line marked as modified and click "Accept single".
   - The code line from the reference project is activated for acceptance into the current project.
5. If available, click on to add a third pane.
   - Below the two panes that contrast the code of the current object and the reference object, a third pane is shown. It shows the result of the actions you have to make to resolve the differences between the current object and the reference object. For these actions, additional buttons are available in the taskbar of the detailed compare view. For further information, see:

     'Compare' command
6. Click on .
   - The project compare view of the entire project is displayed. It is write-protected (read-only) to prevent you from activating differences for acceptance. The link highlighted in yellow above the tree view also indicates this.

7. Click on the link “The project compare view is read-only because there are unconfirmed changes in another view. Click here to switch to the modified view.”.
   ➡ The detailed compare view opens again. The unconfirmed changes are highlighted in yellow.
8. In the view tab, click ✕ and confirm that the changes are to be saved.
   ➡ The detail project view is closed and the POU is overwritten. Now it corresponds to the POU of the reference project. The project view is active again, so you can continue working in the project comparison.

If you do not click on the link and instead close the editor of the project compare view by clicking on ✕, you can also confirm the transfer of the changes to the current project. The detailed changes are applied and the project comparison is closed.

Also refer to

- 'Compare' command
- Creating compare views

10.3 Creating compare views

Prerequisite: You have made changes in your current project and want to compare it with the last saved version, for example. In the meantime, you have, for example, added more programming blocks, removed a POU, changed some code lines in function blocks or the object properties.

1. Click on *“Project → Compare”*.
   ➡ The “Project comparison” dialog is displayed.
2. Specify the path of the reference project, for example, the path of the last saved version of your current project.
3. The comparison option “Ignore spaces” is to remain enabled.
4. Click on “OK”.
   ➡ I/O Engineering opens the comparison view. Title: “Project comparison - Differences” The device trees of the current project and the reference project are compared and the changed objects are highlighted.
5. Select an object highlighted in blue in the reference project tree (right). This object is no longer included in the current project.
   Click on “Accept single”.
   ➡ I/O Engineering adds the object in the tree of the current project (left). The line is highlighted in yellow. The symbol appears in the middle column.
6. Select an object marked in green in the tree of the current project (left). This object is not included in the reference project.
   Click on “Accept single”.
   ➡ I/O Engineering removes the object from the tree of the current project again (left). The line is highlighted in yellow. The symbol appears in the middle column.
7. If I/O Engineering has detected changes in the content for an object that is included in the current project and in the reference project, this is indicated by red text. Double-click on the object to switch to the detailed comparison view of the object.
8. Close the comparison view and select whether or not to save the changes with“Yes”.
   ➡ The changes are implemented in the project.

Also refer to

- ➦Chapter 'Compare' command on page 143
- ➦Chapter 10.2 Open detailed compare view on page 26

11 Protecting and saving the project

11.1 Write and access protection – Basic information

You can protect a project from accidental modification via access and write protection. You can additionally provide it with a read protection (know-how protection).

Write protection:

To apply simple write protection to the entire project, there are the following options:

- When opening the project, select the “Open read-only” option.
- In the “Project information”, set the “Released” status.
- Activate the “read-only" option in the properties of the project file in the local file system.

To protect only certain objects in a project from modifications, or to grant access only to certain users, use user and access rights management (see below). Some target devices also support user and rights management. Thus, access from I/O Engineering to objects and files on the target device might be restricted.

However, write protection and access protection do not serve as know-how protection of the function blocks! Both I/O Engineering, Automation platform plug-ins, and persons with knowledge of the project file format can view or modify blocks created with I/O Engineering.

Know-how protection:

A project is know-how-protected by encrypting the project file. Either by using a project password or a Codesys security key (dongle) or with the help of a certificate. Protection via the key or certificate is recommended as in this case there is no need to share sensible data between authorized users. The desired type of project encryption is activated in the project settings.

Know-how protection of a library is achieved by providing it as a target system-independent "protected library"(`*.compiled-library`, `*.compiled-library-v3`). In this format, the library file no longer contains source code, but only encrypted precompile context. The compiler is still able to interpret this data. Whether access is still possible by other components or additional plug-ins depends on their functionality and has to be taken into consideration on a case-by-case basis. The protection can be further increased by signing.

Related topics

➦Chapter 11.3 User administration and password manager on page 29

➦Chapter 11.8 Protecting projects using a dongle on page 33

➦Chapter 11.7 Assigning passwords on page 33

➦ Chapter 11.10 Protecting Objects in the Project by Access Rights on page 34

11.2 Project encryption with certificates

In I/O Engineering, projects and applications can be encrypted with certificates and signed in order to protect them from unauthorized access.

To do this, you can configure specific security settings for each individual user profile. These settings are always used automatically when the user works with I/O Engineering projects. Therefore, they do not have to be redone for each project. The general configuration of the security features for a user profile is done in the “Security screen” view of I/O Engineering. See the individual instructions below.

You can also encrypt a project file or an application for download or online change directly with a certificate:

- User-independent encryption for the current project is configured in the “Security” category of the “Project settings” dialog.
- User-independent encryption of the application is configured in the “Properties” dialog of the application object.

NOTICE	When you encrypt a project, an application, or online code with a certificate, you will always require the certificate with a private key in order to open the object again.

If the add-on product CODESYS Security Agent is installed, the “Security screen” view provides an additional “Devices” tab. It allows the configuration of certificates for encrypted communication with controls.

Certificates, Windows Certificate Store

All available certificates are located in the Windows Certificate Store (“certmgr”) on your computer. There are two types of keys:

- Certificates with private keys
  - for file decryption
  - for digital signatures
- Certificates with public keys
  - for file encryption
  - for verifying digital signatures

Usually, the local Windows Certificate Store is filled with certificates by the computer's IT administrator. Certificates are either created using special tools or requested from a trusted certification authority (CA).

If you receive a certificate file that you need to install yourself in the Windows Certificate Store, then double-click the file in the store directory. Depending on the type (certificate with private or public key only), the appropriate import wizard will appear.

Also refer to

- Chapter 19.2 General information on page 89
- Chapter 11.12 Encrypting the project with a certificate on page 37

11.3 User administration and password manager

User accounts with different rights can be managed in I/O Engineering. For each account you can define the actions with which the user can access a project object.

You configure the user administration in the “project settings” in the category “Users and groups”.

Before creating users and groups, please note the following:

- Rights can only be assigned to user groups. Therefore, you must assign each user to a group.
- There is automatically always a group `Everyone` and by default every user and every other group is initially a member of this group. Thus each user account is automatically equipped with at least the defined standard rights.

  You cannot delete the group `Everyone`, you can only rename it, and you cannot remove members from this group.

Caution: By default, not `everyone` has the right to change the current user, group and rights configuration!

- Automatically there is always a group `Owner`, which contains a user `Owner`. As of V3.5, in a new project initially only the `owner` has the right to change the current user, group and rights configuration! Hence, only `Owner` can assign this right to another group.
  Initially, the `owner` can log in with the user name `owner` and a blank password.
  You can add further users to the group `Owner` or remove users from it, but at least one member must be retained. Like `Everyone`, you cannot delete the group `Owner` and it always possesses all access rights. This prevents a project from being rendered unusable by denying all access rights to all groups.
  You can rename both the group `Owner` and the user `Owner`.
- If the programming system or a project is restarted, no user is initially logged in to the project. However, the user can then login via a certain user account with user name and password in order to obtain the access rights defined for the account.
- Each project has its own user management! For example, to be assigned certain access rights to a library included in the project, the user has to explicitly log in to the library project.
  Users and groups defined in different projects are not the same, even if they have the same names.
- A user management in a project only recommended if it is connected to the corresponding rights assignment for the access to project and objects. The general rights management for a project is realized in the “Rights” dialog of the “User administration”. You can also change the access rights to an individual project object on the “Access control” tab of the object's “properties”.
- For logging in and logging out to/from a project as a defined user, there are standard menu commands under *“Project → User administration”*. A password manager permits the management of the access data on your computer.

As of V3.5, initially only the `owner` has the right to change the current user, group and rights configuration in a new project! Hence, only `Owner` can assign this right to another group.

NOTICE	I/O Engineering stores the user passwords inaccessibly. If you forget a password, the user account becomes unusable. If you forget the `owner password`, the whole project may become unusable!

Password manager

The password manager allows you to save the access records that you enter during the access data for projects. It is accessible via a button in the login dialog and offers fast access to the access data currently required. This can be useful for example when working simultaneously with multiple library projects protected by different passwords.

The password manager itself is protected by an individual master password. If you want to use the password manager for the first time, I/O Engineering prompts you to define this password in the password manager configuration dialog. I/O Engineering remembers the master password until you end the current I/O Engineering session. You will need to enter the password whenever you want to log into the password manager for the first time during a new session or after you have changed it.

Related topics

➦ Chapter 11.10 Protecting Objects in the Project by Access Rights on page 34
➦ Chapter 11.11 Logging in via user account and password manager on page 35

11.4 Rights management

Rights management for access to a project and objects in a project is necessary in order to make a user management meaningful.

Configure the general rights management for a project in the “Rights” editor of the “User management”. The access rights to an individual project object can also be changed on the “Access control” tab of the “Properties” dialog of the object.

Before assigning rights, please observe the following:

- In a new project, I/O Engineering sets all rights to perform actions to objects with the **default value** "allowed" (default right). The only exception is the right to change the current configurations for users, groups and rights. Initially only the 'Owner' group has this right.
- If you are member of a group that is permitted to change rights, you can do this at any time for each right when working further on a project. You change a right by switching between 'allowed' and 'forbidden' or by resetting to the default.

Also refer to

- ➦ Chapter 11.9 Setting up the user administration on page 34
- ➦ Chapter 11.10 Protecting Objects in the Project by Access Rights on page 34

11.5 Project repository – Saving

Provide the project file with the desired protection before saving it in the file system; see above. For a read-only project file you are given various options so that you can still save the file, depending on the type of write protection.

If the project is to be opened later in an older I/O Engineering version, it makes sense to save the project for exactly this version (file type), since I/O Engineering also informs you immediately about any data loss in the course of this storage action.

If you want to save library projects, follow the rules for creating libraries. Also consider installing a library directly into a library repository.

If you want to continue using a project on another computer, it is recommended to save not only the project file, but to create a project archive from all relevant additional files.

You can make a setting so that a backup copy of this project is created each time the project is saved. In addition you can configure so that projects are generally automatically saved at certain time intervals.

If you want to keep projects in a source code management, consider the corresponding add-ons for I/O Engineering. For example, the link to SVN is supported.

Related topics

➦ Chapter ‘Options’ dialog - ‘Load and save' on page 183
➦ Chapter 11.13 Saving the project on page 39
➦ Chapter 11.13 Saving the project on page 39
➦ Chapter 11.14 Saving/sending the project archive on page 40

➦ Chapter 11.15 Linking a project to the source control system on page 41

11.6 Setting up write protection

A project can be protected against inadvertent changes by means of access and write protection. In addition, however, it can also be provided with read protection (know-how protection). You have the following options.

Open the project with write protection

Prerequisite: a project is not opened.

1. Click on *“File → Open Project”*.
   ➡ The “Open Project” dialog opens.
2. Select the project.
3. At “Open” button, click the arrow button ▾ and select “Open read-only” from the menu.
   ➡ I/O Engineering opens the project. At the top right in the main window a line appears “'The project cannot not be saved...'”. Now choose one of the provided options if you want to save the project file.

Providing projects with the attribute 'Released'

Prerequisite: Project is open.

1. Select *“Project → Project information”* then the“Summary” tab.
2. Activate the option “Released”, confirm with “OK”.
3. Save the project, for example with [Ctrl]+[S].
4. Open the project again by clicking on *“File → Open Project”*.
   ➡ I/O Engineering opens the project. At the top right in the main window a line appears “'The project cannot not be saved...'”. You can now use the option offered to remove the “Released” status again directly if you want to save the project file.

Providing a project in the file system with the property 'Read-only'

1. Provide the project file in its local file system with the property attribute 'Read-only'.
   ➡ If you had already opened the project and you now attempt to save it under the same name, a dialog appears informing you about the existent write protection. This dialog provides you with the following options:

   The “Save as...” button allows to save the project under a different name or path.

   You can deliberately save the project under the same name and path and thus overwrite the existing version in the file system using the button “Overwrite”.

   You can abort the saving procedure using the “Cancel” button, for example to remove the write protection on the disk.

   When opening the project again, the line `The project cannot not be saved…` is displayed in the upper right of the main window.

Also refer to

- ➦ Chapter 11.1 Write and access protection – Basic information on page 28

11.7 Assigning passwords

Prerequisite: The project is open.

1. Select *"Project → Project settings"* then "Security" category.
   ➡ "Project setting/security" dialog opens.
2. Select the "Encryption" option.
   ➡ The option fields "Password", "Dongle", and "Certificates" are selectable.
3. Select the "Password" option.
   ➡ The input fields for the encryption password appear.
4. Enter the encryption password in the "New Password" input field.
5. Enter the encryption password for confirmation in the input field "Confirm new password".
6. Click on "OK".
   ➡ I/O Engineering saves the encryption password for the project.

⚠ CAUTION	If you no longer know the encryption password, you can no longer open or restore the project!

Also refer to

- ➥ Chapter 'Project settings' dialog - 'Security' on page 178

11.8 Protecting projects using a dongle

Prerequisite: The project is open and a Codesys Security Key (dongle) is connected to your computer.

1. Select *"Project → Project settings"*, then the "Security" category.
   ➡ The "Project settings/security" dialog opens.
2. Select the "Encryption" option.
   ➡ The option fields "Password", "Dongle", and "Certificates" are selectable.
3. Select the "Dongle" option.
   ➡ The dialog with the selection list "Registered dongles" and the buttons "Add","Remove","Comment","Blink" opens.
4. Click on "Add".
   ➡ The "Add registered dongle" dialog opens.
5. Select the Codesys security key (dongle) from the "Dongle" selection list and optionally enter a comment.
6. Click on "OK".
   ➡ The added dongle is listed in the list "Registered dongles".
7. Click on "OK".
   ➡ The dongle is now registered for the project.

NOTICE	If the Codesys security key registered for the project is lost, it is not possible to open the project or to restore it.

Also refer to

- ➥ Chapter 'Project settings' dialog - 'Security' on page 178

11.9 Setting up the user administration

This is about user administration for a I/O Engineering project file. Visualizations and devices can have their own user management.

The following guide describes how you can adapt the user administration for the first time in a project. It deals with the definition of a user and a group to which he belongs.

Prerequisite: the project for which the user administration is to be set up is opened. There is no adapted user configuration yet.

1. Select *“Project settings → Users and groups”*, “Users” tab. The user `Owner` is already created by default.
2. Click on “Add”.
   ➡ The “Add user” dialog is displayed.
3. Enter a login name, for example 'Dev1', and a password. Leave the option “Activated” activated. Click on “OK”.
   ➡ Upon the first creation of a group, I/O Engineering now prompts you to authenticate yourself as authorized for this action.
   In this case, enter 'Owner' as the “current user”. Do not enter a “password”, just click “OK”.
   The user `Dev1` appears in the list and is automatically a `member of the group 'Everyone'`.
4. Change to the tab “Groups”, in order to add the user to a new group.
   ➡ The groups `Everyone` and `Owner` have already been created.
5. Click on “Add” to access the “Add group” dialog.
6. Specify at least one name for the new group, for example 'Developers'. In the “Members” field, check the “User 'Dev1'” entry. Click on “OK”.
   ➡ The group “Developers” now appears with `has user member 'Dev1'`.
7. Switch to the “Users” tab.
   ➡ The user “Dev1” now appears as a member of the groups 'Everyone' and 'Developers'.

Use the “Export/import functionality” in the “Project settings” dialog in the “Users and groups” category, to apply the user administration from another project.

Also refer to

- Chapter 11.3 User administration and password manager on page 29
- Chapter 'Project settings' dialog - 'Users and groups' on page 175
- Chapter 'Project settings' dialog - 'Users and groups' on page 175

11.10 Protecting Objects in the Project by Access Rights

Protection of individual objects by setting access rights in the “Rights” **editor**

1. Select *“Project → User Management → Rights”*
   ➡ The window of the “Rights” editor opens. On the left you can see the action categories, on the right the currently existing user groups.
2. Expand the relevant action category and below it the action for which you wish to change a right.
3. Select the goal of the action in the “Actions” window. In the “Rights” window, select the group for which you would like to change the right. Multiple selection is possible.
   ➡ The buttons in the symbol bar are active.

4. Click on the appropriate button in order to change the right of the group for the action on the target object.
   ➡ I/O Engineering updates the symbol in front of the group according to the new right. The right is immediately effective.

See also
- Chapter 21.4.2 Dialog 'Permissions' on page 162

Protection of individual objects by setting access rights in the object properties

Here you can configure whether the members of a group have the right to view, edit or remove the object and to add/remove child objects to/from the object.

1. Select the object in the navigator tree.
2. In the context menu, select the command *"Properties"* and in the dialog box select the category "Access Control".
3. In the table under "Groups, Actions and Permissions", double-click on the symbol of the right that you wish to change.
   ➡ A selection list of the possible rights appears: "Grant", "Deny", "Clear".
4. Select the desired right and click on "Accept" or "OK".
   ➡ The right is immediately effective for the action and group. The symbol changes accordingly.

See also
- Chapter Dialog 'Properties' - 'Access Control' on page 174

11.11 Logging in via user account and password manager

Logging in to a project without using the password manager functions

Prerequisites
- A project is opened
- You have the necessary access data for the project

The following action instruction describes how to log in to a project as a defined user in order to edit it.

1. Select *"Project → User administration → Log in user..."*.
   ➡ The dialog "Login" opens.
2. Select the project file from "Project/library" and enter the required access data "User name" and "Password".
3. Login with "OK".
   ➡ If another user is already logged in, this user will automatically be logged out by the new login.

Setting a master password for the password manager

Prerequisite: A project is opened. You have opened the "Log In" dialog to log in to a project as a defined user. You want to use the password manager to save the access data.

1. Select *"Project → User administration → Log in user..."*.
2. In the "Login" dialog, click on  .
   ➡ If you are working for the first time with the password manager, the dialog "Password manager configuration" opens.

3. Enter a character string as the future master password. Confirm it in the second line and click “OK”.
   ➡ I/O Engineering remembers the master password until you end the current I/O Engineering session. You will need to enter this password whenever you want to log into the password manager for the first time during a new session or after you have changed it.

NOTICE	If you have forgotten your master password, you no longer have any possibility to access the access data already saved! In this case you can only reset the password manager. After that you must start again to save passwords in the manager!

Saving credentials in the password manager

Prerequisites

- A project is opened
- You have the necessary access data for the project
- The access data are not yet stored in the password manager

1. Select *“Project → User administration → Log in user...”* to get the “Login” dialog.
2. Select the project file from “Project/library”.
3. Enter the user name and password for the project.
4. Click on the button.
   ➡ If you are working for the first time with the password manager, you will be requested to define a master password. Refer to the "Set master password for password manager" tutorial above.
   Upon entering the password manager in this I/O Engineering session for the first time, you are prompted to enter the master password.
5. Enter the master password when requested to do so.
   ➡ The password manager menu appears.
6. Select the option “Save access data locally on this computer”.
   ➡ The login takes place. The data are saved in the password manager.

Getting the credentials from the password manager

Prerequisites

- A project is opened
- You have the necessary access data for the project
- The access data are already stored in the password manager

1. Select the command *“Project → User administration → Log in user...”* to get the “Login” dialog.
2. Click on the button.
   ➡ If you are working for the first time with the password manager, you will be requested to define a master password. Refer to the "Set master password for password manager" tutorial above.
   Upon entering the password manager in this I/O Engineering session for the first time, you are prompted to enter the master password.
3. Enter the master password when requested to do so.
   ➡ The password manager menu appears.
4. Select the appropriate entry “Use saved access data for <user name>”.

➡ The login takes place automatically with the data read from the password manager.

Opening the password manager, changing the master password

Prerequisite: A project is opened. You want to open the password manager to view the entries, edit them or change the master password. You have already logged in once with the master password.

1. Select *“Project → User administration → Log in user...”* to get the “Login” dialog.
2. Click on .
   Select “Open password manager”.
   ➡ The password manager window opens.
3. Click “Change Master Password” and make the change.

Logging out from the project

Prerequisite: A project is opened. A user is logged in, which is recognizable by a name entry in the field “Current user” in the status bar.

1. Select *“Project → User management → Logout user...”*. Alternatively, double-click on the field “Current user” in the status bar.
   ➡ If the user is logged in to only one project, he will now be logged out without further interaction. “(nobody)” is displayed again in the field “Current user” in the status bar
   If the user is logged in to several projects, the dialog “Logout” opens. Specifically select the project for which the user is to be logged off.

Also refer to

- Chapter 11.3 User administration and password manager on page 29
- Chapter Command 'User management' – 'Log in User' on page 150
- Chapter 11.9 Setting up the user administration on page 34

11.12 Encrypting the project with a certificate

Configuring a certificate for project file decryption for a user profile

If a project is encrypted with a certificate, this certificate is required for decryption in order to open the project. You can assign such a certificate to specific user profiles. Select the certificate from the Windows Certificate Store on the “Users” tab of the “Security screen”.

1. Double-click on in the status bar or select the command *“View → Security screen”*.
   ➡ The “Security screen” view opens.
2. In the “Users” tab, select the user profile whose communication is to be encrypted. By default, the user profile you used to log in to Windows on your computer is specified. However, you can also create a new user profile via .
3. In the “Project file decryption” section, click on the button.
   ➡ The “Certificate selection” dialog opens.
4. In the “Available certificates” listing in the local Windows Certificate Store, select a certificate with a private key. Certificates with a private key are marked with the symbol .
5. Click on the button

6. The certificate will be added in the upper window of the dialog.
7. Click on “OK” to confirm the entries.
   ➡ The selected certificate is displayed in the “Security screen” in the “Project file decryption” section.

Encrypting the project with a certificate

A project encrypted with a certificate in connection with a user administration allows for a targeted restricted access to the project.

1. Select the command *“Project → Project settings”* and then “Security”category.
   ➡ The “Project settings/security” dialog opens.
2. Select the “Encryption” option.
   ➡ The “Password”, “Dongle” and “Certificates ” radio buttons are available.
3. Select the “Certificates” option.
   ➡ The list of certificates available for the project encryption appears in the lower part of the dialog. If no certificate has been entered yet, select a suitable certificate in the “Certificate selection” dialog after clicking on and return to the “Project settings” dialog. The certificate is now registered for encryption. Now the project can be edited only on computers of users who also have the file decryption certificate.

Deleting the certificate in user profile

You delete the certificate in the “Security screen” view, either directly on the “Users” tab or in the “Certificate selection” dialog. In the other dialog, the certificate is also deleted.

- “Security screen” dialog, “User” tab, “Digital signature” or “Project data decryption”: Select a certificate and click on
- “Certificate selection” dialog: In the “Security screen” dialog, “Users” tab, click on. In the “Certificate selection” dialog, select the certificate to be deleted in the upper field and click on.

Configuring a certificate for digital signature for a user profile

To ensure that the project is not only encrypted with a certificate, but that its authorship and integrity can be verified, add a signature to the project:

1. Double-click on in the status bar or select the command *“View → Security screen”*.
   ➡ The “Security screen” view opens.
2. On the “Users” tab, select the user profile for which you want to create the digital signature. By default, the user profile you used to log in to Windows on your computer is specified. However, you can also create a new user profile via .
3. In the “Digital Signature” section, click on the button .
   ➡ The “Certificate selection” dialog opens.
4. In the “Available certificates” listing in the local Windows Certificate Store, select a certificate with a private key. Certificates with a private key are marked with the symbol.
5. Click on the button.
   ➡ The certificate will be added in the upper window of the dialog.

6. Click on “OK” to confirm the entries.
   ➡ The selected certificate is displayed in the “Security screen” in the “Digital signature” section.

Also refer to

- Chapter 11.2 Project encryption with certificates on page 28
- Chapter 'Project settings' dialog - 'Security' on page 178

11.13 Saving the project

Saving a project under the same name

Prerequisite: The project is open. The project file is not write-protected.

Select *“File → Save”*.
➡ I/O Engineering saves the project file with the current project name, which appears in the title bar of the main window. If the project has been changed since it was last saved, then the project name is provided with an asterisk. A backup copy is created if it is set in the I/O Engineering options in the “Load and Save” category.

Saving a project under a different name or format

Prerequisite: The project is open.

1. Select *“File → Save as...”*.
   ➡ The “Save project” dialog is displayed.
2. Select a storage location in the file system and the desired “file type” and storage version. If you want to open the project later in an older version, then it makes sense to save for precisely this version, as you will then be informed immediately in the message view about possible data loss.
   ➡ If the project file is not write protected, then I/O Engineering saves it in the selected path. Otherwise you will be informed how to proceed.
3. If the current project contains add-ons that are not included in the desired memory version, the “Extend profile” dialog opens.
4. Select the add-ons to extend the memory profile in order for the add-on data to be saved.
5. If you want to save the storage profile permanently click on“Save profile” and enter the desired name in the “Enter profile name” dialog.
6. In the “Expand profile” dialog, select the “Use saved profile” option and confirm the dialog with “Yes”.
   ➡ I/O Engineering saves the project with the saved profile.

Saving a read-only project

Prerequisite: A read-only project is opened.

Select *“File → Save”*.

If the write protection was assigned in I/O Engineering, then it will be displayed by a line in the top right corner of the main window. Depending on the current situation you will be offered one or more of the following actions so that you can still save the project:

- “Save project under a different file name on the disk”: Is always displays and leads to the “Save file” dialog as with the “Save file as...” command.
- “Exit read-only mode”: Is displayed if the option“Open read-only” was selected when opening the project.

- “Remove the write-protect attribute from the project on disk”: Is displayed if the project file was write-protected in the local file system at the time it was opened.
- “Remove the "Released" flag in the project information”: Is only displayed if this attribute is currently set.

If the write protection was assigned outside of I/O Engineering in the properties of the project file in the file system, you will be offered the following options when you attempt to save under the same name and path:

- “Save as...”: By using the “save project as...” command, you can save the project under a different name.
- “Overwrite”: The write protection is removed from the project file and it is saved under the existing name.

1. Click the line in the top right corner of the main window that indicates the write protection.
   ➧ The current options with which you can still save the project appear in a selection menu.
2. Select one of the options offered and perform any necessary actions.
3. Select the command *“File → Save”* or *“File → Save as...”*
   ➧ The project can be saved.

Saving of the project automatically; creating a backup copy

Prerequisite: The project is open.

1. Select *“Tools → Options”*, “Load and save” category.
   ➧ The “Load and Save” dialog opens.
2. Activate the “Create backup copy” option.
3. Enable the option “Save automatically every ... minute(s)” and select a time interval.
4. Click on “OK” to close the “Options” dialog.
   ➧ Upon each time the project is saved, I/O Engineering a backup copy `<Project name>.backup` is additionally created.

   I/O Engineering automatically saves the project at the specified time interval to a file `<Project name>.autosave` in the project directory. If you open the project again after the development system was closed irregularly, then this file will be offered to you as an alternative to the file last saved by the user.

Also refer to

- ➥Chapter ‘Options’ dialog - ‘Load and save' on page 183

11.14 Saving/sending the project archive

You can configure a project archive and then save it in the file system or send it directly in an e-mail.

To send, follow the guide below as far as point 9. Click on the “Send” button instead of on“Save to” to open the default email program directly. Automatically, a new e-mail attached with the project archive file is created.

Prerequisite: a project is opened.

1. Select *“File → Project Archive → Save/Send Archive”*.
   ➧ The “project archive” dialog is displayed.
2. Add a check in the checkbox of each object that should be saved to the archive.

3. If you want to add more files to the archive, click on“Additional files”.
   ➡ The “Additional files” dialog opens.
4. Click on “Add”.
5. Select the files and click “Open”.
   ➡ The files are added to the list of additional files.
6. Click on “OK”.
7. Click on “Comment”.
   ➡ The dialog “Comment” opens.
8. Enter a comment and click “OK”.
9. Click the “Save” button.
10. Select a location and a file name and click “Save”.
    ➡ The project archive is saved in the file directory.

Also refer to

- Chapter Command 'Save project' on page 130
- Chapter Command 'Save project as' on page 130

11.15 Linking a project to the source control system

To link your I/O Engineering projects to a source control system, check the following option:

The CODESYS SVN add-on provides the capability of directly linking to an SVN database. You can get the package at the CODESYS Store and install it with the help of the Package Manager.

Refer to the corresponding help when using CODESYS SVN.

12 Project synchronization

12.1 Introduction

The project synchronization is exclusively available for the ctrlX CORE or ctrlX CORE Virtual target systems.

The “project synchronization” provides the option to save a project in the file system of the engineering PC and to synchronize it with the project on the control by means of definable synchronization settings. The project can also be opened and edited without being connected to the control. As soon as the connection to the control is established again, the two project repositories are compared.

Another advantage of “project synchronization” is that the project can be read directly from the control. This is a useful function if you are not the owner of the project sources. As the project is downloaded to the control, you can connect to the current project on the control, edit the project and upload it to the control again.

If the “project synchronization” is not configured, no project data is stored on the control but only the data required for the control flow is stored. This data can be viewed by opening “Manage app data” in the web interface of the control. They are stored in the configuration data in the io\run... folder.

Functional notes / limitations

If I/O Engineering is run in an VirtualBox environment and the project is saved on a mounted host drive, project synchronization does not work due to the special drive type.

Workaround 1: Use a local directory in the VirtualBox for the project repository.

Workaround 2: Do not mount the host drive in the VirtualBox directly, but share a directory on the host and mount it as a regular network drive in the virtual box.

Status display

In the I/O Engineering status bar (lower right), the status of the project synchronization is visualized via the following icons:

Icon	Meaning
	The synchronization is enabled
	The synchronization is not enabled

Double-click on the icon to open the project synchronization setting dialog.

The project synchronization for the respective project can be enabled or disabled at any time. If the synchronization is deselected and confirmed with “OK”, a prompt is shown if the current synchronization data should be retained on the control or if they are to be removed from the control repository.

12.2 Activation

The “Project synchronization” option can be activated via the “Project synchronization” dialog, see:

➥ Dialog “Project synchronization”

12.3 Options

In the “Project synchronization” dialog, options to store and synchronize project data are available. The dialog is divided into the following sections:

Project storage on PC	
“Name”	Project name
“Location”	Filing of the project data in the file system of the Engineering PC

Synchronization PC and ctrlX CORE (default setting)		Description
☑ **Synchronization of project storages between PC and ctrlX CORE if:**		Enabling/disabling the synchronization function. If active, then under the following subconditions:
	☑ **the project is opened**	Is automatically enabled if the synchronization is active
	☑ **the project is saved on PC**	Synchronization when saving the project
	☑ **a download / OnlineChange is carried out and thus the project is saved**	Synchronization when downloading an application to the control

Synchronization PC and ctrlX CORE (default setting)		Description
	☐ A boot application is created and login data is synchronized	Synchronization occurs when the command “Generate boot application" is executed in the logged-in state .
	☐ the project is closed	Synchronization when closing the project
☐ Synchronize device description files and libraries		Device description files and libraries that are not part of the ctrlX WORKS installation are also saved on the control. This option facilitates direct login to the project, if active.

12.4 Synchronization – Online behavior

To synchronize the project, a connection between the Engineering software and the control is established.

In the next step, the current project data on both sides is compared.

The synchronization cache is used for the comparison, see ➥ Chapter Command “Synchronization cache...” on page 142.

In case of differences between the project repositories, the data transfer is stopped and the dialog “conflict: Project synchronization ...” is opened. In the dialog, the identified differences are listed and options for conflict handling are provided. The provided solutions depend on the conflict and can vary.

Table 1: Conflict handling options

Button	Action
“Use project from ctrlX”	The project currently opened on the Engineering PC is closed. The project data available on the control are applied to a new project on the Engineering PC.
“Use Project from PC”	The project data available on the Engineering PC are transferred to the current configuration on the control.
“Overwrite project on ctrlX”	The project data currently loaded on the control are overwritten by the project data on the Engineering PC.
“Apply ctrlX IP address”	The IP address of the control is applied to the project data on the Engineering PC.
“Open project comparison”	Via the project comparison, the contents of the project data can be compared and matched.
“Cancel synchronization”	The synchronization is canceled. The project data is not synchronized between the Engineering PC and the control. The existing data is retained.
“Continue whithout change”	The suggested conflict solution is dismissed (e.g. applying the IP address of the control). The project data is not synchronized between the Engineering PC and the control. The existing data is retained.

If the addressed control is (temporarily) not available, the project data is saved in a reserved cache ➥ Chapter Dialog “Project synchronization” on page 168 in case of enabled synchronization and is updated on the control once the communication is available again.

If it is detected that the IP address in the project does not match the IP address of the control when opening a synchronized project, the project transfer is stopped and the dialog “Project synchronization” is started. Use the dialog buttons to select whether to apply the IP address of the ctrlX device to the project or whether to continue without change.

13 API for ctrlX I/O Engineering

General information

The API for ctrlX I/O Engineering allows to create and edit projects in ctrlX I/O Engineering based on REST (REpresentational State Transfer)

Optionally, a Client API can be generated in different programming languages for the OpenAPI specification. This allows for an automated project created for commissioning.

Start parameter for use of interface

By default, the ctrlX I/O Engineering software tool automatically generates a communication port via API.

Optionally, a fixed port can be assigned to the API. Starting the software with an additional parameter `--webserverport=<Port number>+`

Please note that a blank space is entered between the storage location path and the parameter.

The ctrlX I/O Engineering API initialization of the standard port can be modified by explicitly calling ctrlX I/O Engineering.

- **Via the Windows input panel**
  <storage location>ctrlX-IO-Engineering.exe --webserverport="*Port number*"
- **Via a link**
  <storage location>\ctrlX-IO-Engineering.exe --webserverport="*Port number*"

The port number is required to start a web server.
The selected port must thus not be assigned by another application.

Documentation of REST commands

The REST API help is provided as interactive documentation.

Opening interactive documentations

In the ctrlX I/O Engineering menu, click on *“Help → API-reference”*

➧ A browser containing interactive documentation opens.
This documentation describes the available commands of the REST API:

- The interactive browser page communicates with the opened ctrlX I/O Engineering
- The default parameter port number is contained in the address line.
  Example: „http://localhost:9003/index.html“

Disabling the IO Engineering API

To start ctrlX I/O Engineering without enabled REST API, please start the software as follows:

<storage location>\ctrlX-IO-Engineering.exe --disablewebserver"

14 Configuring the I/O connection

14.1 Basic information

Use device objects to map hardware to be controlled in a tree structure in your I/O Engineering project. This makes the linking of hardware and application easy to handle.

In the configuration editors of the device objects you can make settings for the communication between I/O Engineering and the controller, and above all for the I/O assignment. The I/O mapping is the linking of the inputs and outputs of the controller with the variables of your application.

Access to control objects at runtime can be device-dependent via an "online user administration". In addition, the communication with the control device depends on the currently valid "Security settings".

Also refer to:

- ➦ Device tree and device editor
- ➦ Mapping the hardware structure in the device tree

14.2 Device tree and device editor

The following descriptions and figures are part of the PLC project engineering in ctrlX PLC Engineering.

However, the general handling within the device tree is also applicable to the configuration of devices, field buses and field bus devices in ctrlX I/O Engineering.

Device tree

In the “Devices” view, also called "device tree", you can compile applications based on target devices. Here, it is possible to map the hardware of one or more control devices or field bus systems, configure the communication with it and assign several applications to each of them.

The root node of the device tree is a symbolic node entry: “<project name>”

Below that, mount the device objects for one or more control devices, also called "target systems". Each device object represents a specific hardware such as a control device, field bus, bus coupler, drive, I/O module or monitor. When inserting the object, an insertion wizard that always offers the matching devices from your local device repository provides help.

If you are already connected to a control network, scan the hardware for existing devices and add them to the current configuration in the device tree.

Certain rules (see below) apply to creating device objects in the device tree, i.e. the mapping of the hardware environment to be controlled. The hierarchical arrangement of application objects and device objects defines the scopes of other objects.

There are programmable devices and purely parameterizable devices. The device type defines the possible insertion position in the device tree and the selection of objects you can insert below the device. Programmable devices automatically receive an additional purely organizational node “PLC logic”

below the device object. Under this node, insert the objects that are needed for programming the device, i.e. the application(s) and, for example, GVLs or text lists.

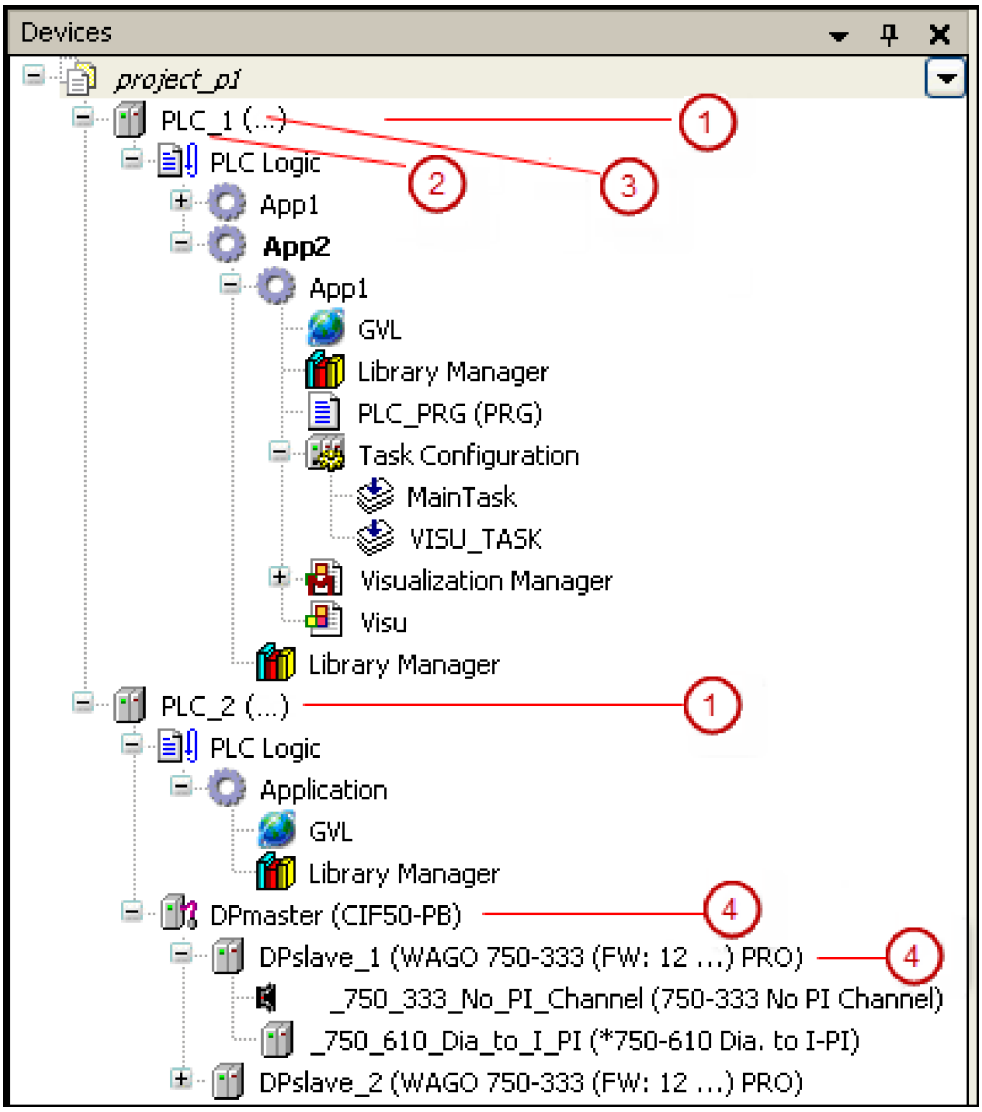
Each device is defined by its device description and has to be installed on the local system to be available for insertion in the device tree. The device description file defines the properties of a device with regard to configurability, programmability and possible connections to other devices.

The “POUs” view contains objects that you can use across the project. Programming objects that are only intended for a specific application have to be inserted below the application object in the “Devices” view (device tree).

Note the possibility of running the active application on a "simulation device", which is automatically available by default within the programming system. Currently, this simulation option exists for the target system ctrlX CORE. In simulation mode, you can also test the online behavior of the application without hardware. Activate the simulation with the command *“Online → Simulation”*.

Furthermore, note the possibility of using the “Online configuration mode” command for an application to connect to the device without having to load this application there first. This is useful for initial commissioning of an I/O system as it allows you to address and test the I/Os in the controller configuration before you have programmed and loaded the actual application.

Example of a device tree:



(1) Programmable device (with applications)
(2) Symbolic device name
(3) Device name, defined in the device description
(4) Purely parameterizable device

A device entry in the device tree consists in each case of a symbol, of the symbolic device name editable in the tree and of the device type (= device name defined in the device description) behind it.

The configuration of a device regarding communication, parameters, I/O image is realized in the dialogs of the device editor. Double-click on the device object to open the editor.

Rules and procedures for arranging and configuring objects in the device tree

- **Insert objects**: To insert a device object, use the “Insert device” or “Attach device” command in the context menu of the device tree. For other objects, use the “Add Object” command. I/O Engineering always provides the objects matching at the currently selected position in the tree for selection. Example: Modules for a PROFIBUS DP slave can only be mounted below a

corresponding slave object, applications only below programmable devices. The selection of device objects additionally depends on which devices are installed in the device repository.

- At the level directly below the root node “<project name>” you can only **mount device objects**. When selecting a different object type such as a text list, I/O Engineeringautomatically inserts it in the “POUs” view, i.e. in the project pool!
- **Insert applications**:
  You can only insert an “Application” object below the “PLC Logic” node, i.e. below a programmable device. All applications have to be named uniquely for each device. Below each application, more objects required for programming, such as POUs, DUTs, GVLs or visualizations can be attached.
  Below each application, you need to insert a task configuration and configure in it the corresponding program calls from application-specific POUs or POU "instances" (from the “POUs” view).
- **Insert devices**: I/O Engineering inserts a device object as a node in the tree. If child nodes are defined in the device description, they are automatically inserted as well. A child node can also represent a programmable device. To order the device objects within the tree from top to bottom: For each level, the programmable devices (PLC logic) are always arranged first, followed by the remaining types, each sorted alphabetically.
- **Update devices**: A device already inserted in the device tree can be replaced by another version of the same device or with a device of a different type“(Update devices”). A configuration tree existing below the device is retained as far as possible.
- **Position objects differently or delete objects**: You can handle the objects with the standard “Cut”, “Copy”, “Paste”, “Delete” commands, or by dragging a selected object with the mouse to another position. When copying objects, the new object is assigned the same name, extended by a consecutive number.
- **Network scan of the current hardware**: By default, creating the control configuration in the device tree is supported by the device editors with a scan functionality: The current hardware environment is searched and the modules found are displayed in a dialog. Subsequently, you can apply the devices directly to the device tree. See the “Search devices” command.

Also refer to

- ➥Chapter 14.3 Mapping the hardware structure in the device tree on page 49
- ➥Chapter Command 'Update device' on page 140

Device tree in online mode

In online mode, an icon in front of a device entry indicates the status of the device.

: PLC is connected, application is running, device is in operation, data is exchanged. The “Update I/Os” option in the “PLC settings” tab can be enabled or disabled.

: PLC is connected and in stop (“STOP”), and the “Update I/Os in stop” option in the “PLC settings” tab is disabled.

: Device does not exchange data; bus error, no configuration, or simulation mode.

: Device runs in demo mode for 30 minutes. After this time, the demo mode expires and the field bus terminates the data exchange.

⚠: Device is configured but not fully operational. No data is exchanged. Example situation: CANopen devices during start-up and in the preoperative state.

⚠: Redundancy mode is active. The field bus master is currently not sending any data as another master is active.

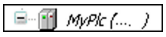
: The device description could not be found in the device repository.

: The device itself is running, but a subordinate device is not running or has a diagnostic message. The subordinate device is not visible due to a collapsed device tree.

: Exclamation mark gray: The device itself is running, but a subordinate device is not running or has a diagnostic message. A diagnostics was pending, the cause of the error is no longer present. This symbol may appear in connection with various other symbols in this list.

: Exclamation mark red: The device is not running or a diagnostics is pending. The cause of the error is still present. This symbol may appear in connection with various other symbols in this list.

The names of all currently connected devices and applications are highlighted in green:

The names of devices running in simulation mode are shown in italics: 

Additional diagnostic information is provided by the “Status” tab of the respective device editor.

When logging in while the device description on the target device is newer than the one in the project, a warning is output as well as the option to cancel the operation.

Device editor

In the Device Editor tabs, select the settings for communication between I/O Engineering and the target device. Double-click on the device object in the device tree to open the editor. The editor contains generic and device-specific tabs. The editor name corresponds to the device name.

14.3 Mapping the hardware structure in the device tree

The following descriptions and figures are part of the PLC project engineering in ctrlX PLC Engineering.

However, the general handling within the device tree is also applicable to the configuration of devices, field buses and field bus devices in ctrlX I/O Engineering.

In the “Devices” view (device tree), you map the hardware that you control with your application. To do this, insert device objects representing real devices in the network into this tree structure until the hierarchy reflects the control network. Device objects are for example a PLC object, a field bus object or a logical device.

Scanning the current hardware and transferring devices to the project

If you use field bus drivers via CODESYS AddOns, you can detect the devices in the network of the hardware environment (scan) and transfer the found devices into the device tree of your project. If the scan functionality is permanently implemented in the PLC, you can perform the scan without any further preparations. I/O Engineering automatically establishes a temporary connection to the control for this purpose.

Scanning is started with the “Find devices” command. It refers to the control currently selected in the device tree and connected to the project. For example, select an inserted PROFINET IO controller and use the command to determine the I/O devices and I/O modules assigned to it.

Prerequisite: You have a project with a device configuration. The communication settings are correct. Gateways and hardware are running.

1. Select a control device object in the device tree.
2. Select the “Search devices” command in the context menu.
   ➧ I/O Engineering establishes the connection to the hardware. The “Search devices dialog” is displayed. Depending on the device type, the dialog provides different functions. However, a table with the devices detected in the hardware is always displayed: Device name, device type, station name, etc.. Please refer to the help for the respective device editor.
3. To show only found devices that are not yet part of the device configuration of the project in the list, enable the option “Show differences to project”.
4. To copy a device to the device tree of the project, select the entry in the table and click on the “Copy to project” button. If no entry is selected, all found devices are applied.
   ➧ The corresponding entries are inserted in the device tree.

15 Using the command line interface

You can start the I/O Engineering.exe from the command line, specifying the switches and options described below.

Syntax:

```
<folder>I/O Engineering.exe --<Schalter oder Option>
```

Enclose paths or option parameters that contain spaces, minus signs, or slashes in single quotes.

Switch `--culture` (national language of the user interface)

Add this switch to the development system call in the command line to set the national language of the user interface.

Syntax:

```
--culture=<Culture>
```

`<Culture>`: Common language abbreviation for the desired national language, for example de, en, fr, it, es, zh-CHS.

Example

Launching the I/O Engineering with English user interface:

```
I/O Engineering.exe --culture=en
```

Also refer to

- Chapter Dialog 'Options' – 'International Settings' on page 182

Switch `--profile` (I/O Engineering profile)

Add this switch to the development system call in the command line to start I/O Engineering directly with a specific profile. When starting I/O Engineering without this specification, the“Select Profile” dialog is opened.

Syntax:

```
--profile="<profile name>"
```

`<profile name>`: Enter the exact profile name, as displayed in the in the *“Help → Information”* dialog of the development system or in the Start menu of your computer.

Example

```
I/O Engineering.exe --culture=de --profile="I/O Engineering
V3.6"
```

Switch `--compare` (initiate project comparison)

Add this switch to the call of the development system in the command line to directly make a comparison of two I/O Engineering projects. After the button, first specify the "current project" and then the "comparison project". I/O Engineering starts and opens the “Project comparison - Differences” view.

Syntax:

```
--compare="<path of project file>" "<path of reference project
file>"
```

Example

```
I/O Engineering.exe --compare  "D:\proj\project1.project"
"D:\proj\project2.project"
```

Also refer to

- Chapter 'Compare’ command on page 143

Option `--project` (I/O Engineering project)

Add this option to the development system call in the command line to immediately open the specified project in I/O Engineering.

Syntax:

```
--project="<path of project file>"
```

`<path of project file>`: Project file path

Example

Open the project `test`:

```
I/O Engineering.exe --culture=de --
project="D:\projects\test.project"
```

Also refer to

- Chapter 'Open project’ command on page 128

Option `--projectarchive` (I/O Engineering project archive)

Add this option to the development system call in the command line to extract the specified project archive immediately in I/O Engineering and then open the project.

Syntax:

`--projectarchive="<path of projectarchive file>"`

`<path of project archive file>`: File path of the project archive

Example

Extract the project archive `test.projectarchive` and open the project in the programming system:

```
I/O Engineering.exe --
projectarchive="D:\projects\test.projectarchive"
```

Also refer to

- Chapter Command 'Extract archive' on page 132

Option `--runscript` (execute scrip)

Add the option to the development system call in the command line to have the specified script file executed by I/O Engineering.

Table 2: “Command line options for --runscript”

`--runscript="<scriptfile>.py"`	I/O Engineering executes the script file `<scriptfile>.py` at startup. Specify the entire path of the script file.
`--scriptargs:'<arg1> <arg2>... <argn>'`	Use the option together with the `--runscript` option. This option triggers the parameter transfer <arg1>... <argn> to the script. The arguments are passed on to the Python variable `sys.argv`.
`--noUI`	Use the option together with the `--runscript` option. The I/O Engineering user interface does not open. I/O Engineering prints all errors, warnings, compiler messages and messages generated by the script on the command line. The script messages (1: Severity Text) can be separated from the other messages (2: Severity FatalError, Error, Warning, Information) with the operator >.
`--enablescripttracing`	Use the option together with the `--runscript` option. It causes each command of the script file to be displayed in the output.

`--textPrompts`	Use the option together with the `--noUI` option. It causes message service methods and standard dialogs to be output to the command line so that user can be input data. If you do not specify `--textPrompts`, all message service prompts are automatically answered with the default value.
`scriptdebugger {="<debugger>"}`	Use the option together with the `--runscript` option. It switches IronPython into debug mode so that external debuggers can be used to debug Python scripts. The following values are defined for <debugger>, case-insensitive: • `auto`: Automatically detects if a debugger is attached to the current process at each script start. Currently, only .NET-based debuggers can be automatically detected. A detected debugger overrides the `--enablescripttracing` flag. • `.NET`: Enables debugging for .NET based debuggers such as the "Python Tools for Visual Studio" (PTVS) and SharpDevelop. With this option, in contrast to "auto", a debugger can also be attached to already running scripts. Please note: This is currently the default value when `--scriptdebugger` is used without specifying a value. • `disabled`: Disables debugging and automatic detection. • `script`: Switches the IronPython script engine into debug mode to enable debugging for settrace-based debuggers. The script itself has to connect to and disconnect from the debugger • `tracing`: Enables simple built-in "script tracing" operation and disables automatic detection. Is equivalent to the `--scripttracing` option. • `$absolute_path.py$`: Absolute path to a Python script that initializes the connection to a Python-based debugger. The IronPython Script Engine is switched to debug mode to enable debugging for settrace-based debuggers. This script is executed once during initialization, and it should define the following parameterless functions: `scriptdebuggersetup` is executed immediately before the user script is executed to connect to the debugger. `scriptdebuggershutdown` is called immediately after the user script is executed, or when the script engine is shut down, and should close the connection to the debugger.

Examples of using transfer parameters in script files using sys.argv

```
start /b /wait CODESYS.exe
--runscript="D:\Script\ArgvAnd__main__Test.py"
--scriptargs:'username password 3.14 "path=\"C:\temp\\"'
```

Script file `ArgvAnd__main__Test.py`

```
from __future__ import print_function
import sys
print("sys.argv: ", len(sys.argv), " elements:")

for arg in sys.argv:
    print(" - ", arg)
print()
print("__name__: ", __name__)
```

Output result `stdout`:

```
sys.argv: 6 elements:
- D:\TestScripts\ArgvAnd__main__Test.py
- username
- password
- 3.14
- path= "C:temp"
__name__: __main__
```

For more information on the `__name__` global variable, see the Python documentation.

Example of message output

```
start /b /wait CODESYS.exe --
runscript="D:\Script\AmpelTest.py" --noUI 1>ScriptMessages.txt
```

All messages generated by the script are redirected to the `ScriptMessages.txt` file by I/O Engineering. The other messages are output in the command line.

```
start /b /wait CODESYS.exe --
runscript="D:\Script\AmpelTest.py" --noUI 2>NUL
```

I/O Engineering suppresses all messages except script messages. The script messages are output in the command line.

Example of use of the option --scriptdebugger

The following `initdebug.py` script has been successfully tested with `pydevd-based` debuggers such as PyDev / LiClipse or PyCharm. To use this script, start I/O Engineering with the following command line:

```
--profile="Fanta Development Build" --
scriptdebugger="D:\test\charmdebug\initdebug.py"
```

`initdebug.py` file:

```
from _future_ import print_function
from _future_ import unicode_literals
import sys
sys.path.append(r"D:\test\Env2\Lib\site-packages\pycharm-
debug.egg")
import pydevd
def scriptdebuggersetup():
pydevd.settrace('localhost', port=51234, stdoutToServer=True,
stderrToServer=True)
def scriptdebuggershutdown():
pydevd.stoptrace()
```

Also refer to

- ➦ http://docs.python.org/tutorial/modules.html

Option `--ignorewhitespace`

Add this option to the call of the development system in the command line after the `--compare <project1> <project2>` option so that spaces are not included in the project comparison. Note: Semantically relevant spaces, such as in `STRING literals`, are nevertheless taken into account in any case.

Syntax

```
--compare="<path of project file>" "<path of reference project
file>" --ignorewhitespace="true"|"false"
```

Example

```
I/O Engineering.exe --compare "D:\proj\project1.project"
"D:\proj\project2.project" --ignorewhitespace="true"
```

Also refer to

- ➦ Chapter 'Compare' command on page 143

Option `--ignorecomments`

Add this option to the development system call in the command line after the `--compare <project1> <project2>` option so that comments are not taken into account in the project comparison.

Syntax:

```
--compare="<path of project file>" "<path of reference project
file>" --ignorecomments="true"|"false"
```

Example

```
I/O Engineering.exe --compare "D:\proj\project1.project"
"D:\proj\project2.project" --ignorecomments="true"
```

Also refer to

- ➦ Chapter 'Compare' command on page 143

Option `--ignoreproperties`

Add this option to the development system call in the command line so that object properties (access rights, translation settings, directories, bitmaps, etc.) are not taken into account in the project comparison.

Syntax:

```
--compare="<path of project file>" "<path of reference project file>" --ignoreproperties="true"|"false"
```

Example

```
I/O Engineering.exe --compare "D:\proj\project1.project" "D:\proj\project2.project" --ignoreproperties="true"
```

Also refer to

- ➥Chapter 'Compare' command on page 143

Option `--skipunlicensedplugins` (license for components missing)

Add this option to the call of the development system in the command line to skip the query whether unlicensed components should be loaded anyway. In this case, I/O Engineering silently does **not** load these components.

Example

I/O Engineering.exe --skipunlicensedplugins

16 Managing Devices

16.1 General

I/O Engineering manages the installed devices in the so-called device repository. A device repository is a defined location in the file system. In the default I/O Engineering installation, it is defined with an absolute path as the system repository. You install or uninstall devices in the "Device Repository" dialog. The system installs a device by reading the device description file. The properties of a device are defined in these files regarding configurability, programmability, and possible connections to other devices.

You can use the devices provided in the device repository by adding them to the device tree of your project.

Also refer to

- ➥Chapter Command 'Device Repository' on page 154
- ➥Chapter 16.2 Installing Devices on page 56

16.2 Installing Devices

Install a device in the device repository in order to include it in your project.

1. Click *"Tools → Device Repository"*.
   ➥ The "Device Repository" dialog box opens.
2. Select the install location. "System Repository" is set by default.
3. Click "Install".
   ➥ The "Install Device Description" dialog box opens.
4. Select the file path of the device description.
5. Select the file type filter of the required device description.
   ➥ All device descriptions of the selected file type are listed.

6. Select the required device description and click “Open”.
   ➡ I/O Engineering adds the device description to the matching category of your device repository.
   If errors occur during installation (for example, missing files that are referenced by the device description), then I/O Engineering displays them in the lower part of the device repository dialog box.

See also

- Chapter Command 'Device Repository' on page 154

17 Managing packages and licenses

17.1 Licensing additional projects (plugins)

In addition to the standard installation, third-party manufacturers such as Codesys GmbH offer additional products for use in I/O Engineering that are subject to a fee and license. The manufacturer provides the licenses for such products as ticket numbers, which can be installed via the License Manager in I/O Engineering.

The license information is stored on the runtime system of the ctrlX device. For more information on the topic "Licensing", refer to the "HOW-TO" section of the ctrlX Automation website.

In the license repository you can retrieve the current information for each ticket number from the central license server. Other information is also provided, such as whether a license can still be activated or must be returned.

License check when starting I/O Engineering

When it starts, I/O Engineering checks the selected profile for plug-ins that are subject to licensing.

- If the profile does not include plug-ins subject to licensing, then I/O Engineering starts as usual without a message.
- If the profile includes plug-ins subject to licensing and a dongle with the required licenses is connected to the USB port of the computer, then I/O Engineering starts without any message.
- If the profile includes plug-ins subject to licensing but no dongle is available or a required license cannot be found, then, I/O Engineering displays the “License Missing” dialog at start-up.
  Note: When starting from the command line, this dialog is skipped when specifying the `--skipunlicensedplugins` option in the command line. In this case plug-ins without the required license are automatically not loaded.

License check during running operation

While I/O Engineering is running, it is checked every five minutes whether the required licenses are available. If a license is missing, for example as the dongle has been removed in the meantime, the “Missing license” dialog is displayed and there are the following options:

- Plug the dongle back in and press “Repeat”: If the dongle is equipped with the missing licenses, you can now continue working.
- “Save Current Project and Exit”: I/O Engineering saves the project and closes.
- “Close”: I/O Engineering closes without saving the project.

When removing the license dongle after the plug-in has been loaded, an error message is displayed.

To fix the problem, plug the dongle back in and press the `Retry` button. If the dongle contains a valid license, the dialog closes and you can continue working. Clicking “Cancel” will exit I/O Engineering.

17.2 Installing/uninstalling add-ons

The installation and management of externally provided software add-ons for ctrlX I/O Engineering and ctrlX PLC Engineering is realized via the "Add-On Installer" software, see:

➦'Add-on installer' dialog

The ctrlX Add-On Installer is based on functions of the CODESYS installer of the CODESYS GmbH.

The complete documentation can be found on the CODESYS website, see: ➦Web documentation

Installing the add-on

1. Open the add-on installer via the menu *“Tools → Add-on installer”*
2. Click on “Install file”
3. In the “Open” dialog, select an add-on from the file directory and click on“Open”.
4. In the next step, the “Confirmation required” dialog shows an overview of the add-on to be installed. Confirm the installation with “Ok”. Optionally, you can also cancel the installation by clicking on the“Cancel” button
5. The “Install packages” dialog opens.

   In the first step, the certificates are displayed for unknown add-ons, including the associated properties. The dialog and displays information on whether and how the selected add-on is signed.
6. If it is a self-signed or unsigned add-on, explicitly allow the installation of this add-on by enabling the “Allow unsigned and self-signed add-ons” option.
7. If you agree with the displayed signing, click on“Continue”. Cancel the installation by clicking on“Cancel”.

   ➧ Add-ons may also include licensing information. This information is displayed in the next step. Via the tabs “License Agreement”, “Signature” and “Third Party Licenses” you can read the corresponding details. You can continue with the installation only after confirming the respective conditions by clicking the box next to it and then confirming with the “Continue” button. I/O Engineering also displays the “checksum” of the add-on.
9. Close the I/O Engineering application so that the add-on can be installed.

   ➧ The add-on installer performs the installation of the add-on in I/O Engineering and displays the installation steps as well as their progress.
10. Finally, the result of the installation is displayed. Finally, confirm the result with“OK” on “Finish”.

Uninstalling the add-on

1. Select the add-on
2. Click on “Uninstall”

   ➧ The wizard opens and is ready to uninstall the add-on.
3. Close the I/O Engineering application so that the add-on can be uninstalled.

   ➧ The add-on installer performs the uninstallation of the add-on.

17.3 Licensing products

The use of many software products is license-protected, so before you start using a product, first activate it. An add-on product that extends the scope of work ofctrlX I/O Engineering or ctrlX PLC Engineering is usually activated with a workstation license. Licensing is realized via the Codesys security key, a dongle that manages all licenses of your workstation. Optionally, some products will soon support licensing on a soft container.

Both licenses of add-on products on the local computer and individual device licenses can be managed uniformly via the I/O Engineering license manager. For devices with a unique serial number, if the license information on the device is lost, the license manager can reactivate the license from an automatically stored license backup file.

If possible, license your products via online activation instead of offline activation.

Online activation

Activation is possible in the “License manager” dialog in ctrlX I/O Engineering System. The prerequisite for this is that your development system has internet access. The target system itself does not require internet access.

Optionally, activate a license on the “License Central” website.

Offline activation

If your workstation computer does not have internet access, activate products using a license activation file. The I/O Engineering license server provides the file. Connect to the server from any web-enabled computer and request the file. Then transfer the file to your workstation computer using any storage device. As usual, the product is activated in ctrlX I/O Engineering System, both for workstation licenses and for single device licenses.

Optionally, if you have access to a web-enabled computer but ctrlX I/O Engineering System is not installed, you can access the “CodeMeter Control Center” from Wibu-Systems to activate the license.

The “CodeMeter control center” controls the CodeMeter service and is included in the I/O Engineeringsetup.

Offline activation of an add-on license on the local computer

The following instruction describes the licensing by a dongle. Licensing via a soft container is done analogously.

Prerequisite: You have internet access on the computer on which you want to install the license and the ctrlX I/O Engineering System is installed on the computer. A dongle is plugged into the computer.

1. Select the command *“Tools → License manager”*.
   ➡ The wizard starts with the “License Manager - Select target device” dialog.
2. Select “Workstation” as the target device, click on “Next >”, then select “Dongle” in the “License Manager - Select container” dialog, and again click on“Next >”.
   ➡ The “License manager” dialog opens and shows the product to be licensed as unlicensed in the “Products” window.
3. Select the product and click on the “Install Licenses” button.
   ➡ The wizard “Install licenses on workstation dongle <Dongle-ID> - Select operation” is displayed.

4. Select the “Activate license” operation and click on“Next >”.
   ➡ The “Install licenses on workstation dongle <Dongle ID> - Activate licenses” dialog is displayed.
5. Enter the “ticket ID” provided by the software provider. The ID consists of 5 blocks of 5 alphanumeric characters each.
   Select the “license server” that provides the license to activate the product. The software provider provides the server URL.
6. Click on the “Next >” button
   ➡ The connection to the license server ➦ (http://license.codesys.com) is established.
   - If the specified ticket contains only 1 license, a dialog confirming the activation is displayed after the server action is successfully completed.
   - If the specified ticket contains multiple licenses, the “Install licenses - Select” licenses dialog is displayed first with the list of licenses managed in the ticket.
7. In case of multiple licenses, select the license to be activated and click on “Next >”.
   ➡ After successful completion of the server action, a dialog is displayed confirming the activation.

Alternatively

1. Open a browser and select the ➦ License central web page.
   ➡ The “License Manager” dialog opens.
2. Enter your license number.
3. Select “Search”.
4. Click on “Next >”
   ➡ The connection to the license server ➦ (http://license.codesys.com) is established.

Restoring licenses

From I/O Engineering V3.5 SP13, a license backup file `*.WibuCmRau` is automatically generated when activating single-user licenses for devices with a unique serial number. The file is stored on your computer and on the license server. In case of loss of the licensing files on the device, the license can be restored from this file with the help of the License Manager.

A device license was activated.

1. In I/O Engineering select the command *“Tools → License Manager”*. Follow the wizard with the appropriate entries for the target device: Device and container (soft container or dongle), and the selection of the device in question.
2. In the “License Manager” dialog, click on the “Additional functions” button and select the “Restore license” command.
   ➡ The “Restore licenses” dialog opens.
3. Enter the “ticket ID” for the device license and click on “Restore”.
   ➡ I/O Engineering searches for the license backup file, first on the local computer, then on the License Central Server. When the file matching the device is found, the license is restored and activated.

List license information for products and device features of a control

Prerequisites:

- ctrlX I/O Engineering System is opened (from version V3.5 SP15)
- The gateway and the control whose information is to be read out are running
- There is no application on the control

1. Select the command *“Tools → Device Reader”*
   ➡ The “Select device” dialog opens.
2. Double-click on the desired gateway to browse the network.
   If no gateway is displayed, first click on“New gateway” and select the desired gateway in the “Gateway” dialog.
3. Select the desired control.
4. Confirm your selection by clicking on the “OK” button.
   Note: If there is an application on the control, a dialog is displayed asking if all applications should be removed from the control. When clicking on“No in”, the license information of the control cannot be read. The “Device Reader” command is aborted.
   ➡ I/O Engineering creates the list with the license information for the products and device features of the selected control and displays the information in the “Device reader” dialog.

17.4 CODESYS Professional Developer Edition

The CODESYS Professional Developer Edition subscription combines all solutions for professional application development according to IEC 61131-3 and their management in a single license.

- **CODESYS SVN**
  Connecting to the software versioning system Apache™ Subversion® for professional source code management.
- **CODESYS Static Analysis**
  Checking the source code using predefined rules and providing characteristic numbers on the code quality.
- **CODESYS Profiler**
  Detailed measurements of runtime performance and code coverage at module level to detect timing issues.
- **CODESYS Test Manager**
  Executing automated tests of applications and libraries.
- **CODESYS UML**
  Two new diagrams of the Unified Modeling Language: the class diagram and the state diagram.
- **CODESYS Git**
  Integrated use of the distributed version management system Git™ for all application objects.

Licensing via Rexroth licenses

Register the PC via the Rexroth license portal before purchasing the CODESYS Professional Developer Edition annual license. The registration function can be found in ctrlX WORKS on the page “Licenses”, see here: ➦ Web documentation

After purchasing the CODESYS Professional Developer Edition annual license (SWL-W-PCx-COSYxPROxDEVxED-Y1NN), import the license into ctrlX WORKS using the “Upload license file (Capability Response)” function. In addition, the add-on „Bosch Rexroth License Provider“ has to be installed via the Add-On Installer in I/O Engineering, see here: ➦ Web documentation

18 Using scripts

18.1 Using scripts– Basic information

With the scripting feature in I/O Engineering, you can automate commands or complex program operations that you would otherwise have to do manually with mouse clicks and text input in the I/O Engineering user interface. You can start these scripts from the I/O Engineering user interface (command or configured toolbar) or from the Windows command line.

Examples of use cases

- Integration of I/O Engineering in automatic build server environments:
  - Continuous Integration (CI)
  - Continuous Delivery (CD)
  - Continuous Testing
- Integration with third-party software, for example:
  - Code generators
  - Creation of projects that are custom tailored to a specific machine configuration
- Creation of documentation
- Updating of libraries:
  - Setting of project information during the release process
- Automatic testing:
  - Mostly in connection with the CODESYS Test Manager
- Outputting variables via monitoring APIs

Also refer to

- ➦ Continuous Delivery (CD)
- ➦ Continuous Integration (CI)
- ➦ Continuous Testing

Scripting language

The I/O Engineering scripting language is modular and based on IronPython. For this purpose, the I/O Engineering "ScriptEngine" component combines the IronPython interpreter with the I/O Engineering development environment. Then you can use the extensive Python framework libraries, which includes file access in networks and much more.

I/O Engineering does not yet include its own Python editor. Create your scripts with any text editor or the Python editor.

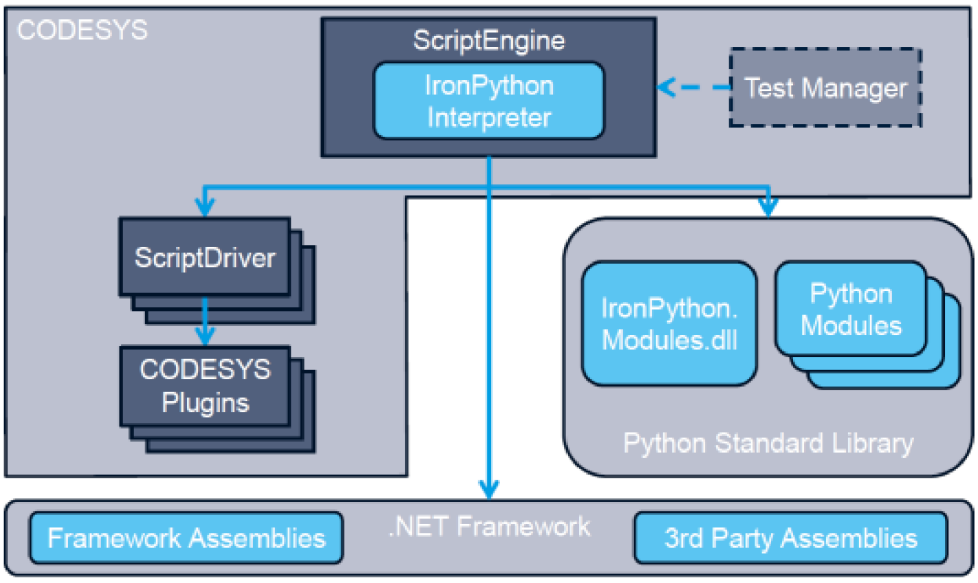
Also refer to

- ➦ Chapter 18.3.1 Python script – Basic information on page 69

Architecture of the ScriptEngine, Extension possibilities

The (Iron)Python scripting language used in I/O Engineering allows for access to the I/O Engineering scripting APIs for controlling I/O Engineering operations. Moreover, it lets users effectively apply both the Python standard library and third-party Python modules, as well as third-party .NET framework libraries and .NET assemblies.

Users can execute the scripts from menu commands or configured toolbars in the I/O Engineering interface or from the Windows command line. Add-ons such as the CODESYS Test Manager also provide ways to execute scripts.



There is not an integrated Python editor in I/O Engineering. Use your favorite text editor or the Python development environment.

With the Automation Platform APIs, the ScriptEngine APIs can be extended. Examples for this are CODESYS Test Manager and CODESYS SVN. Both provide their own objects and methods as an extension to the scripting APIs. In addition, the CODESYS Test Manager allows for the execution of scripts in a test case. Refer to the respective API documentation of the AddOns. Registered Automation Platform users will find more information in the I/O Engineering Developer Network.

Also refer to

- In the offline help: ➦API reference documentation for the ScriptEngine
- In the online help: ➦API reference documentation for the ScriptEngine

18.2 Executing scripts

18.2.1 Executing a script - Basic information

To execute Python script files (<filename>.py) containing a sequence of commands to enable the I/O Engineeringfunctionalities proceed as follows:

- From the I/O Engineering user interface by means of commands in the menu *“Tools → Scripting”*
- From the I/O Engineering user interface by means of a customized, configured toolbar
- From the Windows command-line

18.2.2 Calling scripts from menu commands

Prerequisite: There is a valid Python script file `<filename>.py` in the file system. The I/O Engineering user interface is open.

1. Optional: If you want to monitor the execution of the individual commands used in the script, select the command *“Tools → Scripting → Enable script tracing”*.
2. In I/O Engineering, select the menu command *“Tools → Scripting → Execute script”*.
   ➡ The statements in the script are executed and, if script tracing is activated, listed in the message view.

18.2.3 Starting scripts from the command line

In automated environments such as CI (Continuous Integration) servers or if scripts in I/O Engineering have to be controlled by other programs, menu commands are not appropriate for executing scripts. For these kinds of requirements, you can use the Windows command line to start I/O Engineering and execute scripts.

Prerequisite: There is a valid Python script file `<filename>.py` in the file system.

1. Create a CMD file that starts with `start` I/O Engineering and executes the script file with the option `--runscript`. Other options are possible, such as `--noUI` if the I/O Engineering user interface should not be opened.
2. Open the Windows “Command prompt” and execute the CMD file.

You can pass arguments with additional information to the script. Python scripts can access arguments with the `sys.argv[]` list. The first element (Index 0) is always the name or path of the Python script that is executed, followed by the actual parameters. (This is similar to `argc`/`argv` in C.) In addition, scripts can also access environment variables that are set before I/O Engineering is started with the corresponding Python or .NET APIs.

Example

A CMD batch file `argvtestbat.cmd` has the following contents (all in one line):

```
"C:\Program Files (x86)\3S CODESYS V3.5
SP10\CODESYS\Common\CODESYS.exe" --profile="CODESYS V3.5
SP10" --runscript="D:\Dokumente\Scripting\ArgvTestScript.py"
--scriptargs:'username password 3.14 "path=\"C:\temp\\\""' --
noUI
```

A matching script file `ArgvTestScript.py`:

Example

```
from __future__ import print_function

import sys
print("sys.argv: ",
   len(sys.argv),
   " elements:")

for arg in sys.argv:
    print(" - ", arg)
```

Now when you execute the CMD file, I/O Engineering starts and executes the script without opening the I/O Engineering main window. Then I/O Engineering is exited:

```
C:\Users\user>D:\SVN_Misc\trunk\ScriptEngine\ArgVTestbat

C:\Users\user>"C:\Program Files (x86)\3S CODESYS\CODESYS\Common\CODESYS.exe"
--culture=en --profile="CODESYS V3.5 SP10" --runscript="D:\SVN_Misc\trunk\Script
Engine\ArgvTestScript.py" --scriptargs:'username password 3.14 "path=\"C:\temp\\
\""' --noUI
sys.argv:  5  elements:
-  D:\SVN_Misc\trunk\ScriptEngine\ArgvTestScript.py
-  username
-  password
-  3.14
-  path="C:\temp\"
```

For a complete reference of all possible command line parameters, see the help page for the command line interface in I/O Engineering in the section on "`--runscript`".

Also refer to

- ➦Python API documentation
- ➦Information about the .NET API

18.2.4 Calling scripts from toolbar icons

You can provide your own toolbar in the I/O Engineering user interface with up to 32 icons for calling script files. For this you need an ICO file where the icon is stored, and a PY file where the Python script to be called is stored.

You create the configuration file `config.json` in the installation directory or in the program data directory (`I/O Engineering\Script Commands`). In the file, schematically specify the call information for each icon. A maximum of 16 icons can be configured. You can also store the ICO and PY files in the same directory.

Storage locations

- `<I/O Engineering installation directory>\I/O Engineering\Script Commands\`
- `%PROGRAMDATA%\I/O Engineering\Script Commands\`

Example
Standardinstallation auf Windows 7

```
C:\Program Files (x86)\CODESYS 3.5.14.0\CODESYS\Script
Commands\
C:\ProgramData\I/O Engineering\Script Commands\
```

If you store a `config.json` file with different call information at each of the storage locations, then you can configure up to 32 different icons.

Configuration file

Schema der Konfigurationsdatei für 2 Icons

```
[
  {
    <icon call information>
  },
  {
    <last icon call information>
  }
]
```

Schema der Aufrufinformation `<icon call information>`

```
  "Name": "<tooltip of the symbol button>",
  ( "Desc": "<description of the symbol button>", )?
  "Icon": "<icon file name>",
  "Path": "<path of the script file>",
  "Params":
    [
      ( "<value>", )*
      "<last value>"
    ] // Number of parameters is matching the script
```

Table 3: Call information

`"Name"`	Required; displayed as icon tooltip Example: `"Name": "Pause"`
`"Desc"`	Optional; comment for icon Example: `"Desc": "Operation pause"` Note: Not yet displayed in the user interface
`"Icon"`	Mandatory, file path `<directory path>\<icon name>.ico` of the icon Example: `"Icon": "pause.ico"` Tip: If the file is located in the same folder as the `config.json` file, the file name is sufficient.
`"Path"`	Mandatory, path of the Python script `<directory path>\<script name>.ico` Example: `"Path": "stop.py"` Tip: If the file is located in the same folder as the `config.json` file, the file name is sufficient.
`"Params"`	Optional; only if the script requires parameters Example: `"Params": [ "file1", "file2" ]`

Example

Datei `config.json`

```
[
  {
    "Name": "Start",
    "Desc": "Starts processing",
    "Icon": "start.ico",
    "Path": "goon.py",
    "Params":
      [
        "--continue"
      ]
  },
  {
    "Name": "Pause",
    "Desc": "Pause operation",
    "Icon": "pause.ico",
    "Path": "stop.py",
    "Params":
      [
        "delay:-1"
      ]
  },
  {
    "Name": "Processing",
    "Desc": "Process again",
    "Icon": "VarStatSmall.ico",
    "Path": "process.py",
    "Params":
      [
        "exit"
```

```
        ]
    }
]
```

The `Script Commands` folder contains the following files:

```
config.json
goon.py
stop.py
process.py
start.ico
pause.ico
VarStatSmall.ico"
```

Creating script calls for a toolbar button

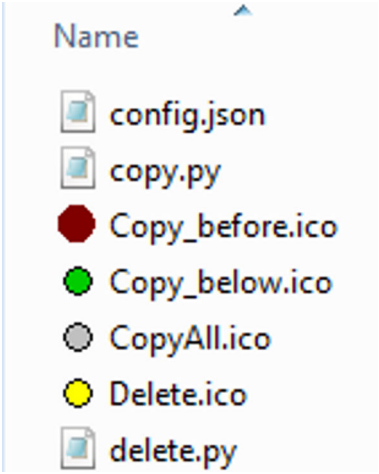
1. Create the `Script Commands` folder in one of the storage locations.
   ➡ `C:\ProgramData\I/O Engineering\Script Commands`
2. Create executable Python files there.
   ➡ Example:

   Datei `copy.py`

   ```
   print("The script COPY.PY is executed")
   ```

   Datei `delete.py`

   ```
   print("The script DELETE.PY is executed")
   ```
3. Create the ICO files for the scripts.
   ➡ Example: `Copy_before.ico`, `Copy_below.ico`, `CopyAll.ico`
4. Create a configuration file `config.json` there.
   ➡ The folder `C:\ProgramData\CODESYS\Script Commands` has the following contents:



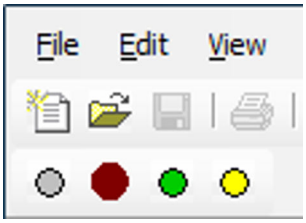
5. Open `config.json` and add the outlined call information.

➡

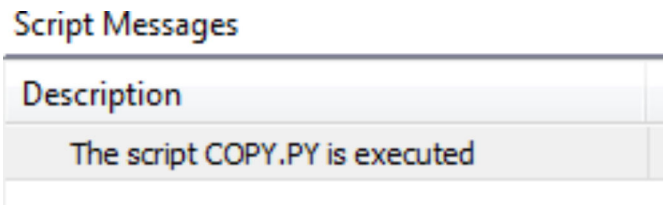
```
[
  {
    "Name": "Copy Before",
    "Desc": "Copy something",
    "Icon": "Copy_before.ico",
    "Path": "copy.py",
    "Params":
      [
        "before"
      ]
  },
  {
    "Name": "Copy Below",
    "Desc": "Copy something",
    "Icon": "Copy_below.ico",
    "Path": "copy.py"
  },
  {
    "Name": "Copy All",
    "Desc": "Copy something",
    "Icon": "CopyAll.ico",
    "Path": "copy.py",
    "Params":
      [
        "all"
      ]
  },
  {
    "Name": "Delete",
    "Desc": "Delete something",
    "Icon": "Delete.ico",
    "Path": "delete.py",
  }
]
```

6. Start I/O Engineering.
   ➡ The script files, the symbol files and the configuration file are read in and are then available in the dialog *"Tools → Customize"* dialog in the "Command icons" tab of the "ScriptEngine commands" command category.
7. Open the dialog *"Tools → Customize"* and click on the "Toolbars" tab.
8. Select the empty toolbar and click on the "Add Toolbar" button.
   ➡ A line editor opens at the empty toolbar.
9. Type in a name (example: `User defined toolbar`).
   ➡ The custom toolbar is displayed in the I/O Engineering window.

10. Add the recently imported commands and close the dialog.
➡



11. Click one of the icons.
➡ The following output is displayed in the message view.



Also refer to
- ⮫Chapter 7.2.2 Customizing Toolbars on page 18

18.3 Creating a python script

18.3.1 Python script – Basic information

Python is a dynamic language. You can start in a simple linear programming style (batch files) and later add the necessary and more powerful means, such as conditions, loops, functions, exceptions, classes, and modules. The focus of the language is on easy and expressive code. Python is more typical at runtime and uses an automatic garbage collector to protect the programmer from accidental damage to the entire system.

IronPython is an implementation of Python for .NET and allows for full access to the .NET framework and classes. The implementation of the IronPython interpreter is based on Python Version 2.7.

There are a variety of free manuals and help pages on the Internet. See the following links for an introduction and detailed introduction about IronPython.

- ⮫http://forum.codesys.com/viewforum.php: "Script language Python..." in the I/O Engineering forum.
  - Especially I/O Engineering-specific questions.
  - Also includes some examples.
- ⮫https://docs.python.org/2/tutorial/index.html:: Python tutorial in the official Python documentation.
- ⮫http://docs.python.org/release/2.7/: Official documentation on Python 2.7
- ⮫http://wiki.python.org/moin/BeginnersGuide: Useful manuals to learn IronPython.
- ⮫http://wiki.python.org/moin/GermanLanguage: Hyperlink collection for German help pages.
- ⮫http://stackoverflow.com/: General community for programming. For general question about (Iron)Python, not I/O Engineering-specific.
- ⮫http://ironpython.net/: IronPython homepage

- ➦http://ironpython.net/support/: Mailing list, FAQ, etc.
- ➦https://gitter.im/IronLanguages/ironpython:: Chat channel for IronPython developers.

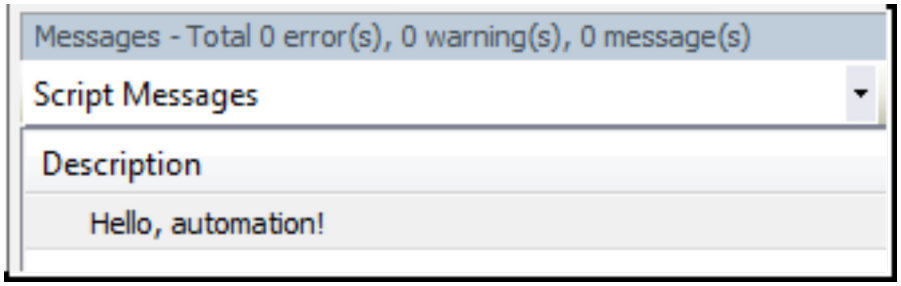
Version incompatibility to Python V3.x

The Python programming language will soon be available in the new version V3.x. Some old program modules were removed. Bosch Rexroth AG plans to upgrade to this new version. Scripting developers should take this into consideration and design their scripts accordingly, for example by using the expression `from __future__ import print_function`. You can find more information about this topic at ➦ http://wiki.python.org/moin/Python2orPython3 and ➦http://docs.python.org/release/3.1.2/whatsnew/3.0.html

18.3.2 Getting Started with Python for I/O Engineering

See below for an simple application of a Python script in I/O Engineering:

1. In any text editor, create a text file `hello.py` with the following contents:
   ```
   print("Hello, automation!")
   ```
2. Start I/O Engineering and click *"Tools → Scripting → Execute Script File"*. Select the file `hello.py` in the file system.
   ➧ See the result in the message view:



For more detailed examples of Python scripts for different use cases with I/O Engineering, refer to the following help pages:

See also

- ➦Chapter 18.3.4 Basic Syntax of Python (with Examples) on page 71
- ➦Chapter 18.3.5 Python Control Structures (with Examples) on page 79

18.3.3 Tips for Python Programmers about .NET API Documentation

The current prerelease of the script interface documentation has been generated automatically from the underlying .NET and C# sources. Therefore, the documentation includes some expressions that are not familiar to Python programmers. The following overview provides some tips about how these expressions can be understood from the Python perspective.

- An interface is the contract that tells an instance of a class that implements the interface which members (methods, properties) it has to prepare. In IronPython, you can implement one or more .NET interfaces in one class by inheriting from a superclass. If a method is needed for the interface but is not available in the class definition, then an exception is thrown. (The `DeviceImportFromSvn.py` example shows a class that implements the `ImportReporter` interface.)
- Each parameter and each method in .NET is typed strictly. The type of parameter is separated by one space character before the parameter name, and the type of return value from a method by one space character before the method name. You can use instances from subclasses when you define a class (or interface). A method without a return value is marked `void`.

- You can overload methods, as multiple methods of the name can exist in one class. However, the number or the types of parameters must be different. IronPython automatically takes care of the most appropriate method overloading being called.
- The data type `int` corresponds to an integer from -2,147,483,648 to 2,147,483,647.
  The data type `bool` corresponds to the Python type `bool` (`True` and `False`).
  The data type `string` corresponds to the Python types `str` or `unicode`, which are identical in IronPython.
  The `IDictionary<Object, Object>` data type corresponds to an ordinary Python dictionary. IronPython automatically converts between Python and .NET data types.
- When a `T` type is inherited from `IBaseObject<T>`, it means that this type can be extended by other plug-ins for additional members. The actual use of this extended type as a parameter or return value is marked by `IExtendedObject<T>`.
- The `IEnumerable<T>` interface to a `T` type means that you can use every Python sequence (generators, lists, tuples, ...) that returns `T` type values (or a subclass). Returns the sequence of incompatible objects, and throws an exception at runtime.
- The `IList<T>` interface to a `T` type identifies a typified list that guarantees to include only elements of type `T` (or a subclass). An exception is thrown at runtime when any attempt is made to add an incompatible object.
- The `params T[] name` id for a parameter of type `T` corresponds to the Python mechanism `*name` for variable argument lists.
- In Python, enumerations (`enum`) do not exist as language constructs. Its purpose is to define an exact number of constant values for a specific purpose, for example the days of the week. Access to .NET enumerations from IronPython works by using "Name.Member", for example via `OnlineChangeOption.Try`.
  (There are many ways of emulating enums in Python. For examples, see ➦ http://pypi.python.org/pypi/enum/ or ➦ http://www.ironpython.info/index.php/Enumerations.)
- The syntax `T name { get; set; }` defines a property with `name` as the name and `T` as the type. If `set;` is missing, then the property is read-only. In Python, the respective construct is `@property` decorator.

See also

- ➦ Documentation for Python API
- ➦ Information about .NET API
- ➦ Chapter 18.3.6 Accessing I/O Engineering functionalities with a script on page 83

18.3.4 Basic Syntax of Python (with Examples)

Python is similar to the languages of the C family, but there are some significant differences and unique properties.

The most obvious syntactic difference between Python and language such as C and ST is that the Python parser recognizes block structures by their indentation. There is no `BEGIN/END` or braces `{}` to identify the blocks of `IF/ELSE` conditions, `FOR` and `WHILE` loops, or functions.

Comments start with `#` and extend to the end of the line. In the first and second line of the source code, you can set a special marker to declare the encoding of the file. We recommend that you use UTF-8 as the encoding if ASCII characters are not required.

For debugging purposes, you use `print` for easy output. With the `%` operator, you achieve functionality similar to the C function `printf()`. The output is displayed in the message view of I/O Engineering.

Example: `print`

```
# encoding:utf-8

# defining a function with the parameter i
def do_something(i):
    # if branch
    if i>0:
        print("The value is: %i" % i)
                sum += i
        print("The new sum is: %i" % sum)

    # else if (optional, there can be none or several elif
branches)
    elif i=0:
        print("The sum did not change: %i" % sum)

    # and the final else branch (also optional).
    else:
        handle_error()

# an endless while loop
while True:
    print("I got stuck forever!")
```

Everything that belongs to the same block has to be indented the same distance. The size of the indentation is irrelevant. Elements such as brackets and braces have a higher priority than indentations. Therefore, the following code segment is completely correct, even if it is written in a poor programming style:

Example: Indentation

```
# warning: bad style below. Kids, don't try this at home!
if foo >= bar:
               print("foobar")
else:
    print(
          "barfoo"
)
```

To avoid ambiguity, you should not mix tabs and spaces in a file.

At this time, mixing tabs and spaces gilt in Python 3 qualifies as a syntax error.

The official Python Style Guide recommends indentation of four spaces and includes some examples of good and poor style. The Python tutorial provides a summary of coding style.

Python is case-sensitive, similar to and in contrast to ST. Keywords, such as `def`, `if`, `else`, and `while`, have to be lowercase (in contrast to the ST rule: keywords are uppercase). Two identifiers, such as "i" and "l", also identify two different variables.

the following keywords are reserved in Python and not permitted for use as identifiers for variables, functions, etc.: `and | as | assert | break | class | continue | def | del | elif | else | except | exec | finally | for | from | global | if | import | in | is | lambda | not | or | pass | print | raise | return | try | while | with | yield`.

Python 3 defined four other keywords: `False | None | True | nonlocal`. While the first three are really new, the first three were already predefined constants in Python 2 and should not be used for any other purposes.

See also

- ➦Python Style Guide
- ➦Python Tutorial

Variables and data types

Python is a powerful, dynamically typed language -- all type information is evaluated at runtime. Variables hold references to objects, and the object knows its type, not the variable. When a programmer attempts to execute an operation that is not possible (for example, adding an integer and a string), Python throws an exception at runtime.

Consequently, there are no declarations of variables and their types. In Python, variables are created only to assign values to them. This is completely different in C and ST where types are strong and static. Every variable is declared with a type, and at compile time the compiler checks that the type and operators are permitted.

Refer to the following examples for dealing with variables:

Example: Variables

```
# assign the integer 1 to the variable i (also "creates" the
variable")
i = 1

# assign the string "foobar" to the variable s
s = "foobar"

# Add 5 to the integer i - this is equivalent to i = i + 5
i += 5
# i now holds the integer 6.

# Try to add i and s - this will throw an exception when
executed
# TypeError: unsupported operand type(s) for +: 'int' and
'str'
result = i + s

# variables can also be "undeclared" by deleting them.
# Further access to the variable i will throw a NameError
exception,
# as the variable does not exist any more.
del i

i += 5 # now throws an exception: NameError: name 'i' is not
defined
```

All existing variables reference one value only. There is not any unassigned or uninitialized variables in Python. To express the absence of a value, Python provides a special object: `None`. In C or ST, you would use a null pointer. Its only purpose is to express "no value here", although `None` is actually an existing instance of the class `NoneType`.

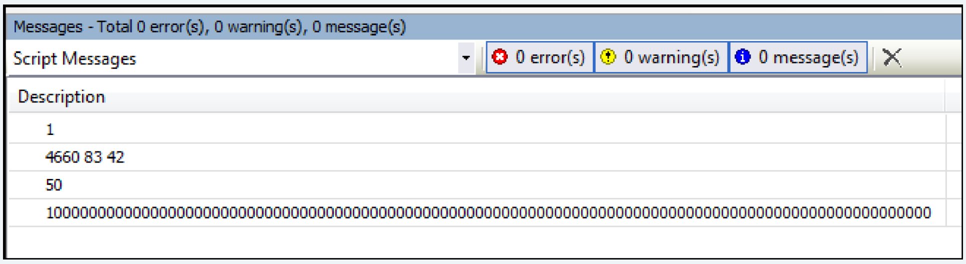
Numeric types and floating points

In contrast to the dozens of integer types in IEC or C, there is only one integer type in Python. Integer types in Python do not have a fixed size. Instead, they grow as needed and are limited only by available memory.

Example: `Integers.py`

```
from __future__ import print_function

i = 1
print(i)

j = 0x1234   # hex number, is 16#1234 in IEC and 4660 in
decimal
k = 0o123    # octal number, is 8#123 in IEC and 83 decimal
l = 0b101010 # binary number, is 2#101010 in IEC and 42 in
decimal
print(j, k, l)

m = (2 + 3)*10 # k is 50 now
print(m)

n = 10 ** 100 # 10 to the power of 100
print(n)
```

Resulting output:



There is also only one floating-point type in Python which is similar to the IEC data type `LREAL`. It provides 64-bit IEEE floating point arithmetic.

The syntax is like C-based languages for the most part:

Example: Floating-point types

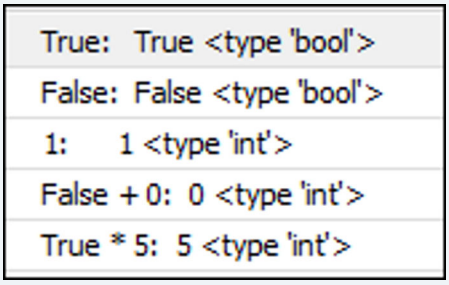
```
# A simple float...
a = 123.456

# A float containing the integral value 2
b = 2.

# Leading zeros can be left off
c = .3 # same as 0.3

# Exponential / scientific representation
d = -123e-5
```

Two special cases are `True` and `False`, two constants that define the Boolean truth values. They behave similar to the integer values `0` and `1`, except when they are converted into strings and return their names.

Example: `Booleans.py`

```
# booleans behave like integers, except when converted to
strings.
# The built-in function "type" can be used to query the type
of a value.
print("True:  ", True, type(True))
print("False: ", False, type(False))
print("1:     ", 1, type(1))
print("False + 0: ", False + 0, type(False + 0))
print("True * 5: ", True * 5, type(True * 5))
```

Resulting output:



Strings

In IronPython, strings are always in Unicode and any length. It does not make any difference if they are enclosed in `'` or `"`. Strings can also have triple quotation marks `"""` or `'''`, which allows for multiline string literals.

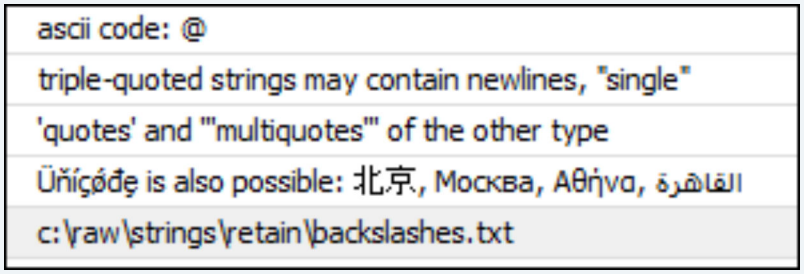
Similar to C, special characters can be excluded by means of backslashes (`\`): As a comparison, the dollar sign (`$`) is used in IEC for this purpose.

There are also raw strings that have other rules for the backslash. This is practical when the string should have literal backslashes. Example: Windows file paths or regular expressions.

Example: `Strings.py`

```
# encoding:utf-8
from __future__ import print_function

a = "a simple string"
b = 'another string'
c = "strings may contain 'quotes' of the other type."
d = "multiple string literals" ' are concatenated ' '''by the
parser'''
e = "Escaping: quotes: \" \' backslash: \\ newline: \r\n
ascii code: \x40"
f = """triple-quoted strings may contain newlines, "single"
'quotes' and '''multiquotes''' of the other type"""
g = "Üňíçóđę is also possible: 北京, Москва, Αθήνα, القاهرة!"
h = r"c:\raw\strings\retain\backslashes.txt"

# we iterate over a sequence of all the variables defined
above:
for i in (a,b,c,d,e,f,g,h):
    print(i) # prints the contents of the variable
```

Resulting output:



Python does not have characters types. Characters are expressed by the use of strings with a length of 1. In this way, iteration via a string, or indexing in a string, returns a single-character string.

See also

- ➦Python Documentation, Strings

Lists and tuples (data sets)

Lists and tuples basically correspond to arrays in C and IEC, but there are some noticeable differences:

- Index access is always checked. Accessing a list or a tuple with an invalid index throws an exception.
- Both lists and tuples can contain elements of different types (also other lists and tuples). In contrast in C and IEC, arrays can contain only elements of a single type.
- Lists are dynamic, and elements can be added, removed, or replaced at any time.
- Tuples are not changeable: Once a tuple is created, it cannot be modified anymore.

Lists are created with the `list()` constructor. As an alternative, you can use brackets `[]`. Tuples are created with the `tuple()` constructor or parentheses `()`.

Example: `list_tuples.py`

```
from __future__ import print_function
print("Testing tuples and lists")

# We define a tuple with the numbers from 1 to 10:
t = (1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
print("Tuple:", t)

# We can access the 6th element of the tuple.
# As in C, index counting starts with 0.
print("Element 5:", t[5])

# Subscription is more powerful using the range syntax:
print("Range[2:5]:", t[2:5]) # lower bound is inclusive,
upper bound is exclusive.
print("Range[2::2]:", t[2::2]) # start with 3rd element, and
print every 2nd element.
print("Range[-3:-1]:", t[-3:-1]) # Start with the 3rd last
element, end just before the last element (upper bound is
exclusive)
print("Range[::-1]:", t[::-1]) # negative step with - print
backwards
```

```
# lists are similar to tuples...
l = [11, 12, 13, "8", t] # contains mixed types: 3 integers,
a string, and the tuple defined above.
print("List:", l)

# ... but elements can be added or removed dynamically.
l.append(9) # Add a 9 to the list.
print("List with 9:", l)
print("List Range[3:6:2]:", l[3:6:2]) # print the 4th and 6th
element.

del l[1] # remove the element at index 1, the 12.
print("Removed[1]:", l)
del l[1:3] # Remove the elements at index 1 and 2, the 13 and
the '8'.
print("Removed[1:3]:", l)
```

Resulting output:

```
List: [11, 12, 13, '8', (1, 2, 3, 4, 5, 6, 7, 8, 9, 10)]
List with 9: [11, 12, 13, '8', (1, 2, 3, 4, 5, 6, 7, 8, 9, 10), 9]
List Range[3:6:2]: ['8', 9]
Removed[1]: [11, 13, '8', (1, 2, 3, 4, 5, 6, 7, 8, 9, 10), 9]
Removed[1:3]: [11, (1, 2, 3, 4, 5, 6, 7, 8, 9, 10), 9]
```

Dictionary

Python also has a hash table type (also "hashmap"). In contrast to the list, it can be indexed with any elements, for example strings. Its constructor is `dict()` and its literals are declared with braces `{}`.

The sample script `dictionaries.py` creates the output displayed below. In the last line, the script is terminated with a "KeyError" exception:

Example: `dictionaries.py`

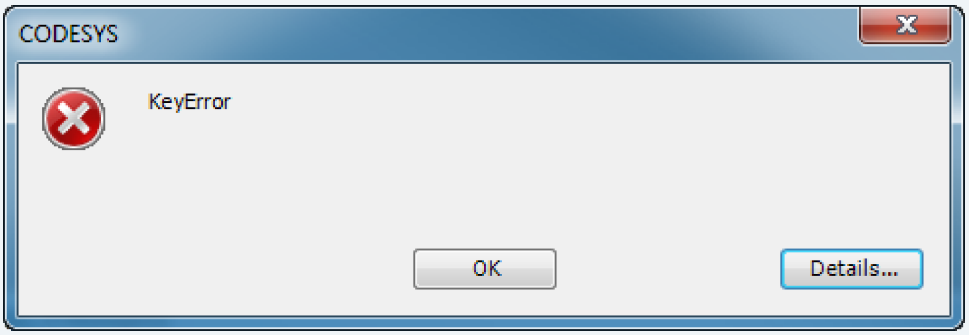
```
from __future__ import print_function
print("Testing dictionaries")

# Declare a dictionary with three entries, the third being a
list
d = {1: "a", 2: "b", "my list": [1, 2, 3]}
print(d)

# print the value of the key 1
print(d[1])

# remove the value with the key "my list"
del d["my list"]

# Add a value 4 with the key 3
d[3] = 4
print(d)
```

```
# The "get" method returns the second argument if the key
cannot be found.
print(d.get(1, 42))
print(d.get(23, 42))

# print all keys in the dictionary
for key in d:
    print(key)

# index access for unknown keys will throw a "KeyError"
exception!
print(d[23])
```

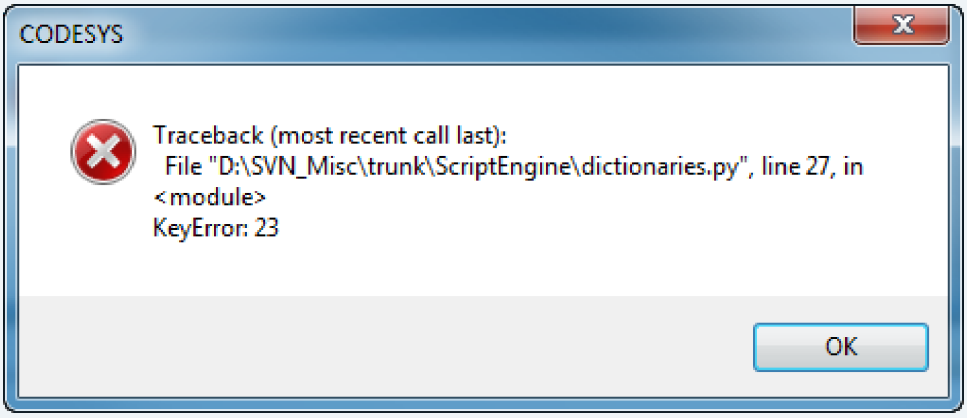
Resulting output:

```
Testing dictionaries
{1: 'a', 2: 'b', 'my list': [1, 2, 3]}
a
{1: 'a', 2: 'b', 3: 4}
a
42
1
2
3
```

And then in the last line, the script terminates:



You can view the stack trace by clicking the “Details” button. Here you find out about line number `27` and the unknown key `23`.



18.3.5 Python Control Structures (with Examples)

Loops

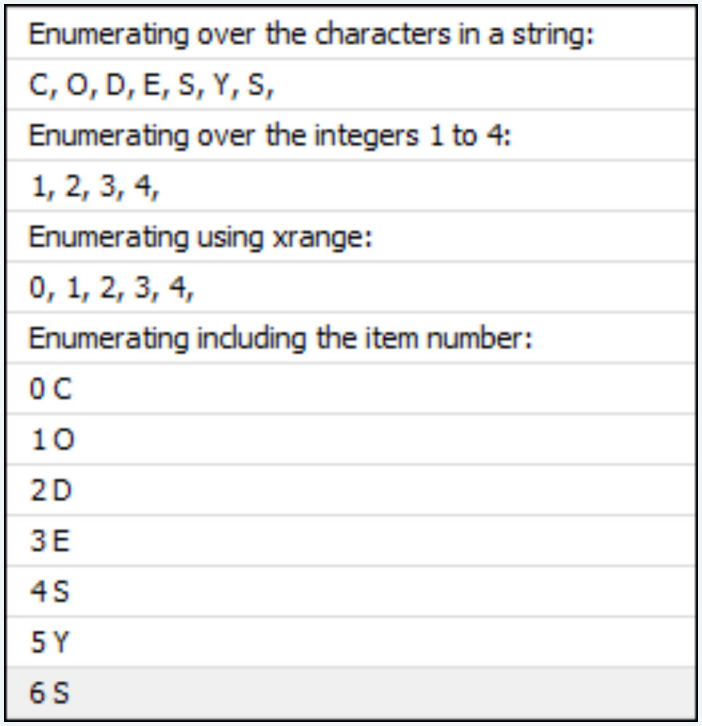
As opposed to C and ST, `for` loops in Python do not count loop variables, but iterate over a sequence. This kind of sequence can be a dictionary, a list, a tuple, the characters in a string, or lines in a file.

The following example shows some `for` loops:

Example: loops.py

```
from __future__ import print_function

print("Enumerating over a simple list:")
for i in (1,2,3,4):
    print(i, end=", ") # end= replaces the newline with ", "
print()                # but we still need a newline at the
end of this case.

print("Enumerating over the characters in a string:")
for i in "CODESYS": # characters are representet as strings
of length 1.
    print(i, end=", ")
print()

print("Enumerating over the integers 1 to 4:")
for i in range(1, 5): # upper bound is exclusive.
    print(i, end=", ")
print()

print("Enumerating using xrange:")
for i in xrange(5): # xrange is similar to range, but needs
less memory for large ranges.
    print(i, end=", ")
print()

print("Enumerating including the item number:")
for i, v in enumerate("CODESYS"):
    print(i, v)
```

Resulting output:



If you require an index or number in addition to the item, then you should use `enumerate` as shown in the last case of the sample script. The following code is considered as poor style:

Example: Poor style

```
text = "CODESYS"

for i in range(len(text)):    # BAD STYLE!
    v = text[i]               # DON'T TRY THIS AT HOME!
    print(i, v)
```

Besides `for` loops, Python also has `while` loops which are very similar to those in C and ST:

Example: "while" loops

```
i = 0
while i < 3;
    print(i)
    i += 1
```

Note: This example is not very practical. You would more likely use a `for` loop with a range.

IF / ELSE

The `if/else` construct is similar to those in other programming languages. Here is a short example:

Example: "if_else.py"

```
from __future__ import print_function
i = int(system.ui.query_string("Please enter an integral
number..."))
if i < 0:
    print("Your number was negative.")
elif i > 0:
    print("Your number was positive.")
else:
    print("It seems your number was zero.")
```

The `else` branch is optional and there can be zero, one, or many `elif` branches.

Functions, classes, and methods

Python allows for defining functions and classes with methods. A class with methods is basically similar to a function block in ST, or classes in languages such as C++, Java, or C#. However, Python does not support interfaces.

For detailed information, refer to the Python documentation for defining functions and classes.

Examples: Functions, classes, and methods

```
#defining a function with name sum and two parameters a and b:
def sum(a, b):
    return a + b # we return the sum of a and b.

# we can now call the function defined above:
print(sum(5,7))

# Now we define a class Foo:
class Foo:
    # The class gets a method "bar".
    # Note: for methods, the first parameter is always "self"
and
    # points to the current instance. This is similar to
"this" in
    # ST and other languages.
    def bar(self, a, b):
        print("bar(%s,%s)" % (a,b))

# We create an instance of the class:
f = Foo()

# We call the method bar on the instance.
f.bar("some", "params")
```

See also

- ➦Python Documentation, Defining Functions
- ➦Python Documentation, Classes

Modules and standard libraries

In IEC, you can import libraries for reuse by other written code. As a pendant, there is the possibility in Python of importing modules.

The Python standard library contains many modules for different purposes, such as:

- String processing
- Date and time handling
- Collections
- Threading
- Mathematical functions
- File handling
- Persistence
- Compression and archiving
- Database access
- Encryption services
- Network and Internet access
- Sending of emails

To create your own modules, write a Python file that defines the functions and classes that you want to provide. Save this file to the same directory as our sample script. If you name the file `mymodule.py`, then you can import it with `import mymodule`.

Here is an example of importing and using the cosine function and the pi constant from the `math` module:

Example: Import mathematical function

```
from math import cos, pi

print(pi) # prints 3.14159265359

print(cos(pi)) # prints -1.0
```

The following contains more examples that access information about the operating system, the Python version, and the interpreter:

More import examples

```
import os
print(os.environ["OS"])

from sys import platform, version, executable
print(platform)
print(version)
print(executable)
```

```
Windows_NT
cli
2.7.1 (CODESYS.exe CODESYS V3.5 SP4, ScriptEngine.plugin 3.5.4.0)
CODESYS.exe
```

There is a special module `__future__` for activating new language features. Above all, it is used when Python developers introduce new functionalities that are backward compatible. These kinds of functionalities have to be activated with special "`__future__` imports". One example that we use in most of our sample scripts here is the activation of the new power syntax of `print` as a function instead of a statement.

Example: "__future__"

```
# make print() a function instead of a statement
from __future__ import print_function
```

The Python documentation provides a complete list of all `__future__` imports.

In addition to the normal Python modules, IronPython code can also access .NET assemblies as if they were Python modules. This opens the access to the .NET framework class library and third-party libraries. Here is an example how to open a dialog box by means of the library `Windows Forms`:

Example: Opening a .NET dialog box

```
import clr
clr.AddReference("System.Windows.Forms")
from System.Windows.Forms import MessageBox

MessageBox.Show("Hello")
```

See also

- ➦Description of the Python standard library
- ➦List of all "__future__" imports
- ➦Overview of .NET framework class libraries

18.3.6 Accessing I/O Engineering functionalities with a script

All objects and commands that I/O Engineering provides for scripts are provided in the Python module "`scriptengine`" . Whenever a script is started, an implicit `<code>from scriptengine import *</code >` is issued. This allows easy access to I/O Engineering. However, if your script imports modules that need access to I/O EngineeringAPIs, these modules have to import the `scriptengine` module!

The main objects (categories) that can be used as entry points in Python scripts are shown in the table below. For detailed documentation of the entry points, see the API reference documentation for the I/O Engineering ScriptEngine.

Objects	Description
system	Access to general I/O Engineering functionalities, for example: • Exiting I/O Engineering • Handling the general user interface • Accessing the message memory • Delay and progress bar control.
projects	• Access to the I/O Engineering project and the device tree • Loading, creating, saving and closing projects
online	Access to online functionalities, for example: • Logging on devices • Managing access data (user name, password) • Performing the network search • Gateway management
device_repository	Handling of the device repository, import and export of device descriptions

In the following, example scripts for accessing I/O Engineering functionalities are described. For detailed information, please see the API reference documentation for theI/O Engineering ScriptEngine.

Also refer to

- ➦Chapter 18.3.3 Tips for Python Programmers about .NET API Documentation on page 70
- In the offline help: ➦API reference documentation for the ScriptEngine
- In the online help: ➦API reference documentation for the ScriptEngine

Example: Print device tree of the current project

The `PrintDeviceTree.py` script is an example of how to navigate in a project. It generates the output of a hierarchical representation of all devices in the currently opened project.

Load a project that contains some device objects and then run the script.

Example `PrintDeviceTree.py`

```
# encoding:utf-8
# We enable the new python 3 print syntax
from __future__ import print_function

# Prints out all devices in the currently open project.

print("--- Printing the devices of the project: ---")

# Define the printing function. This function starts with the
# so called "docstring" which is the recommended way to
document
# functions in python.
def print_tree(treeobj, depth=0):
    """ Print a device and all its children

    Arguments:
    treeobj -- the object to print
    depth -- The current depth within the tree (default 0).

    The argument 'depth' is used by recursive call and
    should not be supplied by the user.
    """

    # if the current object is a device, we print the name
and device identification.
    if treeobj.is_device:
        name = treeobj.get_name(False)
        deviceid = treeobj.get_device_identification()
        print("{0}- {1} {2}".format("--"*depth, name,
deviceid))

    # we recursively call the print_tree function for the
child objects.
    for child in treeobj.get_children(False):
        print_tree(child, depth+1)

# We iterate over all top level objects and call the
print_tree function for them.
for obj in projects.primary.get_children():
    print_tree(obj)

print("--- Script finished. ---")
```

In the message window, the device tree (from the "Devices" view) is displayed, omitting all non-device objects

Example: User interfac/ interaction with the user

In some cases, scripts need to interact with the user. For the most common interactions, we provide some simple APIs. The example script `System_UI_Test.py` shows all available functions.

Example `System_UI_Test.py`

```
# encoding:utf-8
from __future__ import print_function

"""Performs some tests on the messagestore and UI."""

print("Some Error, Warning and Information popups:")
system.ui.error("Fatal error: Everything is OK. :-)")
system.ui.warning("Your bank account is surprisingly low")
system.ui.info("Just for your information: 42")

print("Now, we ask the user something.")
res = system.ui.prompt("Do you like this?",
PromptChoice.YesNo, PromptResult.Yes);
print("The user selected '%s'" % res)

print("Now, the user can choose between custom options:")
res = system.ui.choose("Please choose:", ("First", 2, 7.5,
"Something else"))
print("The user selected option '%s'" % str(res)) # res is a
tuple

print("Now, the user can choose several options:")
res = system.ui.select_many("Please select one or more
options", PromptChoice.OKCancel, PromptResult.OK, ("La
Premiere", "The Second", "Das Dritte"))
print("The returned result is: '%s'" % str(res)) # res is a
tuple

print("Now, the user can select files and directories")
res = system.ui.open_file_dialog("Choose multiple files:",
filter="Text files (*.txt)|*.txt|Image
Files(*.BMP;*.JPG;*.GIF)|*.BMP;*.JPG;*.GIF|All files (*.*)|
*.*", filter_index = 0, multiselect=True)
print("The user did choose: '%s'" % str(res)) # res is a
tuple as multiselect is true.

res = system.ui.save_file_dialog("Choose a file to save:",
filter="Text files (*.txt)|*.txt|Image
Files(*.BMP;*.JPG;*.GIF)|*.BMP;*.JPG;*.GIF|All files (*.*)|
*.*", filter_index = 0)
print("The user did choose: '%s'" % res)

res = system.ui.browse_directory_dialog("Choose a directory",
path="C:\\")
print("The user did choose: '%s'" % res)

print("Now we query a single line string")
```

```
res = system.ui.query_string("What's your name?")
print("Nice to meet you, dear %s." % res)

print("Now we query a multi line string")
res = system.ui.query_string("Please tell me a nice story
about your life!", multi_line=True)
if (res):
  print("Huh, that has been a long text, at least %s
characters!" % len(res))
else:
  print("Hey, don't be lazy!")

print("Username and passwort prompts...")
res = system.ui.query_password("Please enter your favourite
password!", cancellable=True)
if res:
  print("Huh, it's very careless to tell me your favourite
password '%s'!" % res)
else:
    print("Ok, if you don't want...")

res = system.ui.query_credentials("Now, for real...")
if res:
  print("Username '%s' and password '%s'" % res) # res is a 2-
tuple
else:
    print("Sigh...")
```

Example: Manipulating “Project information” object

In the `ProjectInfoExample.py` script, some information is set in the “Project information” object. The most important information parts (such as“title” and “version”) have explicit properties. Any other information fields can be read and written with the `dictionary syntax`. For example, the fields recommended for the properties of a library project.

The example shown below may seem a bit unrealistic, but similar code is used in build servers that automatically create, test and, if necessary, release library projects as well as other projects. ScriptEngine is one of the key elements in creating CI (Continuous Integration) and CD (Continuous Delivery) systems.

Example **`ProjectInfoExample.py`**

```
# encoding:utf-8
from __future__ import print_function

proj = projects.load("D:\Some.library")

info = proj.get_project_info()

# Set some values
info.company = "Test Library Ltd"
info.title = "Script Test Project"
info.version = (0, 8, 15, 4711)
info.default_namespace = "testlibrary"
info.author = "Python von Scriptinger"

# some values recommended in the library toolchain
info.values["DefaultNamespace"] = "testlibrary"
```

```
info.values["Placeholder"] = "testlibrary"
info.values["DocFormat"] = "reStructuredText"

# now we set a custom / vendor specific value.
info.values["SpecialDeviceId"] = "PLC0815_4711"

# Enable generation of Accessor functions, so the IEC
# application can display the version in an info screen.
info.change_accessor_generation(True)

# And set the library to released
info.released = True;

proj.save()
```

Example: Run external commands and import PLCOpenXML files

The example script `DeviceImportFromSVN.py` retrieves a PLCOpenXML file from an external program (in this case an SVN. client) and imports the file into a newly created I/O Engineering project.

If you want to use the script, adjust the paths to your environment.

Example **`DeviceImportFromSVN.py`**

```
# encoding:utf-8
# Imports a Device in PLCOpenXML from Subversion via command
line svn client.

# We enable the new python 3 print syntax
from __future__ import print_function

import sys, os

# some variable definitions:
SVNEXE = r"C:\Program Files\Subversion\bin\svn.exe"
XMLURL = "file:///D:/testrepo/testfolder/TestExport.xml"
PROJECT = r"D:\test.project"

# clean up any open project:
if projects.primary:
    projects.primary.close()

# Fetch the plcopenxml data from subversion.
# We'll catch the output of the program into the xmldata
variable.
# The 'with' construct automatically closes the open pipe for
us.
with os.popen('"' + SVNEXE + '" cat ' + XMLURL, 'r') as pipe:
    xmldata = pipe.read()

# create a new project:
proj = projects.create(PROJECT)

# import the data into the project.
proj.import_xml(xmldata, False)

# and finally save. :-)
```

```
proj.save()

print("--- Script finished. ---")
```

Advanced example: Retrieve library from SVN and install in I/O Engineering

The following sample script can handle retrieving and installing a library as part of a CT (Continuous Testing) environment so that it can then be tested. In addition to the standard I/O Engineering, the add-on CODESYS SVN also has to be installed with a valid license.

Example

```
import tempfile

if projects.primary:
    projects.primary.close()

tempdir = tempfile.mkdtemp()
URL = "svn://localhost/testrepo/trunk/SvnTestLibrary/"

proj = svn.checkout(URL, tempdir, "testlibrary",
as_library=True)
proj.save()

repo = librarymanager.repositories[0]
librarymanager.install_library(proj.path, repo, True)

proj.close()
```

18.3.7 Transitioning from Python 2 to Python 3

With Python version 3, Python developers introduced some incompatible changes and removed some obsolete functionalities. At this time, the Python community is still in the transitional phase from version 2 to version 3.

IronPython does not yet support Python 3, but it is being worked on. As the Python community no longer supports Python 2, we intend to up0grade to Python 3 as soon as it is supported by IronPython. Although we strive for a smooth transition, script writers should take care that their scripts are created in a future-proof style. For example, by using the expression `from __future__ import print_function`.

See also

- ➦ Python 2 or Python 3?
- ➦ New in Python 3

18.3.8 Comparison of IronPython and cPython

There are some small differences and Incompatibilities between IronPython and standard Python ("cPython"). Some are direct errors in IronPython and should be removed in future versions. However, others are considered "implementation details" and will remain. Some of them are very challenging topics.

The difference that is most obvious for users is the handling of strings. Original cPython has two different string types for byte strings and Unicode strings. This concept is similar to the data types STRING and WSTRING in IEC. IronPython simply uses .NET strings that are always Unicode-capable and use UTF-16 internally. However, IronPython implements a trick to hide the difference to cPython from the programmer. (Interesting is that the developers completely

reengineered string handling for the new Python version 3. A module was developed that is much closer to that of IronPython. Afterwards, only Unicode strings are used and there is an individual data type for handling raw bytes.)

Python modules that are written in C cannot be imported into IronPython because cPython uses internal data structures that are completely different from IronPython. Most of the standard library modules were reimplemented in IronPython. However, some modules (such as the TK interface) are not available as long as they are not ported to IronPython explicitly. On the other hand, IronPython provides access to .NET assemblies including the .NET framework (as shown above), which more than compensates for this feature.

While cPython uses reference counting and a deterministic garbage collector for cleaning up cyclic garbage, IronPython relies on the non-deterministic .NET garbage collector. In most cases, this difference does not matter. But when you open files or other resources from the Python standard library or the .NET framework, you should make sure to close them later. It is best to use the `with` statement with the Python context manager or .NET `IDisposable` instances.

See also

- ➦Information about Context Managers
- ➦Information about .NET IDisposable

19 Security

19.1 Security – Preface

Due to the increased networking of controllers and plants, potential threats are also quickly rising. Therefore, you should carefully consider all possible safety measures.

Security measures are absolutely necessary to protect data and communication channels from unauthorized access.

On the following help pages, you can learn more about the safety functions of I/O Engineering and the controller.

19.2 General information

Below, find some general information about security functions ("Security" measures). This information applies regardless of the use in I/O Engineering or a connected control.

Access protection with user management

As a means of protecting against unauthorized access to data, it is necessary to configure user accounts with specific access rights. Only a user with the credentials has access to the data or functions.

Creating passwords according to the general recommendations for achieving a high password strength is a significant contribution to security.

The following types of user management are roughly distinguished:

- Simple user management:
  To access data, only a password or the valid combination of user name and password has to be entered. This means that access can be only granted or denied. Graduated rights cannot be configured.
- Group-based user management:
  The access rights are assigned to user groups. Users who belong to a group can access the data or functions after entering the credentials with precisely these assigned and different rights.

Encryption, Signature

Encryption:

Encryption of data means: The data is converted into a non-readable form and can only be rendered readable again with a matching key. In the simplest case, the key is a password or a key pair.

There are two types of encryption methods:

- Symmetrical method: (the only type of encryption until the mid-1970s)
  Feature: Use of a secret key
  Advantages: Fast, easy encoding
  Disadvantages: The key has to be exchanged secretly!
- Asymmetric method:
  Feature: Use of a key pair (private/secret and public key)
  Advantages: A public key can be made available to anyone; authentication is possible with the key
  Disadvantages: Slow (approx. 1,000-10,000x slower than symmetric methods); complex coding; long key lengths

Key exchange is usually performed by asymmetric methods; encryption and decryption by symmetric methods.

Signature:

In order for the irrefutable ownership and integrity of a message to be verifiable, it should be provided with a signature. The following steps are common:

- Sender: Determines a unique hash value of the data (H)
- Sender: Encrypts the hash value with private key (He)
- Recipient: Also calculates the hash value and decrypts the He with public key and compares the two values. This allows the sender to be identified uniquely and verifies that the sender owns the private key.

In the case of asymmetric encryption, a public key contained in a certificate is first exchanged between the sender and the recipient. In addition, each participant needs a private key with which they can decrypt the data if they have the certificate. So if you want to access a certificate, you need a certificate AND a private key.

Hash methods are necessary for this:

- Hash method:
  Feature: Unique "fingerprint" of the data (for example, checksum of the data)
  As low a collision as possible (it is very difficult to find / construct two different data for a single hash value)

Certificates

To be able to assign the public key to an identity, it is usually embedded in a so-called certificate.

In certificate-based systems, each user receives a digital certificate. The certificate is used for digital identification. It contains information about the identity and the public key of the user. Each certificate is authenticated by an issuing authority, which in turn may be authenticated by higher authorities. The trust system of this PKI (Public Key Infrastructure) is strictly hierarchical. The common trust anchor is a so-called root certificate.

Contents of a certificate:

- Version
- Serial number
- Algorithm ID
- Issuer (authority or company)

- Validity from (not before) to (not after)
- Certificate owner (subject)
- Certificate owner key information (subject public key)
  - Public key algorithm
  - Public key of the certificate owner
- Unique ID of the issuer (optional)
- Unique ID of the owner (optional). The owner has a private key matching the public key
- Extensions
  - Purpose (extended key usage)
  - ...

The certificate consists of 2 parts / files:

- Public X.509 certificate (can be issued to anyone)
- Private key that matches the certificate or its public key only (must not be passed on!).

19.3 Security for the development system

In I/O Engineering, you can provide projects with access protection. In addition to a simple write protection for a project, a user management (credentials, access rights) and encryption using certificates should be used.

20 Use EPLAN data in I/O Engineering

General information on EPLAN

EPLAN Electric P8 is an electrical CAD system for planning electrical systems and machines. The CAD system is distributed by EPLAN GmbH & Co. KG.

The electrical design of the system is planned in detail during project creation in EPLAN. This includes, for example:

- System topology
- Bus topology
- Actuators & sensors
- Drives
- Wiring
- Control cabinet design
- Assigning the PLC variables to the inputs and outputs

EPLAN Electric P8 provides the option of exporting the data of the EPLAN project as an AutomationML file and then using it as a template in other software tools. Importing the AutomationML file into I/O Engineering can simplify project creation by transferring the configuration of the EPLAN project to the I/O Engineering project. This includes, for example

- Applying the system components to the ctrlX project.
- Applying the bus topology.

By importing EPLAN data, the project planning effort in I/O Engineering can be reduced.

20.1 Prerequisites

To be able to process EPLAN data in I/O Engineering, the add-on "Bosch Rexroth EPLAN" has to be installed.

The installation is carried out via the Add-On Installer in I/O Engineering, see also:

➦ Installing/uninstalling add-ons

The add-on installation has to be carried out separately for ctrlX I/O Engineering and ctrlX PLC Engineering.

Installing the add-on "Bosch Rexroth EPLAN"

1. Open the Add-On Installer, see: ➦ Command 'Add-on Installer'
   ➡ The Add-On Installer dialog opens and shows the “Installed” tab with already installed add-ons.
2. Go to the “ Browse” tab.
   ➡ A list of installable add-ons is displayed.
3. Select the add-on "Bosch Rexroth EPLAN".
   ➡ Details about the add-on are displayed on the right-hand side of the dialog.
4. Close the I/O Engineering application, otherwise the add-on cannot be installed in the next step.
5. In the Add-On Installer click on the “Install” button.
   ➡ The installation is started.
   ➡ Confirm the license agreements.

20.2 General information

EPLAN data import

EPLAN data contains information on the configured devices as well as information on the field bus configuration or I/O assignment. By importing the data into ctrlX I/O Engineering or ctrlX PLC Engineering only the data that can be used or is permitted in the respective project is imported into the ctrlX project.

With ctrlX I/O Engineering, the focus is on the field bus configuration and the devices involved.

With ctrlX PLC Engineering, it is primarily about PLC-relevant devices and the associated I/O assignments.

An automatic data check during the import ensures that only usable data is transferred to the project, see also:

➦ Checking the import data and conflict handling

Imported project data

The field buses contained in the AutomationML file and the devices (slaves) with the I/O cards are imported.

The I/O data also contains the symbolic names and comments defined in EPLAN, which are also transferred to the I/O Engineering project and can be subsequently viewed and edited.

In addition, the EPLAN device identifier (BMK) of the imported devices is stored in the I/O Engineering project. The BMK can be displayed in the properties of the respective device.

When EtherCAT I/O data is imported into ctrlX I/O Engineering, the bus configuration is mapped via the DataLayer realtime node in the ctrlX PLC Engineering project. The symbols and comments imported inctrlX I/O Engineering are also transferred if required. The BMK is always included.

Roundtrip engineering

Currently, only the import of EPLAN data into ctrlX I/O Engineering and ctrlX PLC Engineering is possible.

For later versions, the export of project data from I/O Engineering is also planned in order to enable roundtrip engineering between I/O Engineering and EPLAN Electic P8, in which project changes in I/O Engineering can be imported back into the EPLAN project.

20.3 Creating a new project using EPLAN import

This chapter describes the initial import of an EPLAN project into I/O Engineering.

The import is used to create a I/O Engineering project for the first time in order to reduce the project planning effort. Ideally, all required device nodes and substructures are created automatically and configuration settings are applied, such as I/O configuration, bus topology and addresses. A prerequisite for the import is that the devices to be imported are available in the device database of I/O Engineering and that the devices are also correctly typed in the EPLAN project, e.g. by using unique EPLAN device templates (macros).

Prerequisite

For an initial import, there must be no project available in I/O Engineering (no project in the device tree).

1. Open I/O Engineering without creating a project.
2. Execute the *“EPLAN → Create new project via EPLAN import...”* menu command, see also:

   ➥ Command “Create new project via EPLAN import...”

   ➡ A dialog to select the EPLAN import data is displayed.
3. Select the EPLAN file to be imported and assign a name for the new project to be created.
4. Confirm the dialog.
   ➡ The import is started.
   ➡ If conflicts occur during the import, a corresponding message is displayed, see:

   ➥ Checking the import data and conflict handling
5. After a successful import, the device tree is created automatically.

20.4 Import EPLAN data into an existing project

This chapter describes how to import EPLAN project data into an existing I/O Engineering project.

This form of data import is used to synchronize or adapt an existing I/O Engineering project to the configuration of an EPLAN project.

The aim of the import is to automatically merge the EPLAN project configuration with the project configuration in I/O Engineering. The two projects are compared, with the EPLAN project representing the target configuration. When merging different project configurations, device differences are handled according to the following rules:

- Identical parts are retained.
- Differing parts:
  - Devices are added if they have not yet been included in the I/O Engineeringproject.
  - Devices are deleted from the I/O Engineering project if they are not included in the EPLAN project.

Prerequisites

- A project has to be open in I/O Engineering.
- The control device node has to be selected.

1. Open the project in I/O Engineering.
2. Select the control device node in the device tree.
3. Execute the *"EPLAN → Import from EPLAN..."* menu command, see also
   ➦ Command "Import from EPLAN..."
   ➡ A dialog to select the EPLAN import data is displayed.
4. Select the EPLAN file to be imported.
5. Confirm the dialog.
   ➡ The import is started.
   ➡ If conflicts occur during the import, a corresponding message is displayed, see:
   ➦ Checking the import data and conflict handling
6. After a successful import, the device tree is automatically adjusted.

20.5 Checking the import data and conflict handling

When importing EPLAN data, the contained configuration is automatically checked for its applicability in I/O Engineering.

If discrepancies or conflicts are detected during the check, a dialog is displayed listing the individual conflicts and providing information on conflict resolution, as shown in the following image:

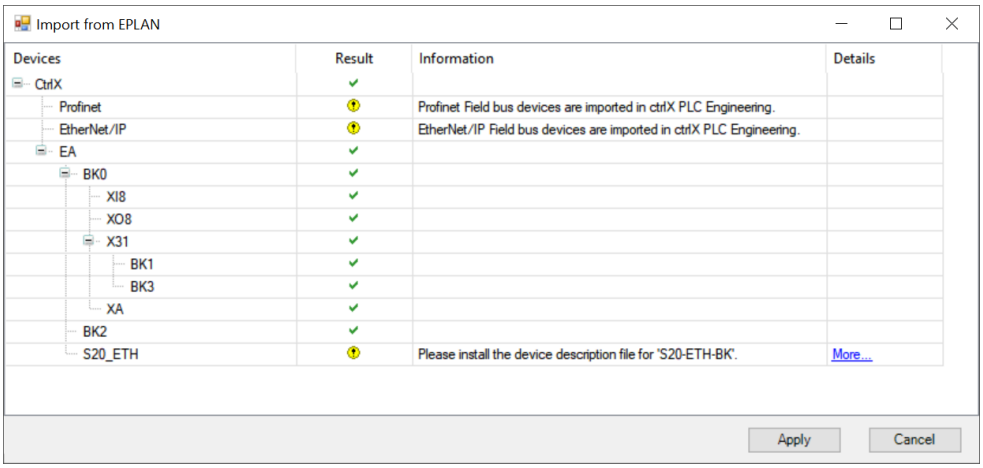


Fig. 6: Example: Import conflict in ctrlX I/O Engineering

Devices	Displays the devices in the import file.
Result	Displays the import status: (✔ = can be imported / ! = conflict)
Information	Describes the cause of the conflict and suggests solutions.
Details	Displays more detailed information about a conflict and the solution in a separate dialog (if available).

Device with multiple versions detected

If it is detected during import that several versions or device description files exist for a device, a query is issued asking which version should be used.

Examples of import conflicts

Conflict	Cause/measure
Please install the device description file for [device]	The device to be imported is not available in the device repository of I/O Engineering. Install the appropriate device description file in the device repository.
Devices are imported into PLC Engineering	The specified device cannot be imported into I/O Engineering, but has to be imported into PLC Engineering.
Devices are imported into I/O Engineering	The specified device cannot be imported into PLC Engineering, but has to be imported into I/O Engineering.
Control ID different	The ctrlX CORE control in the project was not created with the selected EPLAN export file, so an import is not possible. Open the project or create a new EPLAN project.
No suitable field bus found	The selected EPLAN export file does not contain a field bus for {0}. Please check the EPLAN configuration.
No ctrlX control	The selected EPLAN export file does not contain a ctrlX CORE control. Please check the EPLAN configuration.
Exception when reading the EPLAN export file	An exception occurred when generating the AutomationML model.

21 User interface

21.1 Generic Device Editor

21.1.1 Generic Device Editor – general

The generic device editor contains tabs for configuring device properties in I/O Engineering. Additionally there are device-specific tabs, so that the configuration editor consists of many different dialogs, depending on the device.

The editor opens after a double-click the device object in the device tree (“Devices” view).

You can make general settings for a device editor in the I/O Engineering “Options” in the “Device Editor” category. For example, you can show and hide the tabs of the generic device editor.

21.1.2 Tab – “Variables”

Prerequisite

The tab is only displayed for devices with process variables.

Function

The tab shows the variables in output and input direction, which are also available in the Data Layer.

The variables can also be renamed directly in the tab, provided this is supported by the field bus used.

Offline/online

The tab is available in both offline and online modes.

In online mode, the columns “Value” and “Error” are displayed. Furthermore, a note element is displayed when the application is in online mode.

Call

In the generic device editor of the device in the ctrlX I/O Engineering device tree, see ➥ Chapter 14.2 Device tree and device editor on page 45.

Tab elements

Element	Description
“Variable”	Name of the variable, represented in a tree structure. If it is supported for the respective field bus, the variables can be renamed here. The symbol indicates the variable type (input or output).
“Channel”	Name of the channel in the field bus configuration
“Type”	Variable data type
“Description”	Variable description
Additional columns in online mode:	
“Error”	Error that occurred while determining the variable value.
“Value”	Value of the variable.
“Prepared value”	Input field for the value to be written. For variables of type `BIT`, the inputs "`True`" and "`1`" are interpreted as the value `True`. All upper and lower case letters of "`True`" are taken into account. For all other numerical inputs, an input in the format "`0x1234`" is detected as a hexadecimal number. For variables that represent array types, a value can be entered as a comma-separated list of values in decimal or hexadecimal notation. Input in this element is only possible for variables with the direction "`Output`" . Also refer to: ➥ Writing prepared I/O variables (Prepared value)
Optional columns that can be displayed:	
“Bit length”	Data length of the variable in bit
“Direction”	Variable type display
“Data Layer URI”	Variable address in Data Layer

Writing prepared I/O variables (Prepared value)

The “write variables” command can be used to write the values in the “Prepared value” column can be written for all devices at the same time.

The command can be called up in the "Debug" menu or using the keyboard shortcut "Ctrl+F7".

Via the context menu, the column values can be exported to a csv file.

Please note the export behavior in offline or online mode:

- In online mode, all column values are exported.
- In offline mode, for the columns “Value” and “Error” no column values can be entered.
  In this case, the columns are exported but do not contain column values.

21.1.3 Tab – Communication

The tab “Communication” is used to configure the communication settings to the ctrlX device and provides the following functions:

- Scan the network for reachable ctrlX devices.
- Configuring network communication settings to the ctrlX device.
- Executing communication test.
- Displaying device information.
- Configuring the gateway settings (only for ctrlX PLC Engineering).

Call

The tab can be called as follows:

- By double-clicking on the control node in the device tree.
- Via the command “Communication settings...” in the context menu of the control node, also refer to:
  ➦ Command “Communication settings...”
  Alternatively, execute the command from the “Project” menu if the control node is selected in the device tree.

Description

The tab is in ctrlX PLC Engineering and ctrlX I/O Engineering and consists of a toolbar and an overview of the communication settings of the ctrlX device. For ctrlX PLC Engineering, the gateway settings can be configured additionally.

Toolbar	Description
“Scanning network...”	The command scans the network for ctrlX devices and lists accessible devices in a selection dialog.
“Device”	Contains option settings of the ctrlX device: Options: • “Save communication settings to the project” (non-deselectable) • “Edit communication port” disabled in factory setting) If this option is enabled, the communication port can be set in the configuration window.

Communication overview	Description
“PLC gateway”	Field to configure the PLC gateway. GUI elements: • Combobox to select the PLC gateway. • Display of the configured IP address and port number. • Icon to display the communication state: – : Communication OK – : Communication not determined – : Communication disturbed

Communication overview	Description
“ctrlX core”	Field to configure the control (communication to the ctrlX device) GUI elements: • Combobox to select the control. When expanded, the combo box displays all available ctrlX CORE controls on the network. The list of available controls can be refreshed using the “Scanning network...” command in the toolbar. The button to apply the selected control first performs a communication test and then applies the required settings (IP address and ports). Data can also be applied if the ctrlX device is currently not accessible. • Communication test Click on to test the communication to the control. • Icon to display the communication state: – : Communication OK – : Communication not determined – : Communication disturbed • Displays when the control is connected: – ctrlX device name – System status or operating state of the ctrlX device – Processor architecture of the ctrlX device, e.g. amd64 – Serial number – Device type – App version – Available licenses – Available device ports

21.1.4 Tab – Status

The generic device editor tab displays status information, such as the PLC operating status and specific diagnostic messages of the ctrlX device.

21.1.5 Tab – Information

The generic device editor tab displays general information that originates from the device description file and may include the following information:

- Device designation
- Vendor
- Device category
- Version
- Order number
- Description
- Device illustration

21.1.6 EtherCAT Master – Tabs

Tab – “General”

Call

In the generic device editor of the EtherCAT master.

To open the device editor, double-click on the EtherCAT master object in ctrlX I/O Engineering device tree, see ➦ Chapter 14.2 Device tree and device editor on page 45.

Function

In the tab “General”, the general settings of the EtherCAT master can be configured.

Elements of the tab

“General”	
“Cycle time”	Bus cycle time of the EtherCAT communication.

“State machine”	
“Master state after download”	Master target state after configuration download.
“Master state after restart”	Target master state after control startup.
“Slaves have to reach the master state”	If this option is enabled, **all** states have to reach the master state. If this option is disabled, the master can execute a bus startup (e.g. target state OP) if not all slaves have reached the target state or if some slave are missing at the EtherCAT bus.

“Options”	
“Use LRW instead of LWR/LRD”	If the option is enabled instead of separate read (LRD) and write (LWR) commands, combined read/write commands (LRW) are used. **Caution!** • **Not all slaves support LRW** • **LRW cannot be used in case of cable redundancy**

Tab – “Sync Unit Assignment”

Call

In the generic device editor of the EtherCAT master in ctrlX I/O Engineering device tree, see ➦ Chapter 14.2 Device tree and device editor on page 45.

Function

In the tab “Sync Unit Assignment”, SyncUnits can be created manually and assigned to slaves.

The preset SyncUnit “default” is not always available and cannot be deleted. All slaves are assigned to the SyncUnit “default”. Further SyncUnits can then be created and assigned to slaves.

Elements of the tab

“Device name”	Name of the slave of the device tree
“Sync Unit”	Slave assignment to a SyncUnit. Double-click on the SyncUnit field of a slave (on the right of the device name) to select one of the configured SyncUnits.
Add	Use this command to add a new SyncUnit. Enter a name in the text field.
Delete	The command deletes the selected SyncUnit. The preset SyncUnit “Standard” cannot be deleted. If a SyncUnit unit is deleted, but this SyncUnit is still assigned to slaves, these slaves are again assigned to the SyncUnit “Standard”.

Configure Tab – “Distributed Clocks”

Call

In the generic device editor of the EtherCAT master in ctrlX I/O Engineering device tree, see:

➥ Chapter 14.2 Device tree and device editor on page 45

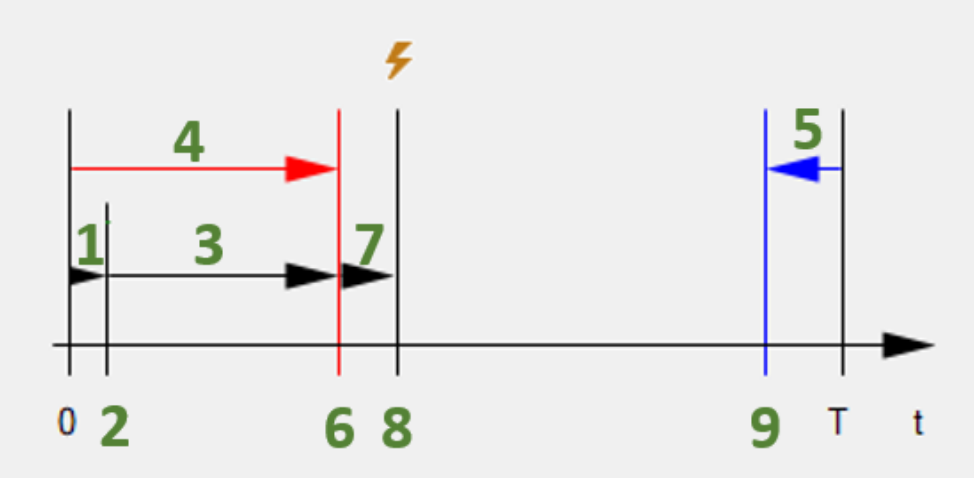
Function

Select the settings of the “Distributed Clocks” of the EtherCAT master in this tab.

When logged in (online), values are displayed that are read out by the control.

Tabs/description

Surface element	Description
Range: “Settings”	“DC mode:” • “Auto”: – Default setting – Automatic selection of the synchronization mode • “Free Run”: – Cyclic communication without clock synchronization – Distributed Clocks are not controlled • “Bus Shift”: – Distributed Clocks active – Slaves are synchronized using the EtherCAT master – The EtherCAT master sets the system time and writes it to the reference clock of the bus (shifting the bus time) • “Link Layer Reference Clock”: – Distributed Clocks active – Slaves are synchronized using the EtherCAT master – The clock integrated in the EtherCAT master is used as reference clock of the synchronization. The setting guarantees the highest synchronization precision but requires special hardware **WARNING** Do NOT use the “Link Layer Reference Clock” if you are simultaneously synchronizing to an external clock, e.g. an NTP server. This can result in unpredictable behavior!

Surface element	Description
	“Sync window monitoring” If this option is enabled, the synchronization of the Distributed Clocks is monitored in all slaves. The “system time difference” (tab `0x092C`) is read out cyclically by each slave. The data entered in the “Sync window:” specifies the maximum system time deviation allowed (reference clock). If - for all slaves - the system time difference is smaller than the monitoring window, the synchronization is considered to be adjusted (“DC in Sync”).
	“Continuous propagation compensation” If the option is enabled, the runtime of the EtherCAT telegram is cyclically measured at the bus. This allows to adapt the runtime compensation continuously. For the EtherCAT bus configurations without a topology change in the operating mode, do not disable this option.
Range: “Timing”	• “Send Offset” Offset between cycle/task start and sending the frames. • “Sync Offset” Offset between sending frames and DC_BASE. • “Output Shift Time” Offset between cycle/task start and DC_BASE. • “Input Shift Time” Offset between cycle /task start and locking of the input-based slaves. The following diagram shows how the values relate to each other:  0 Start of the cycle / start of the task 1 Send offset 2 Frames are sent 3 Sync offset 4 Output shift time 5 Input shift time 6 DC_BASE 7 Switching time of the slave (output-based) 8 Sync0 of the slave (output-based) 9 Latch from input-based slaves T Cycle end/start of next cycle

Surface element	Description
Range: “Master”	Only displayed if the option “Show online data” is enabled. “System time”: System time of the EtherCAT Master. Status display: “DC in sync” applies if the system time difference is smaller than the monitoring window for all slaves
Range “Slaves”	Only displayed if the option “Show online data” is enabled. • “Status” “Green”: OK “ red”: An error was reported, see the “Bus diagnostics” tab. • “Name” Configured device name of the slave. • “Device type” Device type description of the slave (from ESI). • “Address” EtherCAT address • “State” EtherCAT slave state • “System time differenze” System time difference in ns (ESC register `0x092C`) (this value is only valid if “Distributed Clocks” is configured) The first slave at the bus with enabled “Distributed Clocks” provides the reference clock.

Tab – “Bus overview”

Prerequisite

The tab is only displayed if the setting “Show online data” in der ctrlX I/O Engineering toolbar.

Call

In the generic device editor of the EtherCAT master in ctrlX I/O Engineering device tree, see ➦Chapter 14.2 Device tree and device editor on page 45.

Function

The tab is used for display and control the EtherCAT master bus state.

Use the interfaces “Init” / “Pre-OP” / “Safe-OP” / “OP” to switch the bus state of EtherCAT master. The current bus state is also displayed.

Displays in the “Message:” field

Text	Description
“Master not in OP”	The EtherCAT master is not in state
“Slaves not in Master state”	One or multiple slaves report(s) an error (e.g. AL status code).
“Link not available”	No link to EtherCAT master provided
“DC not in-sync”	The synchronization of the distributed clocks (DC) is not within the configured monitoring window (only if DC “Distributed Clocks” is enabled).

Text	Description
“Topology not ok”	The configured slaves do not match the slaves at the bus, e.g. slaves are missing or the configuration of the topology does not match the real topology at the bus. Also refer to the following table of the EtherCAT state “not connected”. This diagnostics is also set if there are more slaves at the bus than configured.
“Master not in target state”	The requested EtherCAT master status could not be reached
“Slaves not in Master state”	One or multiple slaves are not in the master state.
“Slaves in error state”	One or multiple slaves report(s) an error (e.g. AL status code).
“Master not in OP”	The EtherCAT master is not in state OP.
“Running”	The EtherCAT master and all slaves are in a valid cyclic communication (state OP).
“No license available”	An error occurred when the basic license to operate the EtherCAT Master app is checked. Check whether there is a basic license available on your control.
“Port not found”	The EtherCAT master configuration does not match hardware. The configured Ethernet Port does not exist on the hardware.

Two optional views can be toggled under “Slaves”.

- All configured devices are displayed if “Configuration” is selected.
- All currently available slave devices at the bus are displayed if “Online” is selected. This view is especially useful if a topology error was reported, i.e. the configured topology in the device tree does not match the cabling of the slaves at the bus.

The columns of the table have the following meaning:

Status	“Green”: OK, “Red”: An error was reported. The slave is not connected or refer to the column " Diagnostics " "¥
Name	Configured device name of the slave. An unconfigured slave is shown with the name “Slave_x” in the online view.
Device type	Device type description of the slave (from ESI). A slave, whose ESI is not installed in the IndraWorks device database, is shown as “Unknown device” in the online view.
Address	EtherCAT address

State	EtherCAT State of the slave Note: The state “not connected” means that the configured slave could not be assigned to any device at the bus, i.e. the device is either physically not connected to the bus or the topology position of the device at the bus does not match the configuration. Procedure in case of a topology error: • Compare the configuration of the topology in the device tree with to “online” view of the slaves
Diagnostics	Diagnostics, e.g. EtherCAT state error.

Switch slave state individually

The slave state can be switched individually via the context menu.

Select the corresponding slave in the list. Open the context menu via right-click and select the desired state: *“Changing state → ([Init], [Pre-OP], [Safe-OP], [OP])”*

Tab – “Diagnostics”

Prerequisite

The tab is only displayed if the setting “Show online data” in der ctrlX I/O Engineering toolbar.

Call

In the generic device editor of the EtherCAT master in ctrlX I/O Engineering device tree, see ➦Chapter 14.2 Device tree and device editor on page 45.

Function

The tab is used to display messages provided from the CoE object “Diagnosis History” (0x10F3) of the EtherCAT master.

Elements of the tab

Acknowledging/deleting messages

The diagnostic log is operated as ring buffer and can save up to 250 messages (device-specifically it might be less).

Messages can be acknowledged and deleted via the following buttons in the upper area of the tab:

- “Acknowledge messages”
- “Delete messages”

Settings

There are different optional settings that can be set in the slave. If a specific option is not supported by the slave, a writing error is reported while disabling/enabling (parameter value beyond the valid range).

- ☑ “Activate info”
  Setting whether or not information is to be entered in the logbook
- ☑ “Activate warning”
  Setting whether or not warnings are to be entered in the logbook
- ☑ “Activate error”
  Setting whether or not error messages are to be entered in the logbook
- ☑ “Activate emergency”

Setting whether or not the slave sends new diagnostics as CoE emergency messages to the master.

- ☑ “Activate AcknowledgeMode”
  Setting options between “OverwriteMode” and “AcknowledgeMode”.
  In the “OverwriteMode”, if the message buffer is full, a new message overwrites the oldest message in the buffer irrespective of whether the messages were acknowledged.
  Unacknowledged messages are not overwritten in the “AcknowledgeMode”, i.e. if the message buffer is full, the latest messages are not saved and thus lost.

Table list of messages

Column	Description
“Category”	Message category: **Info** / **Warning** / **Error** There is no default setting on how incoming and outgoing errors are reported. Refer to product description of the device. Acknowledged messages are shown by a greyed out icon.
“Time”	Time stamp reported for the device message
“Code”	Diagnostic code
“Text”	The diagnostic texts are taken from the device description (ESI) of the device.

Tab – “Master statistics”

Prerequisite

The tab is only displayed if the setting “Show online data” in der ctrlX I/O Engineering toolbar.

Call

In the generic device editor of the EtherCAT master in ctrlX I/O Engineering device tree, see: ↪ Device tree and device editor.

Function

The tab displays information and error counters about the telegrams on the EtherCAT fieldbus.

Tab elements

The tab is divided into two table sections:

- **Frames**
  Shows sent and lost frames, divided into cyclic and acyclic communication
- **Mailbox requests**
  Shows all sent mailbox requests, subdivided by mailbox and transmission direction

Reset counter readings

Via the “Reset counters” button, the counters in the respective table areas can be reset separately.

Notes on the evaluation

Lost frames are a first indicator for communication problems, but can also occur by plugging or unplugging a bus cable or by switching off/on a slave or the controller.

If the number of lost frames increases continuously during regular operation, check the Slave statistics, see:

➦ Tab – “Slave statistics”

For further notes and details on the procedure, please refer to the documentation of the "EtherCAT Technology Group":

➦ https://www.ethercat.org/.../EtherCAT_Diagnosis_For_Users.pdf

Tab – “Slave statistics”

Error counters to the projected and accessible EtherCAT slaves are displayed in the “Slave statistics” tab.

By analyzing the error counters in connection with the current topology, the position in which the error occurred can be determined.

Tab elements

The upper section, the following buttons are available:

- “Reset error counter” button
  The error counters can be reset to 0 via the button.
- “Advanced” button
  Shows additional setting options to refresh the view.
- Link to the EtherCAT Technology Group documentation:
  ➦ https://www.ethercat.org/download/documents/EtherCAT_Diagnosis_For_Users.pdf

Information about the available EtherCAT slaves is displayed in the lower section in a table:

- **Name**
  Slave name
- **Address**
  Slave address
- **Invalid Frames / Port**
  Content of the ESC register 0x0300, 0x0302, 0x0304, 0x0306
- **RX Errors / Port**
  Content of the ESC register 0x0301, 0x0303, 0x0305, 0x0307
- **Forwarded RX Errors / Port**
  Content of the ESC register 0x0308, 0x0309, 0x030A, 0x030B
- **Proc. Unit** (processing unit error counter)
  Content of the ESC register 0x030C
- **PDI Errors**
  Content of the ESC register 0x030D
- **State**
  Operating state of slave
- **Diagnostics**
  Slave diagnostic messages

Configure Tab – “Slave to slave”

Call

In the generic device editor of the EtherCAT master in ctrlX I/O Engineering device tree, see ➦ Chapter 14.2 Device tree and device editor on page 45.

Function

In the tab "Slave to Slave", connections can be configured. For this purpose PDOs or PDO entries are connected to each other.

The source data is sent to the destination with a delay of one cycle after the data has been copied in the master.

Tab elements

The tab is divided into three segments:

- **Source**
  Displays all inputs in the process data, grouped by slaves and modules.
  Both a PDO and a PDO entry can be selected for a connection.
- **Destination**
  Displays all inputs in the process data, grouped by slaves and modules.
  Only PDOs / PDO entries that match the selected source in terms of length can be selected.
- **Connections**
  Shows a list of configured connections:
  - Check box ☑ for enabling/disabling a connection.
    If the connection is disabled, no data is copied between the slaves and no verification is performed during the compilation.
  - Name and PDO entry of the source module.
  - Name and PDO entry of the target module.
  - Bit length display.
  - [ ] Button or link to the tab page “Variables” where the current variable values are listed.
  - Buttons to delete or move a connection.

The configured connections can be exported to a csv file via the context menu.

Tab – “AoE (EtherCAT Master)”

In the tab “AoE”, the ADS communication settings of the EtherCAT master for ADS over EtherCAT (AoE) are selected.

The tab is divided into two sections:

- **Upper section**: AoE configuration for the EtherCAT master
- **Lower section**: Overview on all slaves supporting AoE

When creating the EtherCAT master and the subordinate slaves in the project, AoE is automatically enabled and unique Net Ids are allocated.

If required, the configuration can be adjusted.

Call

In the generic device editor of the EtherCAT master.

To open the device editor, double-click on the EtherCAT master node in the ctrlX I/O Engineering device tree, see ➥ Documentation

Upper area/AoE configuration

“Settings”	
“Master”	Enables the AoE master functionality (default: enabled)
“Net ID”	Net ID of the EtherCAT master

Lower section/slave overview

The slave overview in the table lists all slaves, supporting AoE, including the AoE Net ID.

The Net ID can be edited in the table.

Also refer to:

- Acyclic communication (mailbox) - Basics ADS over EtherCAT

Tab – "CoE"

Call

In the generic device editor of the EtherCAT master in ctrlX I/O Engineering device tree, see Chapter 14.2 Device tree and device editor on page 45.

Function

In this tab tab, the CoE objects of the EtherCAT master for diagnostic purposes are now displayed.

Elements of the tab

In the right upper drop-down list, select the following settings:

- "Description from ESI"
  The slave description is read out from the device description (ESI)
- "Description from master (online)"
  The slave description is read out of the device (online)

Select the objects to be displayed in the left upper drop-down list:

- "All Objects"
  (RxPDO) objects that can be configured as process data (output data)

The specifications are output in the following table:

Column	Description
"Index"	Via an index, the object is identified. If an object contains subelements, they are written by the subindex
"Name"	Object name
"Value"	Depending on the type, the value can be a text or a number. The number of subindices is displayed for objects with subelements
"Access"	Describes which access to the object is possible: • "RO": Read-only • "RW": Read and write • "WO": Write only
"Type"	The data type of the object

Tab – "EoE"

In this tab, a virtual Ethernet port can be enabled in the EtherCAT master.

The tab is divided into three sections:

- Enabling and configuring the virtual Ethernet port at the EtherCAT master
- Activating and configuring the automatic address allocation for slaves supporting EoE
- Overview on all slaves supporting EoE

Calling the tab

In the generic device editor of the EtherCAT master in ctrlX I/O Engineering device tree, see ➦ Chapter 14.2 Device tree and device editor on page 45.

Virtual network adapter

The EoE communication uses a virtual network adapter on the ctrlX CORE control. The system adds the network adapter automatically to the control if the EoE communication is enabled in the EtherCAT master configuration.

If the IP address is set for the EtherCAT master, the IP routing to the engineering port of the control is enabled at the same time.

In the ctrlX CORE web interface, the network adapter is shown in the window “Network interfaces”. Its name is “eoe0”.

The network adapter settings “eoe0” must not be changed in the window “Network interfaces” (read only).

Elements of the tab

GUI element	Description
“Virtual Ethernet port”	☐: The virtual Ethernet port is disabled on the control (state upon delivery). ☑: The virtual Ethernet port is disabled on the control.
“IP address”	Input field of the IP address of the virtual Ethernet ports
“Subnet mask”	Input field of the subnet mask of the virtual Ethernet ports

Address assignment (slaves)

GUI element	Description
Input field “First IP address”	Defines the area of the IP addresses from which the EoE slaves are assigned
Input field “Last IP address”	
Input field “Subnet mask”	Defines the subnet mask set for each slave
Input field “Standard gateway”	Defines the standard gateway which is set for each slave
Input field “DNS server”	Defines the DNS server set for each slave
Button“Reassign”	The slave addresses are reset and re-allocated according to the definition. By using this functions, inconsistencies can be fixed collectively or the IP address range can be moved.

Address allocation behavior

- Address allocation active (default setting)
  - When enabling EoE of type “IP port”, the IP addresses are automatically allocated from the configured area
  - When adding a slave to the project, EoE is automatically enabled if EoE is supported
- Address allocation disabled
  - When adding a slave, EoE is disabled
  - When enabling EoE, “0.0.0.0“ is specified as default IP address.
    The correct addresses have to be configured individually.

Slave overview

This tab area shows a table, specifying all slaves supporting EoE with their EoE configuration.

The slave configuration can be edited in the table.

Related documentation

- ➦ Acyclic communication (mailbox)

Tab – “Information”

Call

In the generic device editor of the EtherCAT master in ctrlX I/O Engineering device tree, see ➦ Chapter 14.2 Device tree and device editor on page 45.

Function

The tab shows general information of the device description file:

- Device name
- Vendor
- Category
- Version
- Description

21.1.7 EtherCAT Slave – Tabs

Tab – “IO-Link”

Prerequisite

The tab is only displayed if the setting “Show online data” is activated in the I/O Engineering toolbar.

Call

In the generic device editor of the EtherCAT slave.

To open the device editor, double-click on the EtherCAT slave object in ctrlX I/O Engineering device tree, see ➦ Chapter 14.2 Device tree and device editor on page 45.

Function

Objects on IO-Link devices can be read and written in the tab.

Elements of the tab

Element	Description
“IO-Link Port”	Number of the port to which the IO-Link device is connected
“Index”	Index of the IO-Link object
“Subindex”	Subindex of the IO-Link object
“Data”	Editor for reading and writing data (Hex)
“Error code”	Display of pending error codes during transmission
“Cmd Result”	Response of the slave when executing the command (manufacturer-specific)
“Button „Read“”	Reads the data from the slave and displays it in the “Data” editor
“Button „Write“”	Writes the data from the editor to the slave

Please note the documentation of the respective slave regarding the meaning of the parameters.

Related topics

➦ Web-Documentation "EtherCat Master App – Acyclic communication (mailbox)"

Tab – “General”

Call

In the generic device editor of the EtherCAT slave.

To open the device editor, double-click on the EtherCAT slave object in ctrlX I/O Engineering device tree, see ➦ Chapter 14.2 Device tree and device editor on page 45.

Function

In the tab “General”, the general settings of the EtherCAT slave can be configured.

Elements of the tab

“Address”	
“AutoInc address”	**Auto increment address**: Topological address of the slave as negative auto increment (16 bits), for example: 0, -1 (0xFFFF), -2 (0xFFFE) and -3 (0xFFFD) etc. The AutoInc address is automatically specified by the configurator in the sequence in which the slaves were created in the device tree (from top to bottom).
“EtherCAT address”	EtherCAT address of the slave, also called “Fixed Physical Address“. The address is not saved in the slave, but the master assigns this address to the slave upon bus startup. **Note**: The EtherCAT address can currently not be changed. The configurator specifies it automatically.

“Additional”	
“Enable expert settings”	☑: If the option is enabled, additional configuration settings can be selected such as startup checks, timeouts, watchdog and identification. The “Expert process data” tab is enabled.

“Distributed Clock”	
“Select DC”	Dropdown list with predefined settings for Distributed Clocks from the device description file (ESI) of the slave (ESI)
“Enable”	☑ If the option “Enable expert settings” is selected, a user-defined DC setting can be configured
“Enable Sync0”	☑: Use Sync0 Event in EtherCAT slave
“Enable Sync1”	☑: Use Sync1 Event in EtherCAT slave
“Sync unit cycle”	The dropdown list can be used to set a factor for multiplying the EtherCAT communication cycle time. The set cycle time is displayed in “Cycle time”.
“User-defined”	☑: In case this option is enabled, the cycle time (µs) can be manually set in “Shift time”

“Startup Checking” **(only in case of enabled option:** “Enable expert settings”**)**	
“Check vendor ID”	☑: Upon bus startup, it is checked whether the vendor ID of the configured slave matches the physical device
“Check product ID”	☑: Upon bus startup, it is checked whether the product ID of the configured slave matches the physical device
“Check revision number”	☑: Upon bus startup, the revision number of the configured slave is checked with regard to the value of the physical device according to the selection of the drop-down list. Partially different checks can be selected for low word (LW / bits 0-15) and high word (HW / bits 16-31) of the revision number.
“Download expected slot configuration”	☑: For a slave with “modular device profile” (e.g. Rexroth S20-EC-BK), expected or configured slot/ module configuration is loaded to the device to be able to compare them to the actual module configuration in the device.

“Timeouts” **(only in case of enabled option:** “Enable expert settings”**)**	
“SDO access”	Timeout for SDO accesses upon bus startup
“I->P”	Timeout for state transition from Init to Pre-OP
“P->S/S->O”	Timeout for state transition from Pre-OP to Safe-OP and from Safe-OP to OP

“Watchdog” **(only in case of enabled option:** “Enable expert settings”**)**	
“Set multiplier”	The multiplier (register 16#400) divides the local cycle to specify the trigger for the watchdogs: (1 / 25 MHz) * (multiplier + 2) With the multiplier standard setting 2498, the interval 100 µs results (computation: (1 / 25 MHz) * (multiplier + 2) = 40 ns * 2500 = 100 µs) Example of PDI watchdog: With the factor 1,000 results a watchdog timeout of 100 ms. A manual adaptation of the multiplier is for example required for very slow EtherCAT communication cycle times.
“Set PDI watchdog”	The “PDI watchdog (register 16#410)” monitors the process data communication from the ESC to the application (e.g. local micro controller) of the slave
“Set SM watchdog”	The “SM watchdog (register (16#420)” monitors the cyclic EtherCAT communication. The watchdog is reset after each successful EtherCAT process data communication. If the watchdog is enabled and longer than the monitoring time set, no EtherCAT process data communication takes place and the outputs are set to the safe state.

“Identification” **(only in case of enabled option:** “Enable expert settings”**)**	
“Disabled”	◉: There is no identification

“Identification” **(only in case of enabled option:** “Enable expert settings”**)**	
“Configured station alias”	◉: Identification via “Configured station alias” Normally, the value of the station alias is written to the slave EEPROM via the remote assignment. The slave then loads this value to its register ADO 0x0012 when booting (ADO = Address Offset).
“Explict device identification”	◉: Identification via “Explict device identification” Upon request, the slave informs the master on its identification value in the AL status code register (ADO 0x0134).
“Data Word”	◉: Identification via “Data Word” (also called “Input Word”) The identification value is expected at a specific register address offset (ADO) specified from the device description file (ESI) of the slave and arbitrarily set if required in the “ADO” field (refer to the product description of the slave device).
“Value”	Command value specified for the selected method. This value has to match the value set in the slave device.

Tab – “SyncManager”

Prerequisite

The tab is only displayed if the setting “Enable expert settings” is enabled in the tab “General” tab, see ➦ Chapter Tab – “General” on page 111.

Call

In the generic device editor of the EtherCAT slave in ctrlX I/O Engineering device tree, see ➦ Chapter 14.2 Device tree and device editor on page 45.

Function

In this tab, the SyncManager configuration for the selected EtherCAT slave can be displayed and modified manually.

Note that these are expert settings which are not usually required for standard applications!

To change the configuration, select a table entry and click on “Edit”.

Elements of the tab

Display of SyncManager configuration in a table

Table entries	Description
“Start address”	SyncManager Start address
“Length (bytes)”	SyncManager Length in `BYTE`: Default 512 `BYTE` (min 0 <> max 3999999999)

Table entries	Description
“Flags”	Display of the combination of the enabled flags Description of the abbreviation: • **1** or **3** = Number of buffers • **W** = Write • **R** = Read • **E** = SyncManager enabled • **P** = Interrupt to PDI enabled • **F** = Interrupt to EtherCAT enabled
“Type”	**SyncManager Type**: • Inputs • Outputs • Mailbox In • Mailbox Out

Command “Edit”

The command is available once a table column is selected.

The command opens the Edit SyncManager dialog to configure the SyncManager settings.

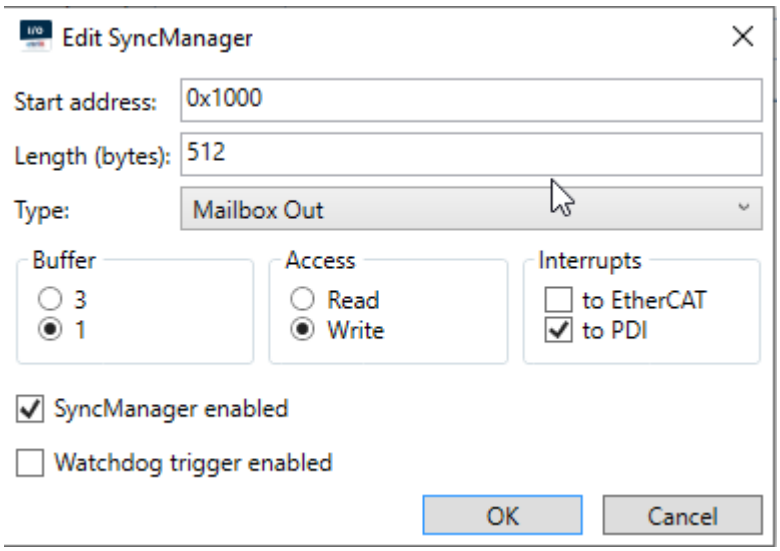


Fig. 7: EtherCAT slave Dialog – Edit SyncManager

Configuration example:

In case of SoE devices such as Rexroth IndraDrive, the SyncManager is set as default with “Buffer 1” (according to the ETG specification). In the "Free Run" operating mode, this can result in "Working-Counter" errors.

Thus, the SyncManager setting “Buffer 3” is recommended for cyclic data (inputs/outputs) for "Free Run" operation.

Tab – “Expert Process Data”

Prerequisite

The tab is only displayed if the setting “Enable expert settings” is enabled in the tab “General” tab, see ➥ Chapter Tab – “General” on page 111.

Call

In the generic device editor of the EtherCAT slave.

To open the device editor, double-click on the EtherCAT slave object in ctrlX I/O Engineering device tree, see ➦ Chapter 14.2 Device tree and device editor on page 45.

Function

The tab “Expert Process Data” tab provides a detailed view and an expert configuration of the process data, which are also displayed in the “Process data” dialog. Additionally, the download of the PDO assignment and the PDO configuration can be enabled in this tab.

Elements of the tab

“Sync Manager”	
“SM”	Sync Manager list
“Size”	Sync Manager data size
“Type”	PDO type (PDO: process data object)

“PDO List”	
“Index”	PDO index (or IDN for SoE)
“Size”	PDO size
“Name”	PDO name
“Flags”	• “F” Fixed content: The contents of the PDO are fixed and cannot be modified. It is not possible to add entries to the PDO content. • “M” Mandatory: The PDO is necessary and cannot be deactivated in the PDO assignment. • “V” Virtual PDO: Reserved for future use
“SM”	ID of the Sync Manager to which the PDO is to be assigned to
Via the commands in the command bar (Add, Delete, Edit), entries can be added or existing entries can be edited or deleted. The dialog “Edit PDO List” opens in Add and Edit, see ➦ Chapter 21.4.5 Configure dialog – Edit PDO List on page 166.	

“PDO Assignment”
PDO list assigned to the selected sync manager (in the “Sync Manager” field). In case the checkbox ☑ is enabled , PDOs are enabled and the I/O channels are generated (comparable to the object selection in the “Process data” dialog)

“PDO Content”
Shows the PDO content selected in the “PDO List” field. Add new entries or edit or delete existing entries by executing the commands in the command bar (Add, Delete, Edit). Requirement: the PDO flag “Fixed content” in the “PDO List” must not be active. You can change the PDO order using the “Move Up” and “Move Down” commands.

“Download”	
“PDO Assignment”	☑ CoE commands for initializing the Sync Manager objects (0x1Cxx) are generated and written to the slave.
“PDO configuration”	☑CoE commands for PDO mapping (0x16xx or 0x1Axx) are generated and loaded to the slave. Usually, the default values are taken from the ESI file and the device has to support it. For example, if a device has a fixed configuration, then these commands are seen as flawed.

“Load PDO Info from the device”
Note: The function is not supported by the Rexroth EtherCAT Master. The current PDO configuration is read out of the slave and is entered in the configuration. Thus, the list at the top and bottom right are completely deleted and filled with read data. This is recommended if the ESI file is incomplete or if the configuration is only available in the slave.

Tab – “Process Data”

Call

In the generic device editor of the EtherCAT slave.

To open the device editor, double-click on the EtherCAT slave object in ctrlX I/O Engineering device tree, see ➥ Chapter 14.2 Device tree and device editor on page 45.

Function

The tab shows the process data for slave inputs and outputs. The device data is part of the device description file (ESI) of the slave.

Elements of the tab

“Select the Outputs”
The table shows the slave outputs defined by Name, Type and Index. ☑ In case of enabled slave outputs, they can be used in the tab“I/O image”. Output objects can be excluded by other (enabled) output objects (these interdependencies are specified in the device description file of the slave).

“Select the Inputs”
The table shows the slave inputs defined by Name, Type and Index. ☑ In case of enabled slave inputs, they can be used in the tab“I/O image”. Input objects can be excluded by other (enabled) input objects (these interdependencies are specified in the device description file of the slave).

Tab – “Startup Parameters”

Prerequisite

The slave device has to support CoE (CAN application layer over EtherCAT) or SoE (Servo drive profile over EtherCAT) so that SDOs (Service Data Object) or IDNs (Identifikations-Nummern) can be entered.

Call

In the generic device editor of the EtherCAT slave.

To open the device editor, double-click on the EtherCAT slave object in ctrlX I/O Engineering device tree, see ➥ Chapter 14.2 Device tree and device editor on page 45.

Function

In the tab “Startup Parameters”, parameters are defined which are written to the slave device during the bus startup (during the transition from Pre-OP to Safe-OP).

Some modules that are inserted below a slave have their own start parameters.
These parameters are also displayed in the list described here but cannot be edited.
The parameters can be modified in the editor of the corresponding module.

Elements of the tab

Table column	Description
“Line”	Line number
“Index:Subindex”	Index and subindex of the object (CoE)
“Name”	Name of the parameter
“Value”	Value of the parameter
Bit Length	Bit length of the parameter
Comment	Comment/input field

Command bar	
“Add”	Opens an input dialog to add parameters of the start parameter table (SDO or IDN)
“Edit”	Opens a dialog to edit the selected parameter (SDO or IDN)
“Delete”	Removes the selected parameter
“Move Up”	Moves the selected line upwards by one line
“Move Down”	Moves the selected line downwards by one line

Adding/editing IDN (SoE)

An entry can be selected from the object directory (requirement: the parameters are described in the device description file (ESI) of the slave device) or a parameter can be defined manually by the input fields.

Table column/input field	Description
“Idn”	ID number
“Base Value”	Basic value of the IDN, can be edited (open with double-click)
“IDN”	Parameter type: S or P-parameters
“PSet”	Parameter block
“Offset”	Data block number
“Bit Length”	Bit length
“Value”	Value
“Is List”	If the option is selected, Sercos lists can be specified. The option is located in the dialog for selecting the object dictionary, see ➥ Example "1 byte list" Lists are specified as byte arrays in the following format: `<Is length (LowByte)>,<Is length (HighByte)>,0,0,<Byte1>,<Byte2>,<Byte3>...`

Adding/editing SDO (CoE)

An entry can be selected from the object directory (requirement: the objects are described in the device description file (ESI) of the slave device) or a parameter can be defined manually by the input fields.

Table column/input field	Description
“Flags”	Display of the access flag: ● “RW” (read write) = read and write ● “RO” (read only) = read-only ● “WO” (write only) = write only
“Base Value”	Basic value of the object, can be edited (open with double-click)
“Name”	Name of object
“Index”	Object index (16#)
“SubIndex”	Subindex of the object (16#)
“Bit Length”	Bit length
“Value”	Value
“Byte Array”	The object is a list

Example "1 byte list"

On the example parameter "Application type" the example string „Hulsch“ is to be written (1 byte list).

The string "Hulsch" corresponds to the following ASCII characters:

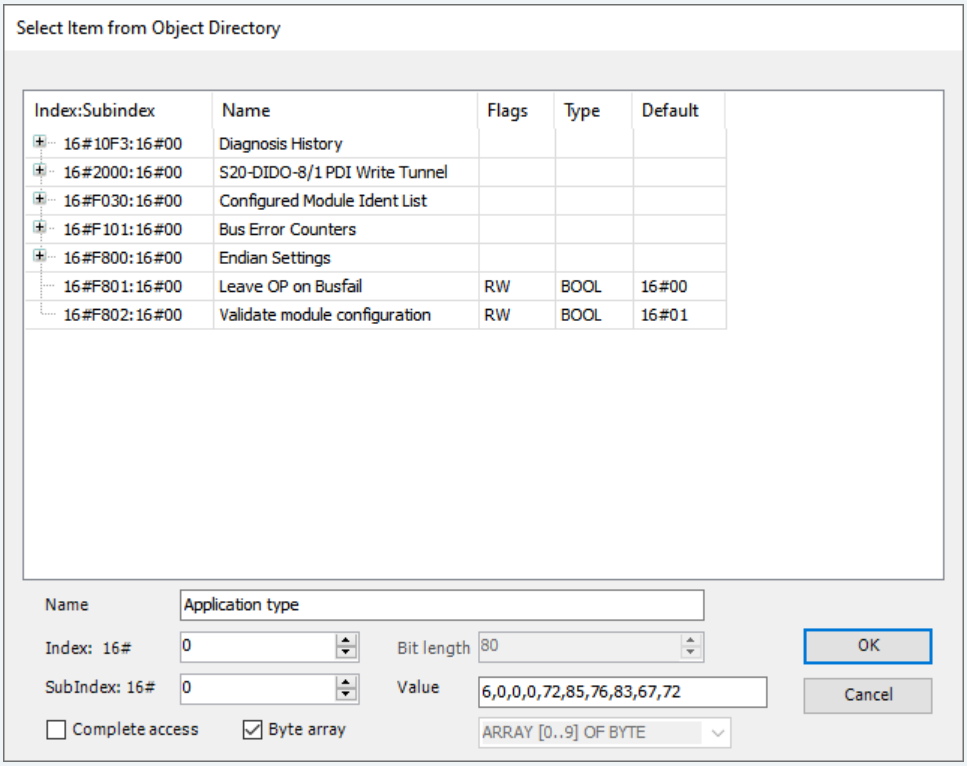
`72(H),85(u), 76(l),83(s),67(c),72(h)`

The string "Hulsch" has 6 characters.

Thus, the "actual length" is 6 bytes and it results for the list the array:

`6,0,0,0,72,85,76,83,67,72`

The array has to be entered in the “Value” field and additionally the “Byte Array” option has to be selected, as shown in the following image:

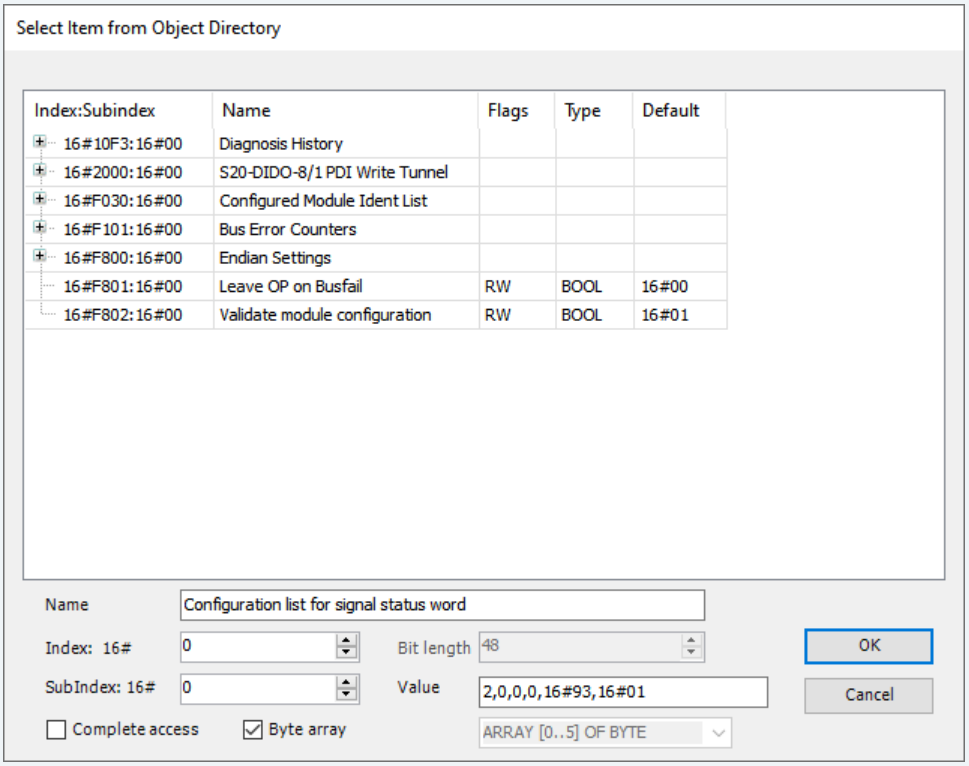


Example "2 byte list"

In the parameter list S26 "Configuration list signal status word", the IDN S-0-403 is written as the first list element.

S-0-403 corresponds to: `0x0193`

For the "Actual length", this results in 2Byte = `0x0002`. Enter the following into the "Value" field: `2,0,0,0,16#93,16#01`



Tab – “Online”

Prerequisite

The tab is only displayed if the setting “Show online data” in der ctrlX I/O Engineering toolbar.

Call

In the generic device editor of the EtherCAT slave.

To open the device editor, double-click on the EtherCAT slave object in ctrlX I/O Engineering device tree, see ➦ Chapter 14.2 Device tree and device editor on page 45.

Function

In the tab“Online”, the bus operating state of the slave can be switched.

The operating state of the slave must not be higher than the operating state of the superordinate master.

Elements of the tab

“State”	
Use the interfaces “Init” / “Pre-OP” / “Safe-OP” / “OP” to switch the bus state of EtherCAT slave.	
“Current state”	Display of the current slave bus state
“Requested state”	Display of the requested slave bus state

“Diagnostics”	
“AL status”	Display of pending errors

“EEPROM”	
Save EEPROM to file	EEPROM Backup slave data in local file

Tab – “Diagnostics”

Prerequisite

The tab is only displayed if the setting “Show online data” in der ctrlX I/O Engineering toolbar.

Call

In the generic device editor of the EtherCAT slave in ctrlX I/O Engineering device tree, see ➦ Chapter 14.2 Device tree and device editor on page 45.

Function

The tab is used to display messages provided from the CoE object “Diagnostic history” (0x10F3) of the EtherCAT slave.

Elements of the tab

Acknowledging/deleting messages

The diagnostic log is operated as ring buffer and can save up to 250 messages (device-specifically it might be less).

Messages can be acknowledged and deleted via the following buttons in the upper area of the tab:

- “Acknowledge messages”
- “Delete messages”

Settings

There are different optional settings that can be set in the slave. If a specific option is not supported by the slave, a writing error is reported while disabling/enabling (parameter value beyond the valid range).

- ☑ “Activate info”
  Setting whether or not information is to be entered in the logbook
- ☑ “Activate warning”
  Setting whether or not warnings are to be entered in the logbook
- ☑ “Activate error”
  Setting whether or not error messages are to be entered in the logbook
- ☑ “Activate emergency”
  Setting whether or not the slave sends new diagnostics as CoE emergency messages to the master.
- ☑ “Activate AcknowledgeMode”
  Setting options between “OverwriteMode” and “AcknowledgeMode”.
  In the “OverwriteMode”, if the message buffer is full, a new message overwrites the oldest message in the buffer irrespective of whether the messages were acknowledged.
  Unacknowledged messages are not overwritten in the “AcknowledgeMode”, i.e. if the message buffer is full, the latest messages are not saved and thus lost.

Table list of messages

Column	Description
“Category”	Message category: **Info** / **Warning** / **Error** There is no default setting on how incoming and outgoing errors are reported. Refer to product description of the device. Acknowledged messages are shown by a greyed out icon.
“Time”	Time stamp reported for the device message
“Code”	Diagnostic code
“Text”	The diagnostic texts are taken from the device description (ESI) of the device.

Tab – “ESC register”

Prerequisite

The tab is only displayed if the setting “Enable expert settings” is enabled in the tab “General” tab, see ➥ Chapter Tab – “General” on page 111.

Additionally muss die settings “Show online data” In the ctrlX I/O Engineering Toolbar aktiviert sein.

Call

In the generic device editor of the EtherCAT slave.

To open the device editor, double-click on the EtherCAT slave object in ctrlX I/O Engineering device tree, see ➥ Chapter 14.2 Device tree and device editor on page 45.

Function

The tab values of the EtherCAT slave controller (ESC) are displayed in the tab “ESC register” tab.

The values are retrieved when opening the tab as well as when clicking on the button“Update”. A tab to write can be selected when clicking on an address within the table.

⚠ CAUTION	**By reading and writing the registers, messages from the slave to the master can be acknowledged which are then not visible anymore for the master stack.** Thus, only experts should read/write registers.

Elements of the tab

GUI element	Description
“Offset”	Offset value of slave controller tab
“Length”	Length value of slave controller tab
“Conpact”	Compact representation of list values
“Update”	Click on the button interface to refresh the displayed values by reading them again.

Table column	Description
“Offset”	Offset value of the tab
“Name”	Tab name
“Value (hex)”	Tab value (hexadecimal)
“Value (dec)”	Tab value (decimal)

Table column	Description
“Type”	Tab type in bytes
“Write register”	
“Offset”	Current offset of the tab
“Length”	Current tab length
“Value”	Current tab value
“Write”	Click on the button“Write” interface to write the values to the selected tab.

Tab – “AoE” (EtherCAT slave)

In this tab, the ADS communication settings for ADS over EtherCAT (AoE) can be selected.

If an online connection is established to the device, parameters can be read and written.

Call

In the generic device editor of the EtherCAT slave.

To open the device editor, double-click on the EtherCAT slave object in ctrlX I/O Engineering device tree, see ➦ Documentation

Elements of the tab

“Settings”	
“AoE”	enables the AoE functionality of the slave and the Net ID configuration is transferred to the drive
“Net ID”	Input field for Net ID

Elements of the “online” area

Input field “Port”	AoE address port to communicate with
Input field “Index group”	AoE Index Group to be accessed
Input field “Index offset”	AoE Offset to be accessed
Editor “Data”	Editor for reading and writing data (hex)
Display “Error code”	Pending error code display during the transfer
Display “Cmd result”	Slave response during the command execution (manufacturer-specific)
Button “Read”	Reads out the slave data and displays them in the “Data” editor
Button “Write”	Writes the data from the editor to the slave

Please note the documentation of the respective slave regarding the parameter relevance

Also refer to:

- ➦ Acyclic communication (mailbox) - Basics ADS over EtherCAT

Configure Tab – “CoE”

Call

In the generic device editor of the EtherCAT slave in ctrlX I/O Engineering device tree, see ➦Chapter 14.2 Device tree and device editor on page 45.

Function

This tab contains information on the CoE objects of an EtherCAT slave with CoE profile.

Elements of the tab

In the right upper drop-down list, select the following settings:

- “Description from ESI”
  The slave description is read out from the device description (ESI)

Select the objects to be displayed in the left upper drop-down list:

- “All Objects”
  All objects supported by the slave.
- “Mappable Objects (TxPdo)”
  - **(RxPDO)**: Objects that can be configured as process data (output data).
  - **(TxPDO)**: Objects that can be configured as process data (input data).
- “Backup Objects”
  Objects that are stored in the slave and are required for a device replacement.
- “Settings Objects”
  Objects that can be configured as start parameters upon startup

The specifications are output in the following table:

Column	Description
“Index”	Via an index, the object is identified. If an object contains subelements, they are written by the subindex
“Name”	Object name
“Value”	Depending on the type, the value can be a text or a number. The number of subindices is displayed for objects with subelements
“Access”	Describes which access to the object is possible: • “RO”: Read-only • “RW”: Read and write • “WO”: Write only
“Type”	The data type of the object

Tab – “EoE”

In tab “EoE”, the communication settings are selected for the Ethernet over EtherCAT (EoE).

The dialog is only displayed if the slave device supports EoE and if the option “Enable expert settings” is enabled in the “General” dialog, see ➦Chapter Tab – “General” on page 111.

Call

In the generic device editor of the EtherCAT slave.

To open the device editor, double-click on the EtherCAT slave object in ctrlX I/O Engineering device tree, see ➦ Chapter 14.2 Device tree and device editor on page 45.

Elements of the tab

“Settings”	
“Virtual Ethernet Port”	☑: Enables the EoE functionality of the slave. A unique virtual MAC ID has to be defined.
“Virtual MAC Id”	Input field for the virtual MAC ID
“Switch Port”	◉: Device operates as switch. No further network settings required in this setting.
“IP Port”	◉: The slave device operates as IP port. IP settings have to be selected in this setting.

IP settings

The IP settings are assigned automatically. The setting can be adjusted at the EtherCAT master or the automatic allocation can be disabled. Valid IP settings have to be specified with regard to the network configuration.

Also refer to:

- ➦ Acyclic communication (mailbox)
- ➦ Tab – EoE (Master)

“IP Settings”	
“IP Address”	IP address of the slave in the network Example: The EtherCAT master is assigned the IP address 172.31.254.254 and if the subnet mask is 255.255.0.0, the IP address of the slave has to be between 172.31.0.0 and 172.31.254.253.
“Subnet Mask”	Subnet mask of the slave
“Default Gateway”	Standard gateway of the slave (e.g. IP address of the master)
“DNS Server”	IP address of the DNS server
“DNS Name”	DNS server name

Tab – “FoE” (EtherCAT slave)

The tab contains the communication settings for the "FoE" data transfer function (File access over EtherCAT).

The function transfers files to the slaves and is usually used to perform firmware updates.

FoE is only possible in state “Bootstrap” (device-specific).

Before downloading the firmware, read the instruction and notes of the slave device manufacturer!

During the file transfer, do not switch off the affected devices!

Prerequisite

The dialog is only displayed if the option “Show online data” is enabled in the tool bar, see:

➦ Command “Show online data”

Call

In the generic device editor of the EtherCAT slave.

To open the device editor, double-click on the EtherCAT slave object in ctrlX I/O Engineering device tree, see ➦ Documentation

Elements of the tab

“State”	Use the interfaces „Init“ / „Pre-OP“ / „Safe-OP“ / „OP“ to switch the bus state of the EtherCAT slave.
“Current state”	Display of the current slave bus state
“Requested state”	Display of the requested slave bus state
Tab “Download”	Input fields: • “Local file name” Selecting the file to be transferred to the slave • “Slave file name” Name under which the file is transferred to the slave
“Details” (can be expanded)	☑“ Required state” (enabled in the default setting) Slave state specification in which the files are transferred. Prior to the transfer, the slave is switched to this state. Possible states: • **Bootstrap** (default setting) • **Pre-OP** • **Safe-OP** • **OP**
	Input field “Timeout”: The selected value specifies how long the transfer may take until an error is reported. Possible settings: • “Auto” (default setting) The time setting is automatically defined by the system • User-defined timeout in seconds
	Option “Password”: If required, a password for the transfer can be entered
	Command “Start” Starts the transfer

Also refer to:

- ➦ Acyclic communication (mailbox) - Basics ADS over EtherCAT
- ➦ FoE error codes

Tab – “Information”

Call

In the generic device editor of the EtherCAT slave in ctrlX I/O Engineering device tree, see ➦ Chapter 14.2 Device tree and device editor on page 45.

Function

The tab shows general information of the device description file:

- Device name
- Vendor

- Category
- Version
- Description

21.2 Objects

21.2.1 Objects – General information

Objects in I/O Engineering provide certain functionalities to create your application. Examples: Application, program, function, library manager, device, image pool. Objects are managed in the views Devices, POUs, Modules in tree structures.

Insert an object into the respective "tree" using the command *“Project → Add object”*. The inserting options depend on the position in the tree.

Properties of each object can be seen and edited via the context menu of the object.

Also refer to:

- ➦Command 'Properties'

21.2.2 ‘Project settings’ object

Symbol 

Function: this object contains the configuration of the project.

Call

- Menu *“Project → Project settings”*
- Double-click on the object in the device tree

I/O Engineering directly saves the project settings in the project. For example, when transferring a project to another system, the “Project settings” object is also transferred. A project archive is not required.

The project settings are valid project-wide and offer setting possibilities for various categories such as “AS” or “Users and Groups”. The available categories vary, depending on which software packages you have installed via the Package Manager.

The following project settings are offered, but have no relevance in I/O Engineering, because these settings are used exclusively in the context of PLC Engineering:

- Compile options
- Compiler warnings
- Monitoring

Also refer to

- ➦Chapter Command 'Add-on Installer' on page 154
- ➦Chapter 'Project settings' dialog - 'Users and groups' on page 175
- ➦Chapter Dialog 'Project Settings' - 'Page Setup' on page 177
- ➦Chapter 'Project settings' dialog - 'Security' on page 178

21.3 Menu commands

21.3.1 Menu commands – General information

By default, the most important commands are available in the user interface of I/O Engineering. To customize the menu configuration, select *“Tools → Customize → Menu”*.

If you have packages or add-ons installed, additional menus and commands may be available.

Also refer to:

- ➦ Customizing Menus

21.3.2 Menu “File”

Command 'New Project’

Symbol: , keyboard shortcuts: [Ctrl] + [N]

Function: The command opens the “New Project” dialog for creating a new project file.

Call: Menu *“File”*

'New project' dialog

Function: Selection of a project category and a project template.

Call: *“File → New project”*

Depending on the selected template, a project is created that is automatically equipped with a certain scope of objects.

“Categories”

“Projects”	Contains templates for project creation

“Templates”

“Projects” category:	
“Empty project”	Contains only the “Project settings” object
“ctrlX CORE I/O”	Contains a ctrlX CORE I/O basic project
“Name”	Name of the project to be created. Depending on the template, a default name is displayed. The numerical addition ensures the uniqueness of the name in the file system. You can change the file name by taking the file path conventions of the operating system into consideration. Dots in the name are not allowed, but the name must not contain any dots. I/O Engineering automatically adds the file extension corresponding to the selected template.
“Location”	Memory location for the new project file. ... opens a dialog for browsing the file system. ▾ shows the history of previously entered paths
“OK”	I/O Engineering opens a new project. An error symbol at the input field indicates missing data. When hovering with the mouse cursor over it, a tooltip informs the user how to proceed.

'Open project’ command

Symbol: Shortcut: [Ctrl] + [O]

Function: The command opens a dialog to load a project. Browse the file system for a I/O Engineering project and open it in the development system.

Call: Menu *“File”*

‘Open project’ dialog

“File type”	Type of the I/O Engineeringproject loaded into the development system
“All supported files”	Filters for all projects that can be loaded by I/O Engineering
File extension `project`	Filters for projects created with I/O Engineering V3

File extension `projectarchive`	Filters for project archives created with I/O Engineering V3
File extension `library`	Filters for library project created with I/O Engineering V3
“Open”	Loads the selected project in I/O Engineering Note: Depending on the state of your I/O Engineering installation, it may be necessary to update or complete the installation. If this is the case, first another dialog “Open project” opens with options for installation management.

Command “Open project from ctrlX CORE...”

Prerequisite

To be able to open a project from the ctrlX device, the project data source has to available on the ctrlX device, see:

➦ Project data source - Introduction

Symbol

Call

In the I/O Engineering menu *“File → Open project from ctrlX CORE...”*

General information about the project data source

The project data can either be saved on the ctrlX CORE device or in the file system of the Engineering PC.

If the project data is stored on the ctrlX device, the command “Open project from ctrlX CORE...” allows to call the project data from the ctrlX device and to open and edit it in I/O Engineering.

Via the command “Project synchronisation...”, it is possible to synchronize the changed project data between Engineering PC and ctrlX device again after editing, see:

➦ Command “Project synchronisation...”

Operating principle

The command opens the “Open project in ctrlX CORE” dialog, see:

➦ Dialog “Open project in ctrlX CORE”

In the dialog, select from which ctrlX device the project data is to be loaded to the Engineering PC. After the project data has been read from the ctrlX device, it is possible to edit the project in I/O Engineering and then synchronize it again with the project store in the ctrlX device. In addition, the project data can also be stored on the Engineering PC.

Command 'Close project'

Function: The command closes the currently opened project. I/O Engineering remains open.

Call: Menu *“File”*. Also, implicitly when opening a new/different project while a project is still open.

If the project contains unsaved changes, a prompt is displayed asking if the project should be saved.

If you have not yet explicitly saved the project, a prompt is displayed asking if you want to delete the project files.

Also refer to

- ➦ Chapter 8.2 Creating a new project on page 21

Command 'Save project'

Symbol: , shortcut [Ctrl] + [S]

Function: this command saves the project file.

Call: *"File"* menu

This command saves the project file with the current project name, which appears in the title bar of the main window. If the project has been changed since it was last saved, the project name is given an asterisk.

The command is not available if the project is read-only.

Write protection exists if

- the project is identified in the project information (summary) as 'Released'
- the option "Open read-only" was selected in the dialog box "Open Project" when opening the project

Write protection is indicated by a line in the top right corner of the main window. A mouse-click on this line brings up a menu with commands for the possible actions:

- "Save project under a different file name on the disk": a mouse-click on this option leads to 'Save file as...'
- "Exit read-only mode": appears only if the option "Open read-only" was selected when opening the project.
- "Remove read-only attribute from the project on the disk": appears only if the project file had been provided with the property 'Read-only' on the disk at the time of opening.
- "Remove identification 'Released' in the project information": appears only if this attribute is currently set.

Backup copy

Optionally a backup copy of the project file can be created. If the option "Create backup copy" is activated in the option dialog box 'Load and Save', the project is additionally copied to a file `<projectname.backup>` each time the project is saved.

See also

- Chapter Command 'Save project as' on page 130
- Chapter 11.13 Saving the project on page 39
- Chapter 'Options' dialog - 'Load and save' on page 183

Command 'Save project as'

The command opens the standard Windows dialog for saving a file. The project can be stored with the desired location and file type.

"File type"	The selection list contains the versions of the development system for which the project can be saved. If the current project contains Add-ons that are not available in the selected storage format (profile), the "Expand profile" dialog is opened. • "Project files (I/O Engineering V<version>) (*.project)": The project is saved as I/O Engineering project file "<projectname>.project " for the currently used or the selected version of the programming system. When opening the project in an older version, it is recommended to save data for precisely this version, as you are informed immediately about potential data loss.

'Expand profile' dialog

In this dialog, the selected profile (memory format) can be extended by the add-ons that are contained in the current project. The profile is saved temporarily and then deleted after being saved or exported.

“Add to profile”	☑: The current profile is extended by the add-on so that the add-on data of the current project is also saved.
“AddOn”	The add-on of the current project that is not contained in the selected memory format.
“Version”	Version of the “Add-on” included in the current profile. If several versions are installed, then the version can be selected.
“Save profile”	Opens the “Enter Profile name” dialog. In this dialog, specify the name for the new profile. The new profile is saved permanently at `$ProgramData$/$PRODUCT$/CustomInformationalProfiles`.
“Use saved profile”	The profile which was permanently saved in “Save profile” is used for saving or exporting the current project.

Also refer to

- ➦ Chapter Command 'Save project' on page 130
- ➦ Chapter 11.13 Saving the project on page 39
- ➦ Chapter ‘Options’ dialog - ‘Load and save' on page 183

Command 'Save/Send Archive'

Function: This command opens the dialog “Project Archive” for the configuration of project archives.

Call: Menu bar: *“File → Project Archive”*

An archive file (*.projectarchive) contains all files contained and referenced in the currently opened project. It can either be saved or dispatched as an e-mail attachment. The dispatch by email is very helpful for providing an employee with all project-relevant files. The file can be simply unpacked again with the command “Extract Archive”.

NOTICE	The archiving function is not intended for the storage of a project, but rather for the simple summarizing of all project-relevant files.

See also

- ➦ Chapter 11.14 Saving/sending the project archive on page 40
- ➦ Chapter Command 'Extract archive' on page 132

Dialog 'Project Archive'

The dialog displays all the categories that can be added to the project archive. In this dialog, complete categories or individual objects from the categories can be added to the project archive by setting a check mark (☑).

Entries that are display as red in the list require your attention. Move the mouse pointer over this library for more information.

“Additional Files”	Opens the dialog “Additional Files”. Here, further files can be added to the archive with the “Add” button.
“Comment”	Opens the "Comment" dialog. Here, comments can be added to the archive.

“Save”	Creates the archive file and saves it. The storage location and the archive name are specified in the subsequent dialog
“Send”	Creates a temporary archive file that is attached to an empty e-mail. A correct installation of the MAPI (Messaging Application Programming Interface) is required for the successful execution of this operation. Failure is documented by the display of a corresponding error message. The temporary archive is automatically deleted after sending the e-mail.

Command 'Extract archive'

Function: The command extracts a project archive created using the “Save/ send archive” command. Before that you need to configure which contents of the archive I/O Engineering should extract and to which directories in the file system they should be copied.

Call: Menu *“File → Project archive”*

By default, an archive has the file extension `.projectarchive`.

After selecting the archive, the “Extract project archive” dialog opens where you configure the extraction.

Dialog Box 'Extract Project Archive'

This dialog box shows the contents of the project archive. You can exclude complete categories or single objects from categories by clearing the check boxes (☑) from the extraction.

Table 4: “Locations”

“Extract into the same folder where the archive is located”	The archive is extracted to the same directory.
“Extract into the following folder”	The contents of the archive are extracted to the given path.
“Advanced”	Opens the “Advanced” dialog box for you to define where special and additional files from the archive are extracted.

Table 5: “Contents”

“Items”	Shows the contents of the archive structured in object categories. ☑: The object is extracted. ☐: The object is not extracted.
“Comment”	Comment that was entered when creating the project archive
“Extract”	If an extracted file has the same name as an existing file in the target directory, then a dialog box opens, prompting whether the local file should be replaced. The decision can be applied automatically to any additional conflicting names. In this case, you have to select the ☑ “Apply to all objects and files” check box.

'Advanced’ dialog

Table 6: “Repositories”

“Install devices in”	Drop-down list with currently available repositories. Select the repositories to which I/O Engineering is to install the devices contained in the archive.

Table 7: “Additional files”

By default, "additional files" are preconfigured with “Do not extract”. Change this by selecting entries in the table and activating one of the following options:	
“Extract to project directory”	Project file directory
“Extract to directory”	User-defined directory
“Do not extract”	Default setting

Also refer to

- ➦ Chapter Command 'Save/Send Archive' on page 131

Command 'Print'

Symbol:

Function: This command opens the default Windows dialog box for printing documents.

Call: Main menu *“File”*

See also

- ➦ Chapter Dialog 'Project Settings' - 'Page Setup' on page 177

Command 'Print Preview'

Function: This command opens a print preview for the currently open element.

Call: Main menu *“File”*

Requirement: An object is open in the editor.

See also

- ➦ Chapter Dialog 'Project Settings' - 'Page Setup' on page 177
- ➦ Chapter Command 'Print' on page 133

Command 'Page Setup'

Symbol:

Function: This command opens the “Page Setup” dialog box for configuring the layout of the printed version of the project contents.

Call: Main menu *“File → Page Setup”*

See also

- ➦ Chapter Dialog 'Project Settings' - 'Page Setup' on page 177
- ➦ Chapter Command 'Print' on page 133

Command 'Last used projects'

Function: Opens the list of recently used projects from which you can select a project to open.

Call: Menu *“File”*

Also refer to

- ➦ Chapter 8.2 Creating a new project on page 21

Command 'Exit'

Shortcut: [<Alt>]+[<F4>]

Function: this command exits from the programming system. If a project is currently opened that has been changed since it was last saved, a dialog box opens asking whether the project should be saved.

Call: “File” menu

21.3.3 Menu “Edit”

Standard Commands

I/O Engineering provides the following standard commands:

- Undo: , shortcut: [Ctrl] + [Z]
- Redo: , shortcut: [Ctrl] + [Y]
- Cut: , shortcut: [Ctrl] + [X]
- Copy: , shortcut: [Ctrl] + [C]
- Paste: , shortcut: [Ctrl] + [V]
- Delete: , shortcut: [Ctrl]
- Select all: shortcut: [Ctrl] + [Ctrl]

Not all editors support the “Insert” command. In some editors it can be used with limitations. Graphical editors only support the command if the pasted elements will create a valid construct.

In object trees like POUs or device view the command refers to the currently selected object. Multi selection is possible.

Command ‘Replace’. 'Replace in project’

Symbol , keyboard shortcut [Ctrl] + [H]

Symbol: ; keyboard shortcut[Ctrl]+[Shift]+[H]

Function: The commands search the project or parts of it for a specific character string and replace it.

Call: Menu *“Edit → Search and Replace”*

Prerequisite: The application is in online mode.

The command opens the “Replace” dialog where the search and replace character strings are specified and the search options are defined.

Table 8: In addition to the options of the “Search” dialog, the following settings are still possible:

“Replace with”	Input field for the new character string.
“Replace”	Each next found character string is highlighted in the editor and replaced (step-by-step replace).
“Replace all”	All found character strings are replaced at one time without them being displayed in the editors.
“Leave changed objects open after "Replace all"”	The editors of the found objects remain open.

Also refer to

- Chapter Command ‘Search', 'Search in project on page 134

Command ‘Search', 'Search in project

Symbol , keyboard shortcut [Ctrl] + [F]

Symbol ; keyboard shortcut[Ctrl]+[Shift]+[F]

Function: The commands search the project or parts of it for a specific string.

Call: Menu *“Edit → Search and Replace”*

The command opens the “Find” dialog where the searched character string is specified and the search options are defined.

‘Search' dialog

“Search for”	Character string to be searched.
“Match case”:	☑: The search considers uppercase and lowercase.
“Match whole word”:	☑: Only character strings are found that exact matches.

“Search up”:	☑: The specified search space is run through upwards. ☐: The specified search space is run through downwards.
“Use regular expressions”:	Use the > button to receive support when specifying regular expressions.
“Search in”	⌄: Drop-down list with the areas of the project to be searched: • “Active editor” • “All open editors” • “Selected objects & subobjects” • “Entire project” • “Selection only” …: Opens a dialog where you set the areas of the project to be searched (see below)
“Find next”	Start the search
“Find all”	All search results are listed in the message view with their object path, project name, object name, and object position. Double-clicking the entry in the list opens the match position in the respective object editor.
“Replace”	Switches to the “Replace” dialog

The color of the search result markings can be customized in the options of the text editor. This is done using the “Selection color” - “Inactive” parameter in the “Text area” tab.

Also refer to

- ➦Tab 'Text Area' on page 187

Dialog for setting the objects to be searched

“Entire project”	All editable positions in all objects of the project are searched.
“Within the following objects”	Only the editable positions within the objects defined here are searched:
	• “Schema”: The “Save” command saves the current search configuration under the specified name. All stored schemes are available in the drop-down list (⌄) . • “Object types”: ☑: Browsing the object • “Name filter”: Name filter for the objects to be browsed. The placeholder "*" can be used. Example: Filter "*CAN*": All objects are browsed containing "CAN" in their name

Also refer to

- ➦Chapter Command ‘Replace’. 'Replace in project’ on page 134

Command ‘Find next’

Symbol , keyboard shortcut [F3]

Function: This command selects the next search result at its position in the relevant editor when searching for a specific character string in the project.

Call: Menu *“Edit → Search and Replace”*

Prerequisite: You have started a search for a certain character string in the project using the “Search” or “Replace” commands.

Also refer to

- ➦Chapter Command ‘Search', 'Search in project on page 134
- ➦Chapter Command ‘Replace’. 'Replace in project’ on page 134

Command 'Find next (selection)'

Keyboard shortcut [Ctrl] + [F]

Function: The command searches for the next string in the project that matches the one that is currently selected or where the cursor is currently located.

Call: Menu *“Edit → Search and Replace”*

Prerequisite: You have placed the cursor in an editable string in the project, or you have selected an editable string.

Also refer to

- Chapter Command ‘Search', 'Search in project on page 134
- Chapter Command ‘Replace’. 'Replace in project’ on page 134

Command 'Search previous’

Symbol  , keyboard shortcut [Shift] + [F3]

Function: This command, when searching for a specific string in the project, selects the previous search result at its position in the relevant editor.

Call: Menu *“Edit → Search and Replace”*

Prerequisite: You have started a search for a certain character string in the project using the “Search” or “Replace” commands.

Also refer to

- Chapter Command ‘Search', 'Search in project on page 134
- Chapter Command ‘Replace’. 'Replace in project’ on page 134

Command 'Search previous (selection)'

Keyboard shortcut [Ctrl] + [Shift] + [F3]

Function: The command searches for the previous string in the project that matches the one that is currently selected or where the cursor is currently located.

Call: Menu *“Edit → Search and Replace”*

Prerequisite: You have placed the cursor in an editable string in the project, or you have selected an editable string.

Also refer to

- Chapter Command ‘Search', 'Search in project on page 134
- Chapter Command ‘Replace’. 'Replace in project’ on page 134

Command 'Next Message'

Keyboard shortcut: [F4]

Function: This command selects the next message in the messages view.

Call: Main menu *“Edit”*.

If the last message in the list has been reached, then the marking jumps to the beginning.

See also

- Chapter Command 'Previous Message' on page 136

Command 'Previous Message'

Keyboard shortcut: [Shift]+[F4]

Function: This command selects the previous message in the messages view.

Call: Main menu *“Edit”*

If the first message in the list has been reached, then the marking jumps to the end.

See also

- Chapter Command 'Next Message' on page 136

21.3.4 Menu “View”

Command 'Devices'

Symbol: , keyboard shortcuts: [Alt] + [0]

Function: The command opens the “Devices” view in the I/O Engineering main window. It contains the "device tree" of the project where you configure your applications.

The standard menu for navigating in the object tree of the view is available via the button.

Call: Menu *“View”*

Also refer to

- Chapter 14.2 Device tree and device editor on page 45

'Messages' command

Symbol:

Function: The command opens the “Messages” view.

Call: Menu *“View”*

View 'Messages'

Message category	The messages are categorized by component or functionality for selection from a list box. Filter the message display by selecting a category.
Message type	Click the symbol of the message type to show or hide messages. I/O Engineering shows the number of messages next to each symbol. • : Errors • : Warning • : Information
	Deletes all messages in the selected message category.
	Deletes all messages in all message categories.
“Description”	Message text with the reported object and the location in the object. Double-click a message in the table to jump to the source text location.
Command icon	In the case of unresolved libraries, it lists commands that can be used to quickly fix the reported error (quickfix) • Updating placeholder '<Library>' to the latest version. • Open the 'Placeholder' dialog of the library manager.
“Project”	Name of the project generating the message
“Object”	Object in which the message was generated
“Position”	Position in code

Table 9: “Commands in context menu”

“Next message”	The source text position of the next message is displayed.
“Previous message”	The source text position of the previous message is displayed.
“Go to source text position”	The source position of the selected message is displayed.

Command 'Start Page'

Symbol:

Function: This command opens the “Start Page” view.

Call: Main menu *“View”*

The view includes some basic commands and a list of recently opened projects. In addition, the I/O Engineering homepage is displayed.

If you access the Internet through a proxy, then you can save the authentication data in the project options (“Proxy Settings”) so you do not have to provide this data every time you use this command.

By moving the mouse pointer over the list of recently opened projects, you can remove or pin individual projects in the list. Pinned projects remain in this list until you remove the pin.

In the project options (“Load and Save”), you can configure whether this start page should open automatically when you start I/O Engineering.

See also

- Chapter ‘Options’ dialog - ‘Load and save' on page 183
- Chapter 'Options' dialog - 'Proxy settings' on page 184

Command 'Full Screen'

Symbol: , keyboard shortcut [Ctrl]+[Shift]+[F12]

Function: This command switches the I/O Engineering display to full screen mode.

Call: Main menu *“View”*

Choosing this command displays the main window of the I/O Engineering user interface in full-screen mode. You can return to the previous setting by choosing the command again or with the keyboard shortcut [Ctrl]+[Shift]+[F12].

Command 'Properties'

Symbol:

Function: The command opens the properties of the currently selected device tree object.

Call: Menu *“View”*

21.3.5 Menu “Project”

Command “Logging in device user...”

Comprehensive information:
Logging in and logging off to/from the control

Symbol:

Call:

The command can be called either via the context menu of the ctrlX device node in the device tree or optionally via the *“Project”* menu

Function:

The command opens the “Login - [Control IP address]” dialog, to establish a connection with a ctrlX device.

Related topics:

- Chapter Command “Log out current device user” on page 139
- Chapter 21.4.3 Dialog “Login - [Control IP address]” on page 164

Command “Log out current device user”

Comprehensive information:
➦Logging in and logging off to/from the control

Prerequisite:

A device user is logged in to the control.

Symbol:

Call:

The command can be called either via the context menu of the ctrlX device node in the device tree or optionally via the *“Project”* menu

Function:

The command terminates the connection to the currently logged in ctrlX device.

Related topics:

- ➦Chapter Command “Logging in device user...” on page 138
- ➦Chapter 21.4.3 Dialog “Login - [Control IP address]” on page 164

Command 'Attach device'

Function: The command opens the “Append device” dialog for selecting a device object to be inserted in the device tree below the currently selected object.

Call: context menu of a device object in the device tree.

Prerequisite: an object is selected in the device tree below which a device object can be inserted.

Also refer to

- ➦Chapter 14.2 Device tree and device editor on page 45

‘Attach device' dialog

Function: depending on the currently selected position in the device tree, the dialog offers a selection of the devices that can be inserted at this point. In addition, it contains the commands that are also available in the context menu: “Attach device”, “Insert device”, “Plug in device”, “Update device”.

Prerequisite: The devices are installed in the device repository on the local system.

If you have opened the dialog, it always displays the selection to suit the object currently selected in the device tree until you click “Close”.

“Name”	Name with which the device is to appear in the device tree. Must be a valid IEC identifier.

Table 10: “Action”

“Attach device”	I/O Engineering inserts the selected device indented below the selected object in the device tree.
“Insert device”	I/O Engineering inserts the selected device at the same level as the selected object below it in the device tree.

“Plug in device”	I/O Engineering inserts the selected device in the selected slot. If the slot is already occupied, the existing module is replaced by the new one.
“Update device”	I/O Engineering replaces the device selected in the device tree by the one selected. Please note: depending on the device, this may cause the configuration already done in the device editor to be overwritten with the default values!

“Character string for full-text search in all devices”	This field is editable after clicking in it. For any character string entered, only those devices that include the character string are displayed in the lower view. The matching character string is highlighted in yellow for these devices.
“Manufacturer:”	Drop-down list with manufacturers whose available devices are displayed.
“Group by category”	☑: The available devices (newest version) are sorted by category. The category is defined in the device description file. ☐: The available devices appear flat and alphabetically sorted.
“Show all versions (for experts only)”	☑: In addition, all other available versions of the devices can also be selected. ☐: Only the newest version of each device is available for selection
“Show outdated versions”	☑: In addition, outdated versions of the devices can also be selected. Outdated versions result, for example, from the update of plug-ins. ☐: Outdated device versions are not displayed.

The information provided by the device description file is displayed: device name, vendor, categories, version, order number and a short description, device-specific bitmap.

Also refer to

- ➦Chapter Command 'Update device' on page 140

Command 'Update device'

Function: As with the “Mount device” command, the command opens the “Mount device” dialog for selecting a device object. This object is inserted in the device tree in place of the currently selected object.

Call: context menu of a device object in the device tree.

Prerequisite: an object is selected in the device tree below which a device object can be inserted.

With this command you can insert either a different version of a device or a different type of device in place of the previous one.

The symbolic device name used in the device tree is retained, but the device type specified in parentheses behind it changes if a different type has been selected. Thus if only the device version is changed, the object entry appears unchanged.

If the device type does not change, the configuration tree indented below the device entry concerned is retained. In this case, the configuration settings also remain the same.

Also refer to

- ➦Chapter Command 'Attach device' on page 139

Command 'Edit Object'

Function: This command opens the object in its editor.

Call: Main menu “Project”, context menu.

Requirement: An object is selected in the device tree or in the “POUs” view.

Command 'Edit Object with'

Function: When multiple objects are available for an object, this command opens a dialog box for selecting an editor.

If only one editor is available for an object, then this command opens the object in that editor.

Call: Main menu “Project” or shortcut menu (right-click)

Requirement: An object is selected in the device tree or in the “POUs” view.

In the standard installation of I/O Engineering, there is no object that has multiple available editors.

Command 'Project settings'

Symbol: 

Function: The command opens the “Project Settings” dialog.

Call: “Project” menu or double-click on the “Project settings” object in the “POUs” view

Prerequisite: A project is open

Also refer to

- Chapter 8.3.2 Selecting project settings on page 23
- Chapter 21.2.2 ‘Project settings’ object on page 127

Command “Communication settings...”

The command opens the tab “Communication” in the generic device editor of the ctrlX device, where the communication settings to the ctrlX device can be configured, see:

Tab – Communication

Call:

The command can be called via the context menu of the ctrlX device in the device tree or optionally via the *“Project”* menu.

Command “Project synchronisation...”

Prerequisite:

The “Project synchronisation...” command is exclusively supported by the ctrlX CORE, ctrlX CORE Virtual and ctrlX CORE I/O devices.

Symbol:

Call:

The command has two access options:

- In the menu *“Project → Project synchronisation...”*
- In the context menu of the “device (ctrlX CORE)” object in the device tree

Function:

The command opens the “Project synchronization” dialog in which basic settings to synchronize the project data between the control and the Engineering PC, see:

Dialog “Project synchronization”.

Without active synchronization, only the data required for the control flow is stored. The data is available in the “Manage app data” window of the ctrlX CORE web interface in the following configuration data folder: *“plc → run...”*

The project synchronization functionality allows to transfer a project from the control to the Engineering PC and to save it on the local file system. Moreover, it can be defined for which actions the project is transferred or synchronized from the Engineering PC to the control.

The data is stored on the control in the configuration data in the following folder: *“plc → run...”*

If you are not the project owner of the project sources, the project can be retrieved directly from the control by means of the project synchronization, you can connect to the running project and to edit it and load it to the control again, see:

➦ Project data source - introduction.

Command “Equalize project PC <-> ctrlX CORE”

Call:

The command has two access options:

- In the menu *“Project → Equalize project PC <-> ctrlX CORE”*
- In the context menu of the “device (ctrlX CORE)” object in the device tree

Function:

By using the command, the project folder on the Engineering PC and the project folder on the ctrlX device are compared, analogously to the check when opening the project.

Please refer to: ➦ Project data source - Introduction

Command “Synchronization cache...”

Prerequisite:

The “Synchronization cache...” command is exclusively supported by the ctrlX CORE, ctrlX CORE Virtual and ctrlX CORE I/O devices.

Symbol:

Call:

The command has two access options:

- In the menu *“Project → Synchronization cache...”*
- In the context menu of the “device (ctrlX CORE)” object in the device tree

Function:

The “Synchronization cache...” command opens the “Project synchronization” dialog to manage the project cache on the Engineering PC, see:

➦ Dialog “Project synchronization” (Synchronization cache...)

The synchronization cache serves as a temporary file repository for project data exchanged between the ctrlX device and I/O Engineering.

Related topics

➦ Project data source - Introduction

Command 'Document'

Symbol: 

Function: This command opens the “Document Project” dialog box, where you can define the project documentation. This includes the selection of objects in the open project that you want to print.

Call: Main menu “Project”

Table 11: “Document Project” dialog box

“Please select the objects which are to be printed”	Project tree view In this view, you can select or clear objects for printing. All objects are selected by default.
“Title page”	I/O Engineering creates a title page named "Project Documentation" with the following information: • File: project file name • Date: Creation date of the project documentation • Profile: I/O Engineering profile of the project
“Table of contents”	I/O Engineering creates a table of contents for the project documentation.
“Preview”	I/O Engineering creates and opens a print preview of the project documentation.
“Select”	I/O Engineering opens a drop-down list of all or single object types for the project documentation.
“Deselect”	I/O Engineering opens a drop-down list of all or single object types that should be excluded from the project documentation.
“OK”	The “Print” dialog box opens.

See also

- ➦Chapter Dialog 'Project Settings' - 'Page Setup' on page 177
- ➦Chapter Command 'Print' on page 133

'Compare’ command

Symbol: 

Function: The command opens the “Project compare” dialog. In this dialog, you define the reference project to compare with the current project. You configure the comparison process by means of options. When closing the dialog, the comparison starts and the result is displayed in the “Project comparison - Differences” view.

Call: *“Comparing → project”*.

Prerequisite: A project is opened.

Also refer to

- ➦Chapter Command 'Commit Accepted Changes' on page 148

‘Project comparison’ dialog

Table 12: “Compare the currently opened project with:”

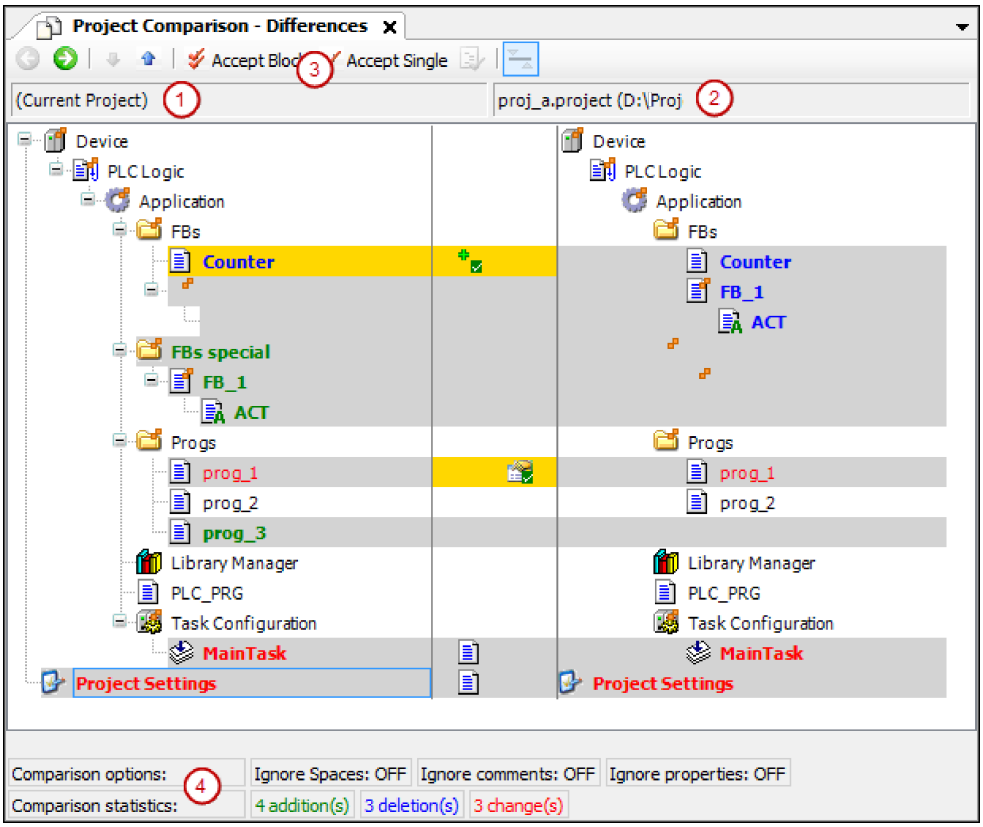
“Project on disk”	Path of the reference project on the file system.
“Project in a source control database”	“Host:” Name of the host where the source code management is located. “Port:” Number of the port for connecting to the source code management “Memory location:” Path of the reference project Prerequisite: The project is connected to a source code management (for example CODESYS SVN) .

Table 13: “Comparison options:”

“Ignore whitespace”	☑: Deviations of the current project from the reference project based only on blanks are ignored.
“Ignore comments”	☑: Comments in the programming code are excluded from the comparison.
“Ignore properties”	☑: Object properties are excluded from the comparison.
“OK”	Starts the project comparison and displays the result in the “ Project Comparison - Differences” view.

View 'Project compare' - 'Differences'

The project compare view appears when closing the “Project compare” dialog with “OK”. The example shown exemplifies the project comparison of a PLC project, but is also applicable to I/O engineering projects in terms of functionality.




```
(1) Objektbaum des aktuellen Projekts
(2) Objektbaum des Referenzprojekts
(3) Befehl 'Block übernehmen', Befehl 'Einzeln übernehmen'
(4) Vergleichsoptionen, konfiguriert im Dialog
'Projektvergleich'
    Vergleichsstatistik: hinzugefügte, gelöschte, geänderte
Objekte
```

Table 14: Toolbar

	Switches to the detailed compare view “Project comparison' - '<Object name> Differences” for the object selected in the tree. Option: Double-click on the object.
	Selects the next bottom object in the device tree where differences were detected.
	Selects the next top object in the device tree where differences were detected.
“Accept Block”	The block (selected object with all subordinate objects and units) is selected for acceptance from the reference block to the current block. Repeated clicking of “Accept Block” undoes the effects of its last change.
“Accept Single”	The object is selected in the current object for acceptance from the reference line.
	Prerequisite: The object selected in the object tree differs with regard to properties, access rights or folder affiliation. Opens the “Accept” dialog.

Table 15: Display of differences with colors, and symbols

Black font	Objects are identical.
Object name with	Child objects of the object are different
Gray highlight	Objects are different.
Gray highlight + bold blue font	Object is only in the reference project.
Gray highlight + bold green font	Object is only in the open project (not in reference project).
Gray highlight + red font +	Object has different properties.
Gray highlight + red font +	Access rights of object and reference object are different.
Gray highlight + bold red font +	Implementation of objects is different. Double-click on the row to display the object-specific compare view.
Yellow highlight	Object is activated for acceptance.
Yellow highlight +	Adding the reference object to the open project is activated.
Yellow highlight +	Deleting the object (in the open project) is activated.
Yellow highlight +	Acceptance of the properties of the reference project is activated.
Yellow highlight + red font +	Acceptance of the access rights of the reference project is activated.
Gray highlight + bold red font +	Acceptance of the implementation of the reference project is activated.
“Comparison options”	The comparison options defined in the “Project comparison” dialog.
“Compare statistics”	Number of additions, deletions, and changes in the current project, as compared to the reference project. “Change” means differences of an object available in both projects.
✕	The dialog prompt opens: “Do you want to commit the changes which you made in the diff view?” “Yes”: The contents, properties or access rights of the yellow highlighted objects are changed in the project. Now they correspond to the reference project. The project comparison view is closed.

View 'Project comparison' - '<Object name> Differences'

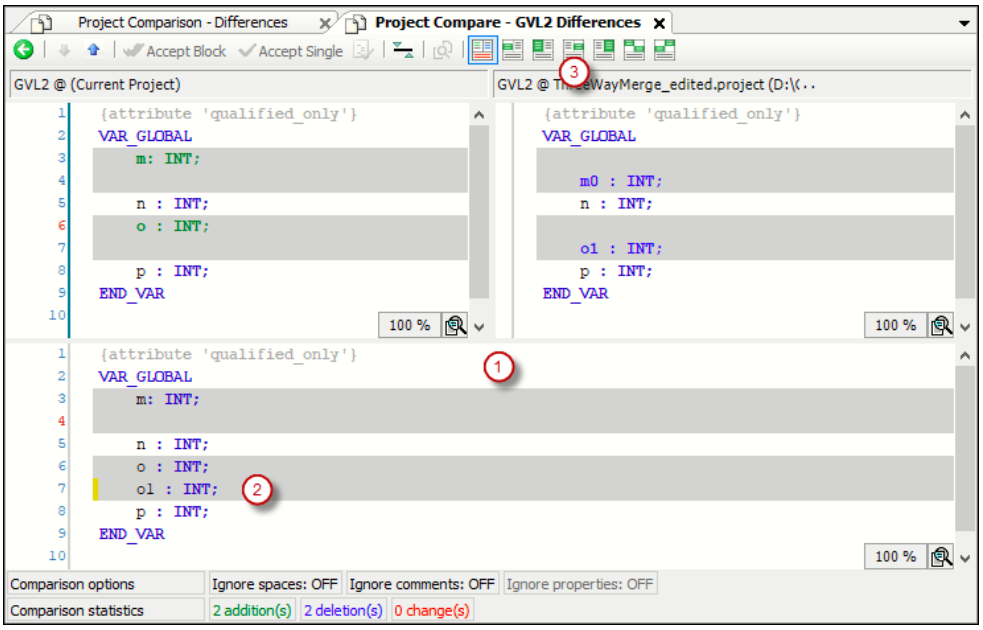
Function: Detailed compare view

Call in the project compare view:

- Select an object that is identified as having different contents which you need to view in detail. Click .
- Double-click the object.

Table 16: Toolbar

	Switches back to the project compare view.
	Selects the next row below in the code where differences were detected.
	Selects the next row above in the code where differences were detected.
"Accept Block"[2]	The block (with all subordinate rows) is selected to apply the reference blocks into the current project. A block in the detailed compare view consists of the unit where the cursor is located and all corresponding units that have the same difference markers. A unit is a row, network, or element. Subsequent rows of a row are examples of corresponding units. Repeated clicking of "Accept Block" undoes the effects of its last change.
"Accept Single"	The row is selected in the current object to apply reference row.
	Switches between the standard display, in which different units (rows, networks, elements) are displayed in red, and another display: In the open project, the units are displayed recently inserted. In the reference project, they are displayed as deleted. Available within the detailed compare view only. Note: Depending on the representation, differences found are counted in the statistics as a change, or as an insertion and deletion!
	Third view in the detailed compare view. The button opens or closes a third pane below the comparison of the current view and reference view (see figure below). This third view shows the result of the actions taken to resolve the differences found. Respective rows are marked by a yellow bar at the beginning of the row.



(1) Third compare view
(2) Result of action "Use right row"
(3) Button "Use right row"

	The left row, that is the implementation in the current project, is used.
	The left block, that is the implementation of the block into the current project, is used.
	The right row, that is the implementation in the reference project, is used.
	The right block, that is the implementation of the block in the reference project, is used.
	Only available if the differences are not directly compared . The left row is inserted above the right one in the third (result) view.
	Only available if the differences are not directly compared . The right row is inserted above the left row in the third (result) view.

Table 17: Display of differences with colors, and symbols

Black font	Objects are identical.
Object name with	The child objects of the object are different.
Gray highlight + bold blue font	Code is only in the reference project.
Gray highlight + bold green font	Code is only in the current project (not in reference project).
Yellow highlight	The object is activated for acceptance.
✕	The dialog prompt opens: “Do you want to commit the changes which you made in the diff view?” “Yes”: The code highlighted in yellow is transferred to the project. The code corresponds to the reference project. The detailed view is closed and the project view is displayed. You can continue working in the project comparison.

'Apply' dialog

Table 18: “Which meta data should be accepted?”

“Access rights”	☑: Access rights that are selected for acceptance.
“Accepted groups”	Grouping with access rights accepted by the reference project. A group is accepted if it exists in both projects with different access rights. Example: `Group_A`
“Unaccepted groups (missing in a project)”	The group is not accepted if it is not present in one of the two projects.
“Properties”	☑: Properties activated for accept Prerequisite: The properties of the reference object and the object are different.
“OK”	Settings are accepted.

Command 'Commit Accepted Changes'

Symbol:

Function: This command commits the accepted differences from the project comparison to the current project.

Call: *“Project → Commit Accepted Changes”*.

Requirement: Changes from the project comparison have been accepted.

The changes are only copied to the project. This command does not save them to the hard disk.

See also

- ➥Chapter 10.2 Open detailed compare view on page 26

Command 'Export'

Function: This command opens a dialog box for exporting objects from a project to an XML file.

Call: Menu bar: *“Project”*.

Dialog 'Export'

This dialog box lists all objects from the device tree, POU tree, and module tree that I/O Engineering can export.

One file per subtree	☑: I/O Engineering generates a separate export file for each subtree that is located directly under the root node and includes selected files. ☐: I/O Engineering generates one export file for all selected objects.
“Saved version”	This version should correspond to the target version where the export file will later be imported. If the current project contains plug-ins or add-ons that are not available in the selected memory format (profile), then the “Extend Profile” dialog box opens. In this dialog box, the selected profile can be extended with the add-ons.

See also

- ➥Chapter Command 'Save project as' on page 130
- ➥Chapter 'Import...' command on page 148
- ➥Chapter 9.2 Exporting and Importing Projects on page 24

'Import...' command

Function: The command opens a dialog for importing objects from an XML file.

Call: Menu *"Project"*

Prerequisite: A project is opened.

Before importing the project, I/O Engineering checks whether the project requires add-ons that are not yet installed. If this is the case, a dialog with the missing add-ons is shown. The add-ons can be installed later via this dialog.

Dialog 'Import'

The dialog lists all objects from the export file that I/O Engineering can import at the current location.

"Currently selected target objects"	Object currently selected in the device tree
"Insertable items"	Displays all objects in the export file that I/O Engineering can insert below the selected object.
"Show Contents"	Displays the contents of the export file in a tree structure

Also refer to

- Chapter Command 'Export' on page 148
- Chapter 9.2 Exporting and Importing Projects on page 24

Command 'Export PLCopenXML'

Function: This command opens a dialog box for exporting objects from a project into an XML file in the PLCopen format.

Call: Menu *"Project"*

Dialog box 'Export PLCopenXML'

The dialog box lists all objects from the Device tree that I/O Engineering can export into an XML file in accordance with the PLCopen format.

The PLCopenXML scheme does not permit VAR_GLOBAL and VAR_GLOBAL CONSTANT POUs to be in the same variable list. Therefore, if you wish to export both, you must first divide the variables into two separate variable lists.

See also

- Chapter Command 'Import PLCopenXML' on page 149
- Chapter 9.2 Exporting and Importing Projects on page 24

Command 'Import PLCopenXML'

Function: This command opens a dialog box for importing objects from an XML file in PLCopen format.

Call: Menu *"Project"*

Requirement: A project is open.

Dialog box 'Import PLCopenXML'

The dialog box lists all objects from the PLCopen export file that I/O Engineering can import at this point.

"Currently selected target object"	Object that is selected in the Device tree
"Insertable items"	Displays all objects of the export file that I/O Engineering can insert below the selected object.

“Conflict Resolution”	When objects are imported which have the same names as existing objects, the conflicts can be resolved for each object as follows: • Replace the existing object: The object which exists in the project is overwritten by the imported object. • Rename the new object: The new object is imported with the changed name. The string _<no> is appended to the end of the name. • Skip the new object: The object is not imported.
“Select”	Opens a list box for selecting object types.
“Deselect”	Opens a list box for deselecting object types.
“Conflicts”	Opens a list box for resolving all conflicts.
“Show Contents”	Opens a dialog where the objects of the XML file are displayed.

The PLCopenXML scheme does not permit VAR_GLOBAL and VAR_GLOBAL CONSTANT POUs to be in the same variable list. Therefore, if you wish to export both, the variables must first be divided into two separate variable lists.

See also

- ➦Chapter Command 'Export PLCopenXML' on page 149
- ➦Chapter 9.2 Exporting and Importing Projects on page 24

Command 'Generate Sercos SCI XML'

The command is not integrated in the standard menu. You can add it via the dialog “Tools → Adapt” from the category “Devices”.

Function: The command opens the standard dialog for saving a file to the local file system. You can define a name and storage location for an XML file in which I/O Engineering should store the EtherCAT configuration of the EtherCAT master currently selected in the device tree. This may be necessary in order to operate an external sercos stack.

Call: context menu of a sercos master device object in the device tree.

Also refer to

- ➦Customizing Menus

Command 'Generate EtherCAT XML'

The command is not integrated in the standard menu. You can add it via the dialog “Tools → Adapt” from the category “Devices”.

Function: The command opens the standard dialog for saving a file to the local file system. You can define a name and storage location for an XML file in which I/O Engineering should store the EtherCAT configuration of the EtherCAT master currently selected in the device tree. This may be necessary in order to operate an external EtherCAT stack.

Call: context menu of an EtherCAT master device object in the device tree.

Also refer to

- ➦Customizing Menus

Command 'User management' – 'Log in User'

Symbol: 

This command opens the dialog box “Login”. Here you specify the project that you wish to edit and enter the login data for a user account with the corresponding rights. In addition, you can open the password manager from this dialog box.

The command is available in the menu *“Project → User Management”*.

See also

- ➦ Chapter 11.11 Logging in via user account and password manager on page 35

Command 'User management' - 'Logout user'

Symbol: 

The user currently logged in to the project is logged out again with this command. This takes place without a dialog or message, unless no user is currently logged in.

The command is available in the menu *“Project → User administration”*.

The status bar always displays the user who is currently logged into the project.

A double-click the field “Current user” in the status bar enables quick access to the “Login” or “Logout” dialog.

Also refer to:

- ➦ Chapter 11.11 Logging in via user account and password manager on page 35
- ➦ Chapter Command 'User management' – 'Log in User' on page 150

Command 'User management' – 'Rights...'

This command opens the dialog box “Rights”, in which you define the actions that may be carried out, the user groups that may carry them out and the project objects on which they may be carried out.

The command is available in the menu *“Project → User Management”*.

See also

- ➦ Chapter 11.10 Protecting Objects in the Project by Access Rights on page 34

Command ‘Insert device'

Function: As with the “Mount device” command, the command opens the “Mount device” dialog for selecting a device object. This object is inserted in the device tree at the same level as the currently selected object.

Call: context menu of a device object in the device tree.

Prerequisite: an object is selected in the device tree below which a device object can be inserted at the same level.

Also refer to

- ➦ Chapter Command 'Attach device' on page 139
- ➦ Chapter 14.2 Device tree and device editor on page 45

Command 'Disable device' - 'Enable device’

Function: this command switches back and forth between the activated and deactivated states of a device in the bus system.

Call: context menu of a device object in the device tree.

Prerequisite: the project is in offline mode. The bus driver must support the function.

A deactivated device is not taken into account and is not addressed. Note that with some bus systems the deactivation of a node can lead to the master stopping.

The entry of a deactivated device in the tree appears in light-gray fonts. When logging in, disabled devices are additionally marked with a red triangle ⚠.

Command 'Scan for Devices'

Function: The command establishes a brief connection to the hardware and determines the devices in the network. Then you can apply the devices found into the device tree of your project.

Call: Menu bar: *"Project"*; context menu of a device object in the device tree

Requirement: The communication settings to the controller are correct. The gateway and the PLC are started. The device supports the scan function.

The following devices provide the scan function: EtherNet/IP Scanner (IEC) and Profinet Controller.

You can perform the device scan immediately if the scan function is permanently implemented in the PLC. When scan function is implemented in a library, you have to log in only one time to download the library to the controller.

The command refers to the master controller selected in the device tree. For example, an already inserted PROFINET IO controller can be selected and the command used to determine the I/O devices and I/O modules assigned to it.

After performing the scan operation, the "Scan Devices" dialog opens and displays the found devices.

Dialog 'Scan Devices'

Table 19: "Scanned Devices"

"Device name, Device type, Address, Station name, etc."	Data about the scanned device depending on network type. When you change a value in the list of scanned devices, the value is shown in italics. This indicates that the new value has been changed in the editor in I/O Engineering, but not in the device. When you download the value to the device, it is shown normally. Value that indicate differences between the project and the scanned device are shown in orange. If multiple device descriptions are available for the scanned device, then the name is displayed in bold. The selection of the matching device description is resolved differently for different fieldbuses. For more detailed information, see the corresponding fieldbus chapters. If a device description cannot be found, then the following message is shown: "Attention! The device was not found in the repository." Depending on the bus system, additional information is displayed, such as manufacturer number and product number. The device cannot be inserted into the project without the installed device description.
"Show differences to project"	☑: The table in the dialog also shows additional configured devices (in the device tree of the project). ☐: The table shows all scanned devices. The configured devices are not shown.
"Scan for Devices"	Starts a new search.
"Copy All Devices to Project"	The device that is selected in the table is inserted into the device tree in the project. If nothing is selected, then all scanned devices are shown.

NOTICE

If you insert devices, which are available in the device tree, to the device tree with "Copy All Devices to Project", then the following should be noted. The data of the "Process Data" and "<...> I/O Mapping" tabs of the existing devices can be overwritten with the data of the recently inserted devices.

Table 20: “Configured Devices”

This part of the dialog is visible only when you select the “Show differences to project” option. Differences between the scanned and configured devices are color-coded. Devices displayed in green are identical on both sides. Devices displayed in red are available only in the view of the scanned or configured devices.	
	If you have selected a device in both views, then the scanned devices are inserted above the selected configured device.
	If you have selected a device in both views, then the scanned devices are inserted below the selected configured device.
	If you have selected a device in both views, then the configured devices are replaced by the selected scanned device.
	All scanned devices are copied to the project.
	Deletes the selected configure device.

The dialogs for the scan differ depending on the type of device. See the help pages for the respective device editor.

See also

- ➦ Scanning the current hardware and transferring devices to the project on page 50

21.3.6 Menu “Online”

Command “Compare project with ctrlX”

Symbol:

Call:

In thectrlX I/O Engineering interface in the “Online” menu

Function:

The command opens a comparison view of the project data sources on the Engineering PC and on the ctrlX device, see also:

➦ 'Compare’ command

➦ Project data source - Introduction

Command “Show online data”

Symbol:

Call:

The command can be called at two positions on the ctrlX I/O Engineering interface:

- Via the context menu of the EtherCAT master node in the device tree
- Via the ctrlX I/O Engineering toolbar when the EtherCATmaster node is selected in the device tree

Prerequisite:

The command is only available, if a communication to the ctrlX device is established, see:

➦ Command “Communication settings...”

Function:

The command enables the online data transfer between the ctrlX device and ctrlX I/O Engineering. For example, in this setting, additional tabs are displayed in the generic device editor.

Additional tabs for the EtherCAT master:

- ➦ Tab – “Bus overview”
- ➦ Tab – “Diagnostics”
- ➦ Tab – “Slave statistics”

Command “Download”

Symbol:

Call:

The command can be called at two positions on the ctrlX I/O Engineering interface:

- Via the context menu of the EtherCAT master node in the device tree
- Via the ctrlX I/O Engineering toolbar when the EtherCATmaster node is selected in the device tree

Function:

The command transfers the EtherCAT configuration of ctrlX I/O Engineering to the ctrlX device.

The configuration can only be transferred if the EtherCAT operating state of the ctrlX device is set to "Init". If the operating state is not on "Init", a dialog to switch the operating state is displayed. Confirm the dialog for status switching to transfer the configuration to the device. A message is output when the configuration has been downloaded successfully.

21.3.7 Menu “Tools”

Command 'Add-on Installer'

The command opens the dialog “Add-on Installer” for managing software add-ons in ctrlX PLC Engineering and ctrlX I/O Engineering, see:

➦ 'Add-on installer' dialog

Symbol

Call

Menu *“Tools → Add-on Installer”*

Further topics:

➦ Chapter 17.2 Installing/uninstalling add-ons on page 58

Command 'Device Repository'

Symbol:

Function: This command opens the “Device Repository” dialog. This dialog is used for managing the devices that are installed on the local system and can be integrated into I/O Engineering projects.

Call: Menu bar: *“Tools”*.

Dialog 'Device Repository'

⚠ CAUTION	**Do not change the internal device repository manually.** Do not copy any files to or from the repository. Always use the device repository dialog to install or uninstall devices.

“Location”	Shows the device repository directory on the local system. The list box shows the currently set save locations. By default, I/O Engineering creates the system repository during installation. The devices of the selected location are listed in the “Installed device descriptions” field.
“Edit Locations”	Opens the “Edit Repository Locations” dialog.

Table 21: Dialog 'Edit Repository Locations'

List of the repositories with “Location” and “Name”.	
“Add”	Creates a new repository. Opens the “Repository Location” dialog. The selected directory (“Location” input field) must be empty or it must be a valid repository.
“Edit”	Opens the “Repository Location” dialog (see “Add”).
“Remove ”	A dialog prompt opens for you to decide whether the respective directory should also be deleted from the hard disk.

Table 22: “Installed Device Descriptions”

List of device descriptions in multilevel tree structure. Shows all device descriptions with “Name”, “Vendor”, and “Version”. The top nodes represent device categories, for example PLCs, fieldbuses, and logical devices.	
“String for full-text search in all devices”	This field is editable after clicking in it. For any character string entered, only those devices that include the character string are displayed in the lower view. The matched string is highlighted in yellow for these devices.
“Vendor”	Drop-down list with manufacturers whose available devices are displayed.
“Install”	Opens the “Install Device Description” dialog. For the default devices with file type "*.devdesc.xml". You can also select manufacturer-specific description files, such as "*.gsd" files for PROFIBUS DP modules, "*.eds" and "*.dcf" files for CAN devices. When you click “OK” to confirm the selection, I/O Engineering inserts the new device into the device repository. If an error occurs during installation (for example, missing files that are referenced by the device description), then I/O Engineering reports the error to the lower part of the device repository dialog.
“Uninstall”	Removes the selected device. If you delete the device from the device repository, then it is no longer available for use in the programming system.
“Renew Device Repository”	Updates all devices in the device repository. When new versions of import plug-ins are available, some device descriptions may be outdated. The affected devices are marked with a warning symbol (⚠). This command opens a dialog to confirm the update.

“Download Missing Device Descriptions”	Opens when you use devices in your project that are not available in the device repository. When you execute this command, a list of missing devices is displayed. There you can select the corresponding devices for download.
“Details”	Opens the “Details” dialog for the selected device description. This dialog provides additional information from the device description file.

NOTICE	
	During installation, I/O Engineering copies the device description files and all additional reference files to an internal location. Therefore, any changes to the original files no longer influence the installed devices. You must reinstall the devices to make any changes effective. We recommended that you change the internal version number of a device description after a modification.

See also

- ➦Chapter 16.2 Installing Devices on page 56
- ➦Chapter 'Options' dialog - 'Download device descriptions' on page 180

Command 'Scripting' - 'Run script file'

Symbol: 

Function: The command opens a dialog to select the script file (*.py) and then executes the selected script file.

Call: Menu *“Tools → Scripting”*

Also refer to

- ➦Calling scripts from toolbar icons

Command 'Scripting' - 'Enable script tracing'

Symbol: 

Function: The command makes I/O Engineering print all commands from the script file to the message view. Use this command for monitoring and debugging scripts. A blue frame around the symbol indicates that the option is active.

Call: *“Tools → Scripting”* menu

Also refer to

- ➦Calling scripts from toolbar icons

‘Scripting’ command - 'Scripts

Function: The command executes a script stored in the `ScriptDir` folder.

Call: *“Tools → Scripting → Scripts”*

Prerequisite: The `ScriptDir` subdirectory is created in the I/O Engineering installation directory. Python scripts are stored in this folder with the file extension `.py`.

All scripts that are contained in the `ScriptDir` folder are executable as menu commands and are sorted alphabetically by file name.

Also refer to

- ➦Calling scripts from toolbar icons

"Customize” command

Function: The command opens the “Customize” dialog. In this dialog you adjust the menus, the toolbars and the keyboard layout to your individual requirements.

Call: Menu *“Tools”*

Also refer to:

- ➦ Customizing Menus
- ➦ Customizing Toolbars
- ➦ Customizing Keyboard Shortcuts

Command 'Options'

Function: The command opens the “Options” dialog for configuring the I/O Engineering options. These options define the behavior and appearance of the I/O Engineering user interface. I/O Engineering saves the settings in your current user profile on your local system. The current profile specifies the standard settings.

Call: Menu *“Tools”*

Also refer to

- ➦ Chapter 'Options' dialog - 'Device editor' on page 181
- ➦ Chapter 'Options' dialog - 'Download device descriptions' on page 180
- ➦ Chapter Dialog 'Options' – 'International Settings' on page 182
- ➦ Chapter ‘Options’ dialog - ‘Load and save' on page 183
- ➦ Chapter Dialog 'Options' - 'PLCopenXML' on page 184
- ➦ Chapter 'Options' dialog - 'Proxy settings' on page 184
- ➦ Chapter 'Options' dialog - 'Refactoring' on page 185
- ➦ Chapter 'Options' dialog - 'Text editor' on page 186

Command 'Import and export options’

Function: The command opens the “Import and Export Options” dialog. Here you can configure the export and import of selected settings of the I/O Engineering options. The settings are saved to an XML file with the default extension (`options.xml`).

Call: Menu *“Tools”*

Dialog 'Import and export options'

“Export selected options”	“Select options”: In the table you can select the categories of machine (computer) or user-related options whose current settings should be exported to the XML file. “File”: Path of the export file in the local file system. Example: `D:\system1.options.xml`. Button …: Opens the standard dialog to search for an existing file or to create a new file in the local file system. The “File type” `option export (*.options.xml)` is preset.
“Import selected options”	“File”: Path of the options export file whose contents are to be imported. Button …: Opens the default dialog to search for an existing file of type `option export (*.options.xml)` in the local file system. After you click “OK” to close the dialog, the settings described in the file are applied to the project.

Also refer to:

- ➦ Setting I/O Engineering Options

21.3.8 Menu “Window”

Command 'Next Editor'

Keyboard shortcut: [Ctrl]+[F6]

Function: This command switches focus from the currently active view to the next view. The next view is identified by the tab to the right of the currently active tab.

Call: Main menu *"Window"*

Requirement: At least one object is open.

See also

- Chapter Command 'Previous Editor' on page 158

Command 'Previous Editor'

Keyboard shortcut: [Shift]+[Ctrl]+[F6]

Function: This command switches focus from the currently active view to the previous view. The previous view is identified by the tab to the left of the currently active tab.

Call: Main menu *"Window"*

Requirement: At least one object is open.

See also

- Chapter Command 'Next Editor' on page 157

Command 'Close all editors'

Symbol: 

Function: The command closes all currently open editor views.

Call: Menu *"Window"*

Prerequisite: At least one editor is open.

Also refer to

- Chapter Command 'Close all editors of inactive applications' on page 158

Command 'Close all editors of inactive applications'

Function: The command closes all editor views for objects that are located directly below a currently inactive application. Object editors in the POU view remain open.

Call: Menu *"Window"*

Prerequisite: At least one object of an inactive application is open.

Also refer to

- Chapter Command 'Close all editors' on page 158

Command 'Reset Window Layout'

Function: This command resets all currently open windows and views to their default positions. You are prompted for a confirmation before the command is executed.

Call: Main menu *"Tools"*

Command 'New Horizontal Tab Group'

Symbol: 

Function: This command moves the currently active view to a new, separate tab group below the existing one.

Call: Main menu *"Window"* or context menu of the tab

Requirement: Several editor views are open as tabs next to each other.

If you open another object in the editor, then this is automatically included in the tab group that is currently in focus.

See also

- Chapter Command 'New Vertical Tab Group' on page 159

Command 'New Vertical Tab Group'

Symbol:

Function: This command moves the currently active view to a new, separate tab group to the right of the existing one.

Call: Main menu *"Window"* or context menu of the tab

Requirement: Several editor views are open as tabs next to each other.

If you open another object in the editor, then this is automatically included in the tab group that is currently in focus.

See also

- Chapter Command 'New Horizontal Tab Group' on page 158

Command 'Float'

Function: This command releases a docked view from its frame in the user interface and repositions it on the screen as a floating window.

Call: Main menu *"Window"*

Requirement: The application is in online mode.

This window can then be positioned outside of the user interface. Use the "Dock" command to return a floating window to the frame of the user interface.

See also

- Chapter Command 'Dock' on page 159

Command 'Dock'

Function: This command returns a floating window, which was released by the "Float" command, to the frame of the user interface.

Call: Main menu *"Window"*

See also

- Chapter Command 'Float' on page 159

Command 'Auto Hide'

Shortcut [F7]

Function: The command shows or hides a view.

Call: Menu *"Window"*

Hide simply means that I/O Engineering minimizes the view to a tab at the bottom of the user interface which is visible only when you move the mouse over the tab. The command works as an option, i.e. when a window is hidden, it appears in the menu with a check mark in front of it. When you click the command again, the check box is cleared and the window is shown.

Also refer to:

- Auto-Hiding Windows

Command 'Next Pane'

Keyboard shortcut: [F6]

Function: This command sets the focus on the next pane.

Call: Main menu *"Window"*

Requirement: An object is open that contains two or more panes.

Example: If an object is open in the ST editor and the cursor is currently in the declaration section, then command sets the focus to implementation section.

See also

- Chapter Command 'Previous Pane' on page 160

Command 'Previous Pane'

Keyboard shortcut: [Shift]+[F6]

Function: This command sets the focus on the previous pane.

Call: Main menu *"Window"*

Requirement: An object is open that contains two or more panes.

Example: If an object is open in the ST editor and the cursor is currently in the declaration section, then command sets the focus to implementation section.

See also

- Chapter Command 'Next Pane' on page 159

Command 'Windows'

Function: This command opens the "Windows" dialog box, which lists all open objects. You can then activate or close any of the listed views.

Call: Main menu *"Window"*

Commands of the Submenu 'Window'

Function: The command activates the selected window.

Call: Main menu *"Window"*

For each opened editor window the menu "Window" contains a command "<n><object name>". Choosing this command activates the corresponding window. In offline mode I/O Engineering adds the extension "(Offline)". To differentiate between the implementation or the instances of a function block the extension "(Impl)" or "<instance path>" is added.

21.3.9 Menu "Help"

Command '3S Homepage'

Function: This command opens the Bosch Rexroth AG home page.

Call: Main menu "Help"

Command 'Find'

Symbol: 

Function: This command opens the I/O Engineering help.

Call: Menu bar: "Help".

In the online help, you can run a full-text search from the input field on the top right of the help page. The "Find" tab opens in the offline help.

Table 23: Tab 'Search'

"Search for"	Combo box for defining the search term or for selecting the 25 most recent search terms.
" Search in titles only"	The search is performed only in the titles of the help pages.
"Display partial matches"	Displays terms also as search results that include the search term.
"Limit to .... matches "	Limits the number of search results. Maximum value: 1000
"Find"	Starts the full-text search.

Command 'Index'

Symbol:  ; keyboard shortcut: [Ctrl]+[Shift]+[F2]

Function: This command opens the I/O Engineering help.

Call: Menu bar: “Help”.

An index search is not possible in the online help. The “Index” tab opens in the offline help.

All index entries of the help are listed alphabetically in the index view.

“ Look for ”	As you type letters into the input field, I/O Engineering searches automatically for matches in the index list.
“Display”	Opens the help page for the highlighted index entry in the list and displays the title of the help page and location of the help file (*.chm) in the “Index results for <index entry>” view. When several pages are found and then displayed in this view, then you view a specific help page by clicking its entry in the list. Clicking an entry in the index list achieves the same result.

Command 'Contents'

Symbol: ; keyboard shortcut: [Ctrl]+[Shift]+[F1]

Function: This command opens the I/O Engineering help.

Call: Menu bar: “Help”.

Command 'About'

Function: This command opens a splash screen with information about the I/O Engineering version and copyright. In addition, buttons are available for detailed information about the version, license, and acknowledgments.

Call: Main menu “Help”.

“Version Info”	Opens the “Detailed Version Information” dialog box with a list of I/O Engineering components and information about the operating system. “Export”: Exports the detailed version information as a *.txt file or in any other format.
“License Info”	Opens the “License Information” dialog box. • “Plug-in”: Drop-down list for the plug-in to display the license information • “Software License”: License information about selected “Plug-in”
“Acknowledgments”	

21.3.10 “EPLAN” menu

Command “Create new project via EPLAN import...”

The command is used to import an existing EPLAN project into a new I/O Engineering project for the first time.

The aim of the import is to transfer the device and configuration data contained in the EPLAN project to a new I/O Engineering project in order to reduce the general project planning effort in I/O Engineering.

For further information, see: ➦Use EPLAN data in I/O Engineering

Prerequisites

The command is only available if the following requirements are met:

- The add-on "Bosch Rexroth EPLAN" is installed in I/O Engineering, see: ➦Installing the add-on "Bosch Rexroth EPLAN"
- No project is open in I/O Engineering (empty device tree).

Call

Via the menu item “EPLAN”

Command “Import from EPLAN...”

The command is used to subsequently import EPLAN project data into an existing I/O Engineering project.

The aim of the subsequent import is to transfer the device and configuration data contained in the EPLAN project to an existing I/O Engineering project.

NOTICE	The import overwrites existing project data and adds new data if necessary!

For further information, see: ➦Use EPLAN data in I/O Engineering

Prerequisites

The command is only available if the following requirements are met:

- The add-on "Bosch Rexroth EPLAN" is installed in I/O Engineering, see: ➦Installing the add-on "Bosch Rexroth EPLAN"
- If a ctrlX project is open in I/O Engineering and the control node is selected in the device tree.

Call

Via the menu item “EPLAN”

21.4 Dialogs

21.4.1 'Import Wizard' dialog

Function: The dialog facilitates applying I/O Engineering options and package installations of an older I/O Engineering installation found on a local computer.

Call: The dialog is displayed when a newly installed I/O Engineering version is started for the first time and an older version is already installed on the computer.

“Program settings”	☑: The user-specific I/O Engineering options are transferred from the older installation to the new installation.
“Packages”	☑: The packages installed with the older I/O Engineering version are transferred to the Package Manager of the new version. See the list of discovered package installations with the “Name”, “Version”, and “Installation date”.
“Importing”	The program settings and/or options are transferred to the current I/O Engineering version.
“Skip”	The program settings and/or options are not transferred to the current I/O Engineering version.

Also refer to:

- ➦Setting I/O Engineering Options

21.4.2 Dialog 'Permissions'

Function: The permissions of user groups are defined here with which they can execute specific actions on specific objects in the project.

Call: Menu bar: *“Project → User Management”*.

Every change made in the dialog is applied immediately.

Actions

All possible actions on objects of the projects are listed in “Actions”. The actions are divided into four categories and assignments to all current objects of the project are listed below each action. For each "action->object" assignment, you can define the permission for each existing user group.

Action categories:

“Commands”	Actions regarding the execution of commands
“Users, groups and permissions”	Actions regarding the configuration of user accounts, user groups, and their permissions
“Object types”	Actions regarding the creation of object types
“Project objects”	Actions regarding the viewing, modification, removal, and child-object handling of objects of the project

Actions in detail:

“Execute”	Execute a menu command
“Create”	Create a new object in the project
“Add or remove children”	Add or remove a child object below an existing object
“Modify”	Modify an object in the editor or modification of user, group, and permission settings in the corresponding editor/dialog
“Remove”	Delete or remove an object
“View”	Open the view of an object in the editor
	Possible target of an action. This can be specific objects of the project, or the user, group, and permission configuration.

Permissions

All defined user groups (except the "Owner" group) are listed in “Permissions” with a toolbar for configuring the permissions of a group.

Granted	The action, which is selected in the actions view, on the selected target(s) is **granted** for the selected group.
Denied	The action, which is selected in the “Actions” view, on the selected target(s) is **denied** for the selected group.
	The permission that executes the actions, which are selected in the “Actions” view, on the selected targets has not been defined explicitly. However, the actions are **granted by default**; for example, because the corresponding permissions have been granted to the parent object. Example: The group has the permission for the object "myplc". As a result, it also has the permission by default for the object "myplc.pb_1".
	The action, which is expanded in the actions view, on the selected targets has not been denied explicitly. However, it is denied by default; for example, because it has been denied to the parent object.
No symbol	There are currently multiple actions selected in the Actions view for which the group does not have the same permission.

Toolbar:

“Grant”	The selected action on the selected target object is granted explicitly for the selected group.
“Deny”	The selected action on the selected target object is denied explicitly for the selected group.
“Clear”	The permission for the selected action on the selected target object is reset to the default value for the selected group.

“Export/Import”	Opens a menu with the commands • “Export all permissions” • “Export selected permissions” • “Import permissions”
“Export all permissions”	Exports all actions and their configured access permissions of the current project to a user-specific file of data type `*.perms`. To do this, the “Export Permissions” dialog opens for you to specify a file name and to select a location in the file directory. The default file type is `Permissions (*.perms)`.
“Export selected permissions”	Exports all selected actions and their configured access permissions of the current project to a user-specific file of data type `*.perms`. To do this, the “Export Permissions” dialog opens for you to specify a file name and to select a location in the file directory. The default file type is `Permissions (*.perms)`.
“Import permissions”	The contents of a `*.perms` file is merged with the actions and permissions of the current project. Groups that are part of the file but not part of the project are ignored. The actions and permissions are aligned by name. To do this, the “Import Permissions” opens for you to select the `*.perms` file from the file system.

See also

- ➦Chapter 11.10 Protecting Objects in the Project by Access Rights on page 34

21.4.3 Dialog “Login - [Control IP address]”

Comprehensive information:
➦Logging in and logging off to/from the control

Call:

Call the dialog using the Logging in device user... command, see ➦ Chapter Command “Logging in device user...” on page 138

Function:

The dialog is used to log in a user to a ctrlX device.

To log in, the user name and the password has to be entered in the dialog.

The Engineering system “remembers” the entered login data until the next restart. Consequently, the login dialog is not displayed each time the user wants to log in.

When enabling the checkbox “Save login data for this device”, the login data in the Windows Credential Manager and are available after a restart of the Engineering system.

21.4.4 'Add-on installer' dialog

The dialog is used to manage additional installable software packages in ctrlX PLC Engineering and ctrlX I/O Engineering, see also:

➦Installing/uninstalling add-ons

Symbol

Call

Menu *“Tools → Add-on Installer”*

Description

The dialog is divided into two sections:

- Header
- Add-on overview

Header

The header contains the following information and setting options:

Information/setting	Description
“Version”	Name of the installation with version number and platform specification
“Location”	Storage location of the installation. It is possible to change the location by clicking on“Browse”.
“Channel for setups”	Service channel for setups
“Channel for AddOns”	Service channel for add-ons

Add-ons overview

The overview is divided into three tabs and contains commands for managing add-ons.

Tabs	Description
“Installed”	Lists all currently installed add-ons.
“Browse”	Lists all available and compatible add-ons that have not yet been installed. To install the add-ons, select the add-on in the list and execute the installation command that appears.
“Updates”	Lists all currently installed add-ons for which at least one update is available. To install the update, select the add-on in the list and execute the installation command that is displayed.

Commands	Description
“Install File”	Opens the file selection dialog, for selecting an add-on. • Confirming the dialog starts the installation of the add-on • Confirm the following license agreements and signature information • The user is informed about the progress during the installation • A message is output once the installation is completed
“Export config”	Opens the file selection dialog so that it is possible to create and store a configuration file. The file contains information about all settings of the dialog.

Commands	Description
“Import config”	Opens the file selection dialog so that you can select a previously stored configuration file. The file contains information that is used to preset all other settings in the dialog.

The ctrlX Add-on Installer is based on functions of the CODESYS installer of the CODESYS GmbH.

The complete documentation can be found on the CODESYS website, see: ➦Web documentation

21.4.5 Configure dialog – Edit PDO List

Call

In the tab “Expert Process Data” of a EtherCAT slave device, see ➦ Tab – Expert Process Data.

Function

This dialog facilitates adding or editing of entries in the PDO list.

Elements of the dialog

Function	Description
“Name”	PDO name
“Index”	PDO index (or IDN for SoE)
“Direction”	• “TxPDO”: PDO is transferred from slave to master • “RxPDO”: PDO is transferred from master to slave
“Flags”	• “F” Fixed content: The contents of the PDO are fixed and cannot be modified. It is not possible to add entries to the PDO content. • “M” Mandatory: The PDO is necessary and cannot be deactivated in the PDO assignment. • “V” Virtual PDO: Reserved for future use
“Exclude PDOs”	It is possible to specify an exclusion list. If a PDO is enabled in the PDO assignment, the others are inactive and cannot be enabled.
“Sync unit”	ID of the SyncUnit in the slave (Sync manager) to assign the PDO to

21.4.6 “Project synchronization" dialog

Dialog “New Project”

Call:

The dialog is called automatically when creating a new project.

A new project can be created using the following menu command: *“File → New Project...”*

Function:

When creating new ctrlX CORE control projects, basic settings have to be selected.

The “New Project” dialog is used to configure the following settings:

- Selecting the control to be configured in the network
- Synchronization settings between control and Engineering PC

Dialog elements:

Element	Description
“Project storage on PC”	Display of: • Project name • Project data storage in the file system of the Engineering PC The displayed settings cannot be changed in this dialog! To change the settings, the dialog has to be canceled using the *“Cancel”* interface. Subsequently, execute the steps to create a new project and specify the settings, see ➦ Chapter 8.2 Creating a new project on page 21.
ctrlX CORE	Configuration of the following settings: • “Control”: Selecting the control to be configured in the network • “Configuration”: Selecting the active configuration on the control on which the project is to be operated on the selected control
Synchronisation PC and ctrlX CORE	There is the option to additionally store all relevant project data at certain points in time on the ctrlX CORE control or to open a project on the control. Setting options: • "☑ **Synchronization of project storages between PC and ctrlX CORE if:**" Enabling/disabling the synchronization. In case of an enabled synchronization, the following conditions have to be configured: – "☑ **The project is opened**" In this setting, the project is synchronized upon opening of the project – "☑ **the project is saved on the PC**" In this setting, the project is synchronized upon saving of the project – "☑ **an application is downloaded on the ctrlX CORE**" In this setting, the project is synchronized upon download of the project to ctrlX CORE – "☑ **the project is closed**"

Element	Description
	In this setting, the project is synchronized upon closing of the project • "☑ **Synchronize device description files and libraries**" With this setting, device description files and libraries (only in the PLC context) that are not part of the ctrlX WORKS installation are additionally stored on the control. This option facilitates the direct project login. For more information about the project synchronization, see: ➦ Project data source - Introduction

Related topics

➦ Chapter 8.2.1 Creating a project on the basis of a template for ctrlX CORE Creating devices on page 22

Dialog “Project data comparison”

Call:

Automatically (no manual access option).

Function:

Upon the I/O Engineering start, the system compares the project configuration on the control to the project configuration in I/O Engineering. If the comparison showed differences, you can display the project data comparison. Additionally, extended project information can be displayed via button “Details...”.

Further information:

- ➦ Chapter Command “Project synchronisation...” on page 141
- ➦ Chapter 'Compare’ command on page 143

Dialog “Project synchronization”

Prerequisite:

The “Project synchronization” dialog is exclusively supported by the ctrlX CORE and ctrlX CORE Virtual devices.

Symbol:

Call:

Open the dialog as follows:

- Via the command “Project synchronisation...”, see: ➦ Command “Project synchronisation...”
- In the lower I/O Engineering status bar, double-click on the  symbol

Function:

The dialog “Project synchronization” is used to configure the project synchronization between the ctrlX device and I/O Engineering on the Engineering PC.

Dialog elements:

Project storage on PC	
“Name”	Project name

Project storage on PC	
“Location”	Filing of the project data in the file system of the Engineering PC

ctrlX CORE	
“Control / IP”	Selecting the target control from which the project is to be opened. Enter the IP address or select a device from drop-down menu list of the ctrlX CORE control accessible in the network (*“Project → New project”*) List of configured ctrlX CORE controls (*“Project → Project synchronization...”*)
“Connection test”	When confirming the button with the “Arrow symbol”, the connection to the control is tested and the current status is displayed.

Synchronization PC and ctrlX CORE (default setting)		Description
☑ **Synchronization of project storages between PC and ctrlX CORE if:**		Enabling/disabling the synchronization function. If active, then under the following subconditions:
	☑ **the project is opened**	Is automatically enabled if the synchronization is active
	☑ **the project is saved on PC**	Synchronization when saving the project
	☑ **a download / OnlineChange is carried out and thus the project is saved**	Synchronization when downloading an application to the control
	☐ **A boot application is created and login data is synchronized**	Synchronization occurs when the command “Generate boot application" is executed in the logged-in state .
	☐ **the project is closed**	Synchronization when closing the project
☐ **Synchronize device description files and libraries**		Device description files and libraries that are not part of the ctrlX WORKS installation are also saved on the control. This option facilitates direct login to the project, if active.

Related topics

➦ Command “Project synchronisation...”

➦ Project data source - Introduction

Dialog “Project synchronization” (Synchronization cache...)

Prerequisite:

The dialog “Project synchronization” is exclusively supported by the ctrlX CORE, ctrlX CORE Virtual devices.

Symbol:

Call:

In I/O Engineering via the “Synchronization cache...” command, see:

➦ Command “Synchronization cache...”

Function:

The dialog is used to manage the cache of all projects on the Engineering PC

Dialog elements:

Element	Description
Display “Size”	Size of the assigned memory for the synchronization cache
Display “Location:”	Storage location of the cache in the file system. The information provides a link. Via the link, the current directory can be directly opened in the Windows file Explorer.
Button “Clear cache”	Button to delete the cache and the contents of the file repository
Button “Close”	Button to close the dialog

Related topics

➦ Command “Synchronization cache...”

➦ Project data source - Introduction

Dialog “Open project in ctrlX CORE”

Prerequisite:

The configured ctrlX CORE device has to be available in the network for the I/O Engineering to connect to the ctrlX device.

Symbol:

Call:

Via the “Open project from ctrlX CORE...” command, see ➦ Chapter Command “Open project from ctrlX CORE...” on page 129

Function:

Use the “Open project in ctrlX CORE” dialog to retrieve a project from the control that was saved on the control via ➦ Chapter Dialog “Project synchronization” on page 168 with the corresponding sources.

The data is applied in two steps. First select the control.

1. Select the control.

ctrlX CORE	
“Control / IP”	Selecting the target control from which the project is to be opened. Enter the IP address or select a device from drop-down menu list of the ctrlX CORE control accessible in the network (*“Project → New project”*) List of configured ctrlX CORE controls (*“Project → Project synchronization...”*)
“Connection test”	When confirming the button with the “Arrow symbol”, the connection to the control is tested and the current status is displayed.

2. Confirm the data application via the “Next” button. Optionally, the operation can be canceled.
3. Before retrieving the data, authentication is required for access to the control. Login to the control with your user data.
4. If you have the required access rights, a dialog to store the data is displayed. The name and the directory under which the project data are stored in the file system of the PC are specified here.

Project storage on PC	
“Name”	Project name
“Location”	Filing of the project data in the file system of the Engineering PC

5. Enter the values in the corresponding fields.
   If no project is stored on the ctrlX device, a new project is created (project synchronization option active).
6. Confirm the data application via the “OK” button. Thus, the project is opened and can be edited.
   Optionally, the operation can also be canceled here.

Dialog “Project synchronization” **(PC > ctrlX)**

Call

Automatically, if the project is called via I/O Engineering and the synchronization function has detected differences in the project files of the ctrlX device and the Engineering PC.

Function

Differences in the project files may occur, for example, if the project is edited on the Engineering PC during offline project planning and there is no connection to the ctrlX device. The project changes cannot be transferred to the ctrlX device in this case. The next time you connect to the ctrlX device, the differences in the project files are detected. In this case, the dialog offers a choice of options on how to handle the project differences between ctrlX device and I/O Engineering, see:

➦ Synchronization – Online behavior

Dialog “Conflict: Project synchronization at open”

Symbol

Call

The dialog is called automatically, for example if differences between the project repository on the Engineering PC and the project repository on the ctrlX device are detected when opening a project.

Function

The dialog offers a choice of options on how to handle project differences between data repositories, see:

➦ Synchronization – Online behavior

Dialog “Conflict: Project synchronization at close”

Symbol

Call

The dialog is automatically called if differences between the project repository on the Engineering PC and the project repository on the ctrlX device are detected when closing the project during project synchronization.

Function

The dialog provides a choice of options how to handle project differences between the repository on the ctrlX device and the Engineering PC, see:

➦ Synchronization – Online behavior

Dialog “Conflict: Project synchronization at save”

Symbol

Call

The dialog is automatically called if the project synchronization detects differences between the project repository on the Engineering PC and the project repository on the ctrlX device when saving the project.

Function

The dialog provides several options on how to handle project differences between the ctrlX device and I/O Engineering, see:

➦ Synchronization – Online behavior

Dialog “Conflict: Project synchronization at equalize project data”

Symbol

Call

The dialog is called up automatically if differences between the project files of the Engineering PC and ctrlX device are detected during the project comparison, see:

➦ Command “Equalize project PC <-> ctrlX CORE”

Function

Differences in the project repositories can occur if the project is edited at the Engineering PC but no connection to the control is established, e.g. in case of offline configuration. In this case, project changes cannot be transferred to the

control. The differences are detected upon the next time, a connection to the control is established. The dialog provides several objects on how to handle project differences between ctrlX device and I/O Engineering, see:

➦ Synchronization – Online behavior

Dialog “Conflict: Project synchronization at open” **(IP address conflict**

Symbol

Call

The dialog is called up automatically if it was detected during project synchronization that the address of the referenced or connected ctrlX device does not match the destination address assigned in the project. This can occur after the following actions:

- If the IP address of the ctrlX device was changed, refer to the ➦ web documentation
- If the project planning data was changed on the ctrlX device via the “Manage app data” window, refer to the ➦ web documentation

Function

The dialog provides a selection of possibilities how project differences between control and PLC Engineering should be handled, see:

➦ Synchronization – Online behavior

21.4.7 “Properties” dialog

'Properties' dialog – General information

The dialog is used to configure the following settings of an object in I/O Engineering. For this purpose, it contains different tabs depending on the object,. Each of the tabs handles a category of properties.

Call: Menu *“View”*, context menu of the object in the “Devices”, “POUs” or “Modules” view.

Dialog 'Properties' - 'Common'

Function: This dialog shows common information about the selected object.

Call: Main menu *“View → Properties”*, or context menu of the object (“Common”).

Requirement: An object is selected in the device tree or POUs view.

“Name ”	Object name as shown in the device tree or POUs view
“Object type ”	Type of object (for example, POU, application, or interface)
“Open with ”	Type of editor to display or edit the object

'Properties' dialog - 'Build'

Symbol 

Function: The dialog contains options for compiling (build process) the object.

Call: Menu *“View → Properties”*, context menu of the object in the device tree

Name	Description
“Exclude from build”	☑: The object and recursively its child objects are not included in the next compilation run. The object entry in the “Devices” or “POUs” view is displayed in green.

Name	Description
“External implementation” “(Late link in the runtime system)”	☑: I/O Engineering does not generate code when compiling the project for this object. The object is not linked until the project is running on the target system, assuming it exists on the target system (for example, in a library). The object name in the “Devices” or “POUs” view is assigned the addition `(EXT)`
“Enable system call”	☑: A system call (runtime system) for functions is possible. Background information: In contrast to CoDeSys V2.3, in V3 the ADR operator can be used with function names, program names, function block and method names. It replaces the INSTANCE_OF operator. BUT: It is not possible to call function pointers within I/O Engineering.
“Always bind”	☑: The object is marked at the compiler and thus always included in the compile information. This means that it is always compiled and loaded on the control. Note: Optionally, the `{attribute 'linkalways'}` pragma can be used to instruct the compiler to always include an object.
“Compiler-Defines”	Here you can enter “Defines" and conditions for compiling the object (conditional compilation). You can also enter the expression `expr` used in such pragmas. Multiple entries are possible in the form of a comma separated list (see `{define}` statements). Example: `hallo, test:='1'`
“Additional compiler definitions from the device description”	
“Defined in the device”	Listing of the compiler definitions that originate from the device description. These compiler definitions are taken into account during the compilation if they are not listed in the “Ignored definitions” field.
“Ignored definitions”	Listing of compiler definitions from the device description that are not taken into account during the compilation.
>	Copies the selected compiler definition from the “Defined in device” field to the “Ignored definitions” field.
<	Moves the selected compiler definition from the “Ignored definitions” field to the “Defined in device” field. The compiler definition is taken into account during the compilation.

Dialog 'Properties' - 'Access Control'

Function: The dialog defines the access rights of user groups for objects.

Call: Main menu *“View → Properties”*, context menu of an object in the view “Device” or “POUs”.

Requirement: An object is selected in the view “Device” or in the view “POUs”.

“Groups, actions and permissions”	A table which displays the following user groups access rights on objects: ● “View” ● “Modify” ● “Remove” ● “Add/remove children” Perform a double click on the access right symbol to open the drop down list with the available rights.

See also

- Chapter 11.10 Protecting Objects in the Project by Access Rights on page 34
- Chapter 21.4.2 Dialog 'Permissions' on page 162

21.4.8 “Project settings" dialog

“Project settings" dialog - General information

The dialogs described below can be called in both ctrlX I/O Engineering as well as in ctrlX PLC Engineering.

However, some of the dialogs are not relevant for working inctrlX I/O Engineering as they are connected to PLC programming in ctrlX PLC Engineering.

'Project settings' dialog - 'Users and groups'

Symbol:

Function: The dialog is for the configuration of the user management for the current project.

Call: Menu *“Project → Project settings”*, “Users and groups” category

'User' tab

Displays the users and their memberships in groups	
“Add”	Opens the “Add user” dialog.
“Edit”	Opens the “Edit User” dialog.
“Delete”	An error message appears if you attempt to delete the last user of a group, since a group must have at least one member.

Table 24: “Add user/edit user”

Input fields for setting up a new user account or changing an existing one	
“Enabled”	☑: You can use the user account which is the default. ☐: The user cannot login. If the user repeatedly tries to log in with incorrect credentials, the account may be deactivated automatically, see below: Settings.
“Memberships”	List of all user groups that you have defined in addition to the group “Everyone” (to which each new user automatically belongs). ☑<Group name> : the new user belongs to the group.

Table 25: “Export/Import”

“Export users and groups ”	The command opens the standard dialog for saving a file to the local file system. You can store the users and groups defined in the project in a `*.users` file in xml format.
“Import users and groups ”	The command opens the standard dialog for searching for a file with the extension `*.users` in the local file system in order to read user and group definitions into the project from the file system.

'Groups' tab

Display of the groups and their members. A group can also be a member of a group.	
“Add”	Opens the “Add group” dialog.
“Edit”	Opens the “Edit Group” dialog.
“Delete”	If you delete a group, the user accounts of the members remain unchanged. You cannot delete the groups “Everyone” and “Owner”.

For the “Export/Import” button, please see “Users” above.

'Settings' tab

Display of the groups and their members in a tree structure. A group can also be a member of a group.	
“Maximum number of login attempts”	☑ (default): If the user has tried to log in with an incorrect password as many times as specified here, the user account is disabled. ☐: The number of the unsuccessful attempts is unlimited
“Log out automatically after inactivity”	☑: If I/O Engineering does not register any user actions via mouse or keyboard for the period of time (minutes) specified here, the user is logged out automatically.

Also refer to

- ➦Chapter 8.3.2 Selecting project settings on page 23
- ➦Chapter Command 'User management' – 'Log in User' on page 150

'Project settings' dialog - 'Compiler options'

Symbol: 

Function: The dialog is used to configure the compiler options.

Call: Menu *“Project → Project settings ”*, “Compiler option” category.

Prerequisite: A project is open

Table 26: “Compiler version”

“Fix Version”	Defines the compiler version that I/O Engineering uses when compiling and during loading for compilation. Example: “3.5.6.0” for version 3.5 SP6

Table 27: “Settings”

“Allow unicode characters for identifiers”	Disabled by default as the use of Unicode characters in identifier names is not allowed in the IEC standard. May be required for some foreign languages (for example, Asian languages).
“Replace constants”	☑ (default): For each constant of scalar type, i.e. not for `STRING`, `ARRAY` or structures, I/O Engineering directly loads the value. In online mode, I/O Engineering identifies the constants in the declaration editor or monitoring window by a symbol preceding the value. In this case, access, for example via an `ADR operator`, forcing and writing, is not possible. ☐: Access to constants is possible, but it prolongs the computation time.
“Enable logging in breakpoints”	For breakpoints defined as execution points, you can enter a message text in the “Breakpoint properties” dialog. I/O Engineering prints this text to the device log when the application halts at the execution point.

Table 28: “Compiler warnings”

“Maximum number of warnings”	Refers to the warnings that I/O Engineering prints to the message view. You define the selection of the displayed compiler warnings in the “ Project settings” dialog in the “Compiler warnings” category.

Also refer to

- ➦Chapter 'Project settings’ dialog - 'Compiler warnings' on page 176

'Project settings’ dialog - 'Compiler warnings'

Symbol: 

Function: The dialog is used to select the compiler warnings that I/O Engineering displays in the message window during a compilation run.

Call: Menu *“Project → Project settings ”*, “Compiler warnings” category.

Prerequisite: A project is open

Set the maximum number of listed warnings in the “Compiler options” dialog.

Also refer to

- Chapter 'Project settings' dialog - 'Compiler options' on page 176

'Project settings’ dialog - 'Download source code'

Symbol:

Function: The dialog defines the compilation and the storage of the source code as a source-code download archive on one or more controllers.

Call: Menu *“Project → Project settings”*, category “Download source code”

A source code download archive is a project archive named `Archive.prj`.

Table 29: “Target device”

Defines the location of the project archive.	
“<name of controller>”	Selected control. I/O Engineering loads the project archive to this control. Prerequisite: The project contains several controls.
“<all devices in the project>”	I/O Engineering loads the project archive to all controls of the project.

Table 30: “Scope”

Defines the content of the project archive.	
“Use compact download”	☑: The project archive contains only that device in the project that contains the active application. ☐: The project archive contains all the devices in the project
“Additional files”	Opens the “Additional files” dialog where you can select additional files for download.

Table 31: “Time”

Defines when I/O Engineering creates a project archive.	
“Implicit in program download and online change”	I/O Engineering loads the project archive to the target device(s) each time an application is loaded and each time an online change is performed, without any further prompting.
“Implicitly when creating a boot application”	I/O Engineering additionally loads the project archive to the target device(s) each time a boot application is created, without any further prompt.
“Implicitly when creating a boot application, download and online change”	I/O Engineering loads the project archive to the target device(s) each time a boot application is created, each time an application is loaded, and each time an online change is performed without any further prompt.
“Prompt during program download and online change”	I/O Engineering opens a prompt every time an application is loaded and every time an online change is made. Select, whether I/O Engineering loads the project archive to the control.
“Only on request”	Only when calling the command *“Online → Load source code on connected control”* opens a command prompt. Select, whether I/O Engineering loads the project archivey to the control.

Dialog 'Project Settings' - 'Page Setup'

Symbol:

Function: This dialog defines the layout for the print version of the project contents. This layout is used for the printout of the project information by clicking *"File → Print"* and the printout of the project documentation by clicking *"Project → Document"*.

Call: Main menu *"Project → Project Settings"* ("Page Setup")

You can change settings the following:

- "Paper"
- "Margins"
- "Header and Footer"
- "Document"
- "Title Page"

Table 32: "Edit Header, Edit Footer"

The headers and footers are structured in table style. You can configure rows and columns, and add text and images to the resulting cells.	
"Row spanning"	Number of rows that I/O Engineering should merge into a single column.
"Column spanning"	Number of columns that I/O Engineering should merge into a single row.
	Opens the list of available placeholders for the "Text" field. When printing the page, I/O Engineering provides the placeholders with the current values.

See also

- Chapter Command 'Page Setup' on page 133
- Chapter Command 'Document' on page 142
- Chapter Command 'Print' on page 133

'Project settings' dialog - 'Security'

Symbol: 

Function: The dialog is used to configure the project protection by a password, a dongle or a certificate.

Call: Menu *"Project → Project settings"*, category "Security"

NOTICE If the encryption password is lost you can no longer open the project. You can also no longer restore it.

"No protection"	: The project file is not protected from unauthorized access and data manipulation. Note: It is strongly recommended to use a security functionality. : The "Password", "Dongle", and "Certificates" options cannot be selected.
"Integrity check"	This option is enabled by default when creating a new project. : The project file is stored in a proprietary format and its integrity is checked each time the project is loaded. The file may be incompatible with older versions of the development system. Please note that the project file is not encrypted. To better protect your data, activate one of the encryption functions.
"Encryption"	: The "Password", "Dongle", and "Certificates" encryption functions can be selected.
"Password"	Entering, changing and confirming the encryption password. If you save the project with these settings you must enter the password later in order to open the project again, even if it is to be loaded as a library reference.

“Dongle”	Prerequisite: You have connected the Codesys security key (dongle) to the computer. “Add”: The dialog “Add registered dongle” is displayed.
“Registered dongles”	Selection list of registered dongles.
“Certificates”	Certificates are used for the encryption of contents of the open project file. Prerequisite: The certificates for all users sharing the project have to be installed in the local storage.  : The “Certificate selection” dialog opens.

Table 33: Adding a registered dongle

“Dongle”	Selection list of all connected dongles.
“Refresh”	I/O Engineering refreshes the drop-down list.
“Flash”	The LEDs of the currently selected dongle flash for two seconds (if it supports this function).

Also refer to

- Chapter 11.7 Assigning passwords on page 33
- Chapter 11.8 Protecting projects using a dongle on page 33
- Chapter 11.12 Encrypting the project with a certificate on page 37

21.4.9 'Options' dialog

'Options' dialog - General information

The options described below can be called in both ctrlX I/O Engineering as well as in ctrlX PLC Engineering.

However, some of the options are not relevant for working inctrlX I/O Engineering as they are connected to PLC programming in ctrlX PLC Engineering.

'Options' dialog - 'Libraries'

Only relevant in the context of PLC programming

Symbol: 

Function: The dialog helps to manage the mappings of library references that I/O Engineering uses during the conversion of an old project. If you have not yet stored any mapping for a certain library, you must redefine the mapping each time when opening an old project in which this library is integrated.

Call: Menu *“Tools → Options”*, category“Libraries”

A mapping defines what a library reference looks like following the conversion of the project to the current format. There are three possibilities:

- You retain the reference. This means that I/O Engineering similarly converts the library into the current format (*.library) and installs it in the local library repository.
- You replace a reference with another reference. This means that one of the installed libraries replaces the library that was integrated until now.
- You delete the reference. This means that the converted project no longer integrates the library.

I/O Engineering applies all the listed mappings to the library references of an old project the next time it is converted. Hence, you must repeat the mapping definition if the same library is integrated again in a project that is to be converted. You can enter a new mapping in the last line.	
“Source library”	Path of the library that is integrated in the project before the conversion. A double-click an entry makes the field editable and the button for the input assistance appears.
“Target library”	Name and location of the library that is to be integrated in the project after the conversion. Double-clicking on an entry opens the “Set target system library” dialog.

Table 34: “Set target system library”

“Search”	“Select library” dialog appears. You can select a library from the library repository here. The dialog corresponds to the dialog in the library repository.
“Ignore....”	When I/O Engineering converts the project, I/O Engineering always removes the existing source library from the project.

‘Options’ dialog - ‘Declaration editor'

Symbol:

Function: The dialog is for the configuration of the display settings for the declaration editor.

Call menu *“Tools → Options”*, category “Declaration editor”

“Textual only”	Textual view of the declaration editor
“Tabular only”	Tabular view of the declaration editor
“Switchable between textual and tabular”	The declaration editor offers two buttons for switching between the textual and tabular views: : text view : tabular view The following option defines the view that appears by default when opening a programming object: ● “Always textual” ● “Always tabular” ● “Save last setting (per object)” ● “Save last setting (global)”

'Options' dialog - 'Download device descriptions'

Symbol:

Function: The dialog is for the configuration of addresses of download servers for device descriptions.

Call: Menu *“Tools → Options”*, category “Download of the device descriptions”.

Also refer to

- ➦ Installing Devices

Table 35: “Download server”

List of download servers containing device descriptions. By default `'https://store.codesys.com/CODESYSDevs'` is entered as the download server. When selecting the “Download missing device descriptions” button in the “Device repository” dialog, I/O Engineering uses the servers entered here and uses the set access data for the proxy server.	
Double-click on “(Enter a new download server here...)”	An input field opens in which you can enter the URL address of a server.
[Del]	Deletes the selected download server.

Also refer to

- Chapter Command 'Device Repository' on page 154
- Chapter 'Options' dialog - 'Proxy settings' on page 184

'Options' dialog - 'Device editor'

Symbol:

Function: The dialog contains settings for the representation of the device editor.

Call: Menu *“Tools → Options”*, “Device editor” category

Tab 'View'

“Show generic device configuration views”	☑: The tab with the list of device parameters is available in the device editors of parameterizable devices.
“Creating cross-references for IEC addresses (cleanup required) ”	☑: I/O Engineering creates the cross-references for unmapped I/Os.
“Communication page”	• “Classic mode”: The “Communication” tab of the device editors is displayed as a two-part window: The left part shows the currently configured gateway channels in a tree structure, the right part shows the related data and information. • “Simple mode”: The “Communication” tab is displayed as described in the corresponding chapter in the help. Additional modes can still be available as customer-specific extensions.
“Show implicit files for application download on the editor of a PLC”	☑: The tab for synchronized files is available in the device editors. Synchronized files are downloaded to the PLC at the time of application download. These can be external files that were added to the application, or implicit files such as a source code archive.
“Show access rights page”	☑: The “Access Rights” tab is available in the device editors. Note: Depending on the device, the device description may overwrite this setting.

'Options' dialog - 'Intelligent coding'

Only relevant in the context of PLC programming

Symbol:

Function: The dialog is used to configure the settings that facilitate the code entry.

Call: Menu *“Tools → Options”*, “Intelligent coding” category

“Automatically declare unknown variables (AutoDeclare)”	☑: The “Declare variable” dialog opens as soon as an identifier that has not yet been declared is entered in a programming language editor and the input line was left. In order for the AutoDeclare function to be available in the ST editor as well, the “Enable for ST editor” option also has to be enabled.
“Enable for ST editor”	Prerequisite: The option “Automatically declare unknown variables (AutoDeclare)” is enabled. ☑: The AutoDeclare function is also available in the ST editor. ☐: The AutoDeclare function is not available in the ST editor.
“Show all variables of an instance in the input assistance”	☑: The "List Components" function also provides the local variables of a function block instance for selection. ☐: The "List components" function only provides the input variables and output variables of a function block instance for selection.
“Display system library symbols in the input assistant”	System libraries are automatically inserted in the library manager and displayed in light gray. ☑: Symbols such as global variables, data types, function blocks, etc are provided in the input help. ☐: The symbols of the system libraries are not available in the input assistance.
“List components after a dot (.) has been entered”	☑: Enables the "List components" function. That means: When entering a dot . at a position where I/O Engineering expects an identifier, a selection list with input options is displayed.
“List components immediately after entering a character”	Prerequisite: “List components after entering a dot (.)” option is enabled. ☑: After entering any character string, a selection list of available identifiers and operators is displayed
“Insert with namespace”	☑: In front of the identifier, I/O Engineering automatically inserts the namespace.
“Automatically convert keywords to uppercase (Autoformat)”	☑: I/O Engineering automatically capitalizes all keywords.
“Update cross-references automatically on selection change”	☑: The cross reference list automatically shows the references of the variables/POUs/DUTs that you are currently selecting or on which the cursor is positioned.
“Underline errors in the editor”	☑: Incorrect or unknown program code is underlined.
“Highlight symbols”	☑: All the places of use of a symbol on which the cursor is positioned are highlighted in the editor. Consequently, cross-references within the editor are detected quickly.
“Maximum degree of parallelism”	Selection list for the number of parallel threads that can be used for precompile processing. I/O Engineering determines the displayed number of threads from the number of CPU cores of the processor. This preset number should only be changed in exceptional cases.

Dialog 'Options' – 'International Settings'

Symbol:

Function: This dialog is for the setting of the language in the user interface and in the help.

Call: Menu bar: *“Tools → Options”*, category “International Settings”.

‘Options’ dialog - ‘Load and save'

Symbol:

Function: The dialog contains settings for the behavior of I/O Engineering when loading and saving a project.

Call: Menu *“Tools → Options”*, “Load and save ” category

“Create backup copy”	☑: I/O Engineering saves the project each time it is saved, in addition to the file `<project name>.project`, also as a `<project name>.backup` file. You can rename the backup file and open it in the programming system.
“Save automatically every ...minutes”	☑: I/O Engineering automatically saves the project at the specified time interval to a file `<project name>.autosave`, which you can reload after a non-regular closing of the programming system. When closing or saving the project regularly, I/O Engineering deletes the `.autosave` file. In case of non-regular termination, I/O Engineering retains the `.autosave file`. When opening a project for which there is an associated autosave file, the “Auto Save backup” dialog opens. In the dialog, choose whether the `.autosave` file or the last version of the project saved by the user is opened.
“Save before compilation”	I/O Engineering automatically saves the project before each compilation run.
“Generate project recovery information”	Prerequisite: In the project settings in the “Security” category, the “No protection” option is enabled. That means: The project is not protected from unauthorized access and data manipulation and there is no integrity check when loading the project. ☑: If a project crashes during editing, the next time you open the project, a prompt is displayed asking if you want to restore the unsaved data and create a new project file. If you click on“Yes”, another dialog is opened. In this dialog, select whether you want to open the restored project or whether you want to open the project comparison. This project comparison shows the differences between the last saved project and the restored project. NOTE: Project Restore records every change on the hard disk when the change is executed. If a power or hard disk error occurs on the hard disk during this process, the last change may be lost.
“Advanced settings”	The “Advanced settings” dialog opens.
“Upon startup”	Selection list for the start screen of I/O Engineering: • “Show start page”: The home page of I/O Engineering is displayed • “Open the last used project” • “Display the "Open project" dialog” • “Display the "New project" dialog” • “Show an empty environment”
“News page”	URL, which can be called with the command *“Help → I/O Engineering Homepage”* is opened. By default, `http://www.codesys.com/startpage` is entered.

Table 36: “Advanced settings” dialog

“Project compression”	
“Step”	Prerequisite: In the project settings in the “Security” category, the “No protection” option is enabled. That means: The project is not protected from unauthorized access and data manipulation and there is no integrity check when loading the project. Selection list for the compression level that is applied when saving the project: • “Lowest compression - highest speed (recommended)” • “Medium compression - medium speed” • “Highest compression - smallest speed”
“Loading behavior” (only relevant in the context of PLC programming)	
☑	Loading libraries and translation information is done in the background while you are already editing the project.

Dialog 'Options' - 'PLCopenXML'

Symbol:

Function:This dialog contains settings for the behavior of I/O Engineering when exporting or importing PLCopenXM.

Call: Main menu *“Tools → Options”*, category “PLCopenXML”

Table 37: “PLCopenXML Export Settings”

“Additionally export declarations as plain text”	By default, I/O Engineering splits the declaration parts in accordance with the PLCopenXML scheme into individual variables and thus loses the formatting and some comment information. ☑: Formatting and comments are retained. I/O Engineering additionally writes the plain text of the exported declaration part into the PLCopenXML file and thus extends the PLCopenXML scheme.
“Export Folder Structure”	☑: I/O Engineering also exports the folders if they contain one of the selected objects. That is a I/O Engineering-specific extension to the PLCopenXML scheme.

Table 38: “PLCopenXML Import Settings”

“Import folder structure”	☑: If the import file contains information about the folder structure of the objects, I/O Engineering also imports this structure. ☐: I/O Engineering imports objects without structure.

See also

- ➦Chapter 9.2 Exporting and Importing Projects on page 24
- ➦Chapter Command 'Export PLCopenXML' on page 149
- ➦Chapter Command 'Import PLCopenXML' on page 149

'Options' dialog - 'Proxy settings'

Symbol:

Function: The dialog is used to save authentication data for the proxy server currently used for accesses from I/O Engineering to the internet.

Call: Menu *“Tools → Options”*, “Proxy settings”Category

Prerequisite: The internet in the network can be accessed via a proxy server

“Enter proxy acess data”	A double-click opens a prompt for user name and password for the proxy server. I/O Engineering uses the access data when connecting to the download server for device descriptions, when connecting to CODESYS Store and for the command *“View → Start page”* when displaying the start page. Prerequisite: The button is available if your workstation computer or network accesses the internet through a proxy server.

Also refer to

- ➦Chapter Command 'Start Page' on page 137

'Options' dialog - 'Refactoring'

Symbol:

Function: The dialog is used to specify the operations in the project for which refactoring is automatically suggested. The refactoring functionality helps you in your improvement endeavors.

Call: Menu *“Tools → Options”*, category: “Refactoring”

'Suggest refactoring for the following operations'

“Autodeclaration”	When changing the name of a variable in a declaration by calling autodeclaration[(][Shift+F2]) , the enabled option “Apply changes using refactoring” is displayed. Then the “Refactoring” dialog opens and you can change the variable throughout the project. • “When adding or removing variables, or for changing the range of validity:” ☑: You delete the names in the “Declare variable” dialog and click on“OK” to close the dialog. Then the “Refactoring” dialog opens for removing the variable throughout the project. • “On renaming variables” ☑: You specify the names in the “Declare variable” dialog and click on“OK” to close the dialog. Then the “Refactoring” dialog opens for renaming the variable throughout the project. See: Chapter "Refactoring", "Changing variable declaration and applying refactoring automatically".
“Unit Conversion Editor”	“ When renaming unit conversions”: • ☑: When changing the name of a conversion in the unit conversion editor,he user is prompted if I/O Engineering should perform "Automatic Refactoring" for renaming.
“Mapping editor”	“On renaming variables” • ☑: When changing a variable name in the device editor (“I/O Mapping” tab), the user is prompted if I/O Engineering should perform "Automatic Refactoring" for renaming.
“Navigator”	“When renaming objects”: • ☑: When changing the name of an object in the device tree or in the POU view, the user is prompted if I/O Engineering should perform "Automatic Refactoring" for renaming.

“Tabular declaration editor”	“On renaming variables” • ☑: When changing the name of a variable in the tabular declaration editor, you are prompted whether I/O Engineering should perform "Automatic Refactoring" when renaming.
“UML class diagram”	Options for supporting refactoring, i.e. project-wide IEC code adaptation, for changes made in the class diagram editor: • “When adding or removing variables”: ☑: When adding or removing variables in the `VAR_INPUT`, `VAR_OUTPUT`, and `VAR_INOUT` sections in the class diagram, refactoring is supported. • “When renaming a block”: ☑: When changing a module name in the class diagram, refactoring is supported. • “When renaming variables or properties”: ☑: When removing a variable or a property in the class diagram, refactoring is supported. Note: If the “Skip refactoring preview” option is enabled in the UML options, refactoring can be performed on all affected locations in the project without first being displayed in the “Refactoring” dialog, depending on the case. See the help page “Dialog 'Options' - 'UML'" in the CODESYS UML help.

'Options' dialog - 'Text editor'

Symbol:

Function: The dialog contains settings for displaying and working in a text editor.

Call: Menu *“Tools → Options”*, “Text editor” category

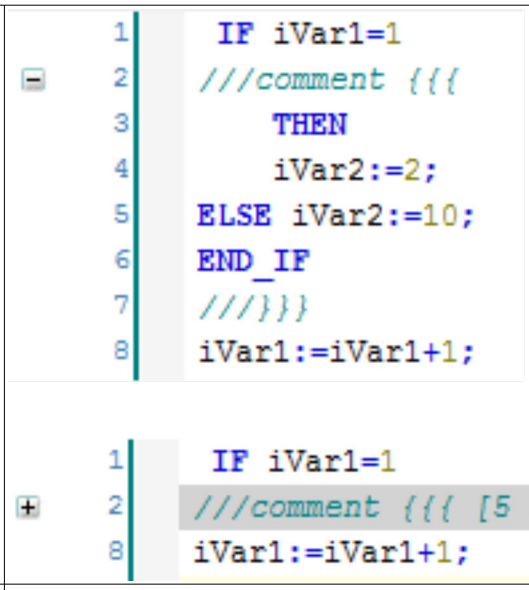
'Theme' tab

Set the desired "Theme" for the interface design of the ST editor.

“Theme”	Color scheme for the text editor. The selected "Theme" is displayed in the “Preview” window. The available color schemes are stored in the installation directory in the `Themes` folder.

'Edit' tab

“Number of undo steps”	Maximum number of editing steps to which you can apply the command *“Edit → Undo”* command.
“Folding”	Defines the structuring of the code by means of indentation. When you select an indentation, you can expand or collapse the indentation section with the help of a plus or a minus sign respectively before the first line of the respective section. • “Indent”: I/O Engineering collects all lines that are indented in comparison with the preceding line in an indentation unit. • “Explicit”: They explicitly identify the code section with comments that should be combined into one indentation unit: The section has to be preceded by a comment containing 3 opening curly braces "{{", the section has to be followed by a comment containing 3 closing curly braces "}}". The comments can contain additional text. Example:

	
“Word wrap”	• “Soft”: The line break occurs at the edge of the editor window if 0 is entered in the “break point”. • “Hard”: The line break occurs after the number of characters specified in the “break point”.
“Tab width”	Number of characters
“Keep tabs”	☑: The empty space that you inserted with the [Tab] key is not resolved into spaces afterwards by I/O Engineering.
“Indent width”	If you have selected the “AutoIndent” option “Smart” or “Smart with code completion”, then I/O Engineering inserts the number of spaces at the beginning of the line.
“AutoIndent”	• “None” • “Block”: A new line automatically takes over the indentation of the previous line. • “Intelligent”: Lines following a line containing a keyword (for example `VAR`) automatically indent by the specified `indent size`. • “Intelligent with code completion”: Indentation as for the “Intelligent ” option , in addition I/O Engineering inserts the terminating keyword (for example `END_VAR`).

Tab 'Text Area'

“Highlight current line”	☑: The line in which the cursor is positioned in is highlighted.
“Matching brackets”	☑: If the cursor is positioned before or after a bracket within a code line, the associated closing or opening bracket is marked by a frame.
“End of line selection”	☑: The end of each editor line is marked by a small slash after the last character (including spaces) of the line.
“Line break”	☑: If a soft or hard line break is enabled, the defined line break location is indicated by a vertical line.
“Font”	Clicking the field opens the standard dialog for the configuration of the font.

Tab 'Margin'

Settings for the left margin of the text editor window, which is separated from the input area by a vertical line:	
“Line numbering”	☑: The declaration and implementation part of the editor have line numbering on the left, starting with 1

“Highlight current line”	☑: The line number of the line where the cursor is located is highlighted.
“Show bracket area”	☑: Bracketing encompasses the lines between the keywords that open and close a construct, for example `IF` and `END_IF`. If this option is enabled and the cursor is located before, after or in one of the keywords of a construct, the bracketed area is indicated by a square bracket in the border.
“Mouse actions”	You can assign one of the following actions to each of the specified mouse actions or mouse button combinations. I/O Engineering performs the selected action when you perform the mouse action on the plus or minus sign in front of the header of a bracketed area: • “None”: The mouse action does not trigger any action. • “Select fold”: I/O Engineering selects all lines of the bracketed area. • “Toggle fold”: I/O Engineering opens or closes the bracketed area or, if there are nested brackets, the first level of the bracketed area. • “Toggle fold fully”: I/O Engineering opens or closes all levels of a nested bracketed area.

Tab 'Monitoring'

Settings for the display of the monitoring fields	
“Enable inline monitoring”	☑: Display of the monitoring fields behind the variables in online mode
“Number of displayed digits”	Number of decimal places in the monitoring field
“String length”	Maximum length of string variable values in the monitoring field

21.4.10 “Customize” dialog

Dialog 'Customize' - General information

The options described below can be called in both ctrlX I/O Engineering as well as in ctrlX PLC Engineering.

However, some of the options are not relevant for working inctrlX I/O Engineering as they are connected to PLC programming in ctrlX PLC Engineering.

➦ Dialog 'Customize' - 'Menu'
➦ Dialog 'Customize' - 'Command Icons'
➦ Dialog 'Customize' - 'Toolbars'
➦ Dialog 'Customize' - 'Keyboard'

Dialog 'Customize' - 'Menu'

Function: With this dialog, you define the structure and contents of the user interface.

Call: Main menu *“Tools → Customize”* (“Menu”).

When you click “OK” to close the dialog, the changes are visible in the menu bar of the I/O Engineering user interface.

Table 39: “Menu”

Display of currently defined menus, submenus, and included commands. In I/O Engineering, a menu or submenu caption is identified by the caption symbol ( ). The layout from top to bottom corresponds to the layout displayed later in the I/O Engineering menu.	
“Add Command”	Enabled when a command is selected. Adds a command above the selected command. Opens the “Add Command” dialog. Use the “Add Command” dialog for selecting one or more commands. Left part: List of categories. Right part: List of commands in the selected category.
“Add Separator”	Adds a separator above the selected command.
“Add Popup Menu”	Adds a popup menu above the selected menu, submenu, or command. Opens the “Add Popup Menu” dialog.
“Edit Popup Menu”	Opens the “Edit Popup Menu” dialog.
“Reset”	Resets the default settings of the entire menu.
“Load”	Loads the settings from a stored file (<file name>.opt.menu).

Table 40: “Add Popup Menu”

In I/O Engineering, a new menu is shown in the menu bar only when the menu contains at least one command.	
“Default text”	Select this check box when localization is available.
“Localized Texts”	List: Languages and localized texts.
“Add Language”	Opens a drop-down list of available languages. In I/O Engineering, the selected language is displayed in the area “Localized Texts”. Use the “Text” column for typing the localized texts.

See also

- ➦Chapter 7.2.1 Customizing Menus on page 16
- ➦Chapter Dialog 'Customize' - 'Toolbars' on page 189

Dialog 'Customize' - 'Command Icons'

Function: This dialog defines the icons of the menu commands.

Call: Menu bar: *“Tools → Customize”* (“Command Icons”).

Table 41: “Command icon”

“Assign”	Opens a dialog for selecting the new icon (*.ico).
“Remove”	Removes the user-defined icon. The default icon is active again.
“Reset”	Resets all default settings of the command icons.
“Load”	Loads the settings from a stored file (<file name>.opt.keyb).
“Save”	Saves the current settings to a file (<file name>.opt.keyb).

See also

- ➦Chapter 7.2.3 Customize Command Icon on page 19

Dialog 'Customize' - 'Toolbars'

Function: Use this dialog for generating new toolbars or customizing existing toolbars.

Call: Main menu *“Tools → Customize”* (“Toolbars”).

When you click “OK” to close the dialog, the changes are visible in the menu bar of the I/O Engineering user interface.

Table 42: “Toolbars”

Display of currently defined toolbars. In I/O Engineering, the associated commands are listed below each toolbar in the order they will appear in the toolbar. Double-clicking a toolbar in the list switches to editing mode.	
“Add Toolbar”	Enabled when a toolbar is selected. In I/O Engineering, this adds a toolbar above the selected toolbar and places the cursor in the name field of the new toolbar.
“Add Command”	Enabled when you select a command or blank command entry below a toolbar. Adds a command above the selected command. Opens the “Add Command” dialog. Use the “Add Command” dialog to select one or more commands. Left part: List of categories. Right part: List of commands in the selected category.
“Add Separator”	Adds a separator above the selected command.
“Hide”	Hide the selected toolbar from the user interface.
“Show”	Shows the selected hidden toolbar in the I/O Engineering user interface.
“Reset”	Resets the default settings of the toolbars.
“Load”	Loads the settings from a stored file (<file name>.opt.tbar).

See also

- ➦Chapter 7.2.2 Customizing Toolbars on page 18
- ➦Chapter Dialog 'Customize' - 'Menu' on page 188

Dialog 'Customize' - 'Keyboard'

Function: This dialog is used for defining keyboard shortcuts (quick access keys or keyboard combinations) for commands.

Call: Main menu *“Tools → Customize”* (“Keyboard”).

Table 43: “Keyboard”

“Shortcuts for selected command”	Keyboard shortcuts for the selected command The drop-down list can include more than one keyboard shortcut for the command.
“Press shortcut keys”	Input field for the keyboard shortcut of the selected field. Permitted combinations include [Ctrl], [Alt], [Shift], and other keys. You clicking “Assign” to assign a recorded keyboard shortcut to a selected command.
“Shortcut keys currently used by”	Command assigned to the currently defined keyboard shortcut
“Reset”	Resets the default settings of the keyboard shortcuts.
“Load”	Loads the settings from a stored file (<file name>.opt.keyb).

See also

- ➦Chapter 7.2.4 Customizing Keyboard Shortcuts on page 19

22 Software add-ons/documentation

In ctrlX I/O Engineering and ctrlX PLC Engineering there is the option to integrate third-party software add-ons into the user interface, see:

➦Command 'Add-on Installer'

Documentation for third-party add-ons have to be obtained from the respective software manufacturer.

Documentation for CODESYS software products

The CODESYS online help can be found at: ➦ https://www.helpme-codesys.com/

Software/link to manufacturer documentation	Description
➦CODESYS Engineering	Development environment: • CODESYS Development System • CODESYS Tools • CODESYS Installer

23 Related documentation

23.1 Overview

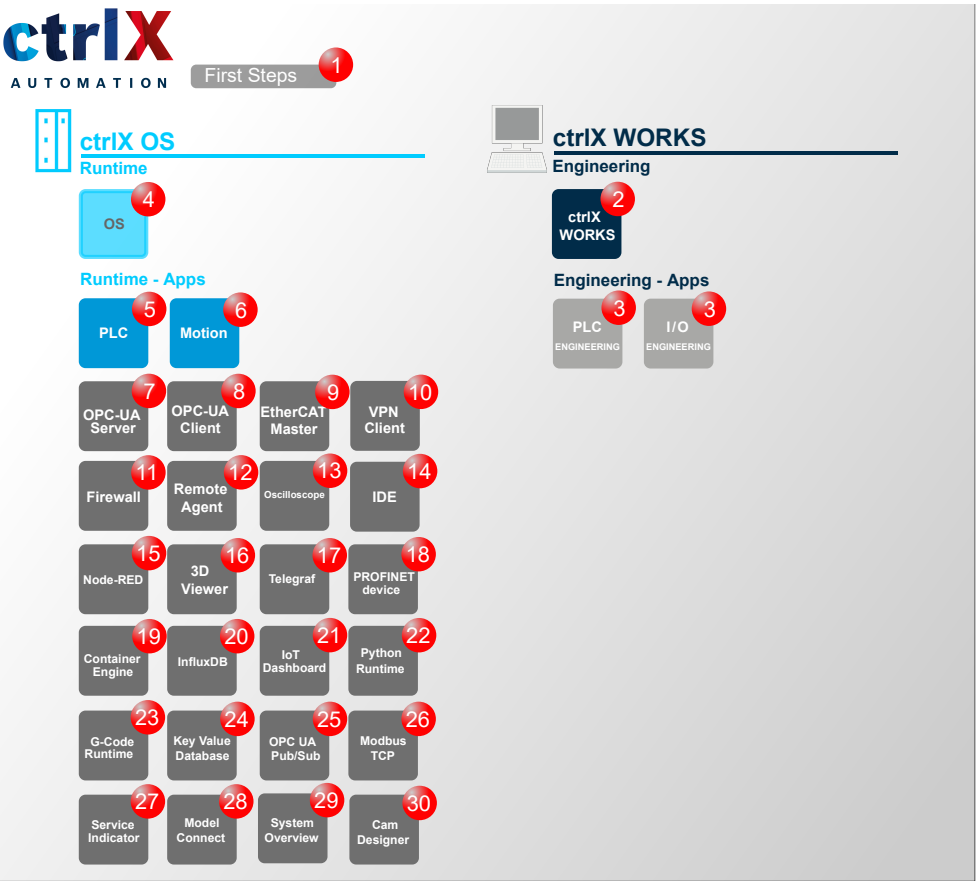


Fig. 8: Overview on further documentations

23.2 ctrlX AUTOMATION

No.	Documentation
1	ctrlX WORKS First Steps 02VRS Quick Start Guide ➦Web documentation link Ordering information: • DOK-XWORKS-F*STEP**V02-QURS-EN-P • R911421574

23.3 ctrlX WORKS

No.	Documentation
2	**ctrlX WORKS Basic System 02VRS** Application Manual ➥ Web documentation link Ordering information: ● DOK-XWORKS-WRK***V02**-APRS-EN-P ● R911421576
3	**ctrlX PLC Engineering - PLC Programming System 02VRS** Application Manual ➥ Web documentation link Ordering information: ● DOK-XPLC**-ENG*****V02-APRS-EN-P ● R911421578
3	**ctrlX PLC Engineering - PLC Libraries 02VRS** Reference Book ➥ Web documentation link Ordering information: ● DOK-XPLC**-LIB***V02**-RERS-EN-P ● R911421580

23.4 ctrlX OS

No.	Documentation
4	**ctrlX OS - Operating System for ctrlX CORE Control Devices 02VRS** Application Manual ➥ Web documentation link Ordering information: ● DOK-XCORE*-XCR***V02**-APRS-EN-P ● R911421590
	ctrlX OS - Data Layer nodes 02VRS Reference Book ➥ Web documentation link Ordering information: ● DOK-XCORE*-DL****V02**-RERS-EN-P ● R911421592
	ctrlX OS - Diagnostics 02VRS Reference Book ➥ Web documentation link Ordering information: ● DOK-XCORE*-DIAG**V02**-RERS-EN-P ● R911421594

23.5 ctrlX OS apps

No.	Documentation
5	**PLC App - PLC Runtime Environment for ctrlX OS 02VRS** Application Manual ➦ Web documentation link Ordering information: • DOK-XCORE*-PLC***V02**-APRS-EN-P • R911421584
6	**Motion App - Motion Runtime Environment for ctrlX CORE 02VRS** Application Manual ➦ Web documentation link Ordering information: • DOK-XCORE*-MOT***V02**-APRS-EN-P • R911421610
7	**OPC UA Server App - OPC UA Server for ctrlX OS 02VRS** Application Manual ➦ Web documentation link Ordering information: • DOK-XCORE*-UAS***V02**-APRS-EN-P • R911421598
8	**OPC UA Client App - OPC UA Client for ctrlX OS 02VRS** Application Manual ➦ Web documentation link Ordering information: • DOK-XCORE*-UAC***V02**-APRS-EN-P • R911421600
9	**EtherCAT Master App - EtherCAT master for ctrlX OS 02VRS** Application Manual ➦ Web documentation link Ordering information: • DOK-XCORE*-ECM***V02**-APRS-EN-P • R911421604
10	**VPN Client App - Remote Support Software for ctrlX OS 02VRS** Application Manual ➦ Web documentation link Ordering information: • DOK-XCORE*-VPN***V02**-APRS-EN-P • R911421596
11	**Firewall App - Security Functions for ctrlX OS 02VRS** Application Manual ➦ Web documentation link Ordering information: • DOK-XCORE*-FRW***V02**-APRS-EN-P • R911421606

No.	Documentation
12	**Remote Agent App - ctrlX Device Portal Connection for ctrlX OS Devices 02VRS** Application Manual ➥ Web documentation link Ordering information: • DOK-XCORE*-RMA***V02**-APRS-EN-P • R911421608
13	**Oscilloscope App - Oscilloscope Function for ctrlX OS Devices 02VRS** Application Manual ➥ Web documentation link Ordering information: • DOK-XCORE*-OSC***V02**-APRS-EN-P • R911421587
14	**IDE App - Integrated Development Environment 02VRS** Application Manual ➥ Web documentation link Ordering information: • DOK-XCORE*-IEN***V02**-APRS-EN-P • R911421612
15	**Node RED App - Graphic Programming for ctrlX OS 02VRS** Application Manual ➥ Web documentation link Ordering information: • DOK-XCORE*-NOENRED*V02-APRS-EN-P • R911421582
16	**3D Viewer App - Browser-based 3D Kinematic Simulation for ctrlX OS 02VRS** Application Manual ➥ Web documentation link Ordering information: • DOK-XCORE*-3DV***V02**-APRS-EN-P • R911421615
17	**Telegraf App - Server Agent for Collecting Data in the Data Layer 02VRS** Application Manual ➥ Web documentation link Ordering information: • DOK-XCORE*-TSA***V02**-APRS-EN-P • R911421623
18	**PROFINET Device App - PROFINET Device for ctrlX OS 02VRS** Application Manual ➥ Web documentation link Ordering information: • DOK-XCORE*-PROFINETV02-APRS-EN-P • R911421617

No.	Documentation
19	**Container Engine App - Using Docker® Images on the ctrlX OS 02VRS** Application Manual ➦ Web documentation link Ordering information: • DOK-XCORE*-DOE***V02**-APRS-EN-P • R911421619
20	**InfluxDB App - Influx Database Connection for ctrlX OS 02VRS** Application Manual ➦ Web documentation link Ordering information: • DOK-XCORE*-IDB***V02**-APRS-EN-P • R911421625
21	**IoT Dashboard App - Data Visualization in Dynamic, Interactive Dashboards 02VRS** Application Manual ➦ Web documentation link Ordering information: • DOK-XCORE*-GDB***V02**-APRS-EN-P • R911421633
22	**Python Runtime App - Python Runtime App Environment for ctrlX CORE 02VRS** Application Manual ➦ Web documentation link Ordering information: • DOK-XCORE*-PYR***V02**-APRS-EN-P • R911421629
23	**G-Code Runtime App - G-Code Interpreter for ctrlX OS 02VRS** Application Manual ➦ Web documentation link Ordering information: • DOK-XCORE*-GCO***V02**-APRS-EN-P • R911421631
24	**Key Value Database App - Data Layer Management 02VRS** Application Manual ➦ Web documentation link Ordering information: • DOK-XCORE*-KVD*****V02-APRS-EN-P • R911418735
25	**OPC UA Pub/Sub App - OPC UA Pub/Sub for ctrlX OS 02VRS** Application Manual ➦ Web documentation link Ordering information: • DOK-XCORE*-UAP***V02**-APRS-EN-P • R911421602

No.	Documentation
26	**Modbus TCP App - Modbus TCP Communication over ctrlX OS 02VRS** Application Manual ➥ Web documentation link Ordering information: • DOK-XCORE*-MOD*TCP*V02-APRS-EN-P • R911421621
27	**Service Indicator App Service Indicator for ctrlX OS 02VRS** Application Manual ➥ Web documentation link Ordering information: • DOK-XCORE*-SIN*****V02-APRS-EN-P • R911421627
28	**Model Connect App Target for Model-based Development and Simulation for ctrlX OS 02VRS** Application Manual ➥ Web documentation link Ordering information: • DOK-XCORE*-MOC***V02**-APRS-EN-P • R911421631
29	**System Overview App - System Topology and System Information 02VRS** Application Manual ➥ Web documentation link Ordering information: • DOK-XCORE*-SOV***V02**-APRS-EN-P • R911424409
30	**Cam Designer - Configuring ctrlX Motion Cams 02VRS** Application Manual ➥ Web documentation link Ordering information: • DOK-XWORKS-CAM***V02**-APRS-EN-P • R911424390

24 Service and support

Our worldwide service network provides an optimized and efficient support. Our experts provide you with advice and assistance. You can contact us **24/7**.

Service Germany

Our technology-oriented Competence Center in Lohr, Germany, is responsible for all your service-related queries for electric drive and controls.

Contact the **Service Hotline** and **Service Helpdesk** under:

Phone:	**+49 9352 40 5060**
Fax:	**+49 9352 18 4941**
Email:	➦ service.svc@boschrexroth.de
Internet:	➦ http://www.boschrexroth.com

Additional information on service, repair (e.g. delivery addresses) and training can be found on our internet sites.

Service worldwide

Outside Germany, please contact your local service office first. For hotline numbers, refer to the sales office addresses on the internet.

Preparing information

To be able to help you more quickly and efficiently, please have the following information ready:

- Detailed description of malfunction and circumstances
- Type plate specifications of the affected products, in particular type codes and serial numbers
- Your contact data (phone and fax number as well as your e-mail address)

25 Index

R

S

rexroth
A Bosch Company

Bosch Rexroth AG
Bgm.-Dr.-Nebel-Str. 2
97816 Lohr a.Main
Germany
Tel. +49 9352 18 0
Fax +49 9352 18 8400
www.boschrexroth.com/electrics

R911421586