Operating instruction manual

netX Studio Engineering Tool

IOT Configuration

V1.400

Hilscher Gesellschaft für Systemautomation mbH
www.hilscher.com

DOC180404OI01EN | Revision 1 | English | 2018-11 | Released | Public

Table of contents

1 General information

1.1 About this manual

This guide describes the IOT configuration by answering the following questions:

- Introduction to IOT configuration with netPROXY: Add IOT functions OPC UA or MQTT for products with netX firmware (or WebServer can be added),
- installation and how to use ,
- Overview and step sequence for IOT configuration,
- Procedure for the IOT configuration based on a use case,

The procedure for IOT configuration recommended in this manual depends on the order of the folders in the user interface of , which is based on the principles of netPROXY.

On other questions about further manuals are available (see section *references* [page 109]), for example, about the Web Builder [5].

1.1.1 List of revisions

Index	Date	Changes
1	18-11-29	Created

Table 1: List of revisions

1.1.2 Online help

The program contains an integrated online help.

➢ Click **View > Help** or press F1, to access to the online help

1.2 Conventions in this manual

Notes, instructions and results of operating steps are marked as follows:

Notes

Important:
<important note you must follow to avoid malfunction>

Note:
<general note>

<note where to find further information>

Instructions

1. Operation purpose
2. Operation purpose
   ➢ Instruction

Results

Intermediate result

Final result

1.3 Legal notes

Copyright

Important notes

Utmost care was/is given in the preparation of the documentation at hand consisting of a user's manual, operating manual and any other document type and accompanying texts. However, errors cannot be ruled out. Therefore, we cannot assume any guarantee or legal responsibility for erroneous information or liability of any kind. You are hereby made aware that descriptions found in the user's manual, the accompanying texts and the documentation neither represent a guarantee nor any indication on proper use as stipulated in the agreement or a promised attribute. It cannot be ruled out that the user's manual, the accompanying texts and the documentation do not completely match the described attributes, standards or any other data for the delivered product. A warranty or guarantee with respect to the correctness or accuracy of the information is not assumed.

We reserve the right to modify our products and the specifications for such as well as the corresponding documentation in the form of a user's manual, operating manual and/or any other document types and accompanying texts at any time and without notice without being required to notify of said modification. Changes shall be taken into account in future manuals and do not represent an obligation of any kind, in particular there shall be no right to have delivered documents revised. The manual delivered with the product shall apply.

Under no circumstances shall Hilscher Gesellschaft für Systemautomation mbH be liable for direct, indirect, ancillary or subsequent damage, or for any loss of income, which may arise after use of the information contained herein.

Liability disclaimer

The hardware and/or software was created and tested by Hilscher Gesellschaft für Systemautomation mbH with utmost care and is made available as is. No warranty can be assumed for the performance or flawlessness of the hardware and/or software under all application conditions and scenarios and the work results achieved by the user when using the hardware and/or software. Liability for any damage that may have occurred as a result of using the hardware and/or software or the corresponding documents shall be limited to an event involving willful intent or a grossly negligent violation of a fundamental contractual obligation. However, the right to assert damages due to a violation of a fundamental contractual obligation shall be limited to contract-typical foreseeable damage.

It is hereby expressly agreed upon in particular that any use or utilization of the hardware and/or software in connection with

- Flight control systems in aviation and aerospace;
- Nuclear fusion processes in nuclear power plants;
- Medical devices used for life support and
- Vehicle control systems used in passenger transport

shall be excluded. Use of the hardware and/or software in any of the following areas is strictly prohibited:

- For military purposes or in weaponry;
- For designing, engineering, maintaining or operating nuclear systems;
- In flight safety systems, aviation and flight telecommunications systems;
- In life-support systems;
- In systems in which any malfunction in the hardware and/or software may result in physical injuries or fatalities.

You are hereby made aware that the hardware and/or software was not created for use in hazardous environments, which require fail-safe control mechanisms. Use of the hardware and/or software in this kind of environment shall be at your own risk; any liability for damage or loss due to impermissible use shall be excluded.

Warranty

Hilscher Gesellschaft für Systemautomation mbH hereby guarantees that the software shall run without errors in accordance with the requirements listed in the specifications and that there were no defects on the date of acceptance. The warranty period shall be 12 months commencing as of the date of acceptance or purchase (with express declaration or implied, by customer's conclusive behavior, e.g. putting into operation permanently).

The warranty obligation for equipment (hardware) we produce is 36 months, calculated as of the date of delivery ex works. The aforementioned provisions shall not apply if longer warranty periods are mandatory by law pursuant to Section 438 (1.2) BGB, Section 479 (1) BGB and Section 634a (1) BGB [Bürgerliches Gesetzbuch; German Civil Code] If, despite of all due care taken, the delivered product should have a defect, which already existed at the time of the transfer of risk, it shall be at our discretion to either repair the product or to deliver a replacement product, subject to timely notification of defect.

The warranty obligation shall not apply if the notification of defect is not asserted promptly, if the purchaser or third party has tampered with the products, if the defect is the result of natural wear, was caused by unfavorable operating conditions or is due to violations against our operating regulations or against rules of good electrical engineering practice, or if our request to return the defective object is not promptly complied with.

Costs of support, maintenance, customization and product care

Please be advised that any subsequent improvement shall only be free of charge if a defect is found. Any form of technical support, maintenance and customization is not a warranty service, but instead shall be charged extra.

Additional guarantees

Although the hardware and software was developed and tested in-depth with greatest care, Hilscher Gesellschaft für Systemautomation mbH shall not assume any guarantee for the suitability thereof for any purpose that was not confirmed in writing. No guarantee can be granted whereby the hardware and software satisfies your requirements, or the use of the hardware and/or software is uninterruptable or the hardware and/or software is fault-free.

It cannot be guaranteed that patents and/or ownership privileges have not been infringed upon or violated or that the products are free from third-party influence. No additional guarantees or promises shall be made as to whether the product is market current, free from deficiency in title, or can be integrated or is usable for specific purposes, unless such guarantees or promises are required under existing law and cannot be restricted.

Confidentiality

The customer hereby expressly acknowledges that this document contains trade secrets, information protected by copyright and other patent and ownership privileges as well as any related rights of Hilscher Gesellschaft für Systemautomation mbH. The customer agrees to treat as confidential all of the information made available to customer by Hilscher Gesellschaft für

Systemautomation mbH and rights, which were disclosed by Hilscher Gesellschaft für Systemautomation mbH and that were made accessible as well as the terms and conditions of this agreement itself.

The parties hereby agree to one another that the information that each party receives from the other party respectively is and shall remain the intellectual property of said other party, unless provided for otherwise in a contractual agreement.

The customer must not allow any third party to become knowledgeable of this expertise and shall only provide knowledge thereof to authorized users as appropriate and necessary. Companies associated with the customer shall not be deemed third parties. The customer must obligate authorized users to confidentiality. The customer should only use the confidential information in connection with the performances specified in this agreement.

The customer must not use this confidential information to his own advantage or for his own purposes or rather to the advantage or for the purpose of a third party, nor must it be used for commercial purposes and this confidential information must only be used to the extent provided for in this agreement or otherwise to the extent as expressly authorized by the disclosing party in written form. The customer has the right, subject to the obligation to confidentiality, to disclose the terms and conditions of this agreement directly to his legal and financial consultants as would be required for the customer's normal business operation.

Export provisions

The delivered product (including technical data) is subject to the legal export and/or import laws as well as any associated regulations of various countries, especially such laws applicable in Germany and in the United States. The products / hardware / software must not be exported into such countries for which export is prohibited under US American export control laws and its supplementary provisions. You hereby agree to strictly follow the regulations and to yourself be responsible for observing them. You are hereby made aware that you may be required to obtain governmental approval to export, reexport or import the product.

Terms and conditions

Please read the notes about additional legal aspects on our netIOT web site under http://www.netiot.com/netiot/netiot-edge/terms-and-conditions/.

1.4 Registered trademarks

Windows® 7, Windows® 8 and Windows® 10 are registered trademarks of the Microsoft Corporation.

All other brands mentioned are property of their relevant rights holder. Any company names and product designations provided in this document may be brands (company names or trademarks) of the corresponding owner and may be protected under trademark or patent law.

2 IOT functions OPC UA, MQTT

2.1 IOT functions with netPROXY

Real-Time Ethernet products for cyclic I/O data communication or for acyclic communication work on the basis of a loadable firmware "Real-Time Ethernet". The Real-Time Ethernet product already contains a developed application that uses this cyclic or acyclic communication.

If this Real-Time Ethernet product is also to support IOT communication with the OPC UA or MQTT protocols, this requires the use of an extended loadable firmware. In addition, an integrated WebServer can be used for the product.

The IOT functions OPC UA or MQTT can be configured or WebServer can be added with the help of . This extension is based on netPROXY. The necessary configuration steps are described in this document. The already developed application can be extended and use data communication via an IOT protocol.

netPROXY object model

netPROXY is an object-oriented data model and provides a protocol-independent application program interface (API) for accessing the objects.

The netPROXY object model allows to map device properties to objects. Each object and thus the device properties defined in the object are identified by an ID. For each object, the user can take settings on how to access to the object and how to process the object data.

On the top level the netPROXY object model includes the object ID, followed by the instance ID and the element ID. Together these three levels characterize an object. The instance ID specifies the occurrancy frequency of each object. The element ID contains the actual value of the device property defined in the object. Several objects are combined to build a group by a group ID.

Application example for IOT configuration

For example, an operating hours counter is assigned to an object via an object ID. The instance ID indicates how often an operating hours counter occurs: operating hours counter 1, operating hours counter 2, etc. The number of operating hours is stored in an element as "Status", or settings could be stored in an element as "Parameters".

Specification for the IOT configuration

The following use cases are conceivable:

- Configuring OPC UA
- Configuring MQTT
- Configuring OPC UA and/or MQTT
- If necessary, additionally configure WebServer

Object properties and functions

In the net PROXY object model every object has properties and functions. A property is characterized by a value, a parameter and a communication attribute.

Value
Parameter
Communication attribute (communication path)

Table 2: Identification of object properties.

An object can include the functions Create, Read, Write, and Delete.

Create
Read
Write
Delete

Table 3: Object functions

The objects properties and functions are given to the object by help of the .

2.2 communication channels

The following graphic gives an overview of the communication between the application and the network. In addition to cyclic and acyclic communication and Ethernet communication to the real-time Ethernet master, this includes IOT communication with OPC UA or MQTT from the object to the application or to the network. In addition, an integrated WebServer can be used.

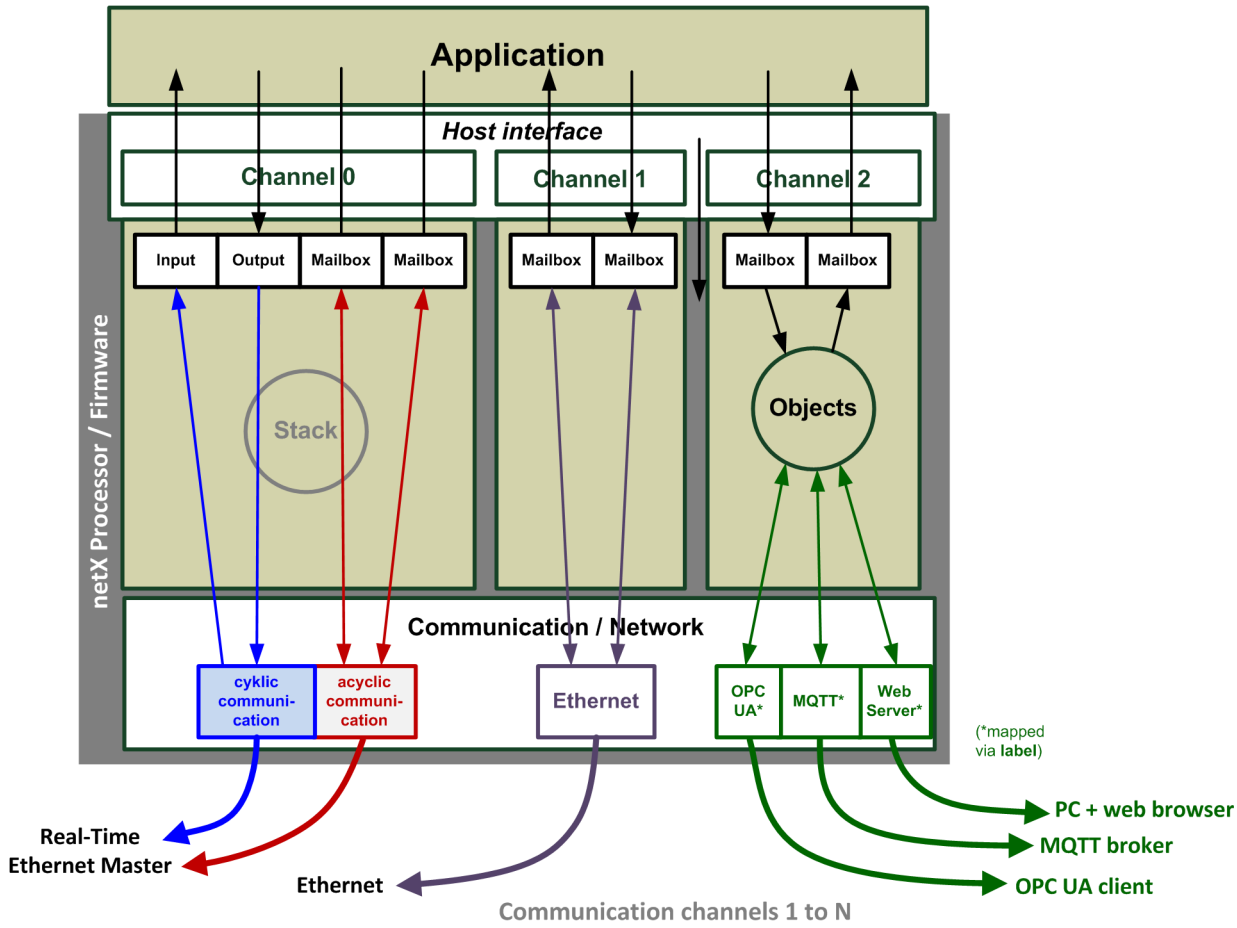


Figure 1: netX Studio Engineering Tool IOT - communication channels

The data transfer between application and communication / network is performed in netX by the device firmware with IOT functions (shown in dark gray) according to the valid configuration and the settings.

In IOT communication, data flows from the object to the network or vice versa via:

- OPC UA*
- MQTT*
- WebServer**

**IOT function, **additional WebServer function*

The objects with the communication attribute **None** can be assigned to OPC UA, MQTT or WebServer via labels. Further information on how to configure access to objects in the object definition can be found in the section *Access to object data* [page 26].

Component	Description
Application program	Cyclic communication: • Input data of the application: The application reads cyclic data from the network. • Output data of the application: The application writes cyclic output data to the network. Acyclic communication: • The application processes the acyclic communication. Ethernet communication • The application processes the Ethernet communication. Access to IOT objects: • The application reads data from the objects received from the network via OPC UA, MQTT or WebServer. • The application writes data to the objects and this data is sent to the network via OPC UA, MQTT or WebServer.
Host interface	Channel 0: Input or output for cyclic and acyclic Real-Time Ethernet communication Channel 1: Mailboxes for standard Ethernet communication Channel 3: Protocol-independent host interface for netPROXY The host interface for netPROXY can be e. g. a serial or a parallel dual-port memory (DPM) based on SPI. Components are mailboxes.
Object dictionary in the netPROXY part	List of objects structured according to the netPROXY data model Via the objects individual device properties are provided. The basic netPROXY data model is independent of the later used communication protocol or the connected hardware.
Communication / network	Hardware-dependent communication interface or connection to the network via the communication channels 1 ... N. • The cyclic, the acyclic and the Ethernet communication to the master are available as components of the netX firmware. • Communication via OPC UA, MQTT or WebServer can also be used by the application.
Object data	Objects can be read or written via OPC UA, MQTT and WebServer. OPC UA, MQTT and WebServer represent an "application" from the point of view of the objects.
Communication paths	Communication takes place from the application to the network or from the network to the application. More details can be found in the here following section and in the section *Access to object data* [▷ page 26].

Table 4: Components netX firmware with IOT

2.3 Labels

Access to the object data is configured via labels. **Labels** define access to the object data to/from the network

- OPC UA,
- MQTT and
- WebServer.

The application accesses the object data via the mailbox.

When defining the objects, it is determined on which communication paths the firmware should transfer the device properties provided by the objects.

Communication attribute "None"

Note:
For the IOT configuration of netX-Fimware products, the objects have the communication attribute **None** as default, since the input and output data are not mapped via netPROXY.

3 netX Studio Engineering Tool

3.1 About the tool

The is an Eclipse-based tool which can be used for products with netX firmware to configure IOT communication, especially for the following purposes:

- Object configuration (creating and managing objects),
- Configuration for OPC UA and MQTT,
- User rights and web content settings,
- Build process of the IOT communication (creating product files, with object configuration and all settings of the IOT protocols),

3.1.1 Scope of application

The scope of application of the is limited to slave devices based on netX, which operate with a loadable firmware. The tool is mainly used for Real-Time Ethernet systems.

3.1.2 Basic and engineering version

can be obtained as a free basic version with limited functionality. For the full version, you must purchase a license. The two versions differ as follows:

- **netX Studio (netPROXY Basic)** allows you to configure
  - *one* application variant and
  - *one* communication protocol (network).
- **netX Studio (netPROXY Engineering)** allows you to configure
  - as many application variants as required and
  - all available communication protocols (networks).

3.1.3 System requirements

- PC with standard equipment (monitor, keyboard, mouse, DVD drive, optionally USB port etc.)
- Windows® 7 (32-Bit) SP1, Windows® 7 (64-Bit) SP1, Windows® 8 (32-Bit), Windows® 8 (64-Bit) or Windows® 10 (64-Bit)

3.1.4 Requirements

Note:
To be able to connect a device via USB (USB/RS232) to , there must be no other serial connection to the device.

3.2 Installation

In the following sections you find descriptions on how to install or uninstall the program under Windows® 7, Windows® 8 or under Windows® 10 to your PC.

Requirements and notes

You need the installation file nxsbuilder-win32.win32.x86.msi.

Note:
Close all programs on your PC, before installing or uninstalling. You also need administrator privileges under Windows®.
If an earlier program version is already installed on your PC, first uninstall this version before you install the current version.
Files and folders created by the user are not deleted during uninstalling the program.

3.2.1 Installation steps

- Double click on the installation file.
- The installation is started.
- Click **Next**.

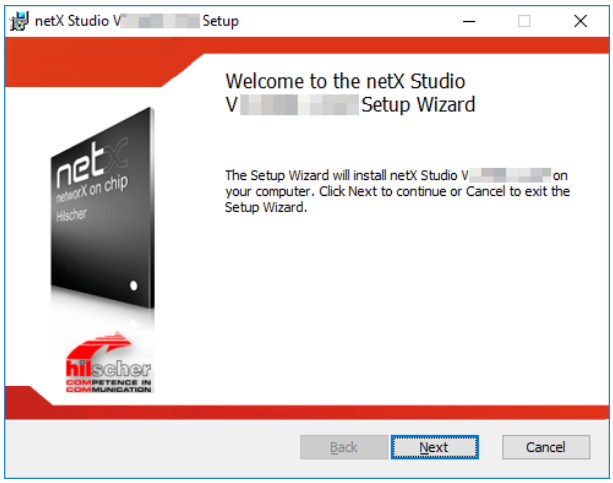


Figure 2: Installation information (exampleWindows® 10)

- Check **I accept the terms in the License Agreement**.

Figure 3: Accepting end user license agreement (example Windows® 10)

➢ Click **Next**.

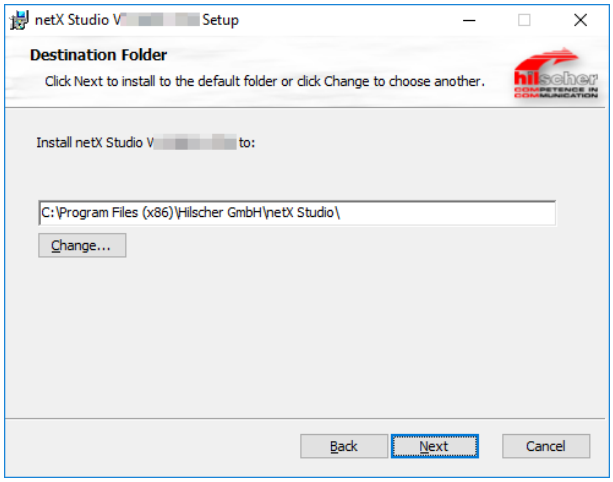


Figure 4: Installation target folder (example Windows® 10)

➢ Select a target folder.

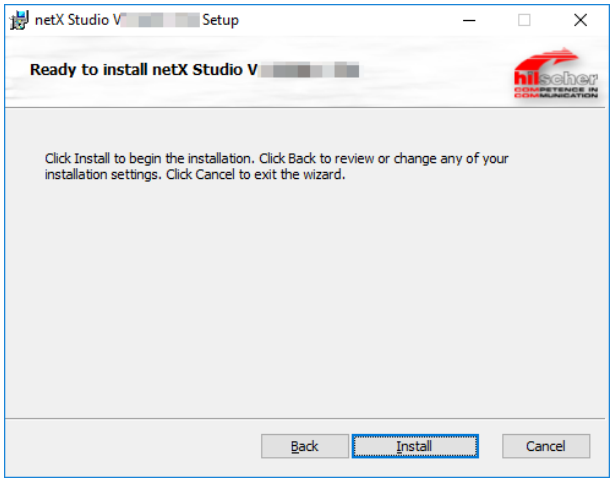


Figure 5: Performing installation (example Windows® 10)

➢ Click **Install**.

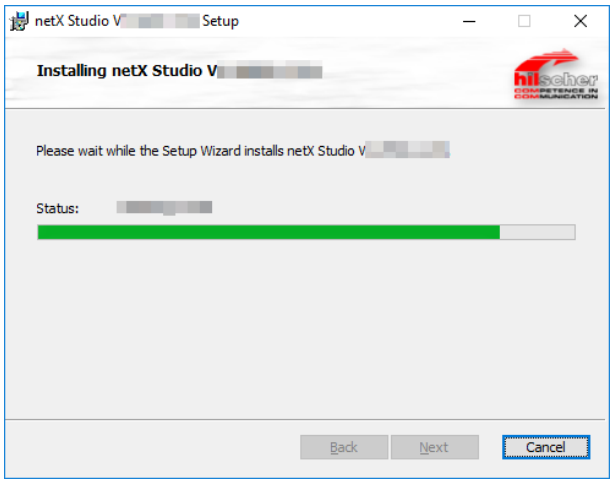


Figure 6: Installation progress (example Windows® 10)

A Windows® safety request is displayed.

➢ Click **Yes**.

The components are installed. This takes a few minutes.

➢ Click **Finish**.

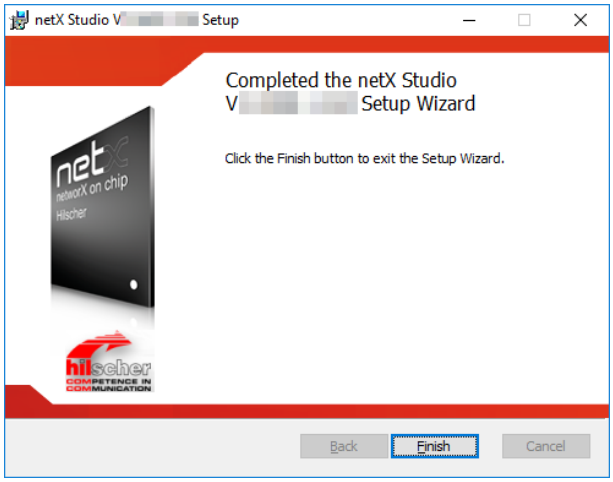


Figure 7: Installation completed (example Windows® 10)

⇨ The installation is complete.

3.2.2 How to uninstall

- Open the Windows® **Uninstall or change programs** menu window.
- Right click on the **netX Studio VX.XXXX.XXXXX** entry.
- Click **Uninstall**.
- The safety request appears asking if you want to uninstall netX Studio.
- Click **Yes**.
- The Windows Installer prepares the uninstallation.

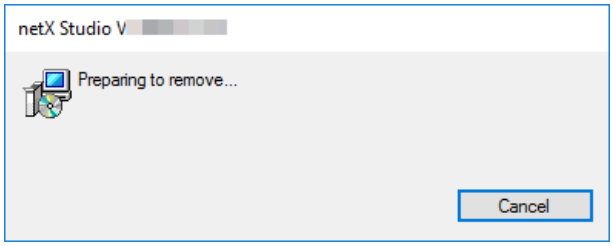


Figure 8: Preparing to uninstall (example Windows® 10)

- A Windows® safety request is displayed.
- Click **Yes**.
- The components are uninstalled. This takes a few minutes.

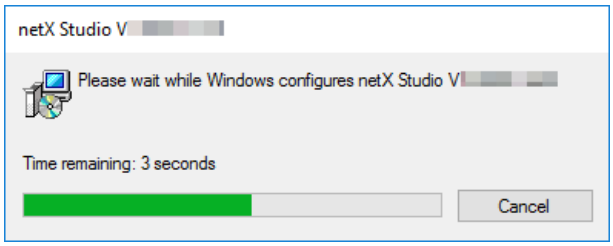


Figure 9: Progress indicator for uninstalling (example Windows® 10)

- The uninstallation is completed. The program was removed from your PC.

3.3 Description

3.3.1 Starting program and creating project

Note:
The device development with is an offline process. A connection to a device is not required.

➢ To start , click **Start > netX Studio**.

⇨ The program opens with the **Welcome** start screen.

In the basic version **netX Studio (net PROXY Basic)** in addition the message appears:

> The free "netPROXY Basic" version of netX Studio limits the device configuration to:
> - one object model (Application)
> - one communication protocol (Communication Settings).
>
> For the unlimited "netPROXY Engineering" version please contact Hilscher GmbH Sales at http://wwww.hilscher.com/en/sales/sales-contact/.

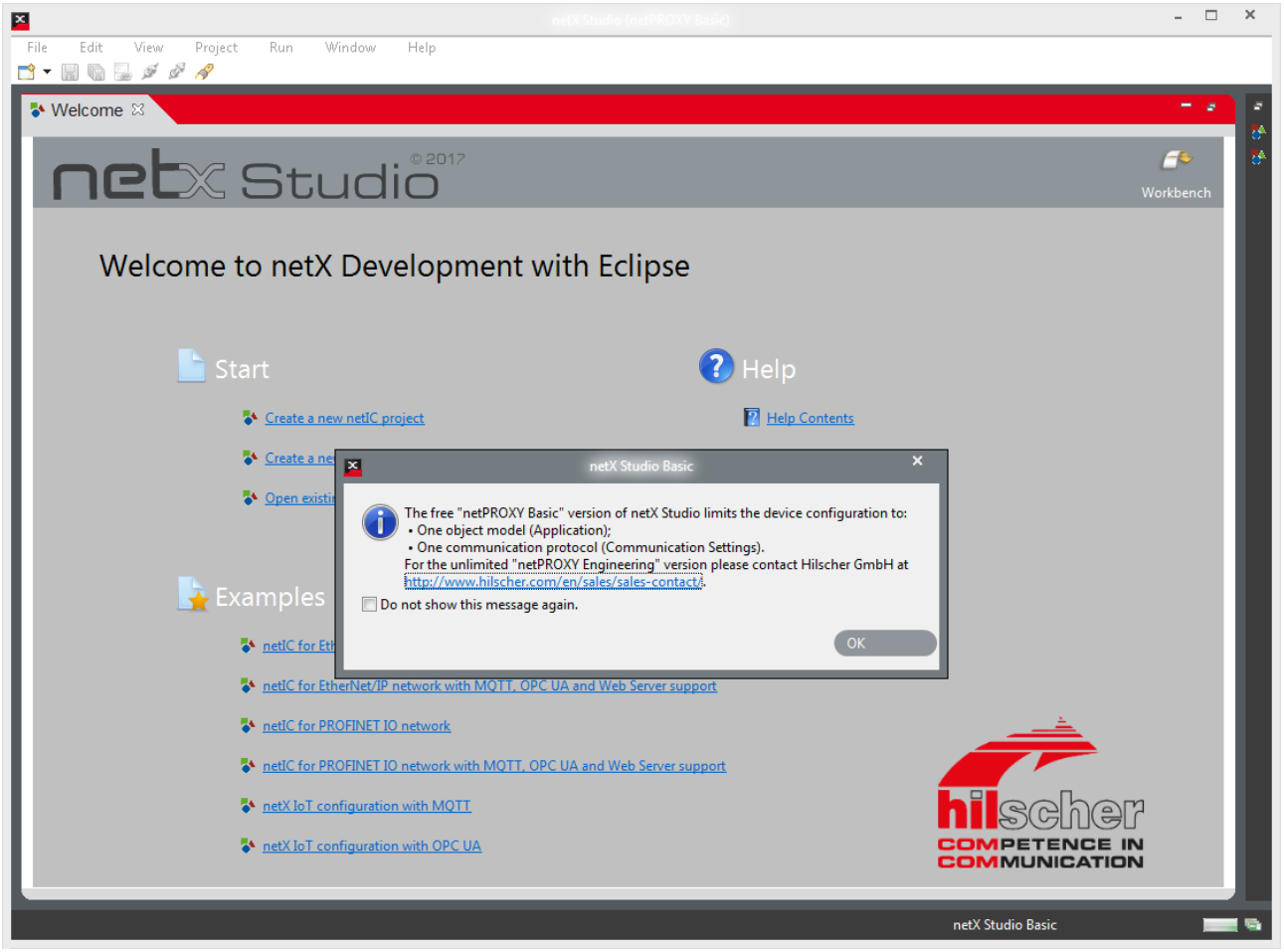


Figure 10: Home screen netX Studio (net PROXY Basic) with message box

➢ Confirm the message with **OK**.

In the full version **netX Studio (netPROXY Engineering)** the home screen appears without a message.

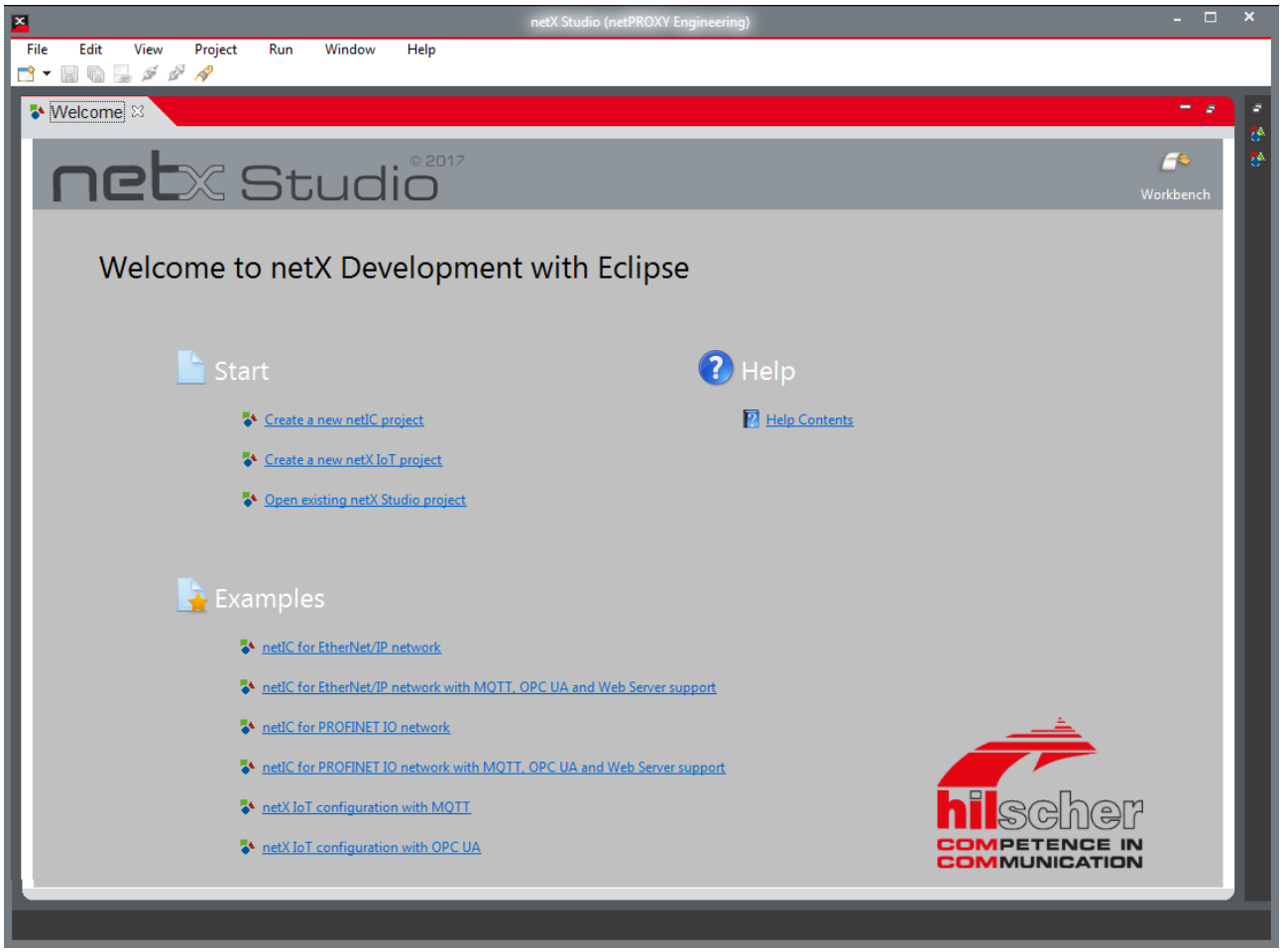


Figure 11: Home screen netX Studio (netPROXY Enigneering)

On the home screen, you can create a project. If you do not close your project before you quit the program, the project is automatically displayed at the next program start.

Creating project

➢ To create a project, click **File > New > New netX IoT project ...** (or on the **Welcome** tab **Start > Create a new netX IoT project**).

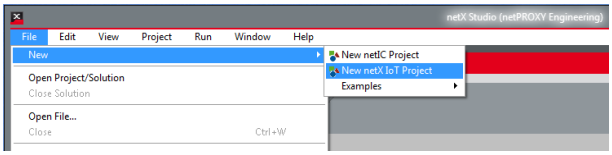


Figure 12: Creating a new project

➢ The **New netX IoT Project** window opens:

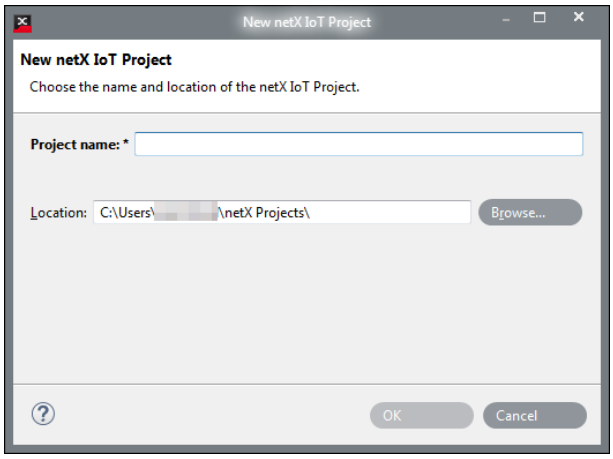


Figure 13: Enter project name and storage path

- Under **Project Name*** (mandatory field) type a project name.
- Set under **Location** the storage path for your project and click **OK**.
- ⇨ The project is created.

Important:
Note that applies all the settings and changes in your project automatically into the project file.

3.3.2 Opening and closing project

Open project

- To open a project in , click **File > Open Project / Solution**.
- Or select on the **Welcome** tab **Start > Open existing netX IoT project**.
- The **Open** window appears.
- Select the file **.project** (type: PROJECT file) or the file **.solproject** (type: SOLPROJECT file).
- Click **Open**.
- ⇨ The project opens.

Note:
Note that in the basic version of "netX Studio (netPROXY Basic)" you only can access to projects, the configuration of which is limited to *one* application variant and *one* communication protocol (network).

Close project

- To close a project in , click **File > Close Solution**.
- ⇨ The project is closed.

3.3.3 Graphical user interface

The following figure shows the graphical user interface of with the opened project:

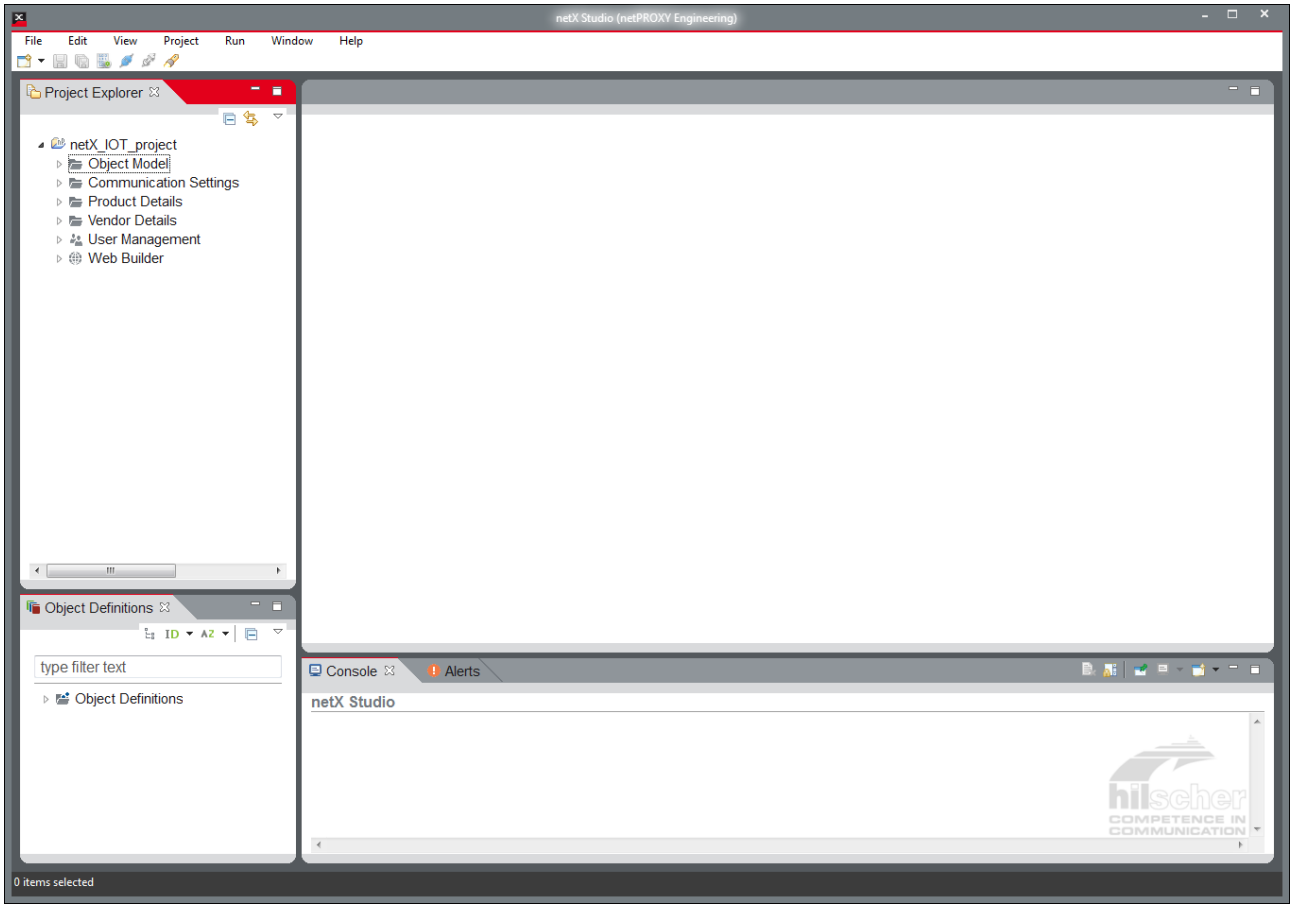


Figure 14: graphical user interface with opened project

Main components

The graphical user interface of the program is divided into:

- the menu,
- the toolbar and
- the movable windows:
  - **Project Explorer** and editor (the shown content depends from the selected folder),
  - **Object Definitions**, **Console** and **Alerts**.

Note:
To perform the IOT configuration correctly, you have to go through the folder structure down from the top in the **Project Explorer** window. Finally, the build process is performed.

Icons and dialogs

There are also the icons to handle the program which are arranged in the single windows, or dialogs that appear during the different steps for the IOT configuration. A description of the icons is given in the respective tooltip.

Further information

A description of the various program components and controls can be found in the descriptions of the IOT configuration steps in the chapter *IOT configuration* [page 34].

3.3.3.1 Toolbar

The toolbar contains icons to access different program options.

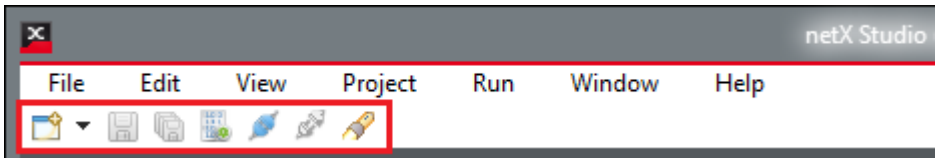


Figure 15: Toolbar (activated)

Icon	Name	Menu path	Meaning	For more information, see section
	New Project	File > New > New Project	Create new project	*Starting program and creating project* [page 20]
	Save	File > Save	Disabled in	-
	Save as	File > Save as		
	Build	Project > Build	Start build process	*Build process* [page 88]
	Connect	Run > Connect	Establish online connection to the device	Without function.
	Disconnect	Run > Disconnect	Disconnect online connection to the device	
	Search	-	Standard search functions in Eclipse	-

Table 5: Icons in the toolbar

Open example projects

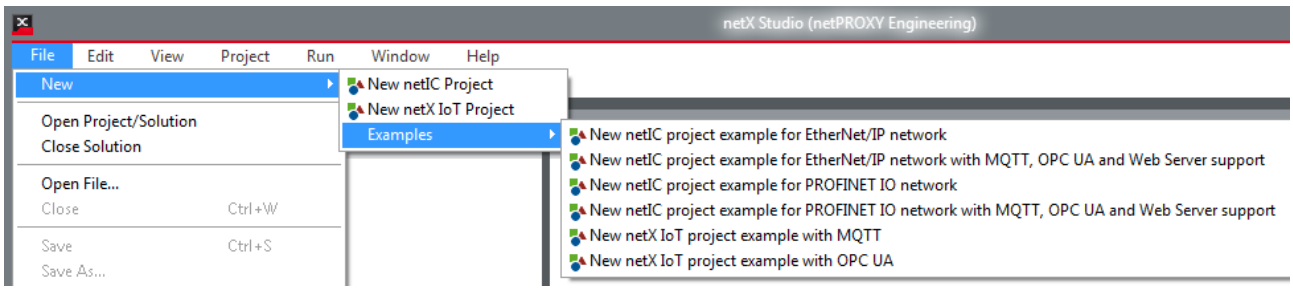


Figure 16: Menu new project

You can open example projects via**File > New > Examples >**

- **New netX IoT project example with MQTT**,
- **New netX IoT project example with OPC UA**.

3.3.3.2 General controls

In the table below you can find explanations of controls that repeatedly appear in in various dialogues.

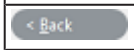
Parameters	Meaning
Filter [type filter text] 	Field to type a filter text.
	Open help information.
	A step back.
	Proceed a step.
	Completing the current step.
	Cancel the operation.

Table 6: General controls

3.4 Access to object data

In you can configure how the object data is to be accessed for IOT communication via OPC UA or MQTT.

OPC UA, MQTT, WebServer

Access to object data via OPC UA, MQTT or WebServer is configured by assigning labels at object level.

Data mapping at the object level

The communication via OPC UA, MQTT or WebServer is configured at object level. This allows access to the data of *all* elements of an object for the respective communication path.

Object Definitions editor and Communication Settings

In the configuration of the access options to the object data for OPC UA, MQTT or WebServer is done in the following way

- with the object definition in the **Object Definitions** editor, by assigning labels on the object level,
- with the communication settings under **Communication Settings**, by supplementing the respective editor or reassigning labels in it and performing the corresponding configuration.

Combinations of different access paths to object data

Important:
The access options to object data via OPC UA, MQTT and WebServer can be combined in different ways. However, OPC UA and MQTT may not be used simultaneously in the communication settings for the same application variant.

Important:
An object or element may only be written by one user (master, OPC client etc.).
An object or element can be read by multiple users.

3.4.1 Examples

The table *Object data by access configuration (examples)* shows examples of possible combinations of different access paths from the network to object data. Considered are the possibilities on how to access **OPC UA**, **MQTT** and **WebServer**.

Object data*	Communication attribute	Label		
		OPC UA	MQTT	WebServ.
• Object data, to be accessed from the WebServer.	None	-	-	www
• Object data, to be accessed from the OPC UA server.	None	OPC UA	-	-
• Object data, to be accessed from the OPC UA server and additionally from the WebServer.	None	OPC UA	-	OPC UA
• Object data, to be accessed via MQTT.	None	-	MQTT	-
• Object data, to be accessed via MQTT and additionally from the WebServer.	None	-	MQTT	MQTT

Table 7: Object data by access configuration (examples)

* To all the data referred in the Object data column, the application has read and write access.

4 Overview of the IOT configuration

The goal of the IOT configuration is to extend products with netX firmware by the IOT functions OPC UA or MQTT. Based on netPROXY an integrated WebServer can be added.

4.1 Create project configuration and specify IOT configuration

As a preparatory step for the mapping of the device properties in objects and the IOT configuration with a listing of the existing functionality, the device properties or the device design and the specification of the IOT configuration of the products to be configured is required (in the example used, 'netX_IOT_project' with the Example device 'C100').

Proceed as follows:

- **Design of the products and the products family**

➢ For the products (or product family) for which IOT functions are to be added, compile a list of existing functions, device properties or device design.

- **Definition of the properties and the assignment to object categories**

➢ Specify object categories (possibly with ID ranges) and objects to map the device properties.

The object categories are used to structure the objects in a way that makes sense.

➢ Set the object identifiers, object names and the assignment of objects to categories.

Also consider the standard objects available in .

- **Access to the objects via the network**

➢ Specify the IOT functions to be supplemented (OPC UA or MQTT) or the WebServer function if necessary, and define the required communication paths (OPC UA or MQTT, or WebServer).

- **Creation of elements**

➢ Specify which elements you require for the individual objects, that is, whether, for example, elements still need to be created for parameters.

4.2 Offline operation and requirements

The IOT configuration with including the build process is performed offline. A connection to a device is not required.

4.3 Steps on how to proceed

The following table shows an overview of the IOT configuration, commissioning or test.

Task	Step	Description	Menu or folder in netX Studio Engineering Tool	For more information see section
IOT configuration (offline process)	1.1	Creating project	File > New > New Project …	*Starting program and creating project* [▷ page 20]
	1.2	Creating object	Object Model / Object Definitions / User	*Creating objects* [▷ page 34]
	1.3	Using standard objects	Object Model / Object Definitions / Standard	*Using standard objects* [▷ page 51]
	1.4	Configuring application variant (scope of functions)	Object Model / Object Instances	Configuring application variant (scope of functions)
	1.5	Performing communication settings	Communication Settings	Performing communication settings
	1.6	Configuring product details	Product Details	*Configuring product details* [▷ page 75]
	1.7	Typing vendor details	Vendor Details	*Entering vendor information* [▷ page 82]
	1.8	Configuring user managment	User Management	*Configuring user managment* [▷ page 84]
	1.9	Creating and managing web content	Web Builder	*Creating and managing web content* [▷ page 88]
	1.10	Creating product files	Product Build	*Generating product files* [▷ page 88]
Loading the product files	2.1	Transfer product files (with the configuration) to the device using the host application.	-	*Downloading product files* [▷ page 94]

Table 8: Step sequence during project configuration or IOT configuration

4.3.1 Explaining the steps

The following section provides explanations of the project configuration steps for mapping the existing products and IOT configuration in .

Project configuration and additional IOT configuration (offline process)

- **Step 1.1: Creating a project**

➢ Create a new project.

- **Step 1.2: Creating object**

➢ Specify the functionality, including device properties, for the products with the IOT functions.

➢ Specify the IOT functions to be added.

➢ Define the objects and the access to the objects and how they are mapped to the network.

Communication to the network can be established via OPC UA or MQTT, or additionally via WebServer.

- **Step 1.3: Using standard objects**

offers the option of using predefined standard objects at object definition level. An assignment of the standard objects to OPC UA Client, MQTT or WebServer is not set up by default and can only be configured manually (if necessary).

- **Step 1.4: Configuring application variant (scope of functions)**

➢ Use the list in step 1.3 to clarify the product classification to be mapped, the range of functions available or the device properties.

➢ Based on the objects defined in step 1.3, configure the functionality for the products intended for IOT configuration.

Note:
Note that in the basic version of "netX Studio (netPROXY Basic)" you can only configure *one* application variant (Application).

- **Step 1.5: Making communication settings (IOT configuration)**

➢ Make the necessary communication settings to configure access to object data via OPC UA, MQTT or WebServer.

- **Step 1.6: Configuring product details**

➢ For each device type set a folder for the product details.

➢ Configure the product data for hardware, firmware and for the boot loader, and the data for product identification and versioning of each single device.

- **Step 1.7: Typing vendor information**
- **Step 1.9: Configuring user management**
- **Step 1.9: Creating and managing web content**

Information on user administration and the WebServer can be found in separate documentation (see section *References* [▷ page 109]).

- **Step 1.10 Generating product files**

➢ Start the build process for the device to create the product files with the configuration.

The product files contain the object definition and the IOT configuration as well as all settings for the product.

Loading the product files

- **Step 2.1: Load product files with the configuration into the device**

➢ Load all product files created with the build under **Product Build > Products > Product (Application)** into the device using netHOST.exe or the host application.

4.3.2 Labels and communication settings for OPC UA, MQTT or WebServer

This section provides an overview of the basic procedure for configuring access to the object data using labels or communication settings for OPC UA, MQTT or WebServer.

Preparation and requirements

First, create a schema on how to access the object data required for your product or to which objects is to be accessed to.

Important:
1. An object or element may only be written by one user (master, OPC client, etc.).
2. An object or element can be read by multiple users.

Principal steps in

1. With the object definition assign one or more labels on object level.
   - In the **Object Definitions** editor via **Assign label** assign a label to the object, see section *Specifying object properties* [page 43] and/or
2. Configure the communication settings for OPC UA, MQTT or WebServer (**Communication Settings**), see section *Adding OPC UA, MQTT or WebServer* [page 65].
   - OPC UA: Configure in the **OPC UA** editor the access via the OPC UA server to object data.
   - OPC UA: In the **OPC UA** editor, define which objects are to be displayed via the OPC UA server.
     Note: Objects for which access is configured from the WebServer, can be displayed simultaneously via the OPC UA server.
   - MQTT: Configure in the **MQTT** editor the access via MQTT to object data.
   - MQTT: In the **MQTT** editor, specify which objects are to be displayed using MQTT.
     Note: Objects for which access is configured from the WebServer, can be displayed simultaneously via MQTT.
   - WebServer: Specify in the **WebServer** editor, which objects are to be displayed via WebServer.
     Note: Objects, for which an access from the OPC UA server or via MQTT is configured, can be displayed simultaneously via WebServer.

4.3.3 Example C100

The example used in this manual includes the C100 product with a counter and a parameter object.

Object data for the example C100

The product used in the C100 example has the function of an operating hours counter and contains a parameter object.

Object identifier	Object name
0x40010001	Counter
0x40020001	Parameter

Table 9: Object data for the example C100

Application variants for the example C100

In the example, the existing product C100 is mapped with a counter and a parameter object (see table *Range of functions, example C100* [page 55]). For the product C100 the operating hours counter 'Counter' records the operating hours, while in object 'Parameter' settings can be made.

Communication settings for the example C100

In the example, the communication settings for OPC UA, MQTT and WebServer are added for the product C100.

5 IOT configuration

To configure IOT, you must create a project for your product, create objects in it, and set IOT parameters.

5.1 Creating objects

To map the device properties in to objects, the following main steps are required:

- Generate for all device properties the categories, the objects and the elements of the objects, so that all device properties are mapped to the created objects.
- Define the access to the objects or elements and their mapping to the application and to the network.

In the following sections you will find a step by step description on how to proceed.

5.1.1 Creating category

➢ To create a category, in the **Project Explorer** window right-click on **Object Model** > **Object Definitions** > **User**.

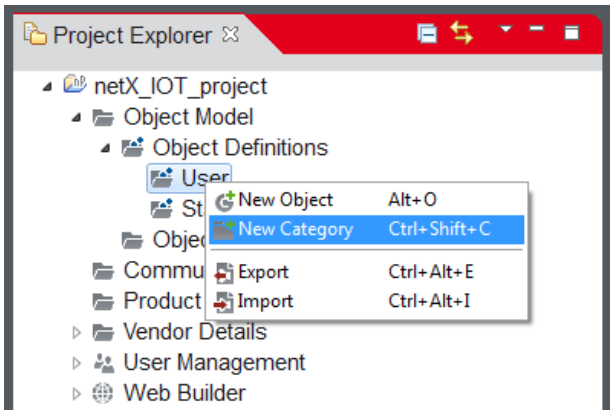


Figure 17: Project Explorer - New Category

➢ Click **New Category**.

↬ The dialog box **Create New Category** appears.

➢ Under **Category name*** (mandatory field), type the name for the category.

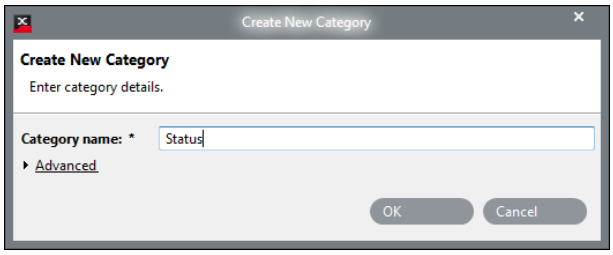


Figure 18: Create New Category

➢ Click **OK**.

⇨ The folder for the category is displayed (in the example **Input data**).

In the same way you can also create additional subcategories.

➢ Also create output data (**Output Data** in the example).

Categories with ID ranges (Advanced)

When you create a category, to reserve object IDs for future extensions, you can define your own ID range for the objects of this category. supports managing object IDs. All object IDs of the objects in the corresponding category are then within this range.

➢ Click **Advanced**.

↬ The **Create New Category** dialog box with the additional input fields for the ID range appears.

➢ Under **Category name*** (mandatory field), type the name for the category.

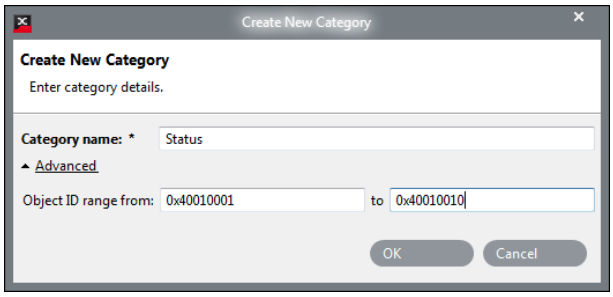


Figure 19: Create New Category – Advanced

➢ Under Object **ID range from** (lower limit) or **to** (upper limit), specify an ID range for the objects of the category.

Note:
The range limits 0x40000000 to 0x4FFFFFFF are valid. The start ID and the end ID must be within these limits. The start ID must be less than the end ID, otherwise an error message will appear.

➢ Click **OK**.

⇨ The folder for the category is displayed (in the example **Input data**). In addition, the defined ID range appears.

Details on categories with ID ranges can be found in the section *ID range for category* [page 37].

Rename, Remove (rename or remove category)

➢ To change the name of a category or to remove the category, right-click in the **Project Explorer** window on a folder of a category (in the example **Input data** or **Output data**).

➢ Click **Rename** or **Remove**.

5.1.1.1 ID range for category

With the full version of **netX Studio (netPROXY Engineering)**, you can assign objects to categories and in addition you can assign objects to defined ID ranges and thereby group them. Fot this the following is valid:

- Optionally, an ID range can be set for each category, which is valid for objects assigned to the category. The range limits 0x40000000 to 0x4FFFFFFF are valid and the start ID must be less than the end ID.
- If an ID range changes, all object IDs of the objects assigned to a category must still be within that ID range.
- To a category with a defined ID range only objects can be assigned the object ID of which is within this ID range.
- For object categories with ID ranges, each object created in a category receives an object ID directly in the ID range defined for this category.
- Two or more categories can have identical or overlapping ID ranges. The object ID remains unique because it can only be used once.
- IP ranges of subcategories are not related to the ID range of the main category and may be outside the ID range of the main category.
- An object can only be moved to a category if its object ID is within the ID range of the target category.
- A category with an ID range can only be imported if it does not contain objects with IDs outside the ID range of the target category. During the import, the existing structure is replaced by the imported structure.

Assigning objects to defined ID ranges

➢ To create an ID range for a category, right-click **Object Model** > **Object Definitions** > **User** > [*Name of category*] in the **Project Explorer** window.

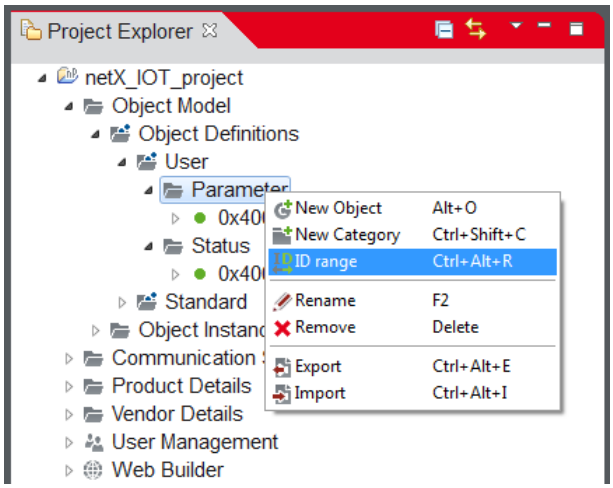


Figure 20: Contextmenu ID range

➢ Click to **ID range**.

➢ The dialogbox **Edit Category Object ID Range** appears.

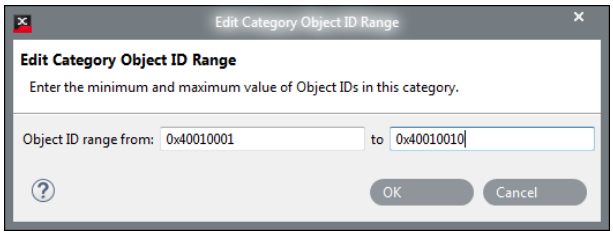


Figure 21: Edit Category Object ID Range

➢ Under **Object ID range from** (lower limit) or **to** (upper limit), specify an ID range for the objects of the category.

Note:
The range limits 0x40000000 to 0x4FFFFFFF are valid and the start ID must be less than the end ID, otherwise an error message will appear.

➢ Click **OK**.

➢ The defined ID ranges (shown in blue in the image) appear in the folder for the category (in the example **Input data** or **Output data**).

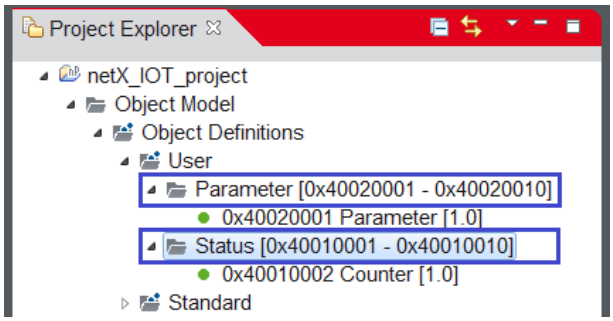


Figure 22: Example of overlapping ID ranges

Importing or exporting categories with defined ID ranges

Categories with defined ID ranges can be imported or exported via **File > Import** or **Export** or via the **Import** or **Export** context menu of a category.

5.1.2 Creating object

You can create objects directly in the **Object Definitions** folder or in a folder of a category or subcategory.

➢ To create an object, right-click in the **Project Explorer** window on a folder of a category (in the example Input data or Output data).

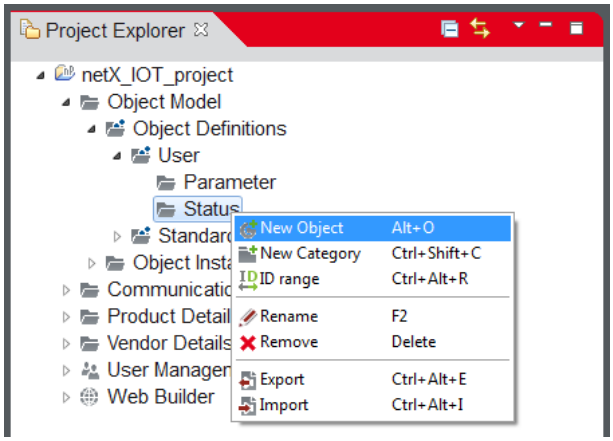


Figure 23: Project Explorer - New Object

➢ Click **New Object**.

↬ The dialog box **New Object** opens.

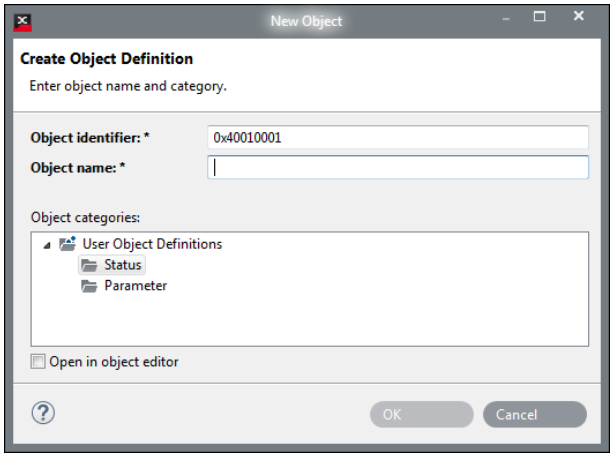


Figure 24: Creating new object

Parameters	Meaning	Range of value/value
Object identifier* (mandatory field)	Each object is uniquely identifiable by its object identifier (object ID). The object identifier is assigned as a consecutive number, no matter whether an object was created by using the **New Object** dialog or by copying. If an object is created within a category with a defined ID range, each object created in this category automatically gets an object ID from the ID range defined for the category. In the case of manual assignment, only object IDs within the defined ID range can be selected. See also section *ID range for category* [▷ page 37].	0x40000000 … 0x4FFFFFFF
Object name* (mandatory field)	Object name	3 … 50 characters
Object categories (optional)	Objects can be assigned to a category and thereby are combined.	

Table 10: New object – parameters

Important:

The object identifier (**Object identifier***) can only be adapted in the **New Object** window. The object identifier (**Object identifier***) cannot be changed later.

If an object has been assigned to a category (**Object categories**) in the **New Object** window once, this assignment may not be changed later.

In the example B100, B150, the specification was made: object identifiers for sensors: 0x4001xxxx object identifiers for actuators: 0x4002xxxx

- ➢ Enter the object identifier (in the example for the 'Counter' object 0x40010001 and for the 'Parameter' object 0x40020001).
- ➢ Add the object name in the mandatory field **Object name***.
- ➢ Under **Object categories** select the object category.

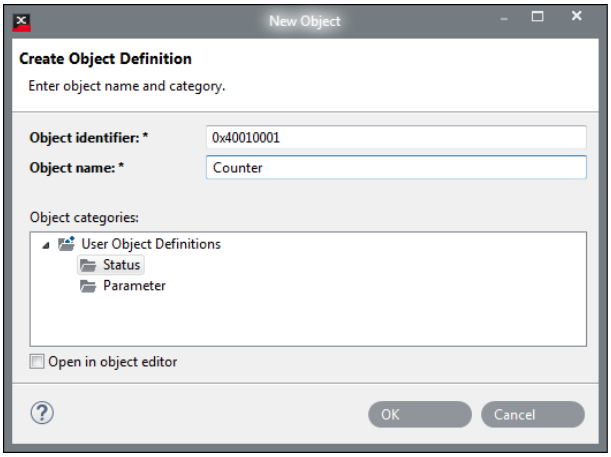


Figure 25: Adapting object identifier, typing object name, selecting object category

- ➢ Click **OK**.

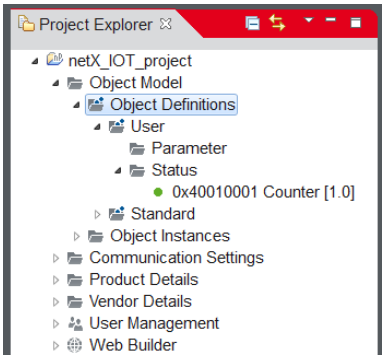


Figure 26: Example object '0x40010000 Sensor - simple [1.0]'

⇨ You have created an object (in the example **0x40010001 Counter**).

Information about how to create an object by copying and pasting or how to rename or remove an object, see section *Copying and pasting object, renaming, removing* [page 41]. It often makes sense to first create the elements of an object and then copy the object afterwards.

5.1.2.1 Copying and pasting object, renaming, removing

Copy (copying object)

➢ To copy an object, right-click in the **Project Explorer** window on an object folder (in this example **0x40010001 Counter**).

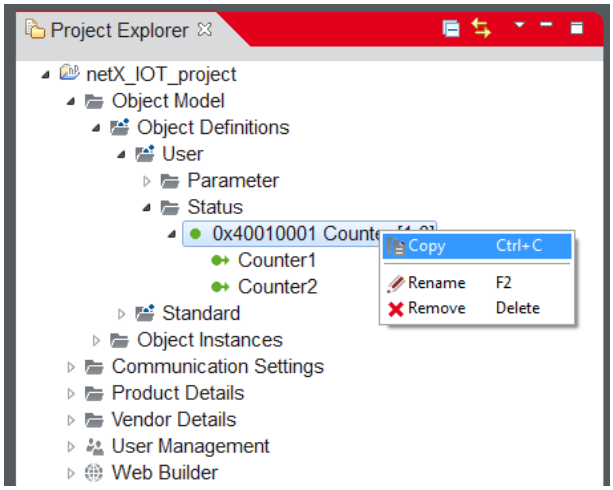


Figure 27: Copy (copying object)

➢ Click **Copy**.

⇨ The object is copied to the clipboard.

Paste (pasting object)

➢ To add an object, right-click on the folder to which you want to add the object and click **Paste**.

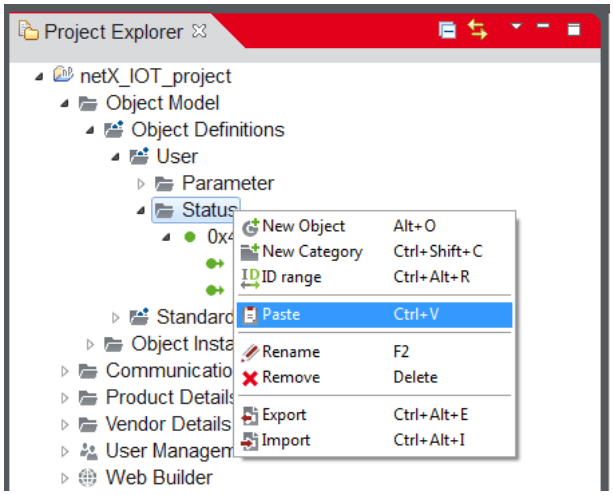


Figure 28: Paste (pasting object)

⇨ The object is inserted at this position.

allows you to copy objects, including elements therein created. If necessary, first create an object with its elements and then generate therefrom further object variants.

Important:

Note that when creating an object by **Copy** and **Paste**, always assigns the object identifiers **Object identifier*** with a consecutive number, starting from 0x40000000. The object identifier cannot be changed.

If the object identifier should be assigned according to additional rules (systematic assignment of identifiers for objects of certain categories, e.g.: 0x4001XXXX or 0x4002XXXX), each object is to be created manually using the command **New Object** and the **New Object** window.

Rename (renaming object)

- To change the name of an object, right-click in the **Project Explorer** window on an object folder (in this example **0x40010001 Counter**).
- Click **Rename**.
- The **Rename** window appears.
- Type the new object name to the **Enter new Name** field and click **OK**.
- The object folder appears with the changed name.

Remove (removing object)

- To remove an object, right-click in the **Project Explorer** window on an object folder (in this example **0x40010001 Counter**).
- Click **Remove**.
- Confirm the question whether you want to remove the object with **OK**.
- The object gets removed.

5.1.3 Specifying object properties

➢ Double click on **Object Model** > **Object Definitions** > **User** and on the object folder (in example **0x40010001 Counter [1.0]**).

↬ In the editor the tab for the object appears, with the subareas **Properties** and **Elements**.

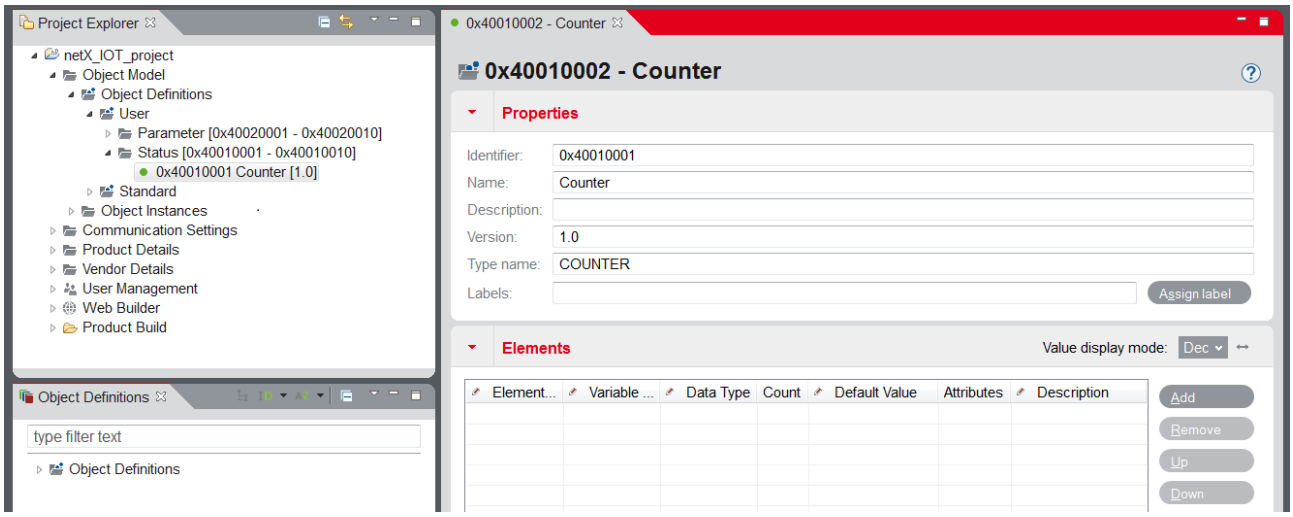


Figure 29: Editor with Properties and Elements - example

The values for **Identifier**, **Name**, **Version** and **Type name**, indicated under **Properties**, are automatically generated or from the data given in the **New Object** window. The object identifier **Identifier** cannot be changed here.

➢ Adapt the parameters under **Properties**.

Assign labels

Via **Assign label** you can create any labels yourself and assign one or more labels to an object. The labels **OPC UA**, **MQTT** and/or **www** already exist in the system.

The assignment of the labels is independent of whether the object is accessed via Real-Time Ethernet (cyclic and/or acyclic communication) or not (see also section *Examples* [page 27]).

By assigning the labels and in a further step by selecting them (see section *Adding OPC UA, MQTT or WebServer* [page 65]), you can define the communication path(s) OPC UA, MQTT or WebServer to be used.

- To assign a built-in or a self-created label, click **Assign label** in the **Object Definitions** editor.
- In the **Assign label** window on the left, select **OPC UA**, **MQTT** and/or **www** (or a self-created label).
- Click on the arrow or double-click on the label, to move the label into the **Assigned labels** list.

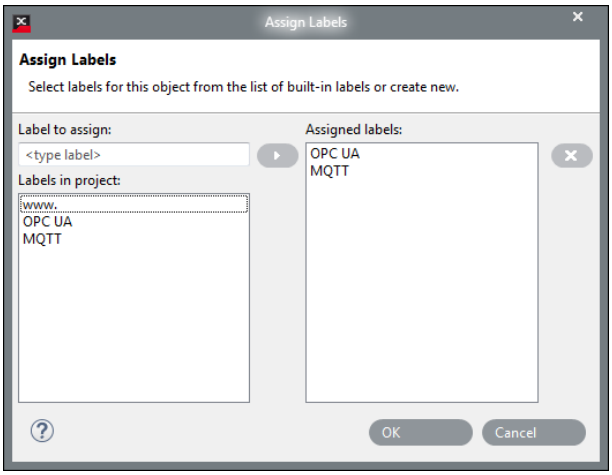


Figure 30: Select label for access via OPC UA, MQTT or www (sample selection www)

- In this way, you can select other labels and move to the right, to assign them to the selected object.
- Click **OK**.
- ⇨ The selected labels (for example **OPC UA**, **MQTT** and **www**) are displayed under **Properties** > **Labels**.

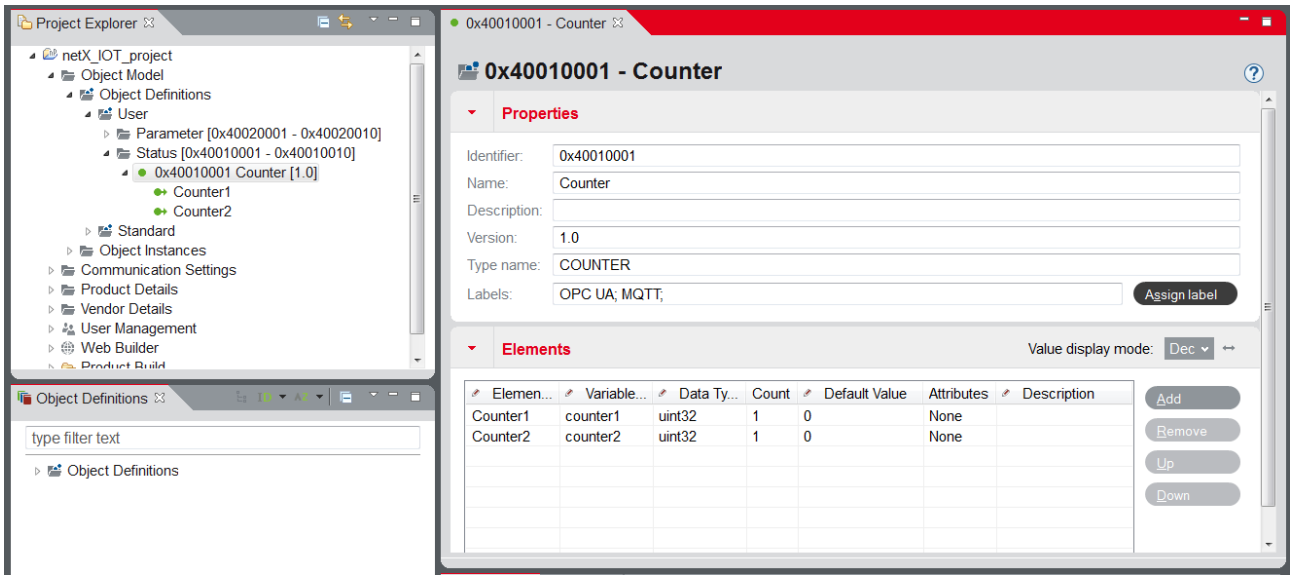


Figure 31: Object Definition editor - selected labels (sample selection OPC UA, MQTT and www)

Label	Meaning
[*custom labels*]	Labels to be defined by the user himself. Such labels are used to configure access to objects from the OPC UA server, via MQTT or from the WebServer. The self-defined label must be assigned in the object definition and selected in the communication settings for **OPC UA, MQTT** or **www**. If required, all objects with a self-defined label can also be displayed via the OPC UA server, MQTT and the WebServer.
OPC UA	From the system predefined label. The label OPC UA serves to configure access from the OPC UA server to objects. The label OPC UA must be assigned with the object definition and can be selected in the communication settings for OPC UA. Objects with the label OPC UA can also be displayed via MQTT and the WebServer if required.
MQTT	From the system predefined label. The label MQTT serves to configure access via MQTT to objects. The label MQTT must be assigned with the object definition and can be selected in the communication settings for MQTT. If required, objects with the label MQTT can also be displayed via the OPC UA server and the Web server.
www	From the system predefined label. The label www serves to configure access from the WebServer to objects. The label www must be assigned with the object definition and can be selected in the communication settings for WebServer. If required, objects with the label www can also be displayed via MQTT and the OPC UA server.

Table 11: Labels for OPC UA, MQTT, www or own labels

The configuration for the access to the object data by the OPC UA Server, MQTT or the WebServer requires the selection of the labels assigned during the object definition in the **Communication Settings**, in each case in the editor **OPC UA** > **Actions**, **MQTT** > **Connections** or **WebServer**, as well as further settings (see section *Adding OPC UA, MQTT or WebServer* [page 65]).

Create custom label

- To assign a custom label, click **Assign label** in the **Object Definitions** editor.
- In the **Assign label** window on the left side in the upper field, enter a name for your own label (example name "test").
- Click on the arrow or double-click on the label, to move the label into the **Assigned labels** list.

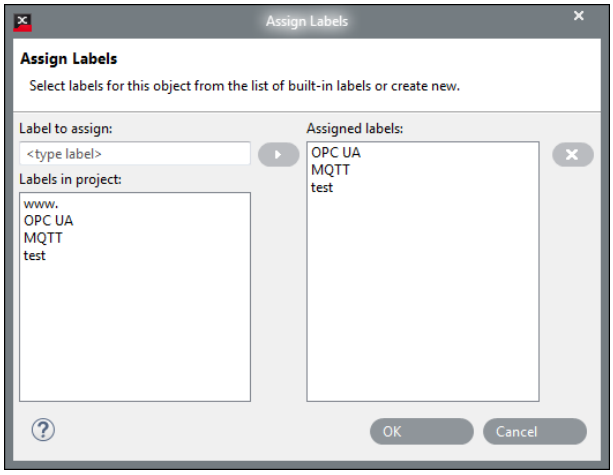


Figure 32: Creating custom label (example name "test")

- In addition you can select other labels and move to the right, to assign them to an object.
- Click **OK**.
- ⇨ The newly created label (example name "test") is displayed under Properties > Labels, next to the other labels.

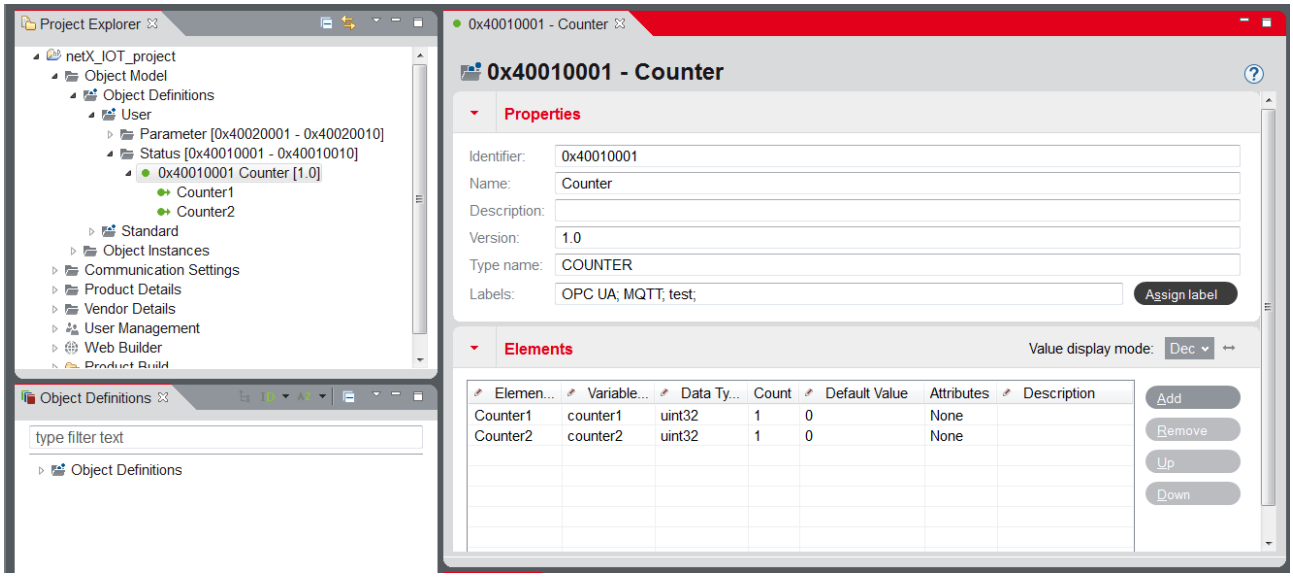


Figure 33: Example assigned labels

Remove label

- You can remove a label by clicking on the label in the list on the right and clicking on the cross .

5.1.4 Creating elements

Elements for example C100

In example C100, for the counters or parameters (objects 0x40010001 or 0x40020001) the elements are

- **counter** (number of operating hours) and
- **parameters** (settings).

How to proceed

➢ Under **Elements** click to **Add**.

↬ A new element is added.

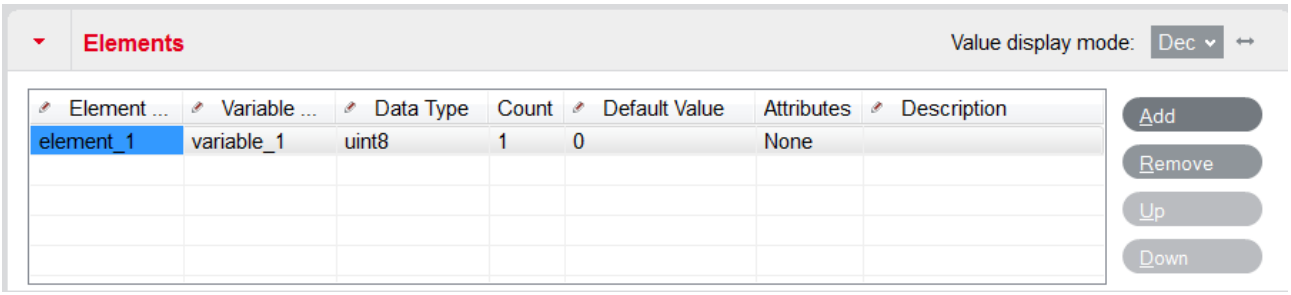


Figure 34: Object Definitions editor - create element (example object '0x40010001 Counter')

For a description on how to use the table **Elements**, see section *Controls of the Object Definitions editor* [page 49].

Parameter	Meaning	Range of value/value
Element Name*	Name of the element that appears also under **Object Instances** with the element of the application variant.	
Variable Name	Here you can define for each element an additional name (does not appear under **Object Instances**). The variable name is used in the header file objects.h for defining the object structure in the C programming language. Recommendation: Use a C-compliant notation, for example bValue.	
Data Type*	Data type of the element Example: Selection uint8	boolean, uint8, uint16, uint32, uint64, int8, int16, int32, int64, string, real32, real64
Count	Maximum number per data type: A single data type is used for the data types boolean, uint8, uint16, uint32, uint64, int8, int16, int32, int64, real32, real64, that is, always count = 1 is valid. For the data type string: Count = 1 ... 1540	For boolean, uint8, uint16, uint32, uint64, int8, int16, int32, int64, real32, real64: Count = 1. For string: Count = 1 ... 1540 See also table *Object element data types, size, number, limits* [page 108].
Default Value*	The default value to be set by the user specifies the initial value for the data area. The elements of the objects require an initial value. After a device reset initially the default value is valid. If for the example of a temperature sensor that can work in the operating modes 0 to 3, a default value of 3 is set, this means that the sensor is operating in the mode 3 when the device starts.	The default value is dependent on the application case.

Parameter	Meaning	Range of value/value
Attributes*	The communication attribute for assigning element data to the application or cyclic network communication is set to **None** by default. During IOT configuration, the I/O data is not processed via netPROXY and the user cannot select anything here.	None
Description*	Here you can add an additional description for the element.	3 … 50 characters

Table 12: Parameters of the elements (Object Definitions)

Note:

*Note that the name, data type, default value and the description set here, are used as defaults when the object is configured in an application variant. A change of the name, data type, default value or of the description in the object definition affects all object instances. In the **Object Instances** editor in the **Element Value** column changes made for the default value will be overwritten by subsequent changes under **Object Definitions** > **User**. Further details you find in the table *Parameters of the elements (Object Instances)* [page 60].

Element name or variable name

- Double click into the table cell for **Element Name** or **Variable Name** to edit the text.
- Customize the name for the element and the variable.

Data Type

- Double click into the table cell for **Data Type** or click in the table cell and press F2.
- The list of data types will appear.
- Select the data type.

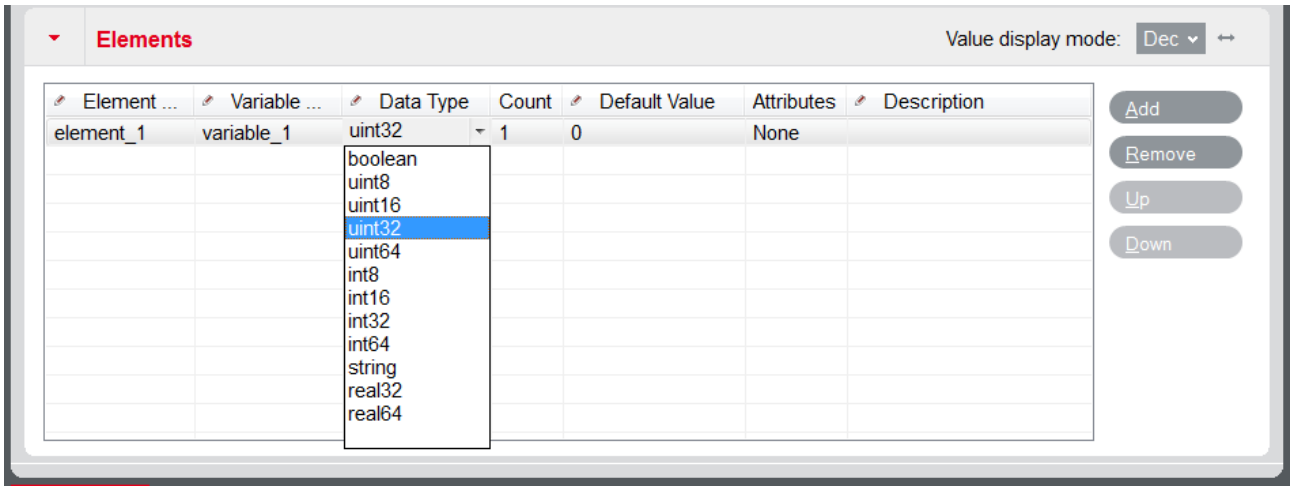


Figure 35: Selecting data type

Count

For the data type "string":

- Double click into the table cell for **Count**.
- Specify the number of data types.

Default Value

Note:
Note that a change of the default value done with the object definition affects all object instances. Values that were set with the configuration of the application variant in the **Element value*** column are thereby overwritten.

* See the table *Parameters of the elements (Object Instances)* [page 60].

- To set the default value, double click into the table cell for **Default Value**.
- Set the default value.

5.1.4.1 Controls of the Object Definitions editor

In the **Object Definitions** editor in the table **Elements** you can perform the actions described below:

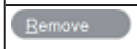
Action	Meaning
	Via **Value display mode** you can choose decimal or hexadecimal data view.
 Auto Resize	Via **Auto Resize** you can reset the width of the table columns to the default setting.
 Help	Via **Help** you can open the online help.
	Via **Add** you can add an element.
	Via **Remove** you can remove an element.
	Via **Up** you can move up an element.
	Via **Down** you can move down an element.

Table 13: Controls of the Object Definition editor

5.1.4.2 Example for IOT communication via OPC UA or MQTT

In the example of the operating hours counter, the number of operating hours 'counter1', 'counter2' and the device settings 'parameter1', 'parameter2' are used.

In the**Attributes** column, the list entry **None** is preset by default for all elements. Descriptions of data assignment using labels can be found in the section *Performing communication settings* [▸ page 63].

Example objects 'Counter' and 'Parameter'

The following figures show the example objects **0x40010001 Counter** and **0x40020001 Parameter** and their elements:

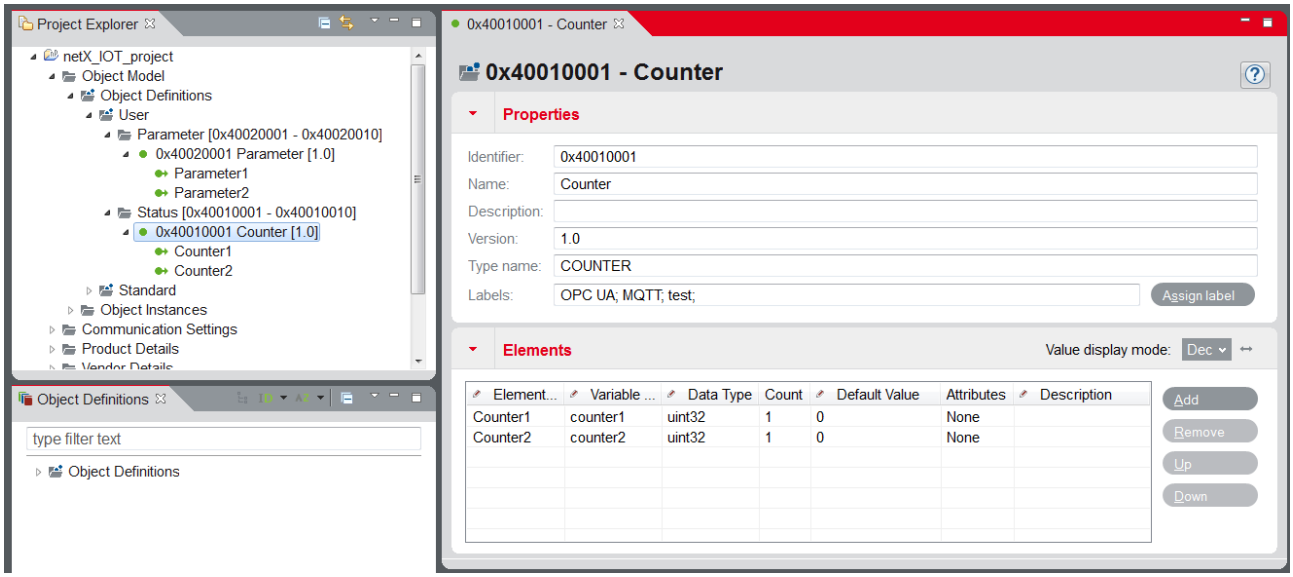


Figure 36: Example object '0x40010001 Counter [1.0]', elements 'Counter1' and 'Counter2'.

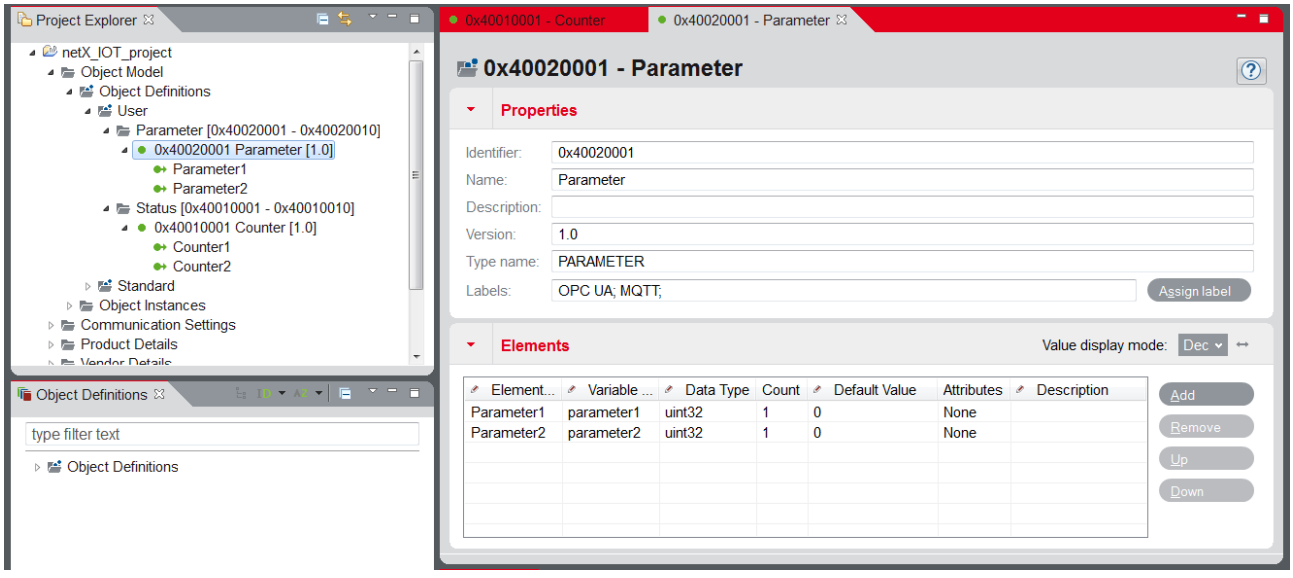


Figure 37: Example object '0x40020001 Parameter [1.0]', elements 'Parameter1' and 'Parameter2'.

The elements for mapping the device properties in the objects are thus defined.

5.2 Using standard objects

has at the level of the object definition the possibility to use predefined default objects. How you can use standard objects is described in the following sections using the example for Generic Device.

For more information about predefined standard objects, see the Technical Reference Guide netPROXY, objects and packages, DOC160204TRXXDE [2].

5.2.1 Adding and displaying Standard objects

➢ When performing the object definition, right-click on the **Standard** folder.

➢ Click **Add**.

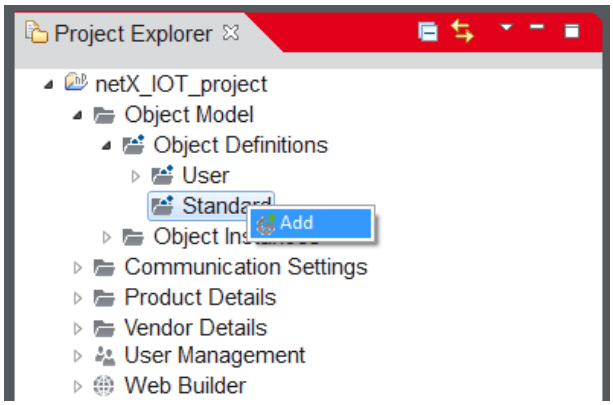


Figure 38: Using standard objects

The Standard objects window appears:

➢ Click on the category for the Standard objects and **OK**.

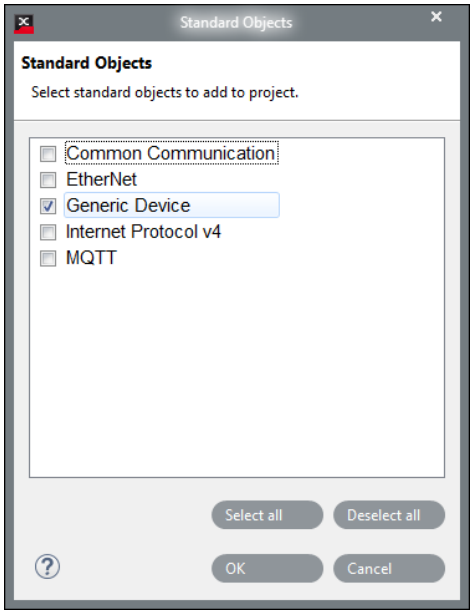


Figure 39: Selection for Standard objects - example Generic Device

In the Project Explorer window the object category Generic Device appears with the Standard objects contained in this folder.

➢ To view an object in the editor on the right side, double click on the object folder.

↬ The appropriate default object is displayed:

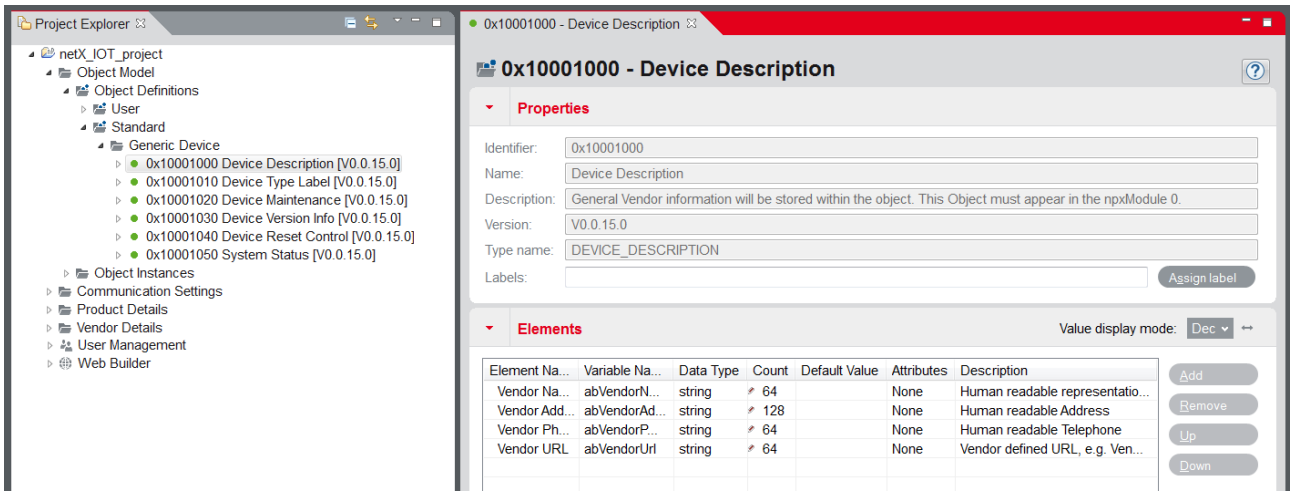


Figure 40: Object editor 0x10001000 - Device Description

5.2.2 Copying or removing Standard objects

Copy Standard objects

Standard objects you can copy and paste under **Object Definitions** > **User**.

- ➢ Right-click on a Standard object.
- ➢ Click **Copy**.

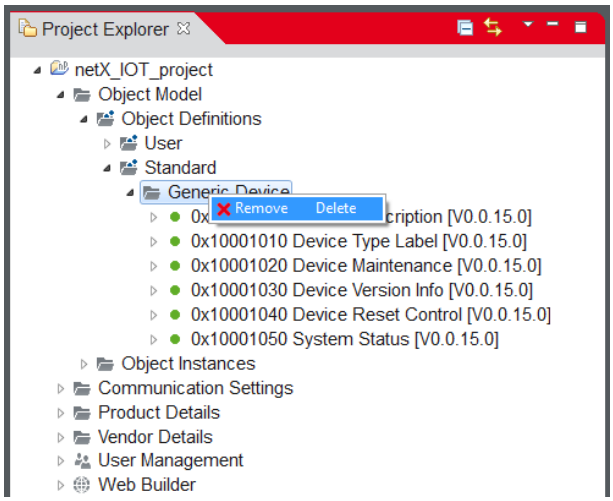


Figure 41: Copying Standard objects (example)

- ➢ Right-click on the **Object Definitions**> **User** folder or subfolder.
- ➢ Click **Paste**.
- ➢ The object is inserted.

Remove category with Standard objects

Standard objects can only be removed by removing the entire object category.

- ➢ Right-click on the category folder (in the example Generic Devices).
- ➢ Click on **Remove**.

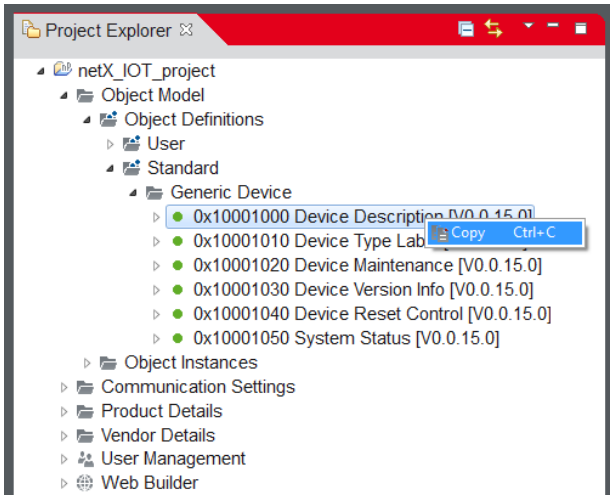


Figure 42: Removing category with Standard objects (example)

- ➢ The selected folder is removed.

5.2.3 Assigning Standard Objects to OPC UA, MQTT, and WebServer

An assignment of the standard objects to OPC UA, MQTT or WebServer is not set up by default and must be configured manually by assigning labels.

In the section *Examples* [page 27] you will find explanations on communication via OPC UA, MQTT or WebServer and in the section *Specifying object properties* [page 43] you will find further details on label assignment.

Proceed as follows:

- To assign an existing or self-created label, click **Assign label** in the **Object definition** editor.
- In the **Assign label** window on the left, select **OPC UA**, **MQTT** and/or **www** (or a self-created label).
- Click on the arrow or double-click on the label to move the label to the list of selected labels.

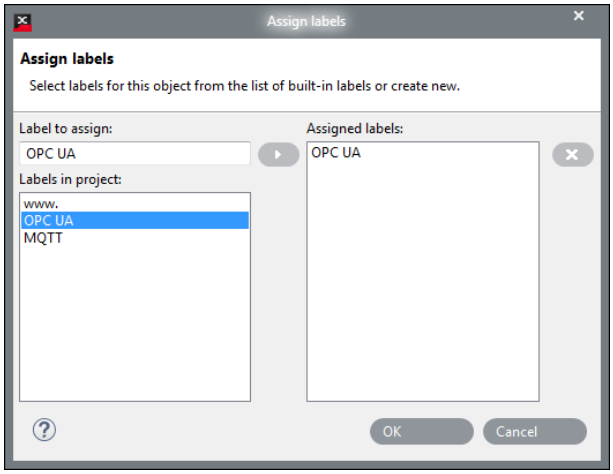


Figure 43: Select label for access via OPC UA, MQTT or WebServer (sample selection OPC UA)

- In this way, you can select other labels and move to the right, to assign it to an object.
- Click **OK**.

⇨ The selected labels are displayed under **Properties** > **Labels**.

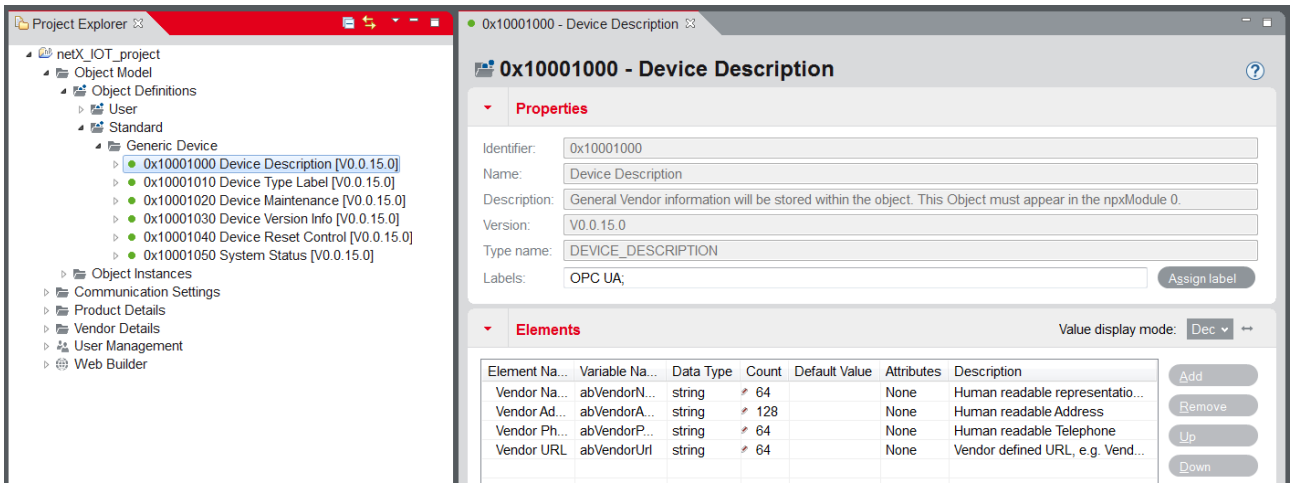


Figure 44: Object Definitions editor - selected labels (example selection OPC UA)

5.3 Configuring application variant (scope of functions)

The products or product variants equipped with IOT functions are compiled or configured as object instances from the objects created in .

The following main steps are necessary for this:

- Define the range of functions of the products or product variants.
- On the basis of the defined objects, configure suitable application variants with the functional scope of the product variants.
- If necessary, adjust the initial value (default value) for all elements used in the application variants.

The following sections provide a step-by-step description of the procedure.

5.3.1 Define scope of functions

In preparation for programming your application or the device functions or the IOT functions, you must define the scope of functions which is to be achieved by supplementing the IOT functions.

In the example C100, the object definition consists of the object for an operating hours counter 'Counter' and an object for device settings 'Parameter'.

Counters or parameters are defined as the functional scope of application variant C100.

Application variant	Scope of functions	Instance ID	Object number	Object name 1 (for instance)	Object name 2 (for object)
Application C100	2 Counter	0:	0x4001001	Counter	Counter
	2 Parameter	0:	0x4002001	Parameter	Parameter

Table 14: Range of functions, example C100

5.3.2 Configure application variant

You can create the application variants and their functional scope required for your products by compiling and configuring the objects defined under **Object Definitions** and the device properties mapped in them in application variants suitable for your products.

To configure an application variant, proceed as follows:

- ➢ Create the application variant.
- ➢ Configure the objects for the application variant by adding the objects.
- ➢ To configure the parameters of the objects, make the following adjustments:

- Object instances tab:
  - Object instance names*
  - Default values of the elements**
  - Element value**
- Details tab:
  - Unit**
  - Min Value and Max Value**

*at the object level, **at the element level

You can create additional application variants by cloning the first application variant or by adding new application variants.

The example used for this manual is shown in the table *Range of functions, example C100* [▷ page 55].

5.3.2.1 Create application variant, add objects

To create the first application variants and configure the objects for them, proceed as follows:

Create application variant

➢ In the **Project Explorer** window, right-click the **Object Model > Object Instances** folder.

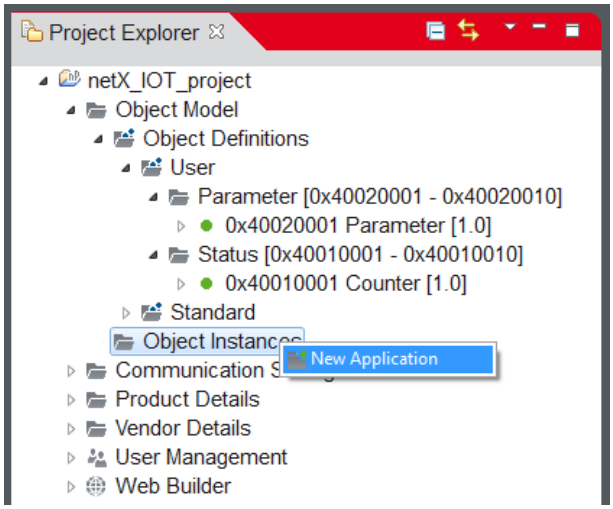


Figure 45: Create application variant

➢ Click **New Application**.

↬ The **New Application** window appears.

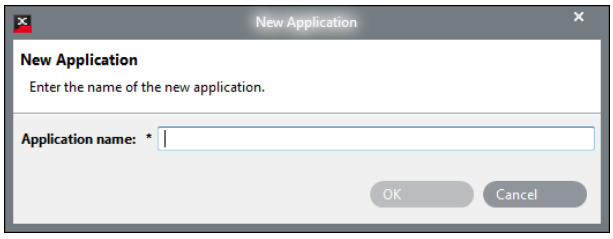


Figure 46: Enter name for first application variant

➢ Under **Application name** type the new name for the first application variant.

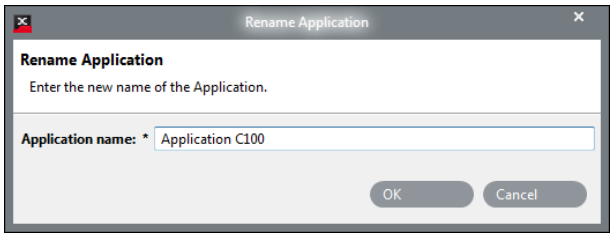


Figure 47: Example Application C100

➢ Click **OK**.

↬ The folder for the application variant appears.

Configure objects of the application variant

To configure the application variant, you must drag the object folders created for your product to the application variant folder. In the example, the folders **0x40010001 Counter [1.0]** or **0x40020001 Parameter [1.0]** are dragged into the folder **Application C100**.

➢ Click on the folders for the objects to be used in the first application variant.

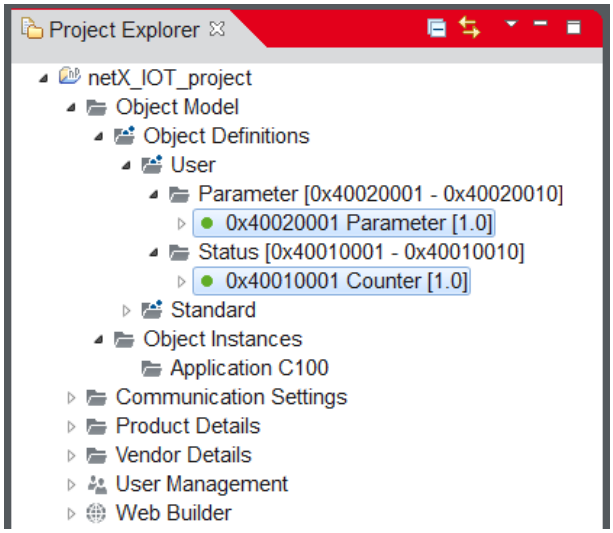


Figure 48: Application Variant (Example Application C100) - Compile Objects Using Drag and Drop

➢ Drag and drop the selected objects into the folder of the application variant.

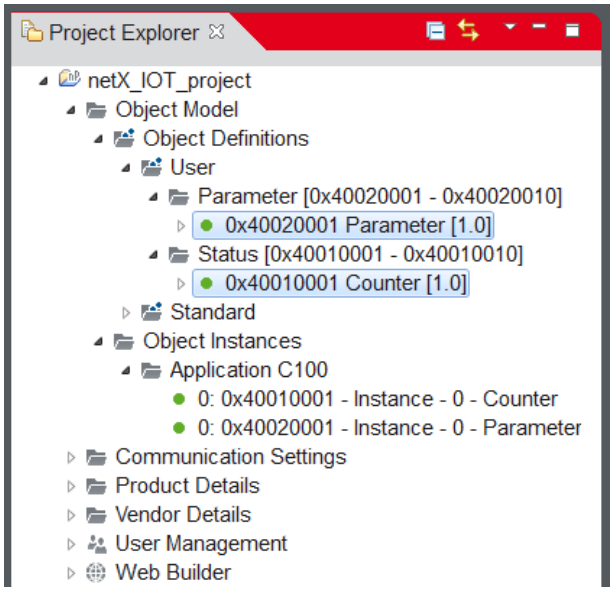


Figure 49: Application variant (example Application C100)

➢ If an object is used more than once in the application variant, repeat the procedure.

5.3.2.2 Customizing object parameters (Object Instances)

To configure the objects, in the Object Instances tab perform the following adaptions:

- Object instance names*
- Default values of the elements**
- Element value**

*at the object level, **at the element level

Customize object instance names and default values of the elements

You can customize the names of the object instances of the application variant, to get a name structuring, that is suitable for your product variants. Furthermore, for each object, configured in the application variant, or for the elements within it, you can individually adapt the default values, where appropriate.

➢ To rename the object instances, you first double-click on an object folder of the application variant.

➢ In the editor, the application variant is displayed (in the example **Application C100**).

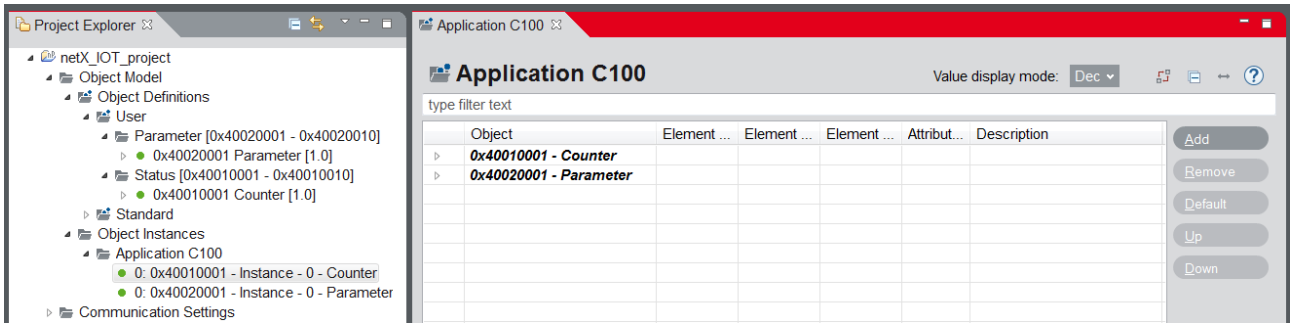


Figure 50: Editor of the application variant (example)

➢ Click in the editor on the line with the object description (in this example **0x40010001 counter**or **0x40020001 parameter**) and expand the lines below, which display the instances and the element data.

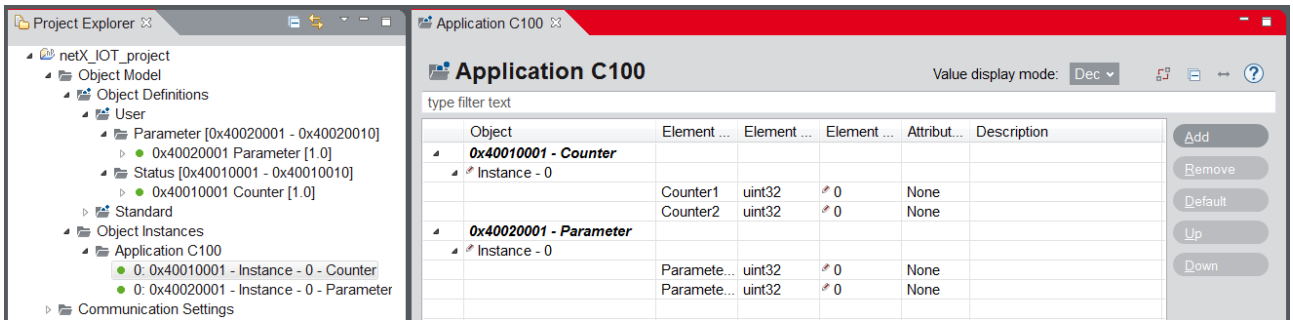


Figure 51: Editor application variant with lines for the instances and elements (example)

For a description of how to use the table, see section Controls of the Application Variant editor.

Parameters of the elements (Object Instances)

Parameters	Meaning	Range of value/value
Object	Object description: In the **Object** column the object descriptions appear, that result from the composed names of the object identifier (Object Identifier) and the object name (Object name); (not editable). The object identifier (Object Identifier) is set in the **New Object** window (see table *New object – parameters* [▷ page 39]). The object name (Object name) is set in the Object Definitions editor. See table Properties – Parameters (Object Definitions).	
	Instance names: Below each object description, the names of the associated object instances appear (editable).	
Element Name	Name of the element that was created under **Object Definitions** (not editable) See table . See table *Parameters of the elements (Object Definitions)* [▷ page 47].	
Element Type	Data type of the element that was configured under **Object Definitions** (not editable here). See table *Parameters of the elements (Object Definitions)* [▷ page 47].	Bit, uint8, uint16, uint32, uint64, int8, int16, int32, int64, string, real32, real64
Element Value	Here editable default value, which applies to the element as the initial value As long as **Element Value** was not customized, the under **Object Definitions** in the **Default Value** field globally configured value appears. See table *Parameters of the elements (Object Definitions)* [▷ page 47]. For each object, configured in the application variant, or for the elements within it, the value for **Element Value** can be adapted individually. This means, in the example C100, the default values for each counter or parameter can be set separately. **Note:** A change of the global default value under **Object Definitions** overwrites all individually set default values.	The default value depends on the application and is defined by the user.
Attributes	Not used (not editable here)	None
Description	Under **Object Definitions** typed description for the element (not editable). See table *Parameters of the elements (Object Definitions)* [▷ page 47].	3 … 50 characters

Table 15: Parameters of the elements (Object Instances)

➢ To change the name for the object instance, move the cursor with a double-click in the cell in the **Object** column that contains the instance name (in the example **Instance 0**).

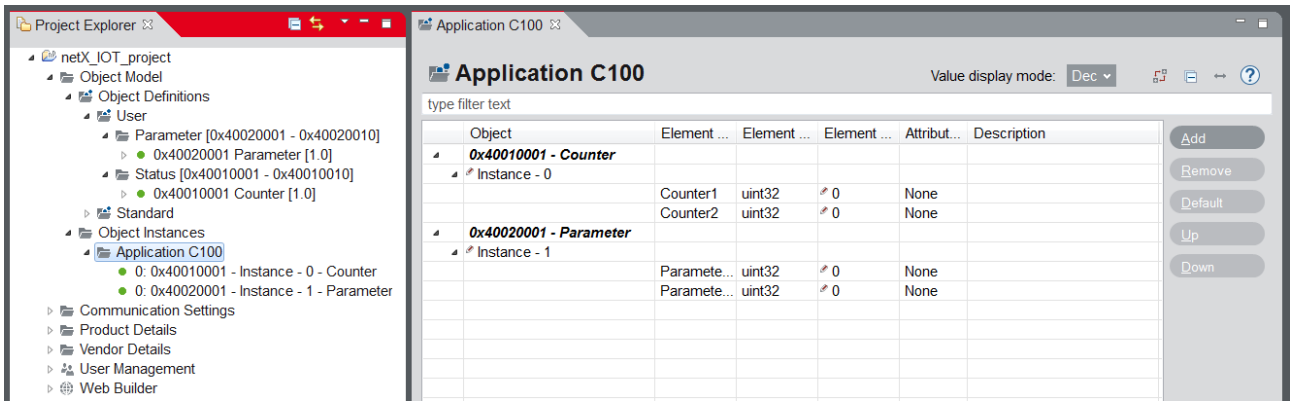


Figure 52: Editor application variant - editing instance name (example)

➢ Enter an instance name suitable for your application variant (in the example Status or Parameter).

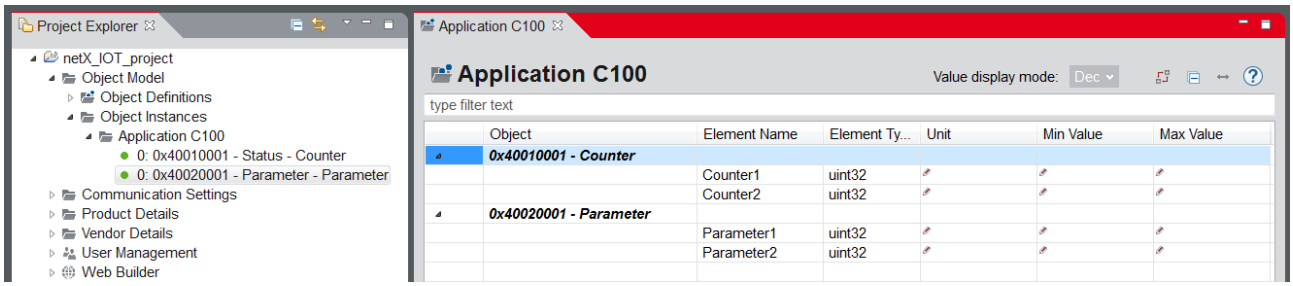


Figure 53: Editor of the application variant with product-related instance names (example)

Customize element value (Element value)

Important:

In the editor for an application variant, you can edit **Element value**.

Note that a change of the global default value in the **Default Value** field, under **Object Definitions**, overwrites all in the **Element Value** field, under **Object Instances**, individually set default values.

5.3.2.3 Customizing object parameters (Details)

To configure the objects, in the **Details** tab perform the following adaptions:

- Unit**
- Min Value and Max Value**

**on element level

Change Unit, Min. Value and Max. Value per element

For the objects configured in the application variant, you can individually customize for each contained element the parameters **Unit** and **Min Value** (minimum value) or **Max Value** (maximum).

➢ In order to make the changes for **Unit**, **Min Value** and **Max Value** per element, click the **Details** tab.

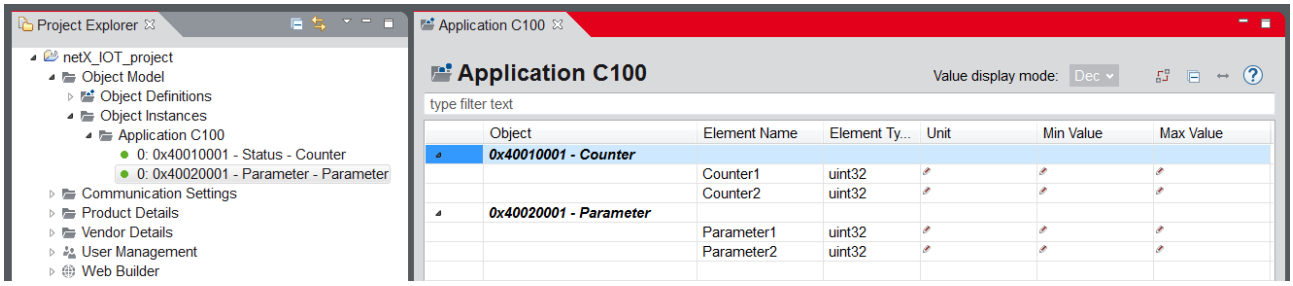


Figure 54: Editor of the application variant - changing Unit, Min./Max. Value (example)

Parameter	Meaning	Range of value/value
Object	In the **Object** column the object descriptions appear that result from the composed names of the object identifier (Object Identifier) and the object name (Object name); (not editable). The object identifier and object name are set in the **New Object** window (see table *New object – parameters* [▷ page 39]). The object name can be adjusted in the Object Definition Editor. See table Properties – Parameters (Object Definitions).	
Element name	Name of the element that was created under **Object Definitions** (not editable) See table *Parameters of the elements (Object Definitions)* [▷ page 47].	
Element Type	Data type of the element, which was configured under **Object Definitions** (not editable) See table *Parameters of the elements (Object Definitions)* [▷ page 47].	Bit, uint8, uint16, uint32, uint64, int8, int16, int32, int64, string, real32, real64
Unit**	Extensive list of units	(e. g. m, kg, Pa, Hz, bit, byte etc.)
Min Value	Freely editable minimum value for the element value	
Max Value	Freely editable maximum value for the element value	

Table 16: Parameters of the elements - Details (Details)

➢ Double-click each in the cell to be adapted and select for **Unit** a physical unit, select for **Min Value** a lower limit or **Max Value** an upper limit for the appropriate value.

5.4 Performing communication settings

To configure the communication settings in :

- Add the communication settings for each application variant and select the IOT protocol and the application (IoT protocol, Application).
- Configure access to object data via OPC UA **or** MQTT (only permitted combinations), or WebServer if necessary.
- For OPC UA or MQTT, make the settings for IOT communication individually.

The following sections provide a step-by-step description of the procedure.

5.4.1 Adding communication settings

You can add communication settings for **one** IoT protocol (OPC UA or MQTT) for each application variant.

Note:
Each application variant can be combined with one IoT protocol.
Note that in the basic version of "netX Studio (netPROXY Basic)", you can configure only *one* communication protocol (network).

Add communication settings

➢ To add communication settings for an application variant, right-click the **Communication Settings** folder in the Project Explorer window.

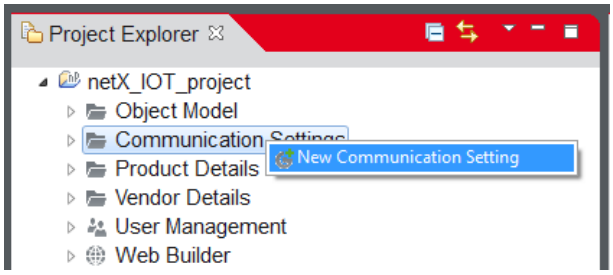


Figure 55: Project Explorer - New Communication Setting

➢ Click **New Communication Setting**.

↬ The dialog box **Add Communication Settings** appears.

➢ Under **IoT protocol**, select the protocol.

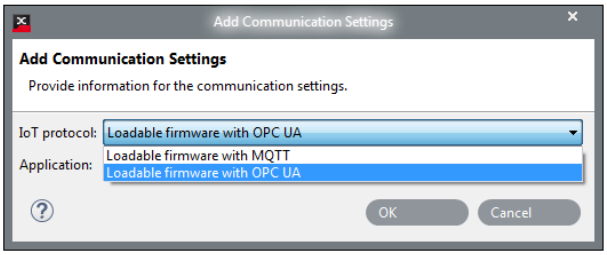


Figure 56: Add Communication Settings - IoT protocol

➢ Under Application select the application variant.

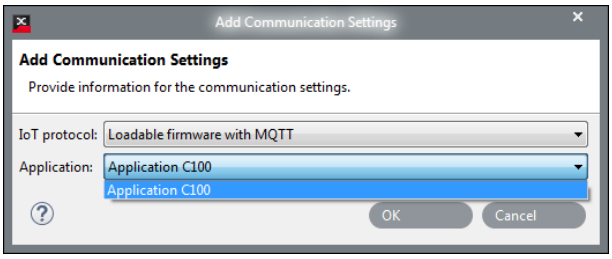


Figure 57: Add Communication Settings - Application (example)

➢ Supplement the communication settings for further application variants.

If you have added the communication settings for each application variant (**one** combination with OPC UA or MQTT), the message appears that no more applications are available for the IoT configuration.

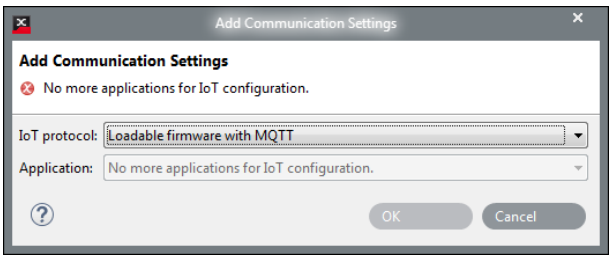


Figure 58: Add Communication Settings – No more applications

Removing communication settings

➢ To remove the communication settings for an application varinate, right-click the corresponding folder in the Project Explorer window below the Communication Settings folder.

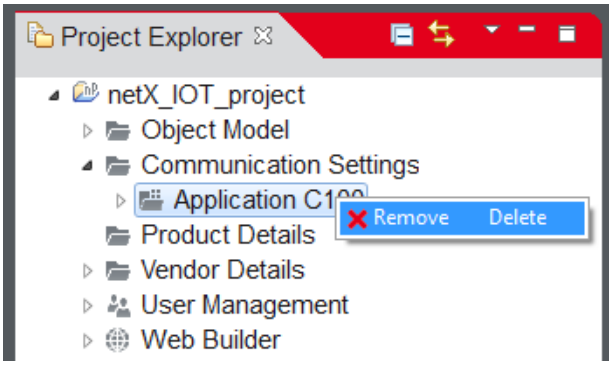


Figure 59: Project Explorer - Remove (example)

➢ Click on Remove.

⇨ The selected communication settings will be removed.

5.4.2 Adding OPC UA, MQTT or WebServer

Object data can be transferred via OPC UA or MQTT or WebServer (see also section *Examples* [page 27]).

For the configuration of OPC UA or MQTT (or WebServer if necessary) it is necessary that you

- Assign one or more predefined labels (**OPC UA** or **MQTT** or **WebServer**) or self-created labels to each object in the **Object Definitions** Editor via **Assign label** (see section *Specifying object properties* [page 43]).
- under **Communication Settings**, add the editor **OPC UA** or **MQTT** or **WebServer** to each application variant, select the assigned labels again in the editor and perform the configuration as described below.

Important:
OPC UA and MQTT must not be used simultaneously in the communication settings for **one and the same** application variant.

Note about the requirements of your end customer

Important:
The is a device development tool. It does not contain any configuration and diagnostic functions for the end user and is not available for the end customer. Because of this reason, note that you must provide to your customer a tool or a solution for the final configuration of the device!

For the final configuration your customer requires a suitable tool with configuration and diagnostic functions. The final configuration includes the indication of the MQTT broker, the indication of the OPC UA port, the user management, as well as simulation steps in preparation for the device commissioning.

To offer to your customer an appropriate solution, you have two options:

- When developing the device with , add the option for communicating via the WebServer.
- Or provide for the end customer an alternative solution to make the final configuration.
  - For example, you can offer to the customer a service for the complete device setup,
  - you can offer to your customers a generic solution,
  - agree with the customer that he must develop his own solution or
  - you schedule a service by the device manufacturer.

Add editor for OPC UA, MQTT or WebServer

➢ Under **Communication Settings**, right-click the folder containing the communication settings of an application variant.

Important:
OPC UA and MQTT must not be used simultaneously in the communication settings for **one and the same** application variant.

➢ Click**Add OPC UA**, **Add MQTT** or **Add WebServer**.

↬ The configuration takes place so that the functions for transferring the object data to the OPC UA server, MQTT or WebServer can be used. The folders **OPC UA**, **MQTT** or **WebServer** are displayed in the **Project Explorer**.

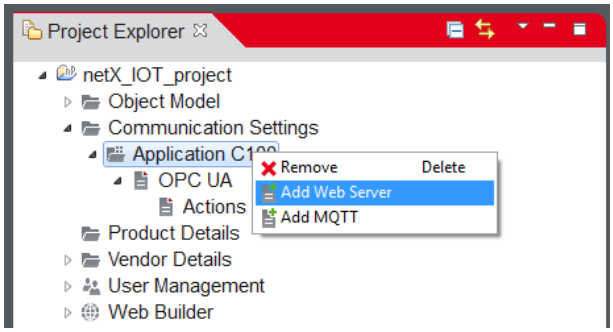


Figure 60: Communication Settings of an Application Variant, Example Add WebServer

5.4.2.1 Communication settings for OPC UA

To configure the communication settings for OPC UA, proceed as follows:

➢ Double-click on the **OPC UA** folder.

↬ The **OPC UA** editor appears.

Under **OPC UA Server Configuration** you can configure various settings.

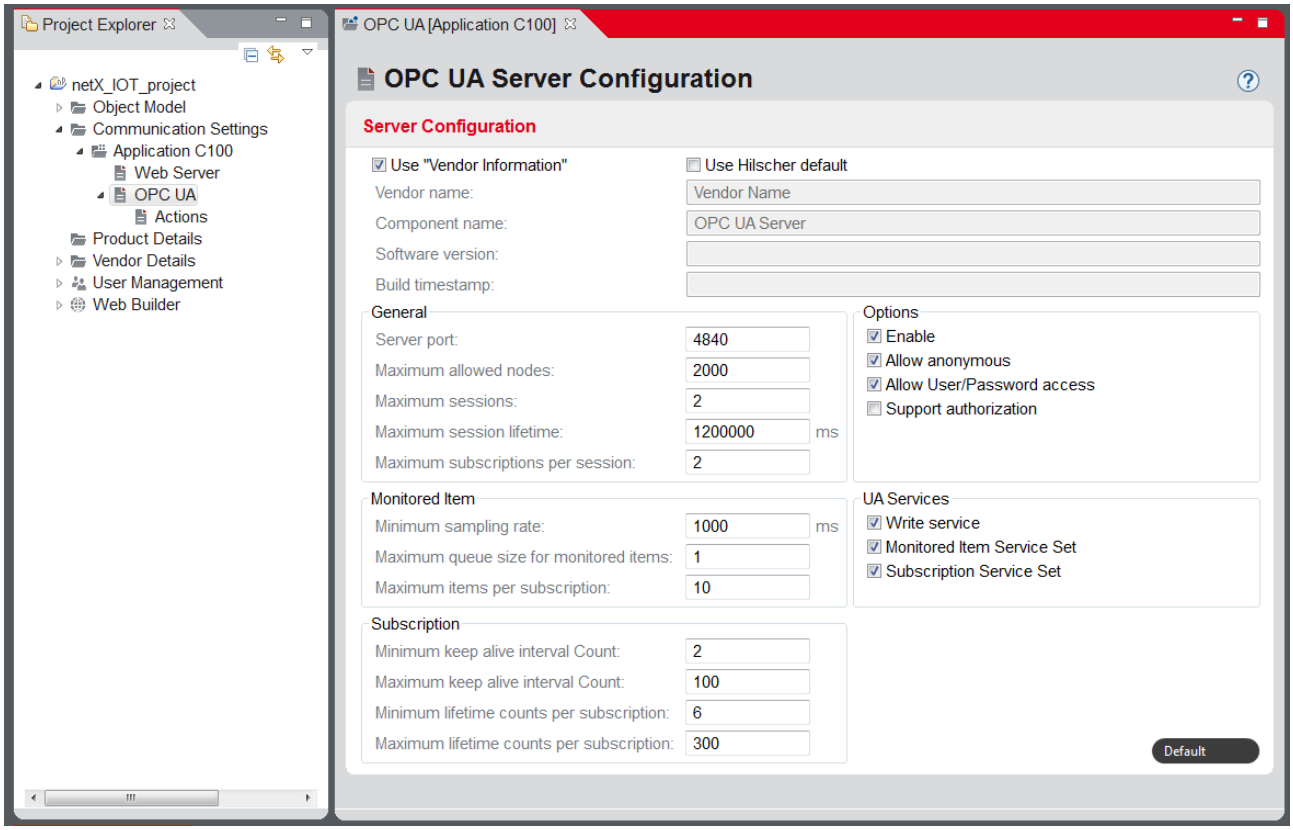


Figure 61: Communication Settings > OPC UA Editor - OPC UA Server Configuration (example)

Parameters OPC UA Server Configuration

Parameter	Meaning	Range of value/value
Use „Vendor information“ Use „Hilscher default“	The default values of the OPC UA server are used for **Vendor name** and **Component name**, or Hilscher standard values for **Software version** and **Build timestamp**.	Unchecked, unchecked, Default: Use "Vendor information" checked
Vendor name	Here you can enter the vendor name freely. This one may be identical or different, as typed under **Project Explorer** > **Vendor Details** > **Vendor Information** (see section *Entering vendor information* [page 82]).	String (max. 63 characters.), Default: *Hilscher Ges. für Systemautomation mbH*
Component name		String (max. 31 characters.), Default: *OPC UA Server*
Software version	String for storing the software version: 'MAJOR.MINOR.REVISION.BUILD TEXT'	-

Parameter	Meaning	Range of value/value
Build timestamp	String to store the build timestamp Format: 'yyyy-MM-dd'T'HH: mm: ss.SSSXXX ' yyyy: year, MM: month, dd: day, T: fixed character (for time), HH: hours, mm: minutes ss: seconds SSS: milliseconds, XXX: time zone; additionally: Z for UTC +00: 00 Example: 2016-04-07T12:34:11.123Z	-
General		
Server port	IP port number of the OPC UA server Port 0 and 1 are not supported.	Valid port, 0, 2, 3, ... 65535; Default: 4840
Maximum allowed nodes	Allowed maximum number of nodes which the server can represent in the defined address space. These are self defined nodes that are configured by the as well as the the mandatory server nodes.	400 ... 2000, Default: 2000
Maximum sessions	Allowed maximum number of UA session which the server has to handle. The nano profile requires only 1. UA communication is divided into sessions.	1, 2, Default: 2
Maximum session lifetime	Allowed maximum lifetime for a UA session in ms.	30000 ms ... 1200000 ms (30 s to 20 min), Default: 1200000 ms (20 min.)
Maximum subscriptions per session*	Allowed maximum number of subscriptions per session	Min: 1, Max: 2, Default: 2
Options Options for configuring the OPC UA server communication.		
Enable	Enable	Checked, unchecked, Default: checked
Allow anonymous	Allow anonymous access	Checked, unchecked, Default: checked
Allow User/Password access	Allow User/Password access	Checked, unchecked, Default: checked
Support authorization	Support authorization	Checked, unchecked, Default: unchecked
Monitored Item		
Minimum Sampling rate in ms*	The sample rate defined for a monitored item may be faster than the publishing rate of the subscription. For this reason, the monitored item may be configured to either queue all notifications or to queue only the latest notification for transfer by the subscription. In this latter case, the queue size is one.	Min: 50, Default: 1000
Maximum queue size for monitored items*	Supported maximum queue size for monitored items	Max: 1, Default: 1
Maximum items per subscription*	Supported maximum items per subscription	Max: 10, Default: 10
UA services		
Write service	Enable write access to OPC UA objects (default)	Checked, unchecked, Default: checked
Monitored Item Service Set	Enable Monitored Item Service Set (default)	Checked, unchecked, Default: checked
Subscription Service Set	Enable Subscription Service Set (default)	Checked, unchecked, Default: checked
Subscription		

Parameter	Meaning	Range of value/value
Minimum keep alive interval Count*	Supported minimum number of 'Keep Alive Interval Counts'.	Min: 1, Max: Subscription: Maximum 'Keep Alive Interval Count' Default: 2
Maximum keep alive interval Count*	Supported maximum number of 'Keep Alive Interval Counts'	Min: Subscription: Minimum 'Keep Alive Interval Count', Max: 1431655765 (1/3 * UINT32_MAX) Default: 100
Minimum lifetime counts for Subscriptions*	Supported minimum Lifetime counts for Subscriptions	Max: 4294967295 (UINT32_MAX), Default: = Min: 3 x Subscription: Minimum 'Keep Alive Interval Count'
Maximum lifetime counts for Subscriptions*	Supported maximum Lifetime counts for Subscriptions	Min: Subscription: Minimum Lifetime Counts for Subscriptions, Max: 4294967295 (UINT32_MAX); Default: 3 x Subscription: Maximum 'Keep Alive Interval Count'
*only available in Micro Server		

Table 17: Parameters OPC UA Server Configuration

Parameters OPC UA Actions Configuration

➢ Double-click on the **OPC UA** > **Actions** folder.

➢ The **OPC UA Actions** editor appears. Under **Filter** the selection for the label **OPC UA** is pre-configured by default. In the table below, you can configure the type of access **read** or **write** for each object instance configured for OPC UA.

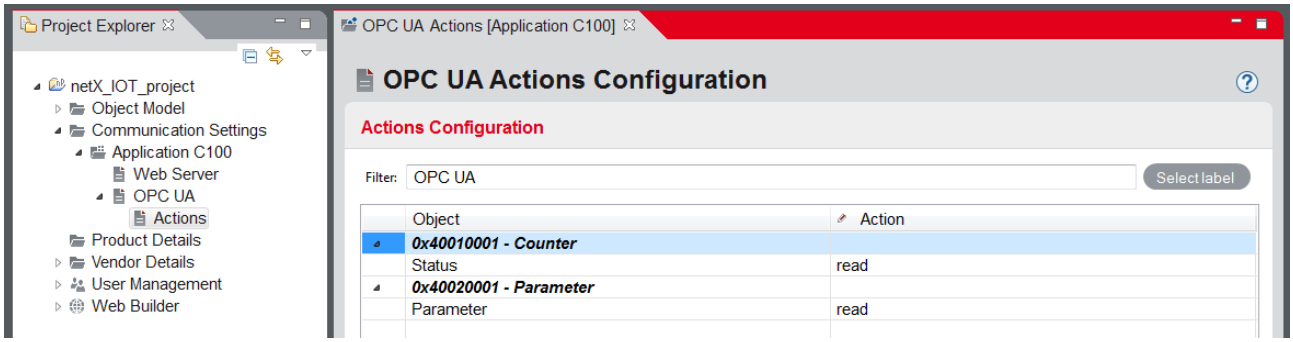


Figure 62: Communication Settings > OPC UA Editor - OPC UA Actions Configuration (Example)

Parameter	Meaning	Range of value/ value
Object	Object name, which was specified in the **New Object** window, and which was adapted if necessary, in the **Object Definitions** editor (see table Properties – Parameters (Object Definitions), cannot be edited here).	Name of the object
Action	Read and write services; sub-function of the overall OPC-UA functionality; see also table*Object functions* [page 11] .	read, write, Default: read

Table 18: Parameter OPC UA Actions Configuration

Note:

For objects with access via WebServer, you can additionally configure access via the OPC UA Server and display the object data configured for the WebServer on the OPC UA Server.

5.4.2.2 Communication settings for MQTT

To configure the communication settings for MQTT, proceed as follows:

➢ Double-click on the **MQTT** folder

↬ The editor **MQTT Component Information** appears.

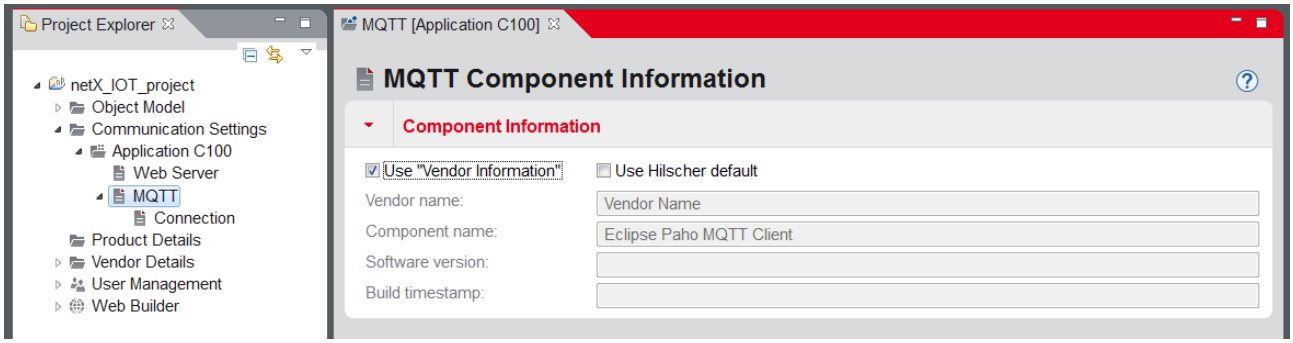


Figure 63: Communication Settings > MQTT Editor - MQTT Component Information (Example)

Parameter	Meaning	Range of value/value
Use „Vendor information“ Use „Hilscher default“	The default values of the Eclipse Paho MQTT Client are used for **Vendor name** and **Component name**, or Hilscher default values for **Software version** and **Build timestamp**.	Checked, unchecked, Default: Use "Vendor information" checked
Vendor name	Vendor name. Here you can enter the vendor name freely. This one may be identical or different, as typed under **Project Explorer** > **Vendor Details** > **Vendor Information** (see section *Entering vendor information* [page 82]).	String (max. 63 characters.), Default: *Hilscher Ges. für Systemautomation mbH*
Component name	Component name	String (max. 31 characters.), Default: *Eclipse Paho MQTT Client*
Software version	String to store the software version: 'MAJOR.MINOR.REVISION.BUILD TEXT'	-
Build timestamp	String to store the build timestamp Format: ‚yyyy-MM-dd’T’HH:mm:ss.SSSXXX‘ yyyy: year, MM: month, dd: day, T: fixed character (for time), HH: hours, mm: minutes ss: seconds SSS: milliseconds, XXX: time zone; additionally: Z for UTC +00: 00 Example: 2016-04-07T12:34:11.123Z	-

Table 19: Parameters MQTT Component Information

Parameter MQTT - Connection

➢ Double-click on the **MQTT** > **Connections** folder.

➮ The **MQTT** editor appears.

MQTT Connection

Parameter	Meaning	Range of value/value
Client ID	Unique name of the MQTT client in UTF-8 format. All devices that are connected to a broker, must have a unique name. If the field is empty, then the broker assigns a name.	max. 23 Bytes for max. 23 chaacters, Default: [Client ID]
Broker address	IP address of the broker	Valid IP address; Default: [BrokerAddress]
Topic prefix	Text that is prefixed to each topic, e. g. 'StationA'. For each single topic can be configured if this prefix is to be preceded or not.	Text of uppercase and lowercase letters and underscore, Default: [*not specified*]
User name	User name to login to the broker.	[*not specified*] or user name, Default: [*not specified*]
Password	Login password to the broker.	[*not specified*] or password, Default: [*not specified*]
Will topic	Will Topic	Text of uppercase and lowercase letters and underscore; Default: [*not specified*]
Connect timeout	Time for trying to establish a connection (MQTT Connect) to the broker. If the connection could not be established, then the MQTT client waits for the duration of 'Connection Timeout' until a new connection is established to the broker.	Specified in s. = 0 MQTT client constantly tries to establish a connection to the broker. Default: 0
Connection idle timeout	Time after an unused connection (no data transmission) is closed.	Specified in s. 0 = no timeout, i. e. connection remains open. Default: 0
MQTT keep alive interval	Interval in which the MQTT client sends a sign of life to the broker. The set value for the MQTT client must be less than the monitoring time set in the broker.	Specified in s. 0 = send no sign of life to the broker. Default: 0

Parameter	Meaning	Range of value/value
Handling of an MQTT connection	Clear session: Setting whether all topics are to be transferred to the broker after establishing a connection or not. unchecked (recommended): After a connection to the broker has been established, all topics of the type 'publish' are transmitted from the MQTT client to the broker . Unchecked: Only those topic are transmitted to the broker, which have changed since the last connection. Note that if you use this setting, the broker must support the 'preserve context' function.	Checked, unchecked, Default: checked
	Reliable: Parallel or sequential transmission. unchecked: parallel transmission of MQTT services. checked: sequential transmission of MQTT services.	Checked, unchecked, Default: unchecked
	Will: unchecked: Will function disabled. checked: Will function enabled.	Checked, unchecked, Default: unchecked
	Prefix will: unchecked: Will-Topic unprefixed. checked: Will-Topic with prefix.	Checked, unchecked, Default: checked
	Secure: unchecked: unsecured data transmission. checked: secured with SSL data transfer.	Checked, unchecked, Default: unchecked
	Verify certificate: Without function.	Checked, unchecked, Default: unchecked

Table 20: Parameters MQTT - Connection Settings

MQTT Configuration

Under **MQTT Configuration > Filter** the selection for the label **MQTT** is preconfigured by default. In the table below, the parameters Action, Retained, Prefix, Encoding QoS or Topic transferred to MQTT for the objects of the corresponding application variant appear.

Parameter	Meaning	Range of value/value
Object	- Object name - cannot be edited - which was specified in the **New Object** window, and which was adapted if necessary in the **Object Definitions** editor (see table Properties – Parameters (Object Definitions)). Below - the name of the object instance -, as set in the **Application Variant** editor and below - name of the element -, which was set under **Object Definitions** (see table *Parameters of the elements (Object Definitions)* [page 47]).	Name of the object
Action	Publish: MQTT client should transmit this topic to the broker. Subscribe: MQTT client should receive this topic from the broker.	Publish, Subscribe, Default: Subscribe
Retained	Setting whether the broker shall store the history of a data value or not. Unchecked: Broker shall not store history. Checked: Broker shall store history.	Checked, unchecked, Default: checked
Prefix	Unchecked: The topic prefix shall not be proceeded to this topic. Checked: The topic prefix shall be proceeded to this topic.	Checked, unchecked, Default: unchecked
Encoding	Data format of the topic. No encoding: Topics are transmitted in binary form. JSON: data is transmitted in JSON format, i. e. together with information on data type, data size and data value. UTF-8/MBCS: data is transmitted as text in UTF-8/MBCS format.	No encoding, UTF-8/MBCS, JSON; Default: JSON
QoS	Quality of Service. Setting: 0 = At most once: data reception is not confirmed via MQTT. 1 = At least once: data reception is confirmed via MQTT. 2 = Exactly once: data is transmitted via MQTT exactly once by state machine and confirmed.	At most once, At least once, ;Default: At most once
Topic	Unique name for the topic, editable.	Max. 128 characters of text from uppercase and lowercase letters and underscore.

Table 21: Parameters MQTT Configuration

Note:

For objects with access via WebServer, you can also configure access via MQTT and display the object data configured for the WebServer via MQTT.

5.4.2.3 Communication settings for WebServer

To configure the communication settings for WebServer, proceed as follows:

➢ Double-click on the **WebServer** folder.

➢ The **WebServer** editor appears. Under **Filter** the selection for the label **www** is pre-configured by default. In the table below, the object data transferred to the WebServer appears.

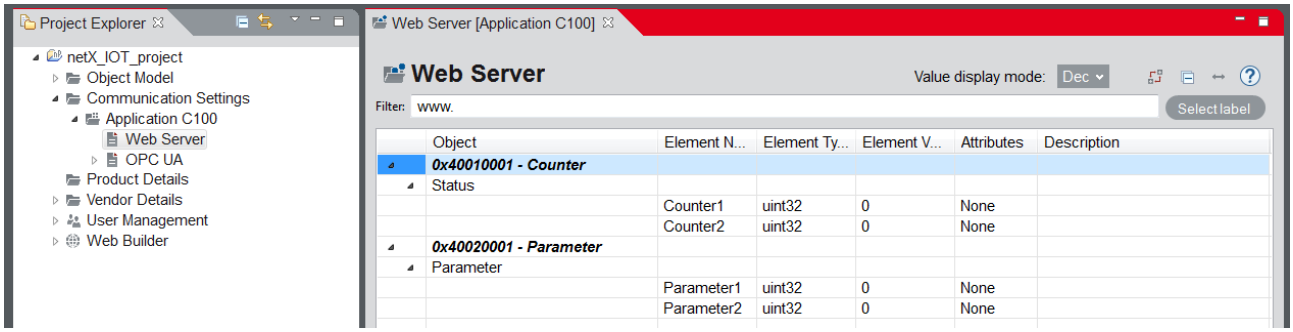


Figure 64: Communication Settings > example WebServer editor (default selection www)

For a description for the table in the **WebServer** editor, refer to the table *Parameters of the elements (Object Instances)* [page 60]. But the values for Element Value cannot be edited here.

Information on **Value display mode** and the icons **Toggle Expand** or **Collapse All** etc., see section Controls of the Application Variant editor

Note:

For objects with access via OPC UA or MQTT, you can also configure an access via the WebServer and display the object data configured for the OPC UA server or MQTT in addition at the WebServer. For this, the label "www" must be selected in the object configuration!

5.5 Configuring product details

allows to enter data for product identification under **Product Details**. The following main steps are necessary for this:

1. Preparation

➢ Write down the product details for your devices.

These product details include:

- Product information
  (product name, product ID, product type, order ID, profile ID, profile type) and
- Product version information
  (product revision, hardware name, hardware version, software name, software version, bootloader name, bootloader version).

2. Configuration in netX Studio Engineering Tool

➢ Create for each device a folder for the product details.

➢ Configure the product details data.

5.5.1 Creating folders for product details

➢ To create a folder for product details, right-click on the folder **Product Details**.

➢ Click **New Product**.

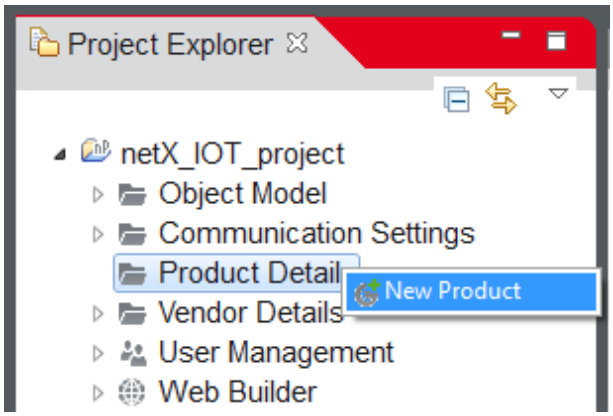


Figure 65: Product Details – New Product

↬ The dialog box **Add New Product** appears.

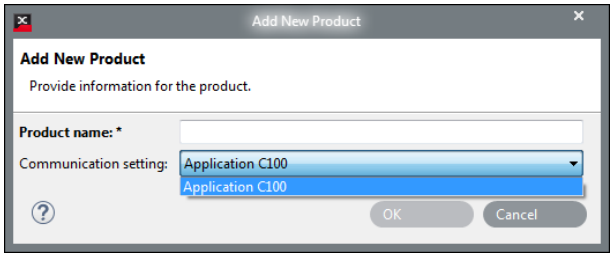


Figure 66: Enter the name and select the communication settings

➢ In the **Product name *** field type a short term for the product.

↬ The used product name appears under Product Details in the folder name for the product.

Note:

Always use the same product name for a product, so you can always recognize the data of this product!

Use the same product name also in the Product Information editor under **Product Type** in the field **Product name ***.

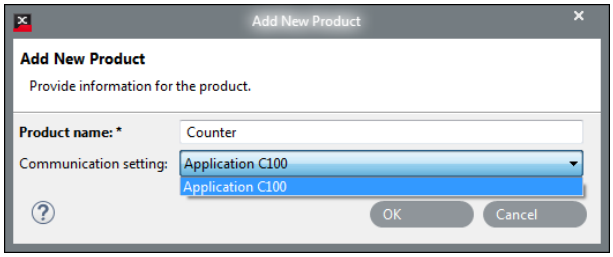


Figure 67: Enter the name and select the communication settings

➢ Under **Select Communication Settings** select the communication settings for the product.

➢ Click **OK**.

↬ The new product folder appears with the subfolders **Product Information** and **Product Macros**.

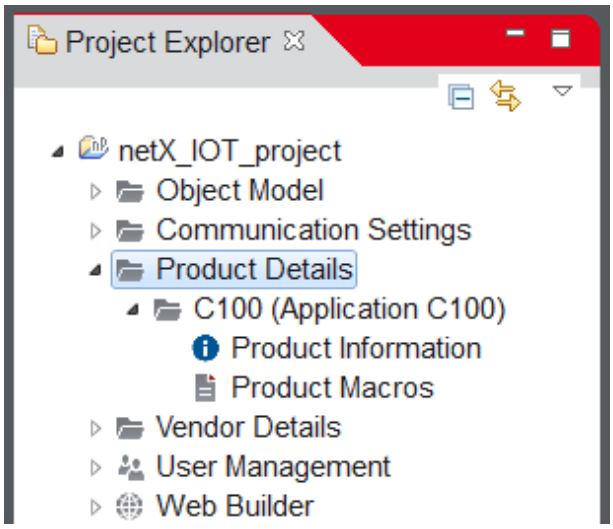


Figure 68: Product Details – example

For detail about the **Product Macros** folder, refer to the application note Web Builder [5].

➢ This way, create all your products.

You can rename the folder for product details or remove (see also *Renaming or removing product details* [page 78]).

5.5.1.1 Renaming or removing product details

Product details, you can rename or remove.

Rename product details

➢ Right-click on the folder **Product Details**> [*product name*].

➢ To rename your product, click **Rename**.

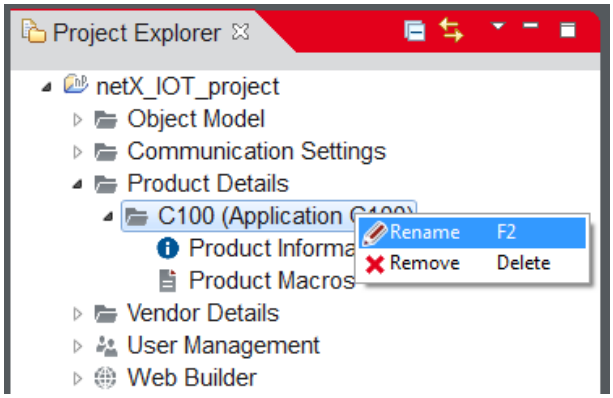


Figure 69: Product Details – renaming (example)

➢ Type the new product name in the dialog box **Rename Product**in the **Product name *** field.

↬ The changed product name appears under Product Details in the folder name for the product.

Note:

Always use the same product name for a product, so you can always recognize the data of this product!

Use the same product name also in the Product Information editor under **Product Type** in the field **Product name ***.

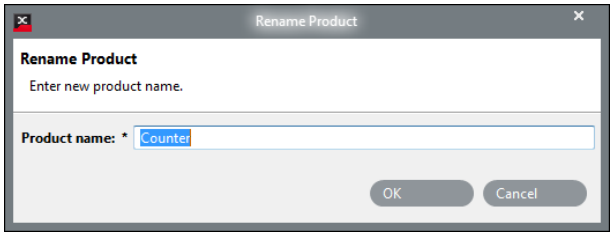


Figure 70: Rename Product

➢ Click **OK**.

⇨ The product folder appears with a new name.

Remove product details

➢ Right-click on the folder **Product Details** > [*product name*]

➢ To remove your product, click **Remove**.

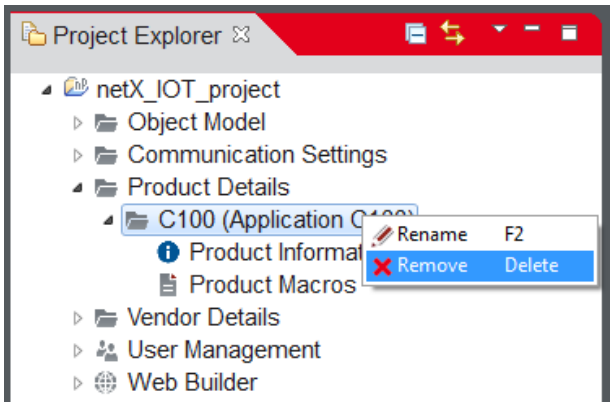


Figure 71: Product Details – removing (example)

⇨ The product is removed.

5.5.2 Entering product information

Note:
Entries in the Product Information editor affect in various standard objects.

➢

⇨ The **Product Information** editor (on the right side) appears.

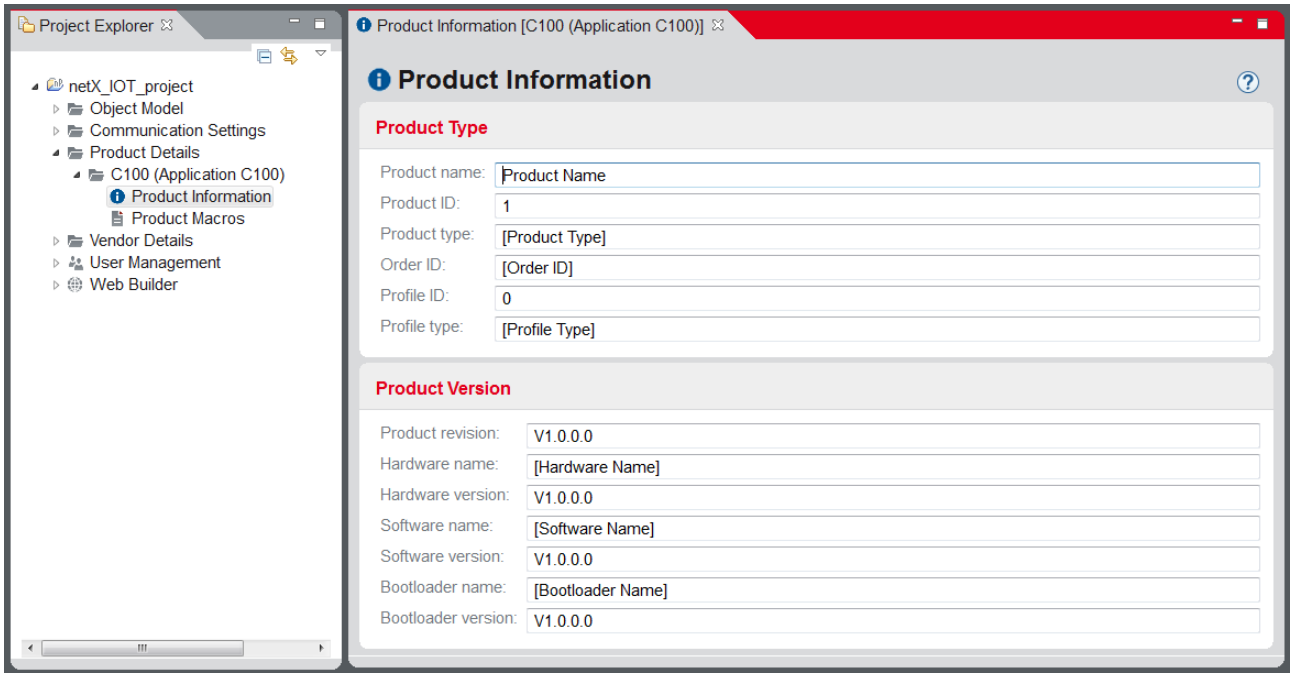


Figure 72:

➢ Select in the Product Information editor under Product Type the product type, or add under Product Version the product version details.

Note:
Always use the same product name for a product, so you can always recognize the data of this product!

The product name typed under **Product Type** in the **Product Name** field is used in the device description file. Use the same product name as the entry used in the dialog box **Add New Product** in the **Product name** field which appears under Product Details in the product folder name.

⇨ The product information is used in the device description file or has effect in different Standard objects. Via the Standard objects, the product information can be mapped to specific locations.

Example C100

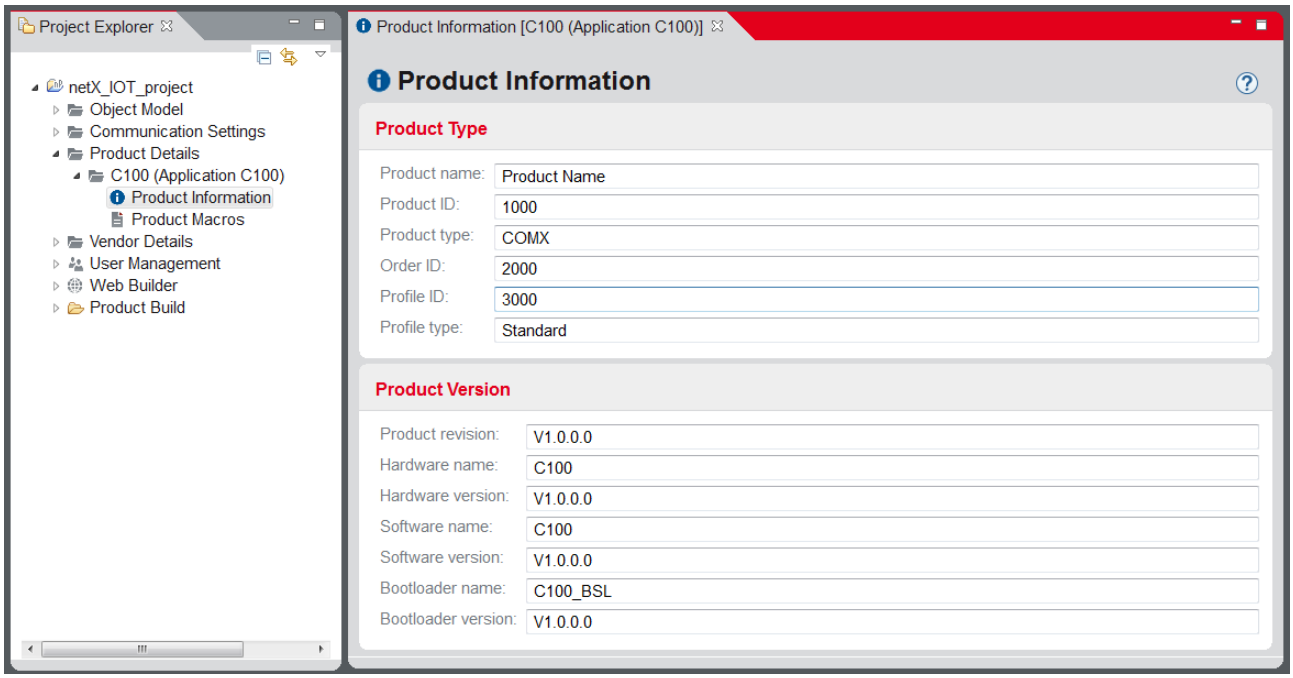


Figure 73:

5.6 Entering vendor information

Vendor Information

The vendor information is identical in the overall project and for all devices.

➢ To enter the vendor information, in the **Project Explorer** window, double-click on the folder **Vendor Details > Vendor Information**.

⇨ The Vendor Information editor (on the right side) appears.

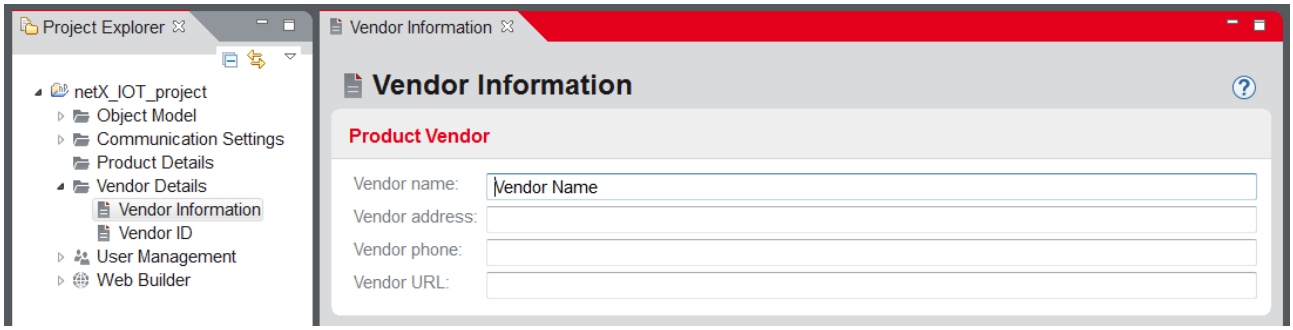


Figure 74: Vendor Details > Vendor Information.

In the table below, you can see, which entries in the Product Information editor are set in which reference object in in the **Project Explorer** under the **Object Model > Object Definitions > Standard > Generic** folder.

Herstellerinformation	Max. number of characters	Character type	netPROXY object			CIP object
Vendor name	64	Printable ASCII string; only the following characters are allowed: 32 (0x20) … 126 (0x7E)	0x10001000, Element 0	[Device] VendName	Hilscher Gesellschaft fuer Systemautomation mbH	-
Vendor address	128		0x10001000, Element 1	[Device] VendAddresse	Rheinstrasse 15, D-65795 Hattersheim	-
Vendor phone	64		0x10001000, Element 2	[Device] VendPhone	+49(0)6190/9907-0	-
Vendor URL	64		0x10001000, Element 3	[Device] VendURL	http:// www.hilscher.com	-

Table 22: Vendor Information (example)

To configure the vendor information, proceed as follows:

➢ Under **Vendor Name** (mandatory field), type the manufacturer's name.

➢ Under **Vendor address**, **Vendor phone** and **Vendor URL** enter the address, phone number and web address.

⇨ The data appear in the Vendor Information editor.

Vendor ID

The vendor ID you must specify separately for each communication system.

➢ To enter the vendor ID, in the **Project Explorer** window, right-click on the **Vendor Details > Vendor ID** folder.

↬ The Vendor Information editor appears.

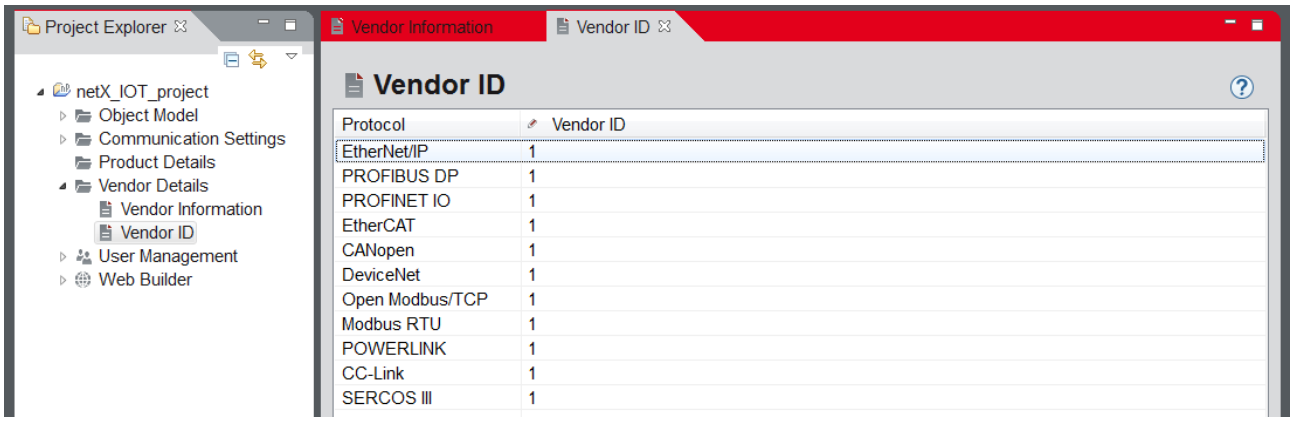


Figure 75: Vendor Details > Vendor ID (example with vendor identifiers of the Hilscher company)

Note:
You need your own vendor identifiers (ODVA)!

➢ In the **Vendor ID**column enter your vendor identifier for the communication systems used in your project.

5.7 Configuring user managment

Under **User Management** in you can complete the scope of functions of your device by user names and roles for the user management at the WebServer or the FTP server for your device. You can also view an overview of the distribution of access rights to your device for the WebServer or FTP server. The settings for user administration are stored during the build process in the configuration file security.cfg. The set names and roles for the roles Administrator, Supervisor and User can be used at the WebServer or the FTP server.

➢ To create an account, in the **Project Explorer** window double-click on the **User Management > User Accounts** folder.

↬ The User Accounts editor appears.

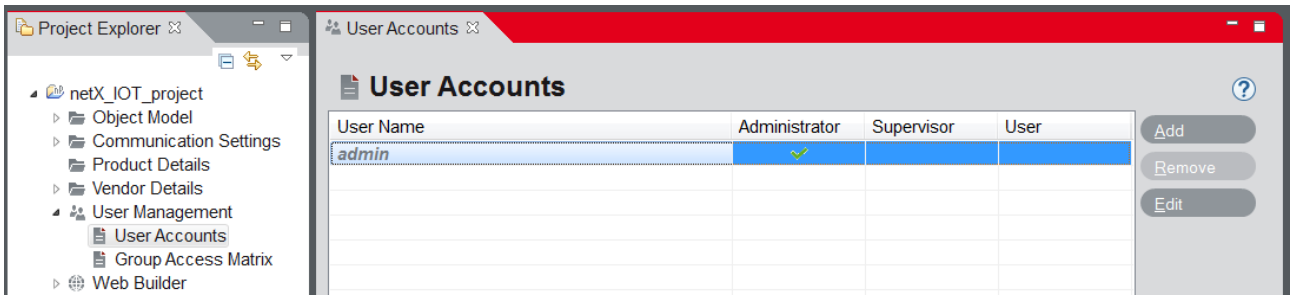


Figure 76: User Management - User Accounts editor

Action	Meaning
Add	Via **Add** you can add a new user account.
Remove	Via **Remove** you can remove a user account.
Edit	Via **Edit** you can edit a user account.

Table 23: Control of the User Accounts editor

User account for default administrator

The user account for the administrator is pre-configured by default. This account can not be deleted. The administrator password can be individually configured.

Note:
Change the default password for the administrator.

➢ Click **Edit**.

➢ The User Accounts dialog appears.

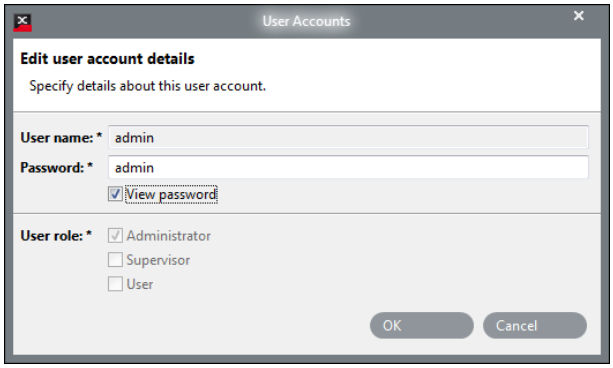


Figure 77: User account Admin edited

➢ Check **View password***, to see the password.

➢ Adapt the password in the **Password*** field and leave the dialog with **OK**.

Create user accounts

You can create user accounts for the roles Administrator, Supervisor and User. In addition to the default user account "admin" you can create additional user accounts for additional administrators.

➢ Click **Add**.

⇘ The User Accounts dialog appears.

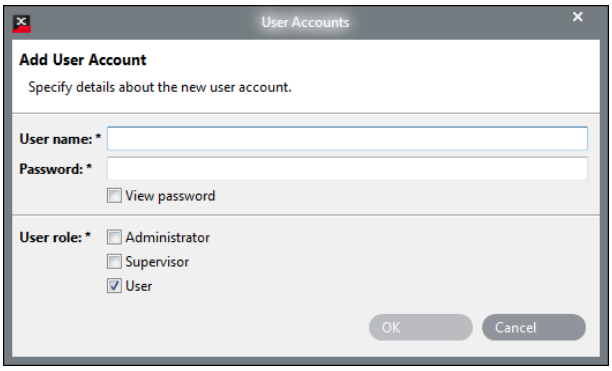


Figure 78: User Accounts – add details to the new user account

Parameter	Meaning	Range of value/value
User name*	User name (mandatory)	String
Password*	Passwort for the user (mandatory)	String of uppercase and lowercase letters, numbers and special characters
User role*	Role of the user (mandatory) A user can have only one role or several roles in combination.	Administrator, Supervisor, User Default: User

Table 24: Details to the new user account

➢ In the **User name*** field, enter a username.

➢ In the **Password*** field, enter a password for the user.

➢ Under **User role*** check, which role or which roles the user shall have.

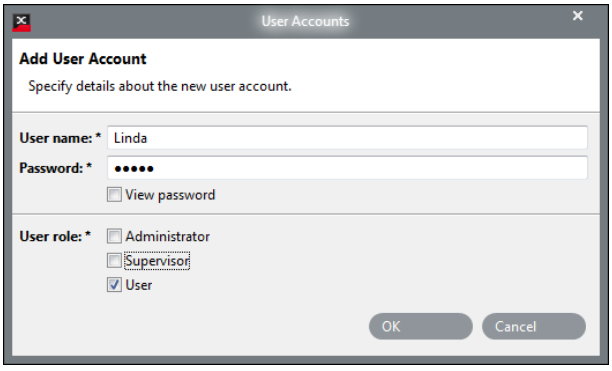


Figure 79: Create a new user account (example)

➢ Exit the user account dialog every time via **OK**.

⇨ In the User Accounts editor, the names and details of the specified roles appear.

Group Access Matrix

➢ To open the overview of the distribution of the access rights for the WebServer or FTP server, in the **Project Explorer** window, double-click on the folder **User Management > Group Access Matrix**.

⇨ The Group Access Matrix editor appears. Therin you see an overview of the roles and rights at the WebServer and the FTP server.

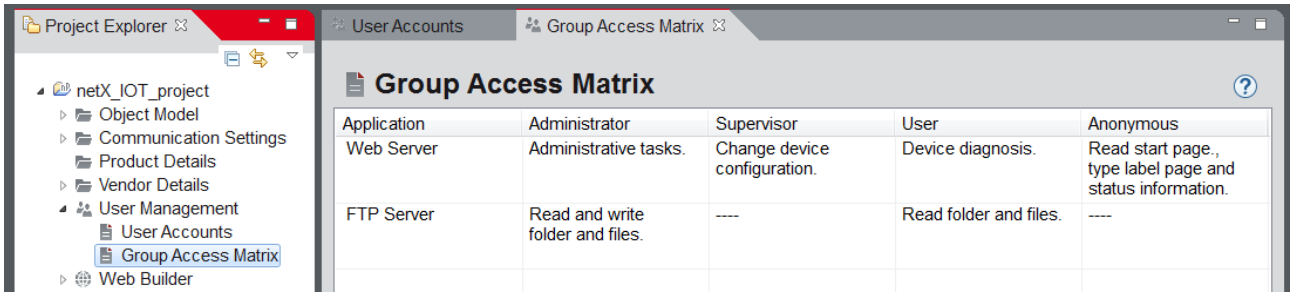


Figure 80: Overview of the distribution of the access rights for the WebServer or FTP server

5.8 Creating and managing web content

For information about creation and management of web content, refer to the application note, **netX Studio Engineering Tool**, Web Builder [6].

5.9 Generating product files

In the build process the product files are created, including the configuration file, or the files for the object configuration, plus web content files and the header files. The product files include the configuration (definition of objects) and any additional settings (scope of functions, communication, product details, vendor information).

A firmware image file can be created via the export function of . See section Creating firmware image. The firmware image file contains the individual files from the build process, without the header files, but in addition the second-stage boot loader (SSBL).

5.9.1 Build process

Start build process

After you have finished the object definition and added settings and data for your products, you can start the build process.

➢ For this select in **Project** > **Build**.

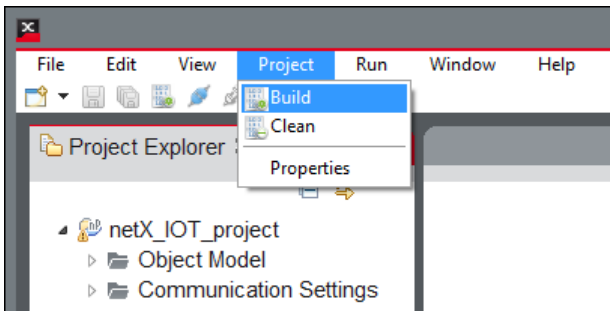


Figure 81: Start device creation process (build process)

➢ For seconds, a dialog appears which displays the progress.

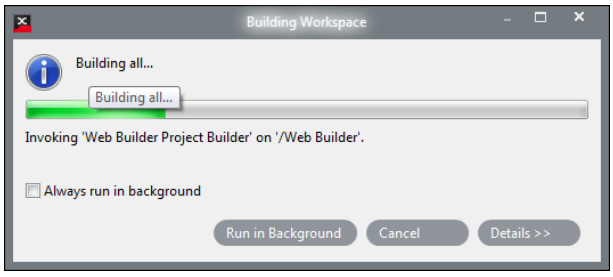


Figure 82: Running build process

⇨ The **Product Build** folder with the subfolders **Application Interface** and **Products** appears in the Project Explorer.

Directory Product Build with the generated product files

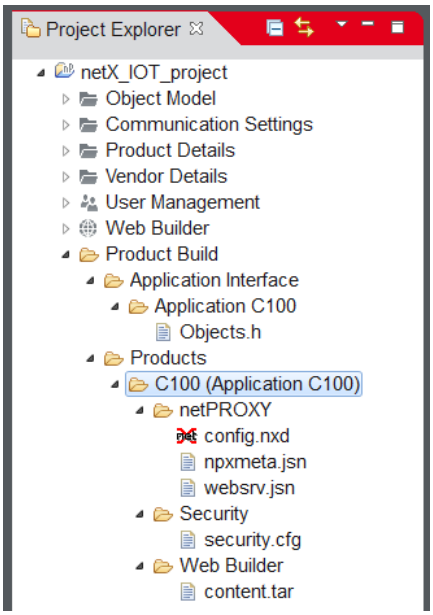


Figure 83: Directory Product Build

Folder	File name	Description
Application Interface subfolder		
Header file	Objects.h	Definition of user-specific objects for C programming
Products subfolder		
netPROXY	config.nxd	File for netPROXY object configuration (objects, OPC UA server parameters or MQTT client parameters)
	npxmeta.jsn	File with meta information
	npxopcu.jsn	File with object parameters for OPC UA
	websrv.jsn	File with WebServer object list
Security	security.cfg	Configuration file with content for user management on the FTP server respectively on the WebServer
Web Builder	content.tar	Pack file with web content for the WebServer (optional)

Table 25: Product files in the build process of netX Studio Engineering Tool

- The **Application Interface** folder contains a folder for each application variant, in which the header file **Objects.h** is located. Double-click the file to open it.

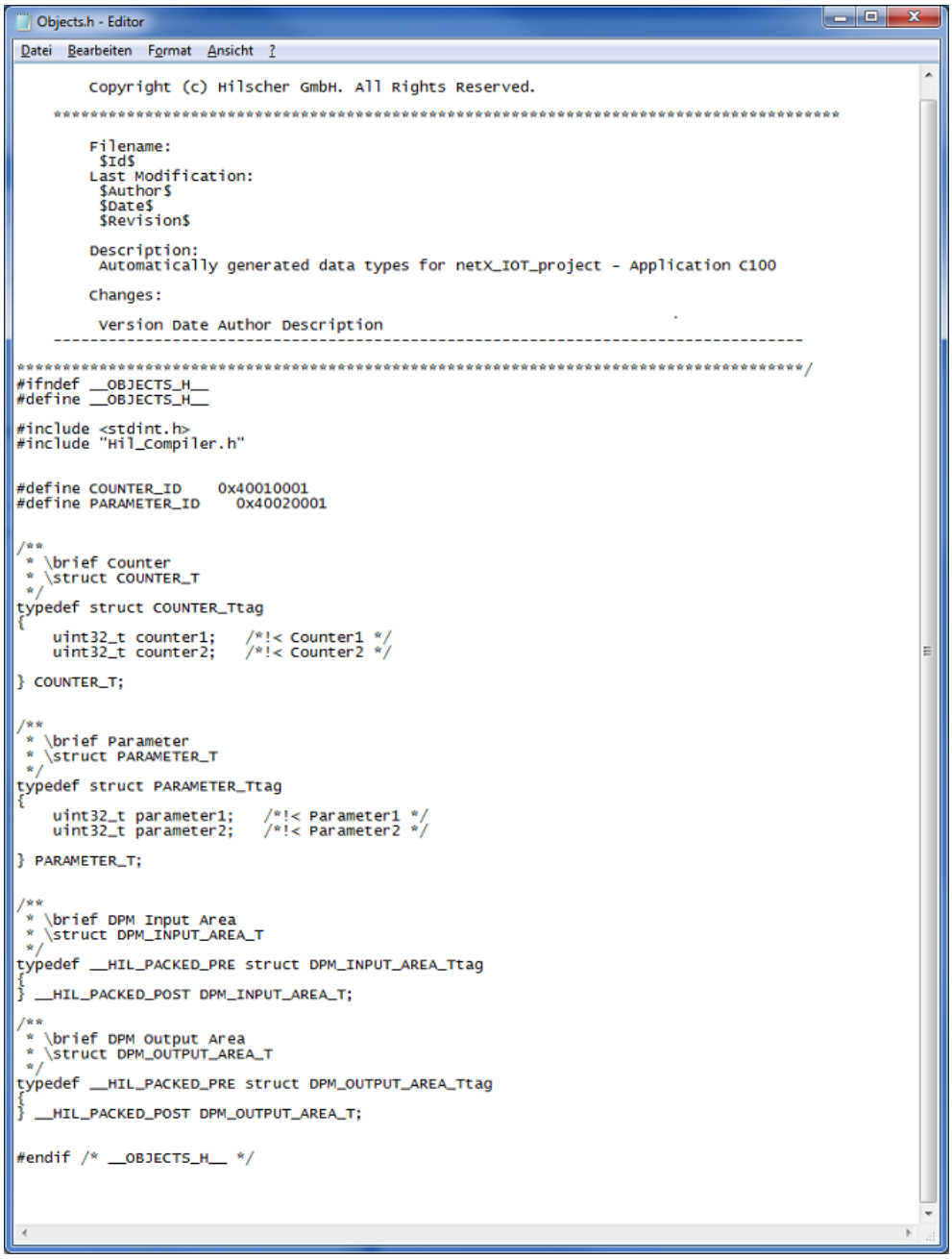


Figure 84: Example, Header file Objects.h

Note:
The header file **objects.h** contains only objects from the Object ID range 0x4xxxxxxx (see section *Object IDs* [page 107]).

- Below the folder **Products** there is a subfolder for each device with a name of the application variant with the further subfolders **netPROXY**, **Security** and **Web Builder**.
  - The subfolder **netPROXY** contains the file **config.nxd** with the object configuration and the communication settings for the corresponding device, and the file **npxmeta.jsn** with meta information, as well as the file **websrv.jsn** with the WebServer object list.
  - The subfolder Security contains the file security.cfg with settings for the user management.

– The subfolder Web Builder contains the file content.tar with content for the WebServer for the corresponding device.

Clean - eliminate build process problems

Using the Clean option, you can eliminate build process problems and rebuild a project build from scratch.

➢ Click **Project** > **Clean**.

↬ The **Clean** dialog appears.

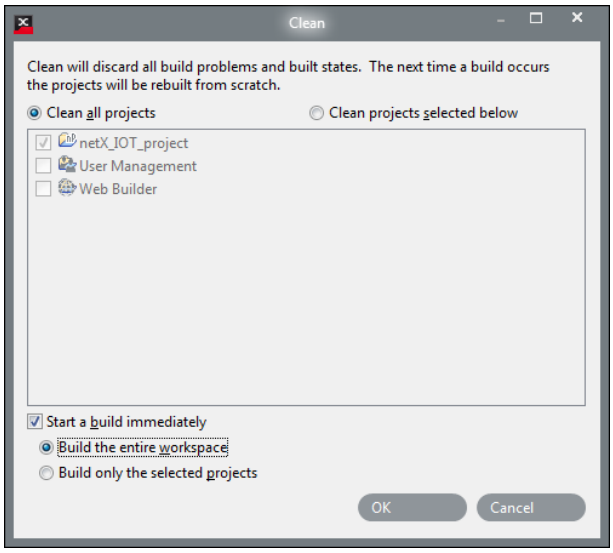


Figure 85: Project > Clean

- In the Build dialog, you can configure whether all projects should be cleaned (Clean all projects) or only certain projects which you have selected (Clean projects selected below).
- In addition, you can configure whether to start a new build process immediately (Start a build immediatelly) and, if so, whether the build process shall be performed for all projects (Build the entire workspace) or only for selected projects (Build only the selected projects).

5.9.2 Copy, remove product files

Open the Windows directory with product files

➢ Right click in on **Product Build**.

➢ The context menu **Show in Windows Explorer** appears.

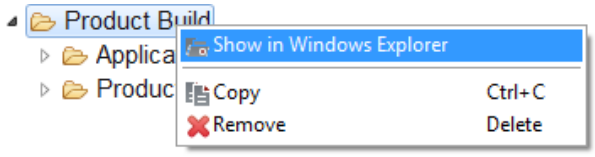


Figure 86: Context menu Product Build > Show in Windows Explorer, Copy, Remove

➢ Click on **Show in Windows Explorer**.

➢ The Windows Explorer with the **Product Build** folder opens. The product files can be found in the subdirectories Application Interface and Products.

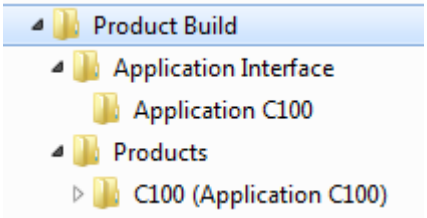


Figure 87: Folder Product Build, subdirectories Application Interface and Products

Copy, remove product files

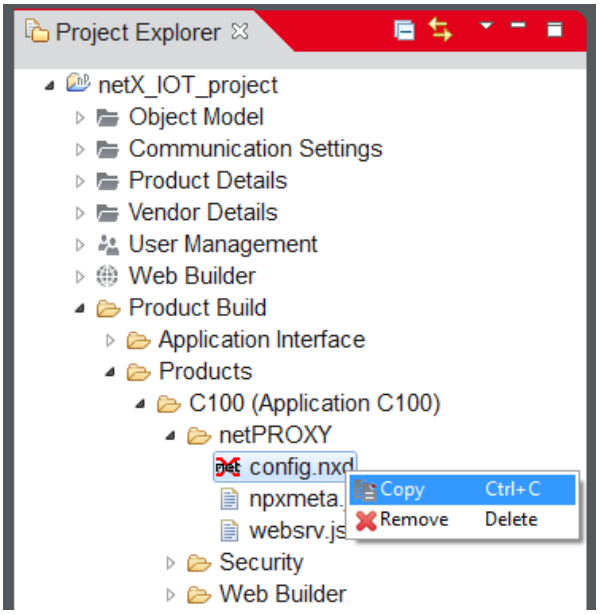


Figure 88: Context menu commands Copy, Remove

➢ Using**Copy**, you can copy a subdirectory or a file from the Product Build directory and place it in Windows Explorer.

➢ Use **Remove** to remove a subdirectory or file from the Product Build directory.

5.9.3 Meaning of the product files

The product files created in the build process are individual files, wherein the object definition and the device configuration and all settings for the device are included.

File	File name	Description
Header file	Objects.h	Contains information for application developers on which objects are present in the device and how to access the objects.
net PROXY object configuration	config.nxd	This includes the configuration information for the object definition with the definition of the device properties and the specifications on the object instances for the scope of functions of the devices, and the communication settings.
	npxmeta.jsn	File with meta information
	npxopcu.jsn	File with object parameters for OPC UA
	websrv.jsn	File with WebServer object list
User management	Security.cfg	Configuration file with content for user management on the FTP server respectively on the WebServer
Web content (optional)	Content.tar	Pack file with web content for the WebServer (including corporate logos, counters, etc.); finds application in Ethernet systems

Table 26: Product files of the build process

5.10 Downloading product files

All files created during the product build under **Product (Application)** (see table *Product files in the build process of netX Studio Engineering Tool* [page 89]) must be transferred to the device. Each file must be downloaded individually.

Note:
During a product build, not all the files listed in the table *Product files in the build process of netX Studio Engineering Tool* [page 89] are created in every case.

To transfer the files to the device, proceed as follows:

- Right-click on **Product Build** and select **Show in Windows Explorer**.
- Using netHOST.exe or the host application, transfer all product files created during the product build under **Product Build > Products > Product (Application)** to the device one by one.

Details about the netHOST user interface or the host application can be found in the corresponding software manual.

Important:
After all product files created during product build under **Product Build > Products > Product (Application)** have been loaded into the device, a device reset is required.

- After downloading the product files, perform a device reset.

⇨ Your device is then prepared. This means that the device can be accessed with OPC UA, MQTT or the web browser.

6 Import, Export

includes import and export functions which you can use for various purposes. The following sections provide more details about.

You can call the dialogs on the import and export functions at various positions in the user interface.

- General on **File > Import ...** or **File > Export ...**
- In the **Project Explorer** window via a right-click on the project folder
- After executing the **Build** function by right-clicking on one of the product files (objects.h, config.dat, security.cfg etc.) which are located in subdirectories of the **Product Build** folder.

6.1 Import

About the import function in you can import the from an existing project exported object definitions to one or several new projects.

The descriptions on how to proceed you will find here below.

Note:
Before importing, it is recommended that you save an existing object definition by exporting it (see section *Export* [page 100]) to avoid accidental overwriting of object definitions.

Open the import dialog and specify import content

➢ In the **Project Explorer** window, right-click on the project folder.

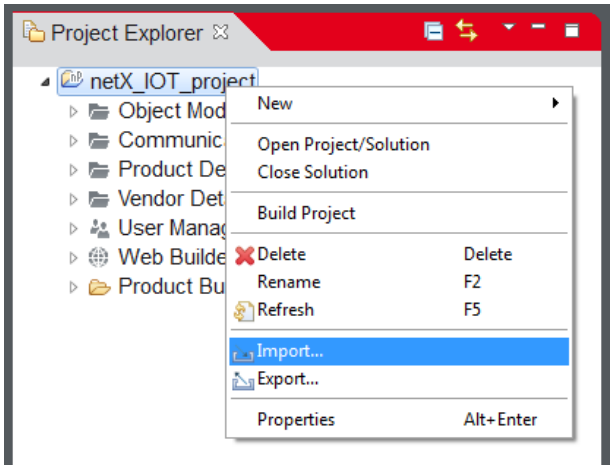


Figure 89: Call import function (example)

➢ Click **Import …** .

➢ The **Import**(**Select**) window appears.

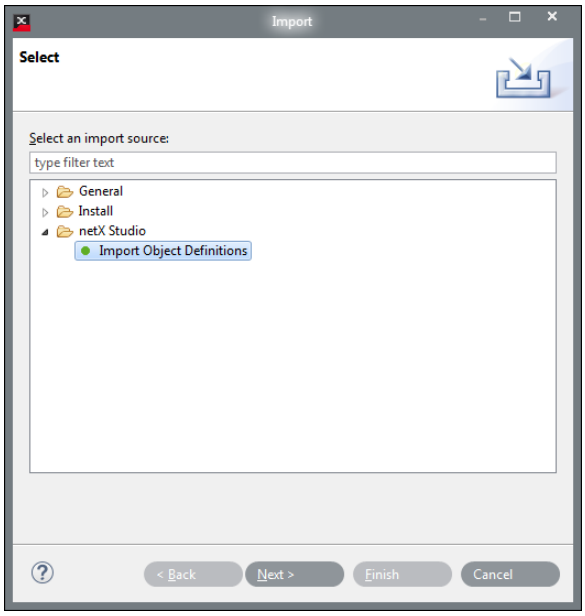


Figure 90: Specify import content (example)

Parameter	Meaning	Range of value / value
Select an import source	A display window with a folder structure for selecting the import content. Relevant is the folder netX Studio. The other folders are not relevant. In netIX Studio folder, you can specify the import content: • Import Object Definitions (object definitions)	General, Install, netX Studio Default: folder netX Studio
Filter [type filter text]	Field for typing a filter text to select a subfolder.	-

Table 27: Notes to the Import (Select) window

➢ Select the folder **netX Studio**.

➢ Select **Import Object Definitions** and click **Next >**.

➢ The **Object Definitions import** window appears.

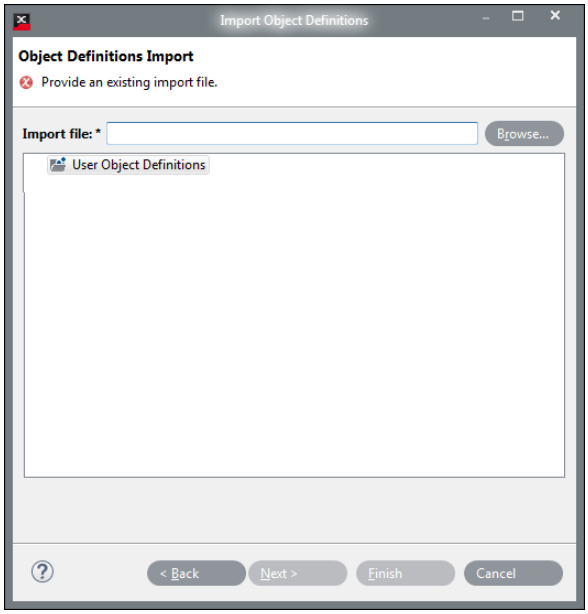


Figure 91: Select import file

Parameter	Meaning	Range of value / value
Import file (mandatory)	Displays after selecting the import file the path to the selected file with the object definitions inclusively the name of the zip file.	
Browse...	Via **Browse ...**, the import file can be selected with the object definitions *.zip (e. g., User.zip).	*.zip, e. g. User.zip
User Object Definitions	If only the folder 'UserObject Definitions' appears, an export file with any object categories and object definitions can be imported. If other folders of object categories appear under the 'User Object Definitions' folder, and you select one of these folders, you must pay attention to warnings so that you do not inadvertently overwrite object definitions.	Folder 'User Object Definitions', or folders of subordinate object categories
Errors (optional)	If you try to replace a folder of an object category with an import file with object definitions, some of the object definitions to be imported already exist in another object category of the project, the error message appears: Can not import object with ID 0x4XXXXXXX because such an object exists outside of the destination category. In this case, select another folder, for example 'User Object Definitions', or import only object definitions that occur in the project not yet outside the target category.	

Table 28: Note to the window Object Definitions import

➢ Click **Browse…**.

➢ The **Open** window appears.

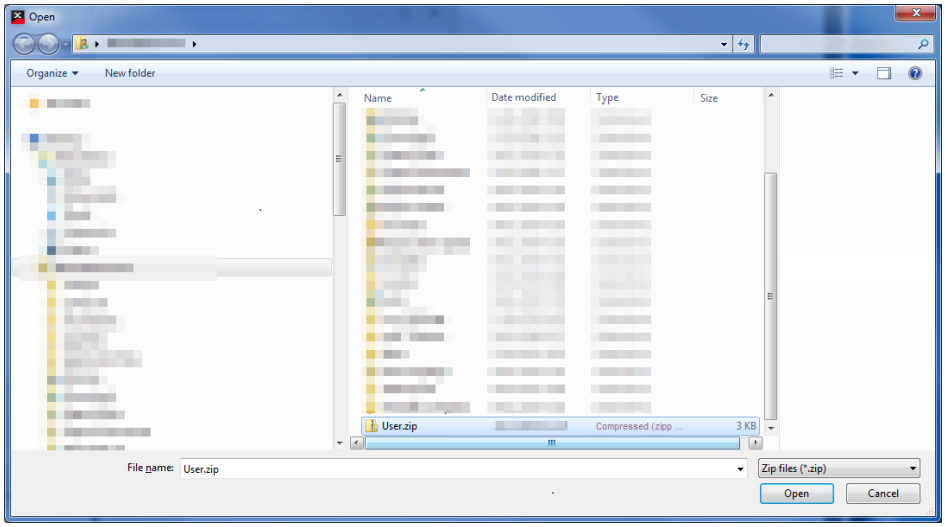


Figure 92: File selection dialog for import file (example User.zip)

➢ Select the import file with the object definitions (e.g. User.zip) and click **Open**.

➢ The path to the selected file with the object definitions (e. g. User.zip) is displayed in the window **Import Object Definitions**.

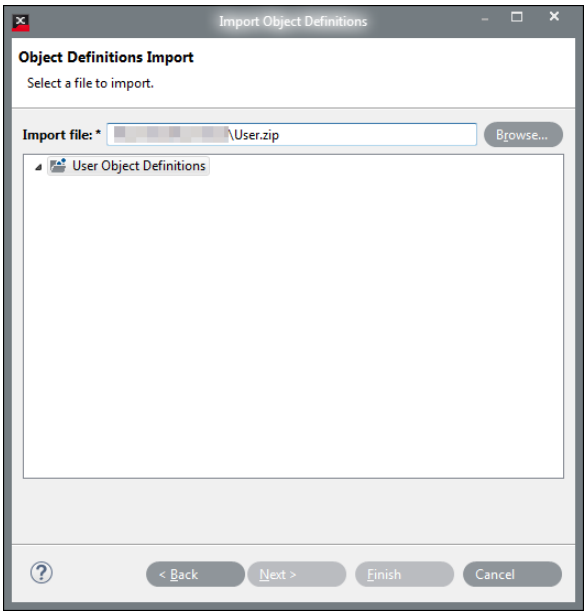


Figure 93: Import file selected (example User.zip)

➢ In the **Import Object Definitions** window click **Finish**.

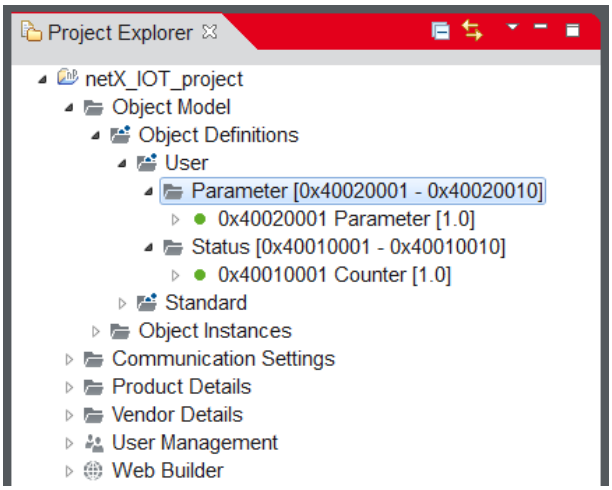


Figure 94: Newly imported object definitions (example)

Warning: Object category will be overwritten

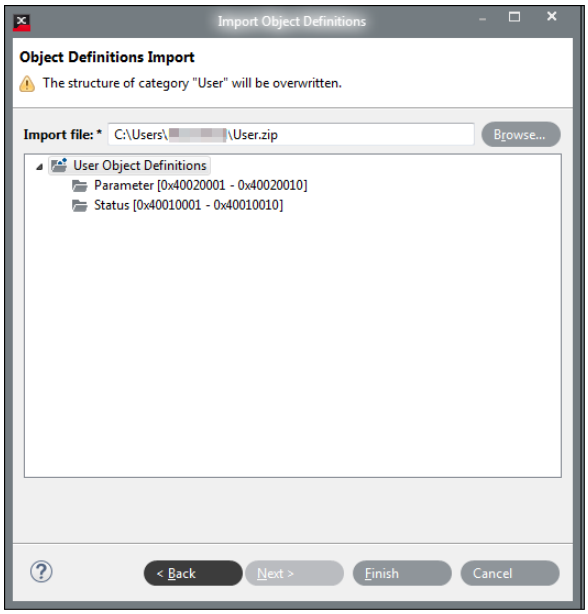


Figure 95: Import file selected (example User.zip) - Warning, category "User" will be overwritten

6.2 Export

Using the export function in you can

- export the object definitions of an existing project (in order to be able to import them subsequently to one or several new projects)
- the project documentation of a project.

The descriptions on the procedure in detail, you find in the following sections.

6.2.1 Exporting object definitions

In the dialog for exporting the object definitions (Export Object Definitions), you specify the output folder for the file containing the object definitions. Proceed as follows:

Openl Export dialog and define export content

➢ In the **Project Explorer** window, right-click on the project folder.

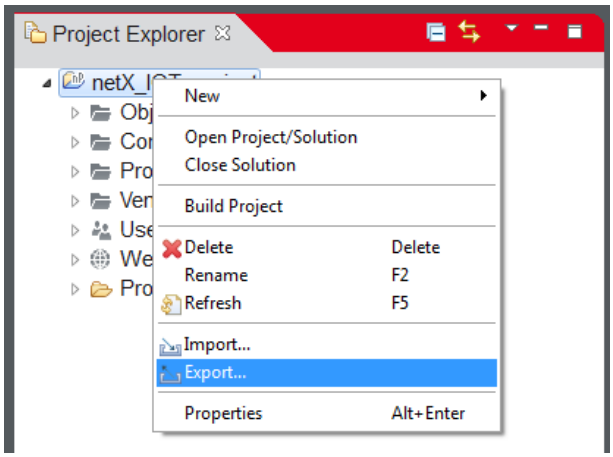


Figure 96: Call Export function (example)

➢ Click **Export ….**

↬ The **Export** window (**Select**) appears.

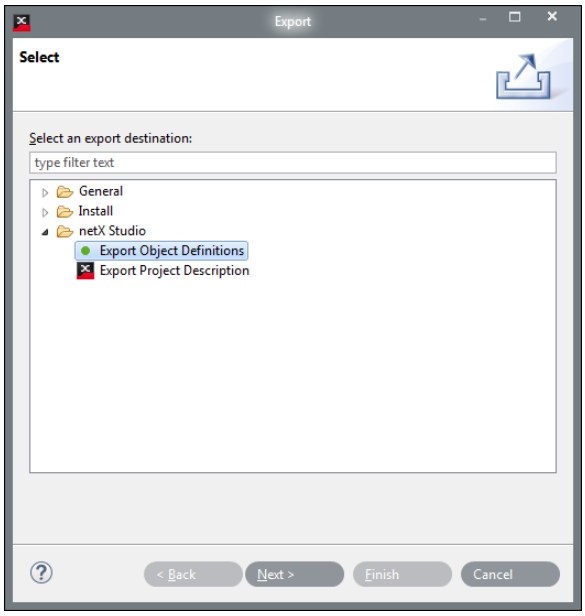


Figure 97: Define export content - Export Object Definitions

Parameter	Meaning	Range of value / value
Select an export destination	Display window with a folder structure for selecting the export content. Relevant is the folder netX Studio. The other folders are not relevant. In netX Studio folder, you can specify the export content: • Export Object Definitions (object definitions), • Export Project Description (project documentation)	General, Install, netX Studio Default: folder netX Studio
Filter [type filter text]	Field for typing a filter text to select an export function.	-

Table 29: Notes to the Export (Select) window

➢ Select the folder **netX Studio**.

➢ Select **Export Object Definitions** and click **Next >**.

↬ The **Export Object Definitions** window appears.

Set output folder for the file containing the object definitions

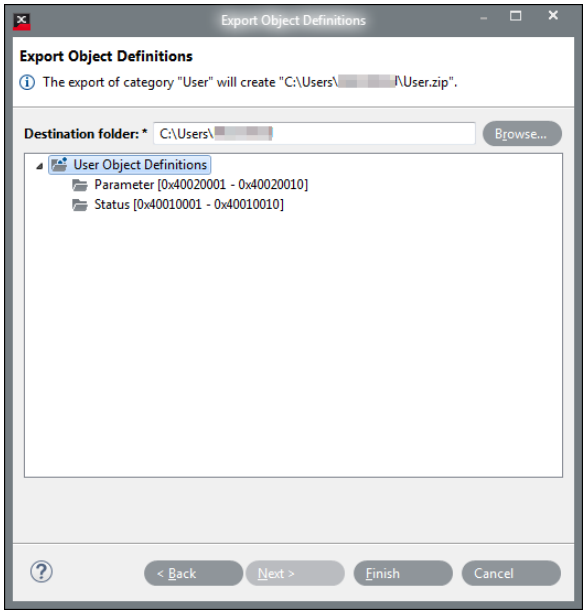


Figure 98: Set output folder

Parameter	Meaning	Range of value / value
Export Object Definitions	Instruction texts: Provide an existing destination folder. The export of category „XXX“ will create [*path and export file name*].	
Destination folder* (mandatory)	Shows after the selection the path to the current output folder for the export file with the object definitions.	
Browse...	Via **Browse ...**, an output folder for the export file with the object definitions can be changed or can be created newly (by default on your PC).	C:\User\[*user name*]

Table 30: Notes to the window Export Object Definitions

➢ Via **Browse ...** you can search for the output folder or create a new output folder.

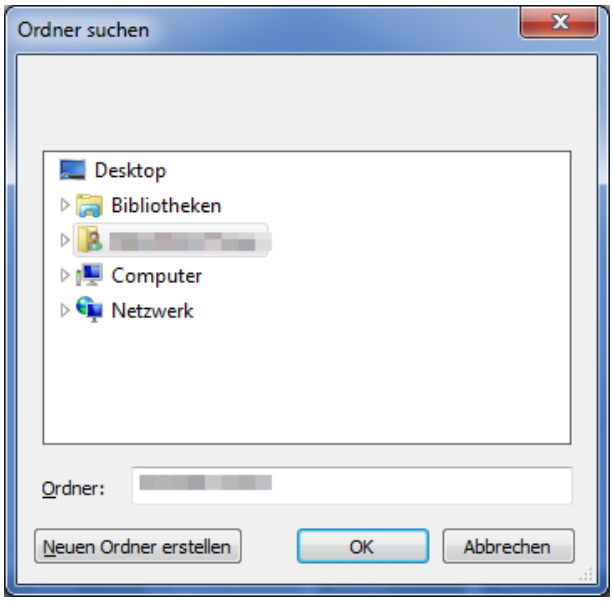


Figure 99: Search folder

- ➢ Select the path and click **OK**.
- ➢ In the **Export Object Definitions** window, click **Finish**.
- ⇨ The file with the object definitions (e. g. User.zip) is stored in the selected storage location. In case the same storage location is used more than once, *no* query appears if you want to overwrite the file with the object definitions.

6.2.2 Exporting project descriptions

In the dialog for exporting the project descriptions (Export Project Description), you specify the content for the project documentation and the output folder. Proceed as follows:

Open Export dialog and define export content

- ➢ In the **Project Explorer** window, right-click on the project folder.

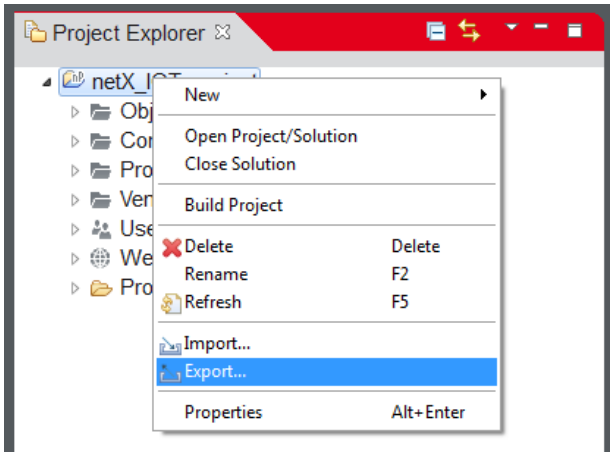


Figure 100: Call Export function (example)

- ➢ **Click Export ….**
- ⇨ The **Export** window (**Select**) appears.

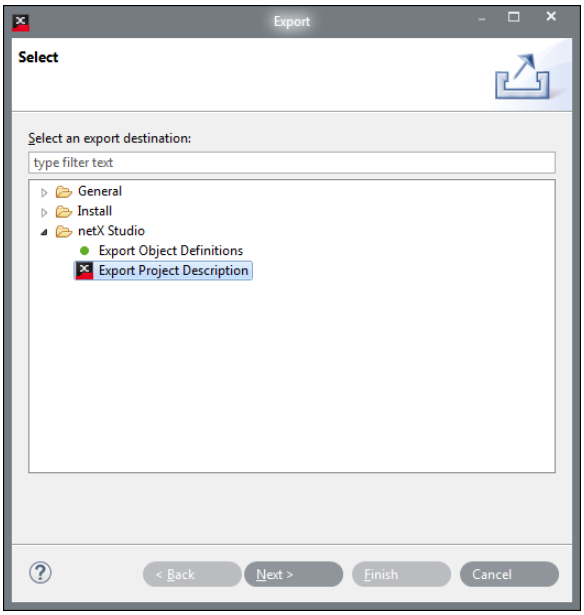


Figure 101: Define export content - Export Project Description

Parameter	Meaning	Range of value / value
Select an export destination	Display window with a folder structure for selecting the export content. Relevant is the folder netX Studio. The other folders are not relevant. In netX Studio folder, you can specify the export content: • Export Object Definitions (object definitions), • Export Project Description (project documentation)	General, Install, netX Studio Default: folder netX Studio
Filter [type filter text]	Field for typing a filter text to select a subfolder.	-

Table 31: Notes to the Export (Select) window

➢ Select the folder **netX Studio**.

➢ Select **Export Object Definitions** and click **Next >**.

↬ The **Export Project Description (Select products to export and destination folder)** window appears.

Specify content for the project documentation

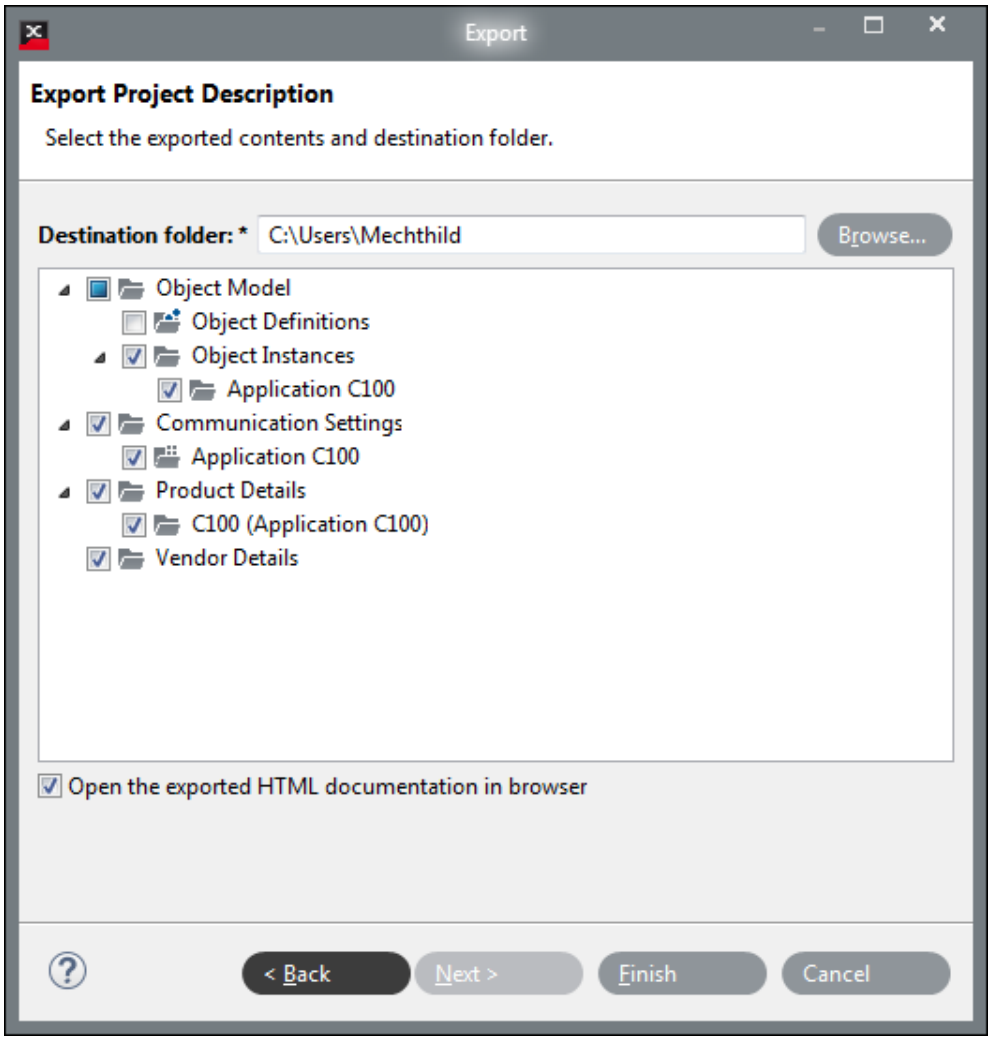


Figure 102: Specify content for the project documentation and output folder

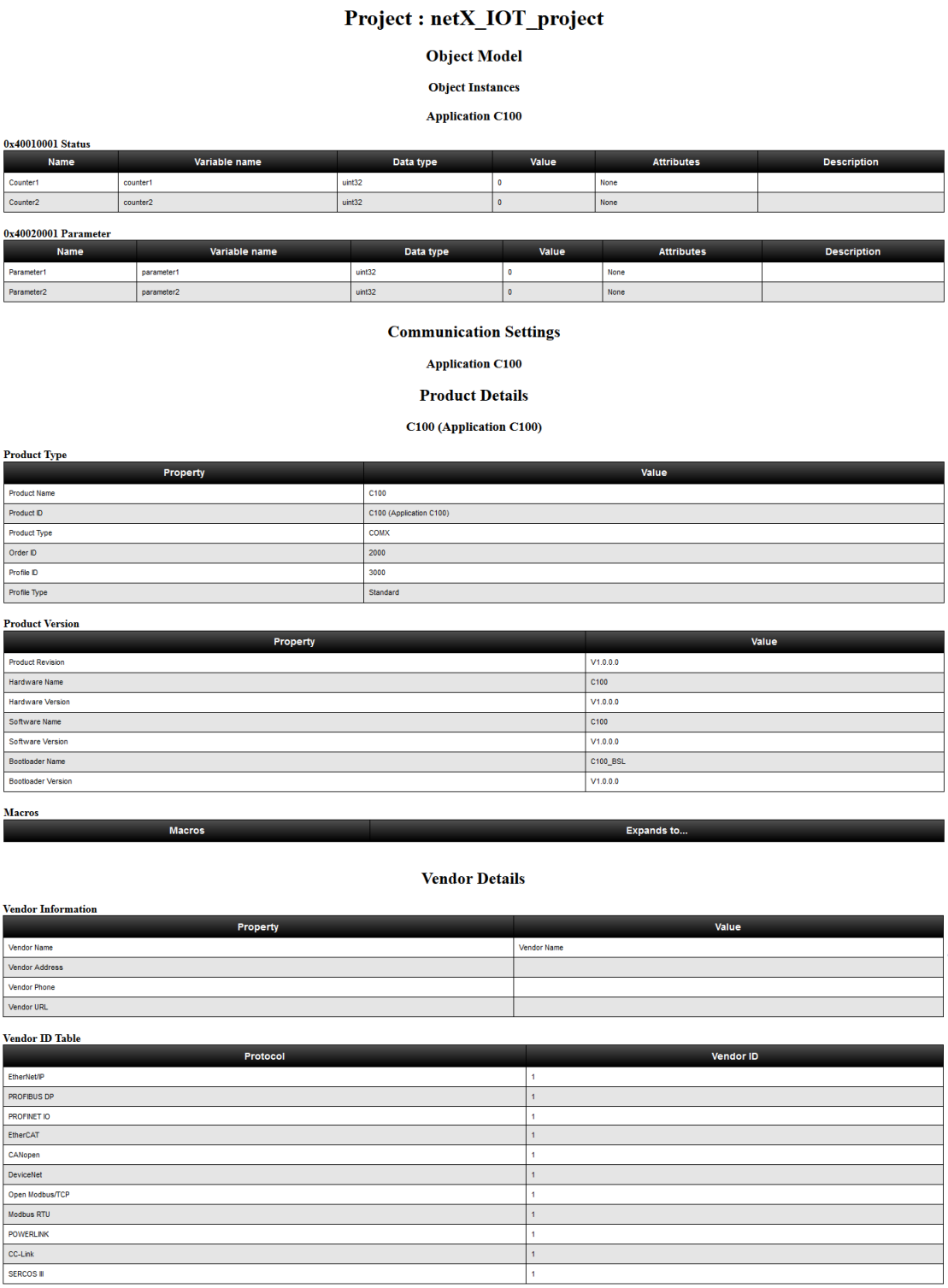
Parameter	Meaning	Range of value / value
File system	Instruction texts: Select projects to export and destination folder.	
Destination Folder* (mandatory)	Displays the path to the current output folder for the file with the project documentation. The file name [*Project name*].html (in the example series B.html) and cannot be adapted here.	
Browse...	Via **Browse ...**, an output folder for the file with the project documentation can be selected or can be created newly (by default on your PC).	C:\User\[*user name*]
Selection dialog for the project content	The selection dialog includes the total project content, • to the Object Model, • to the Communication Settings, • to the Product Details, • to the Vendor Details and By default, except the object definitions, but these can be selected manually.	
Open the exported HTML documentation in browser	The exported project documentation will be displayed directly in a browser after each export.	Checked, unchecked, Default: checked

Table 32: Notes to the window Export Project Description

➢ Click **Finish**.

⇨ The project documentation appears in your browser.

The project documentation includes all in the selection dialog selected project content. By scrolling in the browser, you can view all content.



Project : netX_IOT_project

Object Model

Object Instances

Application C100

0x40010001 Status

Name	Variable name	Data type	Value	Attributes	Description
Counter1	counter1	uint32	0	None	
Counter2	counter2	uint32	0	None	

0x40020001 Parameter

Name	Variable name	Data type	Value	Attributes	Description
Parameter1	parameter1	uint32	0	None	
Parameter2	parameter2	uint32	0	None	

Communication Settings

Application C100

Product Details

C100 (Application C100)

Product Type

Property	Value
Product Name	C100
Product ID	C100 (Application C100)
Product Type	COMX
Order ID	2000
Profile ID	3000
Profile Type	Standard

Product Version

Property	Value
Product Revision	V1.0.0.0
Hardware Name	C100
Hardware Version	V1.0.0.0
Software Name	C100
Software Version	V1.0.0.0
Bootloader Name	C100_BSL
Bootloader Version	V1.0.0.0

Macros

Macros	Expands to...

Vendor Details

Vendor Information

Property	Value
Vendor Name	Vendor Name
Vendor Address	
Vendor Phone	
Vendor URL	

Vendor ID Table

Protocol	Vendor ID
EtherNet/IP	1
PROFIBUS DP	1
PROFINET IO	1
EtherCAT	1
CANopen	1
DeviceNet	1
Open Modbus/TCP	1
Modbus RTU	1
POWERLINK	1
CC-Link	1
SERCOS III	1

Figure 103: Detail product documentation in HTML

7 Appendix

7.1 Reference data for netPROXY

7.1.1 Object IDs

The following summary table lists the defined value ranges for objects, which play a role in the use of the object-oriented data structuring with netPROXY with the device development tool and must be distinguished or which Object IDs are used by in the ranges 0 to 3.

From the user in created objects are always in object ID area. 4

Range of value object ID	Name of the range	Note
0x0xxx xxxx (without 0x0000 0000)	netPROXY	e. g. directory, event log etc. in
0x1xxx xxxx	System	e. g. general services. These include among others objects that have been created in under **Product Details**.
0x2xxx xxxx	Communication protocol	Object ID range that is reserved for objects of the communication protocol.
0x3xxx xxxx	Hilscher	Objects created by Hilscher, e. g. Host interface, peripherals, etc.
0x4xxx xxxx	User	User-created objects in . These are included in the objects.h file.
0x5xxx xxxx … 0xFxxx xxxx	Reserved	

Table 33: Fixed ranges of value for object IDs

For further information about the assignment of the Object ID ranges listed in the table *Fixed ranges of value for object IDs* [page 107], refer to the manual **Technical reference, netPROXY, Objects and packages** [2] (see section References).

7.1.2 Element data type, size, number, limits

The following table *Object element data types, size, number, limits* [page 108] lists the for

- the in configurable elements valid element data types,
- the related netPROXY data types,
- the size for each data type in bytes
- the maximum number of data types per element for each data type and
- the minimum and the maximum number of viewable values for each data type.

The default value for all numeric data types is 0.

The default length for all numeric data types is 1 (0 is not allowed).

A "string" has an empty string as default value. The default length "string" is 255 bytes.

netX Studio Element data types	netPROXY data types	Size in bytes	Max. number of data types = Max. Default value	Minimum number of viewable values	Maximum number of viewable values
boolean	NPX_TYPE_BOOLEAN	1	1	0	255
uint8	NPX_TYPE_UNSIGNED	1	1	0	255
uint16	NPX_TYPE_UNSIGNED	2	1	0	65535
uint32	NPX_TYPE_UNSIGNED	4	1	0	4294967295
uint64	NPX_TYPE_UNSIGNED	8	1	0	1,84467E+19
int8	NPX_TYPE_INTEGER	1	1	-128	127
int16	NPX_TYPE_INTEGER	2	1	-32768	32767
int32	NPX_TYPE_INTEGER	4	1	-2,147,483,648	2,147,483,647
int64	NPX_TYPE_INTEGER	8	1	-9,223,372,036,854,775,808	9,223,372,036,854,775,807
real32	NPX_TYPE_REAL	4	1	-3.4E+38 (7 digits)	+3.4E+38 (7 digits)
real64	NPX_TYPE_REAL	8	1	-1.8E+308 (15 digits)	1.8E+308 (15 digits)
string	NPX_TYPE_STRING	varies	1540	-	-

Table 34: Object element data types, size, number, limits

7.2 References

1] Technical Reference, netPROXY, Objects and Packages, DOC160204TRXXDE, netPROXY - Objects and Packages TR XX EN.pdf

[2] Protocol API, netPROXY, Function Interface, DOC160205APIXXEN, netPROXY Function interface API XX EN.pdf

[3] Operating instruction manual, netX Studio Engineering Tool (V1.400), IOT configuration, DOC180404OIXXDE, netX Studio Engineering Tool OI IOT XX DE.pdf (this manual)

[4] Application note, netX Studio Engineering Tool, Web Builder, DOC160207ANXXDE,
netX Studio Web Builder AN 01 DE.pdf

List of figures

List of tables

Glossary

MQTT	Message Queue Telemetry Transport: publish-subscribe based "light weight" M2M (machine to machine) messaging protocol for use on top of the TCP/IP protocol and for telemetry data transmission with a very low footprint. As protocol of the Internet of Things (IoT) it is standardized by OASIS (Organization for the Advancement of Structured Information Standards) since 2013. Additional features: ease of use, efficient use of resources, security mechanisms, support of several cloud-systems, session-aware, data-agnostic (i.e. flexible structures allow to transmit multiple types of information
netPROXY	Application framework for efficient development of networked devices based on Hilscher netX. It provides an object model as a generic abstraction for services or data. The term is also used as a general term related to the netPROXY technology of Hilscher.
netX	networX on chip, Hilscher network communication controller. High integrated network controller with optimized system architecture for communication and maximum data transfer.
OPC UA	OPC Unified Architecture: industrial M2M (machine to machine) communication protocol developed by the OPC Foundation for interoperability, with a cross-platform service oriented architecture (SOA) for process control, while enhancing security and providing an information model
Second Stage Boot Loader	started by the netX ROM loader, it creates a standard Hilscher dual-port memory and offers basic functionalities. It enables the download of firmware files via the dual-port memory.
WebServer	Firmware subset of a device developed according to the netPROXY model and allows the user in the field among other things, using a standard web browser and Ethernet interface, to retrieve status parameters from the automation device, to perform a firmware update or to reset the device via HTTP and optionally to read and write data or to configure the IP address of the device.

Contacts

HEADQUARTERS

Germany

Hilscher Gesellschaft für
Systemautomation mbH
Rheinstrasse 15
65795 Hattersheim
Phone: +49 (0) 6190 9907-0
Fax: +49 (0) 6190 9907-50
E-mail: info@hilscher.com

Support

Phone: +49 (0) 6190 9907-99
E-mail: de.support@hilscher.com

SUBSIDIARIES

China

Hilscher Systemautomation (Shanghai) Co. Ltd.
200010 Shanghai
Phone: +86 (0) 21-6355-5161
E-mail: info@hilscher.cn

Support

Phone: +86 (0) 21-6355-5161
E-mail: cn.support@hilscher.com

France

Hilscher France S.a.r.l.
69500 Bron
Phone: +33 (0) 4 72 37 98 40
E-mail: info@hilscher.fr

Support

Phone: +33 (0) 4 72 37 98 40
E-mail: fr.support@hilscher.com

India

Hilscher India Pvt. Ltd.
Pune, Delhi, Mumbai
Phone: +91 8888 750 777
E-mail: info@hilscher.in

Italy

Hilscher Italia S.r.l.
20090 Vimodrone (MI)
Phone: +39 02 25007068
E-mail: info@hilscher.it

Support

Phone: +39 02 25007068
E-mail: it.support@hilscher.com

Japan

Hilscher Japan KK
Tokyo, 160-0022
Phone: +81 (0) 3-5362-0521
E-mail: info@hilscher.jp

Support

Phone: +81 (0) 3-5362-0521
E-mail: jp.support@hilscher.com

Korea

Hilscher Korea Inc.
Seongnam, Gyeonggi, 463-400
Phone: +82 (0) 31-789-3715
E-mail: info@hilscher.kr

Switzerland

Hilscher Swiss GmbH
4500 Solothurn
Phone: +41 (0) 32 623 6633
E-mail: info@hilscher.ch

Support

Phone: +49 (0) 6190 9907-99
E-mail: ch.support@hilscher.com

USA

Hilscher North America, Inc.
Lisle, IL 60532
Phone: +1 630-505-5301
E-mail: info@hilscher.us

Support

Phone: +1 630-505-5301
E-mail: us.support@hilscher.com