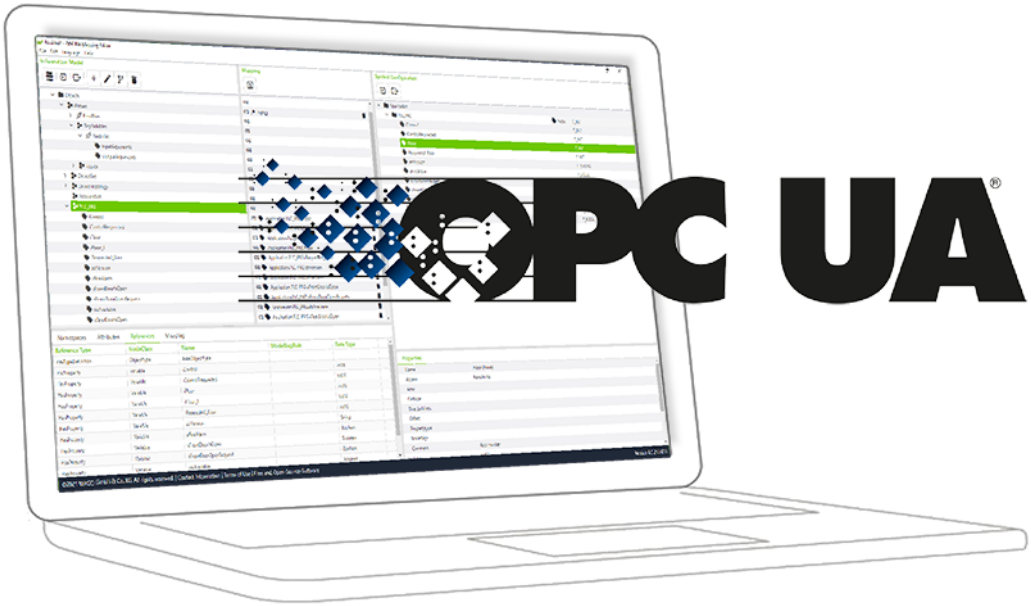
WAGO Software

OPC UA Mapping Editor



Product manual | Version 1.1.0

WAGO GmbH & Co. KG
Hansastraße 27
D - 32423 Minden
Phone: +49 571/887 – 0
Fax: +49 571/887 – 844169
E-Mail: ✉ **info@wago.com**
Internet: **www.wago.com**

Technical Support
Phone: +49 571/887 – 44555
Fax: +49 571/887 – 844555
E-Mail: ✉ **support@wago.com**

Every conceivable measure has been taken to ensure the accuracy and completeness of this documentation. However, as errors can never be fully excluded, we always appreciate any information or suggestions for improving the documentation.

E-Mail: ✉ **documentation@wago.com**

We wish to point out that the software and hardware terms as well as the trademarks of companies used and/or mentioned in the present manual are generally protected by trademark or patent.

WAGO is a registered trademark of WAGO Verwaltungsgesellschaft mbH.

Table of Contents

Terms

This documentation applies to the OPC UA Mapping Editor software, version 1.1.

Note

Observe the product documentation!

The software must only be installed and operated according to the instructions of the complete Instructions for use. Knowledge of the complete Instructions for use is required for proper use.

1. Carefully read the Product Manual.
2. Also read the product manuals for the products you use (e.g., fieldbus couplers, controllers and panels).

1.1 Intended Use

The OPC UA Mapping Editor is an independent, free software to map program structures to various OPC UA models, regardless of the platform.

Improper Use

Improper use of the products is not permitted. Specifically, improper use occurs in the following cases:

- Non-observance of the intended use
- Use of the products in areas with special risk that require flawless continuous operation and in which failure or operation of the software can result in an imminent risk to life, limb or health or cause serious damage to property or the environment (such as the operation of nuclear power plants, weapon systems , aircraft and motor vehicles)

Warranty and Liability

The terms set forth in the General Business and Contract Conditions for Delivery and Service of WAGO GmbH & Co. KG and the terms for software products and products with integrated software stated in the WAGO Software License Contract – both available at **www.wago.com** – shall apply. In particular, the warranty is void if:

- The product is improperly used.
- The deficiency (hardware and software configurations) is due to special instructions.
- Modifications to the hardware or software have been made by the user or third parties that are not described in this documentation and that has contributed to the fault.

Individual agreements always have priority.

Obligations of Installers/Operators

The installers and operators bear responsibility for the safety of an installation or a system assembled with the products. The installer/operator is responsible for proper installation and safety of the system. All laws, standards, guidelines, local regulations and accepted technology standards and practices applicable at the time of installation, and the instructions in the the products' Instructions for Use, must be complied with. In addition, the Installation regulations specified by Approvals must be observed. In the event of non-compliance, the products may not be operated within the scope of the approval.

1.2 Typographical Conventions

Number Notation

100	Decimals: Normal notation
0x64	Hexadecimals: C-notation
‘100’	Binary: In single quotation marks
‘0110.0100’	Nibbles separated by a period

Text Formatting

italic	Names of paths or files
bold	Menu items, entry or selection fields, emphasis
`Code`	Sections of program code
>	Selection of a menu point from a menu
“Value”	Value entries
[F5]	Identification of buttons or keys

Cross References / Links

	Cross references/links to a topic in a document
	Cross references / links to a separate document
	Cross references / links to a website
	Cross references / links to an email address

Action Instructions

✓ This symbol identifies a precondition.

1. Action step
2. Action step

   ⇨ This symbol identifies an intermediate result.

⇨ This symbol identifies the result of an action.

Lists

- Lists, first level
  - Lists, second level

Figures

Figures in this documentation are for better understanding and may differ from the actual product design.

Notes

⚠ DANGER

Type and source of hazard

Possible consequences of hazard that also include death or irreversible injury

- Action step to reduce risk

⚠ WARNING

Type and source of hazard

Possible consequences of hazard that also include severe injury

- Action step to reduce risk

⚠ CAUTION

Type and source of hazard

Possible consequences of hazard that include at least slight injury

- Action step to reduce risk

NOTICE

Type and source of malfunction (property damage only)

Possible malfunctions that may restrict the product's scope of functions or ergonomics, but do not lead to foreseeable risks to persons

- Action step to reduce risk

Note

Notes and information

Indicates information, clarifications, recommendations, referrals, etc.

1.3 Legal Information

Intellectual Property

Unless barred by applicable legal provisions, unauthorized copying and distribution of this document, as well as the use and communication of its content are strictly prohibited unless expressly authorized by prior agreement. Third-party products are always mentioned without any reference to patent rights. WAGO GmbH & Co. KG, or for third-party products, their manufacturer, retain all rights regarding patent, utility model or design registration.

Third-party trademarks are referred to in the product documentation. The "®" and "TM" symbols are omitted hereinafter. The trademarks are listed in the Appendix (**Protected Rights [▶ 45]**).

Subject to Change

The instructions, guidelines, standards, etc., in this manual correspond to state of the art at the time the documentation was created and are not subject to updating service. The installer and operator bear sole responsibility to ensure they are complied with in their currently applicable form. WAGO GmbH & Co. KG retains the right to carry out technical changes and improvements of the products and the data, specifications and illustrations

of this manual. All claims for change or improvement of products that have already been delivered – excepting change or improvement performed under guarantee agreement – are excluded.

Security

2.1 General Safety Regulations

- This documentation is part of the product. Therefore, retain the documentation during the entire service life of the product. Pass on the documentation to any subsequent user of the product. In addition, ensure that any supplement to this documentation is included, if necessary.
- Any actions related to the use of WAGO software may only be performed by qualified staff with sufficient knowledge to use the respective PC system.
  Steps in which files are created or changed on a PC system may only be performed by qualified employees with sufficient knowledge in the administration of the PC system used in addition to file creation or modification.
  Steps that change the PC system's behavior within a network may only be performed by qualified employees with sufficient knowledge of administration of the responsible network.
- Comply with the laws, standards, guidelines, local regulations and accepted technology standards and practices applicable at the time of installation.

Overview

“OPC Unified Architecture" (OPC UA) is a platform-independent and service-oriented architecture. It is used to describe and transport data. Because the services are independent, devices from different manufacturers can be interconnected. Known as “companion specifications,” these were defined to cope with the demands of different industries with similar products and machines and contain information models (so-called base models) adapted to the application.

By default, WAGO controllers use the “PLCopen” information model to provide data for other applications. The OPC UA Mapping Editor can be used to map this information model to any other OPC UA model, for example to OPC UA base models such as “Robotics” or “Euromap77”.

The mapping functionality is supported by all PFC200 controllers of the 2nd generation (750-821x), TP600 Control Panels (762-430x) and the Edge Controller (752-8303 / 8000-0002).

The OPC UA Mapping Editor offers the following functions:

Editing of an OPC UA Information Model

The editor can be used to create and edit the OPC UA information model. Optionally, existing information models can be imported as a base, e.g., “Robotik.xml“ or “Euromap77.xml”, but you can also create your own company or application-specific models.

Importing the Symbol Configuration (Application Data)

The symbol configuration describes the data source with the application variables. They are created with a ***e!****COCKPIT* project and imported into the OPC UA Mapping Editor.

Linking the Data Sources to the Information Model

Application data is linked with the variables of the information model. Thus, the corresponding OPC UA variable returns the value of the application variable at runtime.

Exporting the Mapping File

A mapping file results from the link between the OPC UA information model and the data sources. It is then exported to the controller.

Accessing Controller Application Data

After the mapping file has been exported to the controller, the controller provides the application data in the corresponding information model. The data and the new structure according to the information model can then be further processed by the corresponding OPC UA clients.

Note

More information on OPC UA at WAGO is available here:

“WAGO OPC UA Server” Manual
The manual describes the functions of the WAGO OPC UA server, establishing a connection using an OPC UA client and configuring from web-based management.

PFC200, 2nd Generation (750-821x) Manuals
Control Panel TP600 Manuals (762-430x)
Edge Controller Manual (752-8303/8000-0002)
The manuals describe the respective products for which the OPC UA Mapping Editor can be used. Information on downloading licenses for these devices is available on the website **http://www.wago.com**.

You can find more information about information models on the Internet at:
https://www.plcopen.org/technical-activities/communication
https://www.euromap.org/i40/overview
https://opcfoundation.org/about/opc-technologies/opc-ua/ua-companion-specifications/
Download the node sets from the OPC UA Foundation:
https://github.com/OPCFoundation/UA-Nodeset

Requirements

System requirements

Table 1: System requirements

Components	Requirements
Operating system	Windows 10
Memory	4 GB
Free hard disk space	800 MB
Processor	Dualcore CPU
Screen resolution	Minimum: 1366 x 768 pixels Recommended: 1920 x 1080 pixels

Licenses

The OPC UA Mapping Editor, which is used to adapt the information model, is available free of charge and license-free from the WAGO website **https://www.wago.com/de/offene-automatisierung/connectivity-hero/opc-ua**.

Generally, the mapping functionality is supported by all 2nd Generation PFC200 Controllers (750-821x), TP600 Control Panels (762-430x) and Edge Controllers (752-8303/8000-0002).
However, to use the adapted information model via the imported mapping file on the controller, an extended ***e!****RUNTIME* license of the WAGO OPC UA Server “OPC UA Server Extended” is required on the device (Item Number for PFC200(G2): 2759-2233/0210-1000 and Item Number for TP600/Edge Controller: 2759-2236/0210-1000).

Install

The WAGO OPC UA Mapping Editor is installed as a setup.

1. Open the WAGO website for OPC UA:
   https://www.wago.com/de/offene-automatisierung/connectivity-hero/opc-ua
2. To get a download link for the WAGO OPC UA Mapping Editor, click the link **[Registration]** and fill out the form.
3. Download the software.
4. Start the installation process by double-clicking the installation file.
5. Follow the steps in the installation wizard and select a target directory for the installation.
6. Click **[Install]** to start the installation.
7. Click **[Finish]** to complete installation.
8. Go to **WAGO Software** > **WAGO OPC UA Mapping Editor** to launch the software.

Graphical User Interface

The graphical user interface is structured in the following areas.

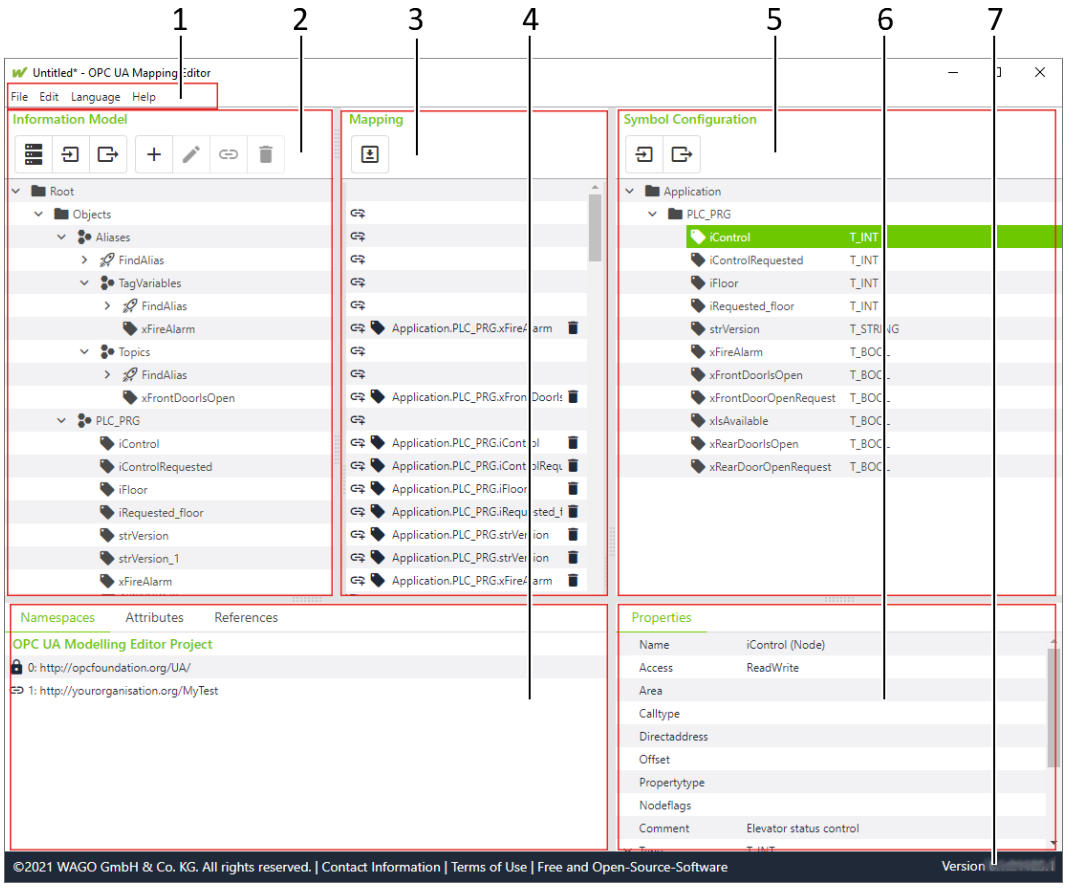


Figure 1: Graphical User Interface

Table 2: Graphical User Interface

No.	Description
1	**File and Project [▶ 15]** The buttons are used for general settings.
2	**Information Model [▶ 16]** Information models are displayed and adapted here.
3	**Mapping [▶ 18]** In this area, variables from the symbol configuration are mapped to variables of the information model and exported as a mapping file.
4	**Information Model and Mapping Properties [▶ 19]** Information on a selected element from the information model is displayed in this area.
5	**Symbol Configuration [▶ 22]** In this area, application data from an ***e!****COCKPIT* project is displayed, imported and exported.
6	**Symbol Configuration Properties [▶ 23]** Information on a selected element from the symbol configuration is displayed in this area.
7	**Status bar** The contact details of WAGO GmbH & Co. KG as well as the terms of use and license conditions of the software are accessed via links. The software version is also displayed.

6.1 File and Project

The menu offers the following functions for general settings relating to files and projects:

Table 3: “File” Menu

Menu Item	Function
File	
New project	A dialog for creating a new project opens. A model according to the OPC UA standard specification is initially created in the “Information Model” area (see **Create a New Project [▶ 24]**).
Open project	A project opens (*.opcprj) that was previously created with the OPC UA Mapping Editor.
Save project (as...)	The project is saved (*.opcprj).
Exit	The OPC UA Mapping Editor is closed.
Edit	
Select base models	A dialog opens in which saved base models can be selected or new base models can be added (see **Select Base Models [▶ 25]**). Base models are models that are accessible via open platforms and are already adapted to various areas of application.
Import information model	A dialog opens to import information models in *.xml format (see **Import Information Model [▶ 29]**).
Export information model	A dialog opens to export customized information models in *.xml format (see **Export Information Model [▶ 29]**).
Information model settings	A dialog opens to change the model URI, version and date of the information model. You enter this data for the first time when creating a new project.
Add instance	A dialog opens to add instances (variables or objects) (see **Add Instance [▶ 26]**).
Edit instance	A dialog opens to edit instances (variables or objects) (see **Edit Instance [▶ 27]**).
Delete instance	The selected instance is deleted (see **Delete Instances [▶ 28]**).
Add reference	A dialog opens to add references (see **Add References [▶ 28]**).
Language	
German/English	The user interface language of the mapping editor is changed.
Help	
Display help	The online help of the mapping editor opens in a separate window.

6.2 Information Model

The OPC UA model that is used for the mapping is displayed in the “Information Model” area.

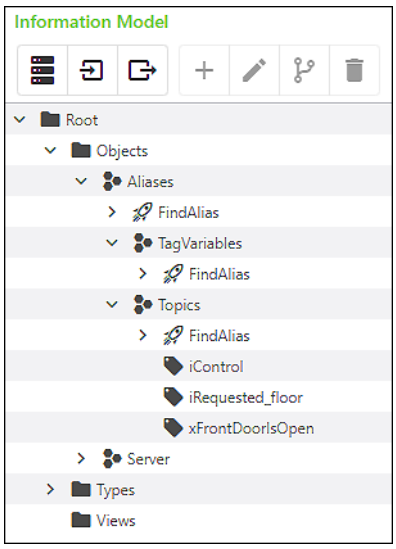


Figure 2: “Information Model” Area

The information model is created in this area as follows:

- In **“New project”**, an initial model is first created in the default structure per OPC UA.
- In **“Select base model”**, base models with predefined structures are applied to the model according to the area of application.
- In **“Import information module”**, other models that are available as files can be imported.

The following icons are used:

- Folder
- Object
- Object type
- Method
- Variable
- Data type

The following functions are available via buttons in the “Information Model” area:

Table 4: Functions in the “Information Model” Area

Button		Function
	[Select base models]	A dialog opens in which saved base models can be selected or new base models can be added (see **Select Base Models [▶ 25]**). Base models are models that are accessible via open platforms and are already adapted to various areas of application.
	[Import information model]	A dialog opens to import information models in *.xml format (see **Import Information Model [▶ 29]**). **Note:** Note that importing a new information model overwrites the existing one.
	[Export information model]	A dialog opens to export customized information models in *.xml format (see **Export Information Model [▶ 29]**).
+	**[Add instance]**	A dialog opens to add instances (variables or objects) (see **Add Instance [▶ 26]**).

Button		Function
	[Edit instance]	A dialog opens to edit instances (variables or objects) (see **Edit Instance [▶ 27]**).
	[Add reference]	A dialog opens to add a reference (see **Add References [▶ 28]**).
	[Delete instance]	The selected instance is deleted (see **Delete Instances [▶ 28]**).

You can also access functions for adding instances and references as well as cutting, copying, pasting and deleting elements from the context menu.

6.3 Mapping

The link between the application data and the information model is displayed in the “Mapping” area. Drag and drop the application variable of the symbol configuration into the “Mapping” area next to the corresponding variable or the object of the information model (see **Map Application Variables to Information Module Variables/Objects [▶ 31]**).

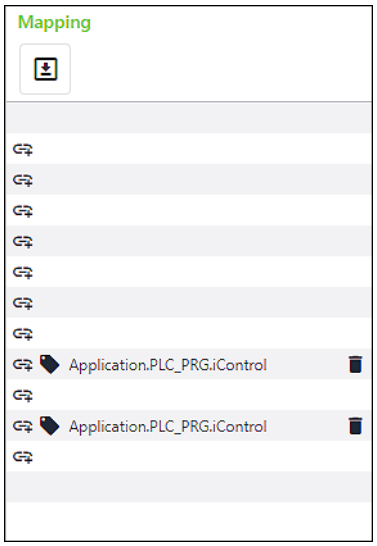


Figure 3: Variables Mapped to the Information Model in the “Mapping” Area

The following icons are used:

- Linkable element (mapping possible)
- Variable
- Static Value
- Regular Expression

The following functions are available in the “Mapping” area:

Table 5: Functions in the “Mapping” Area

Button		Function
	[Export mapping]	A dialog for exporting the mapping as an XML file opens (see **Export Mapping [▶ 39]**). The mapping file is used to load it into the controller via the WBM.
	[Delete mapping]	You can delete individual mappings between the symbol configuration and the information model using the trash can icon in the respective line (see **Delete Mapping [▶ 39]**).

6.4 Information Model and Mapping Properties

Depending on the element selected in the “Information Model” area, additional information is displayed in this area based on the context.

“Namespaces” Tab

This tab shows the namespace on which the information model is based.

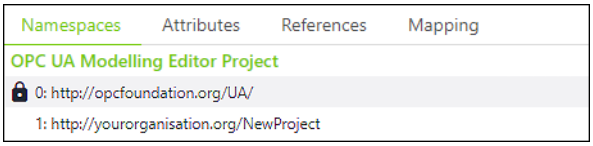


Figure 4: “Namespaces” Tab

“Attributes” Tab

This tab shows attributes and values of the selected element.

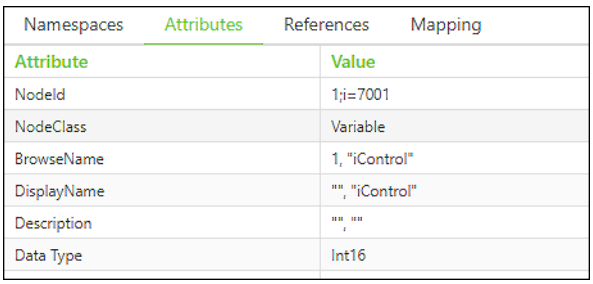


Figure 5: “Attribute” Tab

“References” Tab

This tab shows the reference types used with node class, name, modelling rule and data type for the selected element (see also **Add References [▶ 28]**).
If the reference can be deleted (this applies, for example, to references that you have created yourself), a trash can icon is displayed to delete the entry (see **Delete References [▶ 28]**).

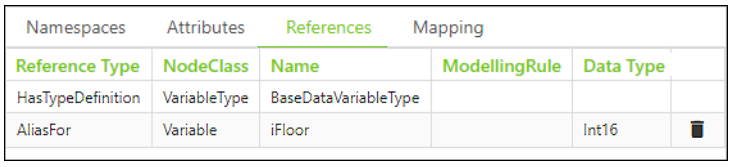


Figure 6: “References” Tab

“Mapping” Tab

This tab is only displayed if an element of the information model is selected for which mapping with application variables is possible.

When selecting a **variable**, the “Mapping” tab shows the following settings:

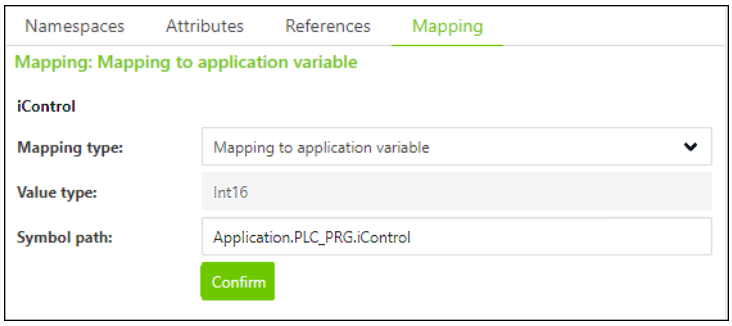


Figure 7: "Mapping" tab (when selecting a variable)

Table 6: Display and selection of the mapping type

Mapping Type	Description
No mapping	"No Mapping" is displayed if the variable selected in the information model is not (yet) mapped to an application variable. **Note:** If a mapping already exists and you switch to this option and click **[Confirm]**, the existing mapping is deleted and the entry in the "Mapping" area is removed.
Mapping to application variables	"Mapping to application variables" is displayed if the variable selected in the information model is mapped to an application variable (see **Map Application Variables to Information Module Variables/Objects [▶ 31]**). In this case, the data type and symbol path of the linked application variable are displayed. **Note:** If there is no mapping before and this option is selected, the symbol path can also be entered manually at this point (see **Map Application Variables to an OPC UA Variable Manually [▶ 33]**). Click **[Confirm]** to check the validity of the symbol path.
Mapping to a static value	A static value is assigned to the variable (see **Assign Static Values to Variables [▶ 34]**). **Note:** Note that the value entered must match the data type used. The values are not checked for validity. Please note the OPC UA attribute reference under **https://reference.opcfoundation.org**.

When selecting an **object**, the "Mapping" tab shows the following settings:

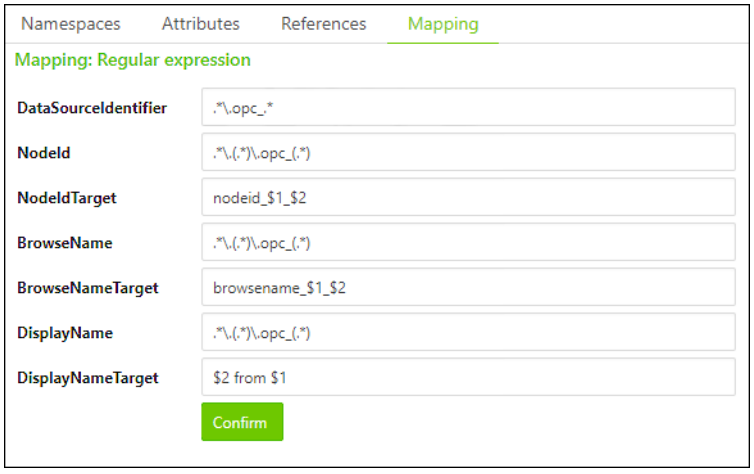


Figure 8: "Mapping" tab (when selecting an object)

Table 7: Settings for Regular Expressions

Parameters	Description
DataSourceIdentifier	The symbol paths of all available application variables are filtered using this regular expression.

Parameters	Description
NodeId/ NodeIdTarget	For each symbol path already filtered using the “DataSourceIdentifier”, a NodeId string is generated using the “NodeId” and “NodeIdTarget” properties. **Note:** Note that this NodeId string must be unique according to the OPC UA specification. The “NodeId” property specifies a regular expression that is applied to the symbol paths that have already been filtered. The “NodeIdTarget” property compiles the final NodeId value of the OPC UA variable from the components of the regular expression of the “NodeId”.
BrowseName/ BrowseNameTarget	Similar to the “NodeId” property, the OPC UA BrowseName is generated from the already filtered symbol path using these properties.
DisplayName/ DisplayNameTarget	Similar to the “NodeId” property, the OPC UA DisplayName is generated from the already filtered symbol path using these properties.

The description for using regular expressions is available at **Use Regular Expressions to Filter Application Variables [▶ 35]**.

6.5 Symbol Configuration

In the “Symbol Configuration” area, the contents of an imported symbol configuration (XML) resulting from an application are displayed. The structure for controls is based on the PLCopen structure as standard.

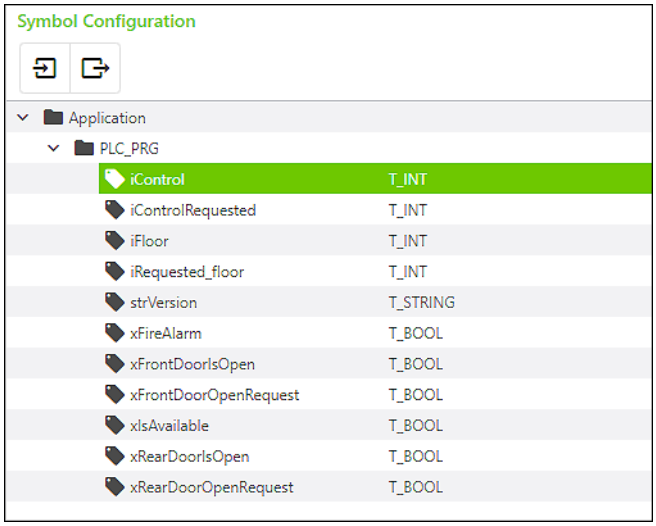


Figure 9: “Symbol Configuration” Area

The following functions are available in the “Symbol Configuration” area:

Table 8: functions in the “Symbol Configuration” Area

Button		Function
	[Import symbol configuration]	A dialog opens to import symbol configurations in *.xml format (see **Import Symbol Configuration with Application Variables [▶ 30]**).
	[Export symbol configurations]	A dialog opens to export the symbol configuration in *.xml format. Although no changes are made to the symbol configuration itself in the OPC UA Mapping Editor, the export function can be used, for example, if the symbol configuration is to be edited in another program. It can happen that entire projects have been passed on from the mapping editor and the original symbol configuration is therefore no longer available.

6.6 Symbol Configuration Properties

The properties of selected variables of the symbol configuration are displayed in this area.

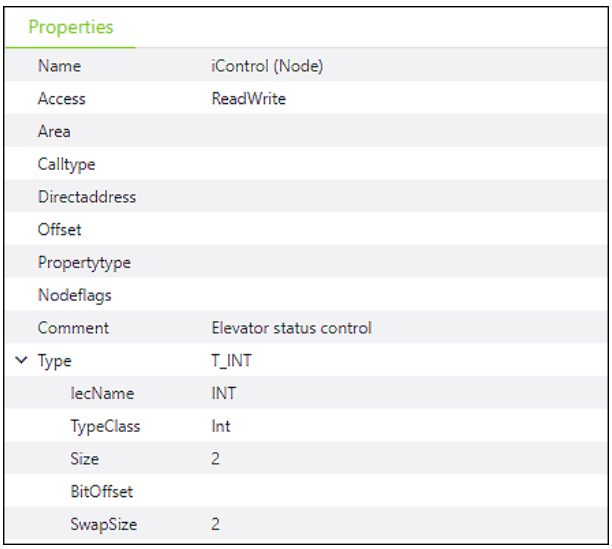


Figure 10: Symbol Configuration Properties

Operation

7.1 Preparation: Create symbol configuration in *e!COCKPIT*

1. Open ***e!****COCKPIT*.
2. Open the program structure and build your program.
3. Right-click the application (offline) and select **Symbol Configuration** from the context menu.
4. Select the option “Support OPC UA functionalities” in the dialog.
5. Click **[Add]** to create the symbol configuration.
   ⇨ The “Symbol Configuration” tab appears.
6. Click **[Create]** because the program must be compilable for the next steps.
7. Select those variables from the list that are to be published by the server via OPC UA.

⇨ The selected symbol configuration is automatically transferred to the controller via download / online change the next time it is connected. The parameters are initially available on the controller in the standard format according to the PLCopen specification.
The symbol configuration is saved as a file in the project directory in which the “.ecp” project file is also located.

7.2 Create a New Project

1. Open the OPC UA Mapping Editor.
2. In the menu bar, click **File** > **New project**.
3. Enter a namespace in the “Model URI” field in the dialog. This is used to filter data in a client program.
4. Also enter a version and date for the model that you want to create.
5. Click **[OK]**.
   ⇨ The basic structure of the information model is shown.

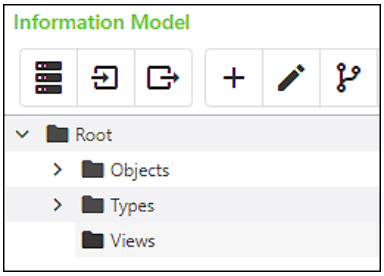


Figure 11: Information Model Basic Structure

You can continue working with this model. For various industrial areas of application, however, base models are also available that already contain data models adapted to the respective area of application (see **Select Base Models [▶ 25]**).

7.3 Select Base Models

Base models are published by the OPC UA organization as “Companion Specifications” for the respective application areas.

1. To use a base model, click the **[Select base model]** button.

   ⇨ The “Select Basic Models” dialog opens.

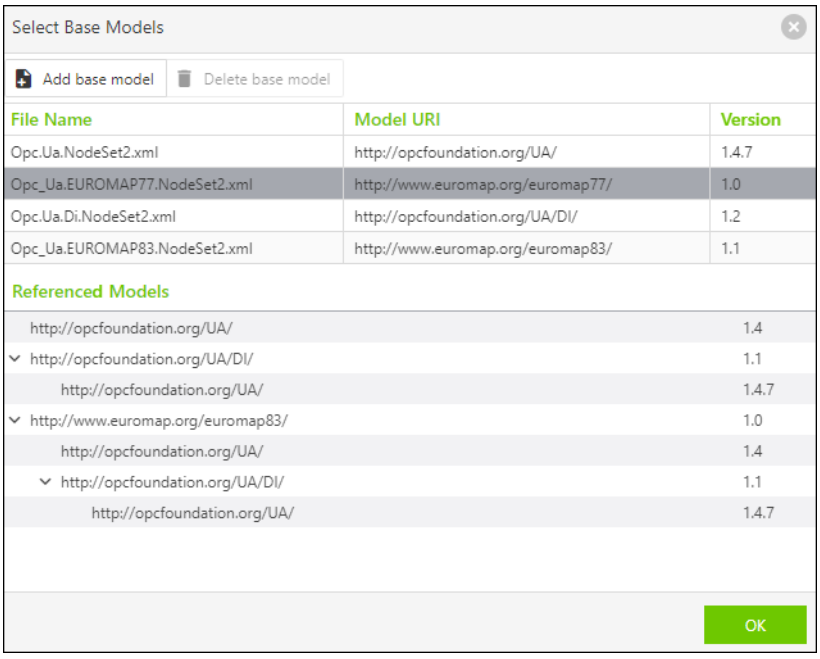


Figure 12: “Select Basic Models” Dialog

2. From the list of saved base models, select the model that you want to use for your project, e.g., “EuroMap”.
   Optionally, you can first download the desired base model (see **Add New Base Models [▶ 26]**)
3. Click **[OK]**.

   ⇨ Your information model is now adapted to the respective area of application by the referenced base model.

You can use the information model or supplement it individually (see **Edit Information Model [▶ 26]**).

7.4 Add New Base Models

1. To expand the list of selectable base models in the “Select Base Models” dialog, first load additional base models at https://opcfoundation.org/ua/.
2. Click the **[Add base model]** button to select the base model previously saved.
3. Click **[Open]**.

⇨ The list in the “Select Base Models” dialog is expanded accordingly.

7.5 Edit Information Model

In the “Information Model” area, you can edit the elements of the OPC UA information model.

At the top level, the model consists of the “Objects”, “Types” and “Views” folders as standard in accordance with the OPC UA base specification. In the “Objects” folder you can add, delete and edit instances (objects and variables).
Note: The “Types” and “Views” folders cannot be modified. If you want to create your own type definitions here, you can create / change them using a different XML tool. Then import the changed model back into the OPC UA Mapping Editor.

7.5.1 Add Instance

1. In the information model, select the “Objects” folder or an underlying object node.
2. Click the + **[Add instance]** button.

   ⇨ The “Add Instance” dialog opens.

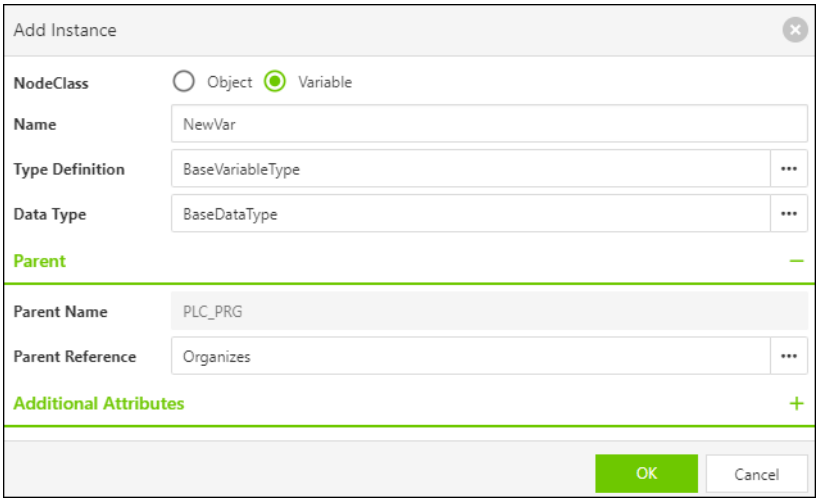


Figure 13: Add Instance

3. Under “NodeClass,” select whether you want to add an object or a variable.
4. Enter the name of the object or the variable in the corresponding text field.
5. Select the type of object or variable.

   ⇨ The types available depend on the OPC UA base models selected in the project (e.g., a “Companion Specification”).
   Without adding a base specification, only the data types that are defined as standard for OPC UA are offered.

6. You can also edit other OPC UA properties of the new instance:
   In the “Parent” area, you can change the reference that defines the relationship to the parent node, e.g., to the reference type “Organizes” or “HasComponent”.
   You can set additional OPC UA attributes in the “Additional Attributes” area.
7. Click **[OK]** to close the dialog.

7.5.2 Edit Instance

You can edit the properties of an already existing object or a variable later.

1. Select the instance to be edited in the information model.
2. Click the **[Edit instance]** button.
   ⇨ The “Edit Instance” dialog opens.

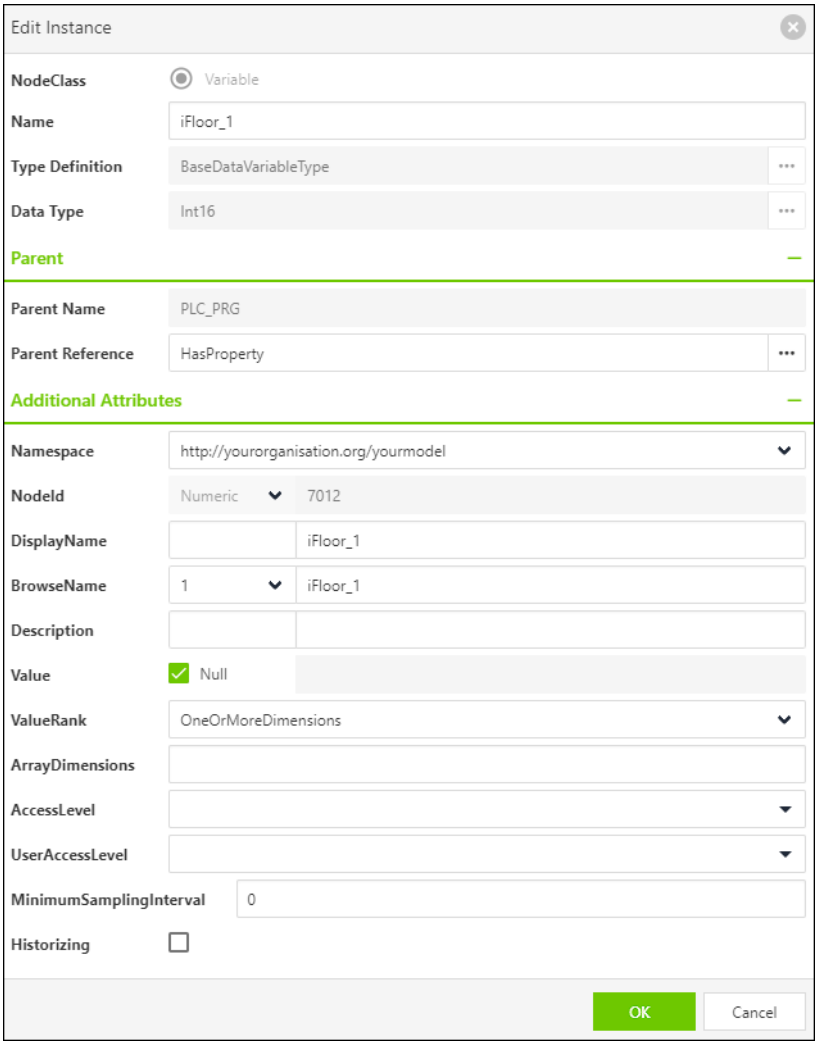


Figure 14: Edit Instance

3. Edit the required properties of the instance.
   Note: The “NodeClass” property of an instance cannot be changed subsequently. If you want to change this, create the instance (object or variable) again.
4. Click **[OK]** to close the dialog.

7.5.3 Delete Instances

You can delete created instances (objects and variables) as follows:

1. Select the object or variable to be deleted in the information model.
2. Click **[Delete instance]** in the context menu.

⇨ The object or variable is deleted from the information model.

Note: Objects or variables that are defined by default in the base model cannot be deleted.

7.5.4 Add References

You can add references to an instance in the OPC UA information model.

1. Select an object or variable in the OPC UA information model that is not part of the base model.
2. Click the **[Add reference]** button.

   ⇨ The “Add Reference” dialog opens.

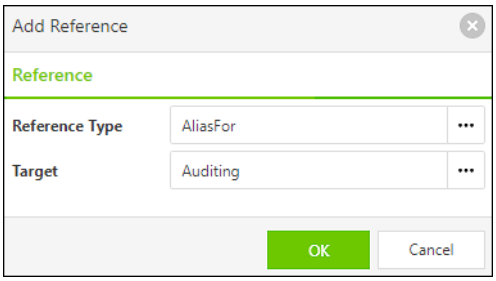


Figure 15: Add Reference

3. Select the type of reference.

   ⇨ The types available depend on the OPC UA base models selected in the project (e.g., a “Companion Specification”).
   Without adding a base specification, only the reference types that are defined as standard for OPC UA are offered.
4. Select the reference target. The selection consists of all nodes in the current information model.
5. Click **[OK]** to confirm.

⇨ The reference just added appears in the “References” tab.

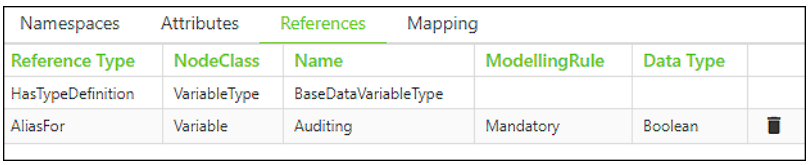


Figure 16: Newly Created Reference

7.5.5 Delete References

Added references can be deleted again.

1. Select the object or variable in the information model for which you have created a reference.

2. In the “Information Model and Mapping Properties” area, select the reference to be deleted in the “References” tab.

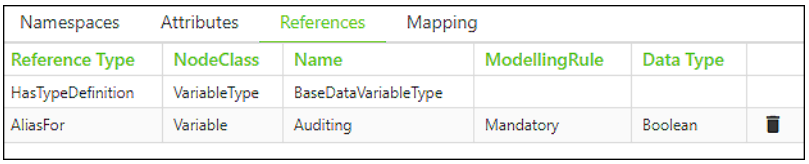


Figure 17: Delete Reference

3. Click the 🗑 icon in the right column of the entry.

⇨ The reference is deleted.

Note: If no trash can icon appears, the reference cannot be deleted. This is the case, for example, for references defined by default in the base model.
References to subordinate (child) structures are also shown as non-deletable. Editing is not permitted from the higher-level structure. To change or delete, switch directly to the subordinate (child) instance.

7.5.6 Import Information Model

1. To import an information model, click the **[Import information model]** button.
   Note: Note that importing a new information model overwrites the existing one.
2. Select an information model in *.xml format. These can be models that were created in other applications (e.g., UaModeler).
   Note: The XML file must contain a valid OPC UA node set according to the XML schema of the OPC Foundation.
3. Click **[Open]**.

⇨ The information model opens.

7.5.7 Export Information Model

1. To export an information model, click the **[Export information model]** button.
2. Select a location and enter a file name for the information model.
3. Click **[Save]**.

⇨ The information model is saved in *.xml format.

7.6 Import Symbol Configuration with Application Variables

- To open the XML file created in ***e!****COCKPIT* with the application variables (symbol configuration), click the **[Import symbol configuration]** button in the “Symbol Configuration” area.

  ⇨ The content of the XML file appears.
  Note: No validation is carried out.

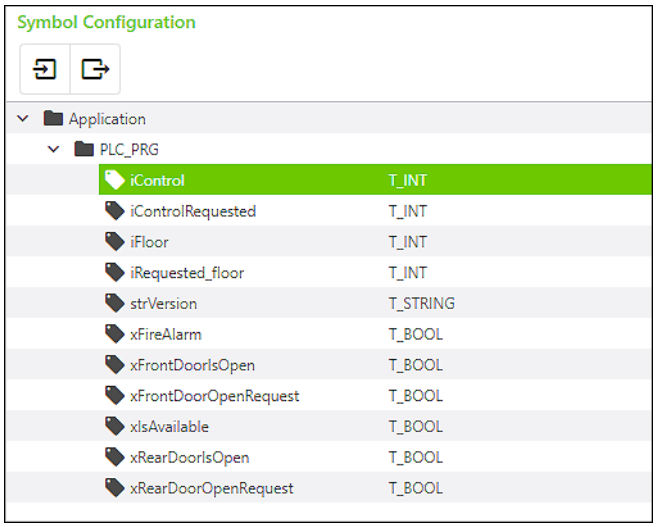


Figure 18: Import Symbol Configuration

Tips:

- Multiple selection is possible for mapping in this area, provided that the application variables are dragged next to an OPC UA object.
- If you select individual variables, you will find more information in the “Properties” area below.
- You can use **regular expressions [▶ 35]** to filter the symbol configuration at runtime.

7.7 Map Application Variables to Information Module Variables/Objects

The OPC UA Mapping Editor offers various options for mapping between application variables and the OPC UA model. You can link variables with each other both automatically by drag & drop and manually by entering the target variables. If there is no suitable variable for mapping in the information model, a new variable or corresponding folder structure is automatically created in the information model. In addition, you can use regular expressions to define a selection of application variables from the symbol configuration, for which OPC UA variables are generated at runtime.

Note: Note that application variables can only be mapped to variables / objects from the information model that are marked by the icon in the “Mapping” area.

7.7.1 Map Application Variables to an OPC UA Variable Directly

An already created variable in the OPC UA model is linked to an application variable. The data types of the application variable and the OPC UA variable must be compatible.

- To do this, drag and drop the variable from the symbol configuration into the “Mapping” area next to the respective OPC UA variable.

⇨ The application variable is displayed in the “Mapping” area next to the information model variable.

Figure 19: Direct Mapping

The data types must match for the mapping. The check for matching of the data types is done according to **Overview and mapping of OPC UA and CODESYS data types [▶ 42]**.

Note

Note about mapping enumeration types (ENUM) and structure definitions (STRUCTS)!

Variables in the symbol configuration that are of the “ENUM” or “STRUCT” data type cannot be mapped directly to a variable in the OPC UA information model. Variables of these data types can, however, be dragged and dropped into the “Mapping” area next to an object of the information model (see **Create OPC UA Variables Automatically [▶ 32]**). The OPC UA Mapping Editor then creates a new variable in the OPC UA information model. Furthermore (if the “ENUM” or “STRUCT” data type is used for the first time), a corresponding OPC UA data type is created and assigned to the new variable.

7.7.2 Create OPC UA Variables Automatically

You can automatically create OPC UA variables in the information model as follows:

- Drag and drop a variable from the symbol configuration into the “Mapping” area next to an object in the OPC UA information model.

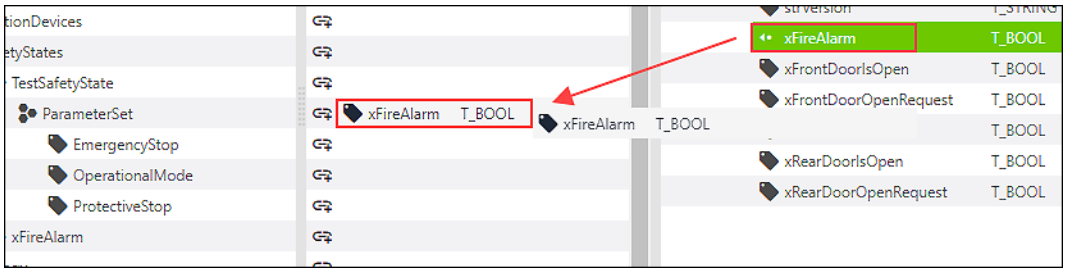


Figure 20: Drag Variable into the “Mapping” Area

⇨ A new variable is then added to the OPC UA object.

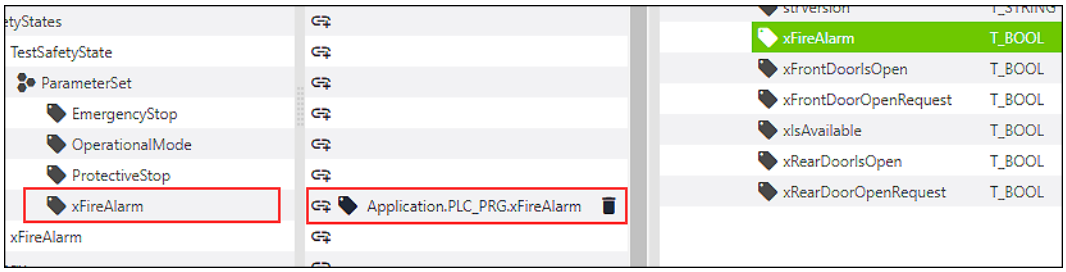


Figure 21: Newly Created OPC UA Variable

The name and the data type of this variable are created according to the variable from the symbol configuration (see **Overview and mapping of OPC UA and CODESYS data types [▶ 42]**).

Tip: You can create multiple variables at the same time via multiple selection.

7.7.3 Create OPC UA Object Structure Automatically

The symbol configuration is organized as a tree structure. This tree also contains nodes that do not directly correspond to a variable in an application. Instead, these nodes represent a folder such as a PLC program (here “PLC_PRG”), which in turn contains variables.

The OPC UA Mapping Editor allows you to create an object in the OPC UA information model for such a node. All variables and possibly subfolders are also created as variables or further child objects.

- To do this, drag and drop the folder from the symbol configuration into the “Mapping” area next to an OPC UA object.

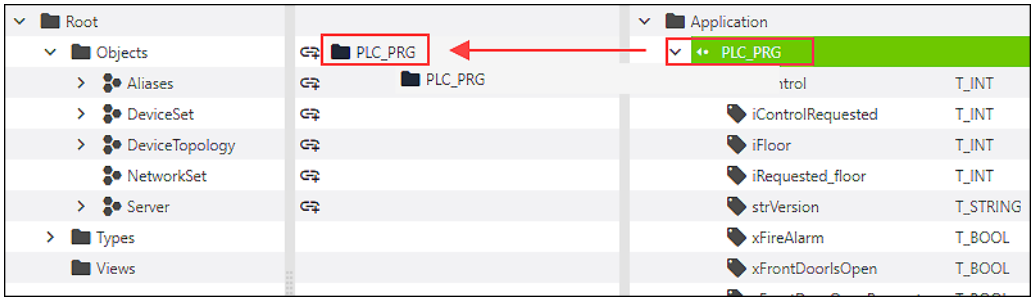


Figure 22: Drag the Entire Folder onto an OPC UA Object

⇨ The new object is created in the information model with all variables it contains. Mapping is automatic.

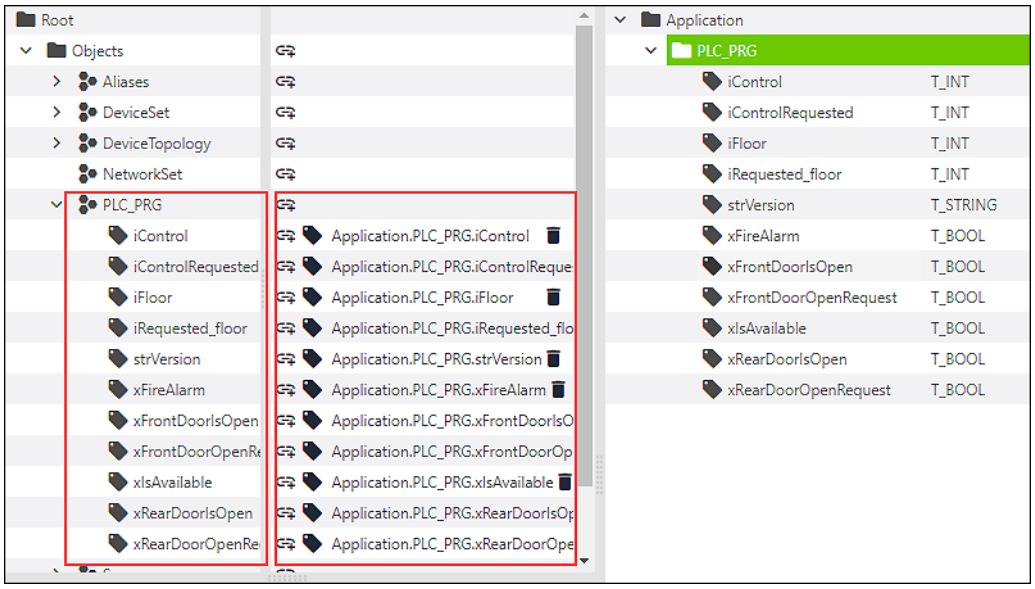


Figure 23: Created OPC UA Objects with Variables

Note: Objects in the OPC UA information model of the “Folder” object type are shown in the tree as folders, but are generally also objects in the sense of the OPC UA specification. It therefore makes no difference whether you drag the folder from the symbol configuration next to an OPC UA folder or next to another OPC UA object.

7.7.4 Map Application Variables to an OPC UA Variable Manually

In addition to being able to link an application variable with an OPC UA variable using drag & drop, you can also enter the path of the application variable manually.

1. Select the OPC UA variable in the OPC UA information model.

2. Open the “Mapping” tab in the information model.
3. In the “Mapping type” selection box, select the “Mapping to application variable” option.

   ⇨ **Note:** The data type of the OPC UA variable only serves as additional information at this point. If you want to change it, use the “Edit instance” function.
4. In the “Symbol path” field, enter the path of the application variable in the symbol configuration.

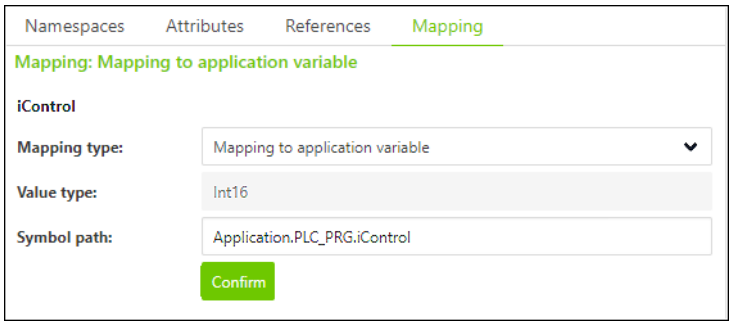


Figure 24: Map Application Variables to an OPC UA Variable Manually

5. Click **[Confirm]** to apply the changes.

⇨ The entered value is displayed in the “Mapping” area next to the OPC UA variable.

Note: When the entry is applied, the validity of the path is checked. If the path entered does not refer to a valid variable in the symbol configuration, an error is displayed. The changes are not applied.

7.7.5 Assign Static Values to Variables

A static value can be specified for the value of a variable in an OPC UA information model.

1. Select the OPC UA variable in the OPC UA information model.
2. Open the “Mapping” tab.
3. In the “Mapping type” selection box, select the “Mapping to static value” option.
   Note: The data type of the OPC UA variable only serves as additional information at this point. If you want to change it, use the “Edit instance” function.
4. Enter the static value.
   Note: Note that the value entered must match the data type used. The values are not checked for validity. Please note the OPC UA attribute reference under **https://reference.opcfoundation.org**.

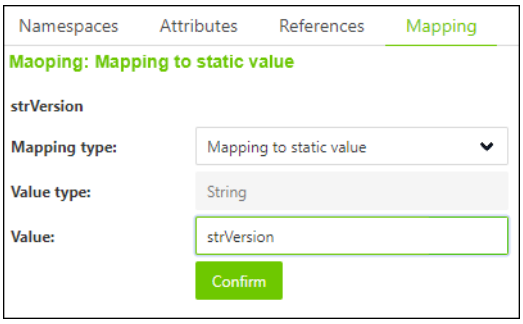


Figure 25: Enter Static Value

5. Click **[Confirm]** to apply the changes.

⇨ The entered value is displayed in the “Mapping” area next to the OPC UA variable and identified by the ⓢ icon.

Figure 26: Variable in the “Mapping” Area

7.7.6 Use Regular Expressions to Filter Application Variables

The OPC UA server allows you to generate OPC UA variables at runtime and to map them to application variables. The application variables to be mapped are filtered out of all available application variables using regular expressions.

To create such a mapping rule, proceed as follows:

1. Select an object (not a variable) in the information model.
2. Open the “Mapping” tab in the information model.
   ⇨ For OPC UA objects, the context-sensitive mapping configuration is offered here using regular expressions.

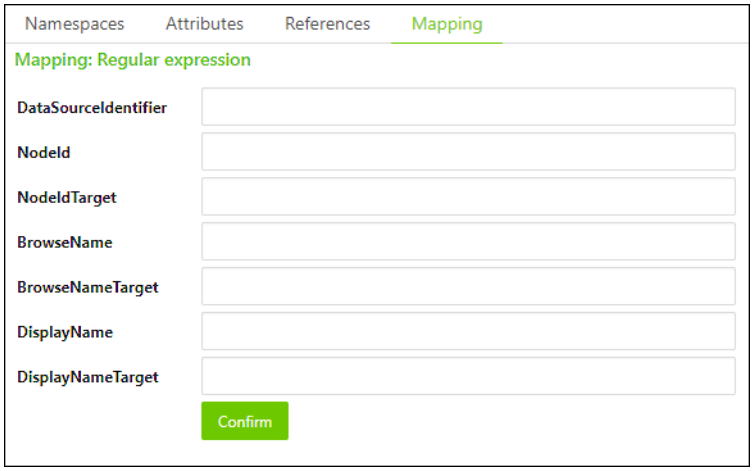


Figure 27: Regular Expressions in the “Mapping” Tab

3. Configure the regular expressions with which variables are automatically generated under the selected OPC UA object at runtime (see example below).
4. Click **[Confirm]** to apply the changes.

Example of Using Regular Expressions

In this example, you map all application variables below an OPC UA variable whose name is preceded by the prefix “opc_”. It should be irrelevant whether these are defined in the “PLC_PRG” program or the “GVL” global variable list. The following symbol configuration serves as the starting point for the example:

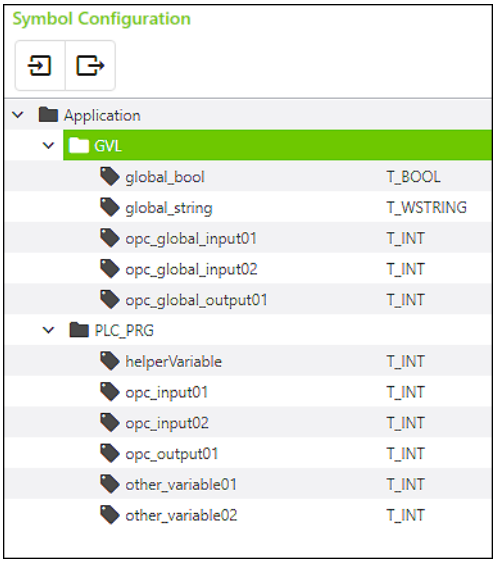


Figure 28: Symbol Configuration (Example)

1. First, create an OPC UA object instance (e.g., with the name “RegExpMapping”). For the example, you can leave the object type at the default value (“BaseObjectType”).
2. Select the newly created object and edit the regular expressions in the “Mapping” tab under the information model as follows:
3. **DataSourceIdentifier**
   Enter a regular expression under “DataSourceIdentifier” with which the symbol paths of all available application variables are filtered.
   Note: The symbol paths contain periods as separators (“`.`”). In the regular expression, precede the period with a backslash (“`\`”). Otherwise, the period will be interpreted as a wildcard (unless you actually want to use the period as a wildcard).
   ⇨ **Example:** `.*\.opc_.*`
4. **NodeId**
   For each symbol path already filtered using the “DataSourceIdentifier”, a NodeId string is generated using the “NodeId” and “NodeIdTarget” properties.
   Enter a regular expression as the “NodeId” property that will be applied to the symbol path that has already been filtered.
   ⇨ **Example:** `.*\.(.*)\.opc_(.*)`

5. **NodeIdTarget**
   The “NodeIdTarget” property compiles the final NodeId value of the OPC UA variable from the components of the regular expression of the “NodeId”.

   ⇨ **Example:** `nodeid_$1_$2`
   For the symbol path `advanced.GVL.opc_global_input01`, `nodeid_GVL_global_input01` would result as the “NodeId”.

   ⇨ **Note:** Note that the NodeId string must be unique according to the OPC UA specification. To ensure this, part of the symbol path (GVL or PLC_PRG) is included in the example, as otherwise there would be a risk of a duplicate NodeId.

6. **BrowseName / BrowseNameTarget**
   Enter the properties “BrowseName” and “BrowseNameTarget” to create the OPC UA BrowseName from the symbol path that has already been filtered.

   ⇨ **Example:**
   BrowseName: `*\.(.*)\.opc_(.*)`
   BrowseNameTarget: `browsename_$1_$2`
   The result for `advanced.GVL.opc_global_input01` is `browsename_GVL_global_input01`.

7. **DisplayName / DisplayNameTarget**
   Enter the properties “DisplayName” and “DisplayNameTarget” to create the OPC UA DisplayName from the already filtered symbol path.

   ⇨ **Example:**
   DisplayName: `*\.(.*)\.opc_(.*)`
   DisplayNameTarget: `$2 from $1`
   The result for `advanced.GVL.opc_global_input01` is `global_input01 from GVL`.

8. Click **[Confirm]**.

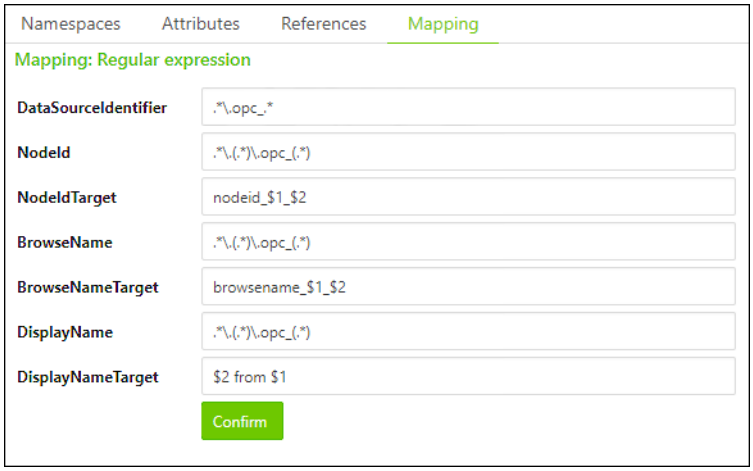


Figure 29: Use Regular Expressions

9. You can view the result of the variables filtered by regular expressions with an OPC client (view here as an example for the OPC UA client “UAexpert”).

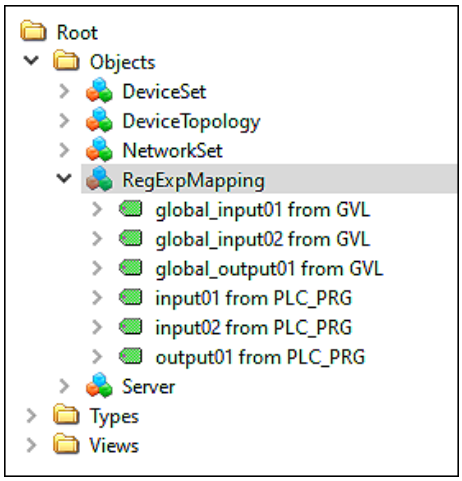
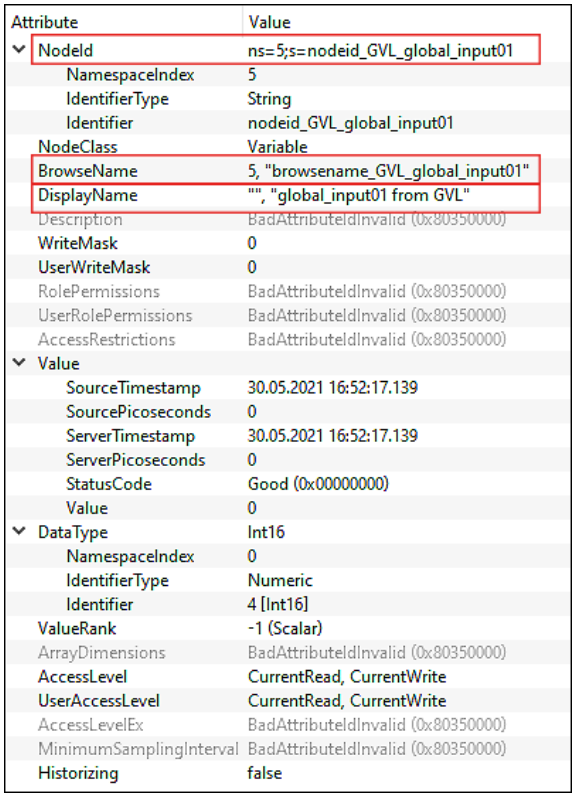


Figure 30: Hit List from the Regular Expressions (View in the OPC UA Client "UAexpert")

10. If you view the attributes, you will find the NodeId, BrowseName and DisplayName, for example. These were formed from the regular expressions that you entered for NodeID, NodeIdTarget, etc.



Attribute	Value
NodeId	ns=5;s=nodeid_GVL_global_input01
NamespaceIndex	5
IdentifierType	String
Identifier	nodeid_GVL_global_input01
NodeClass	Variable
BrowseName	5, "browsename_GVL_global_input01"
DisplayName	"", "global_input01 from GVL"
Description	BadAttributeIdInvalid (0x80350000)
WriteMask	0
UserWriteMask	0
RolePermissions	BadAttributeIdInvalid (0x80350000)
UserRolePermissions	BadAttributeIdInvalid (0x80350000)
AccessRestrictions	BadAttributeIdInvalid (0x80350000)
Value	
SourceTimestamp	30.05.2021 16:52:17.139
SourcePicoseconds	0
ServerTimestamp	30.05.2021 16:52:17.139
ServerPicoseconds	0
StatusCode	Good (0x00000000)
Value	0
DataType	Int16
NamespaceIndex	0
IdentifierType	Numeric
Identifier	4 [Int16]
ValueRank	-1 (Scalar)
ArrayDimensions	BadAttributeIdInvalid (0x80350000)
AccessLevel	CurrentRead, CurrentWrite
UserAccessLevel	CurrentRead, CurrentWrite
AccessLevelEx	BadAttributeIdInvalid (0x80350000)
MinimumSamplingInterval	BadAttributeIdInvalid (0x80350000)
Historizing	false

Figure 31: Result from the Regular Expressions (View in the OPC UA Client "UAexpert")

ⓘ Note

The OPC UA variables are generated at runtime!

The OPC UA variables are only generated by the OPC UA server at runtime. For this reason, they are not displayed in the OPC UA Mapping Editor under the selected OPC UA object in the information model.

Note

You can also filter using regular expressions without importing a symbol configuration!

The regular expressions are always used by the OPC UA server at runtime on the symbol configuration of the currently active application. For this reason, you can also use the OPC UA Mapping Editor directly to enter regular expressions without having to import a symbol configuration. You just have to export the mapping and load it into the controller. The regular expressions are applied to the currently active application at runtime.

7.7.7 Delete Mapping

- In the “Mapping” area, click the trash can icon for the mapping you want to delete.

⇨ The mapping is deleted.

Alternatively, select the variable in the OPC UA information model and select “No mapping” as the mapping type in the “Mapping” tab. Clicking **[Confirm]** deletes the mapping.

7.7.8 Export Mapping

1. Click the **[Export mapping]** button.

   ⇨ A dialog with export settings opens.

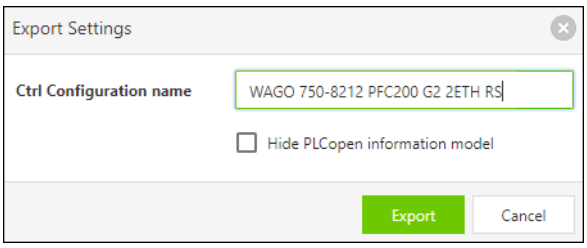


Figure 32: Export Mapping

2. Enter the name of the controller for which you are saving the mapping as “Ctrl Configuration name”.
   Note: Use the same name as in the WBM!
   You can find the name to be used in the WBM in the “OPC UA” area, “Ctrl Configuration name” field. Use the same name exactly when exporting!
3. To save the mapping in an XML file, click **[Export]**.
4. Select the location and click **[Save]**.

⇨ The controller can use the exported XML file to display the variables for its application in the modeled format.

⇨ **Note:** The XML structure of the export corresponds to the structure of the OPC UA information model, which has been expanded to include the corresponding structure from the symbol configuration (PLCopen-compliant).
If you do not want this, enable the option “Hide PLCopen information model” in the dialog to ensure that only the structure of the information model is used.

7.8 Load Mapping in the Controller

Open the Web-based Management of the controller.

1. Open the “Fieldbus” tab > “OPC UA” page.
2. If necessary, adjust the configuration of the controller under “Configuration”.
3. For example, enable the “Security Policy - None” and “Trust all clients” options if your client is working unencrypted or make changes to certificates (for more information on configuration, see the PFC200, 750-821x manual).
4. Open the “OPC UA” > “Information Model” page.
5. For the controller to use the new information model, first enable the “Feature enabled” option.
6. Click **[Submit]**.
7. In the field below, use **[Upload]** to select the XML file that you exported from the OPC UA Mapping Editor.

   ⇨ The server now displays the data in the new structure, which you can now also find in your client.

7.9 Use Shortcuts

Use key combinations to quickly access frequently required functions.

- To enter key combinations, press the keys stated in the following list. Keys to be pressed simultaneously are marked with a plus sign (+).

Table 9: Shortcuts

Function	Shortcut
New project	[CTRL] + [N]
Open project	[CTRL] + [O]
Save project	[CTRL] + [S]
Save project as...	[ALT] + [CTRL] + [S]
Exit	[ALT] + [F4]
Select base models	[CTRL] + [B]
Import information model	[CTRL] + [I]
Export information model	[CTRL] + [E]
Information model settings	[CTRL] + [comma]
Add instance	[ALT] + [A]
Edit instance	[ALT] + [E]
Delete instance	[DEL]
Add reference	[ALT] + [L]
Show help	[F1]

Appendix

8.1 Overview and mapping of OPC UA and CODESYS data types

In order to map exported IEC variables from CODESYS in the OPC UA Server, the compatibility of the different, available data types must be ensured. The following two tables list standard OPC UA data types and standard CODESYS data types. A third table shows the mapping between OPC UA and CODESYS data types.

Standard OPC UA data types

Table 10: Standard OPC UA data types

Data type	Description	Comment
Logic types		
Boolean	Truth value	TRUE, FALSE
Integral types		
SByte	Signed 8 bit integer	[-128, 127]
Byte	Unsigned 8 bit integer	[0, 255]
Int16	Signed 16 bit integer	[-2^15, 2^15 - 1]
UInt16	Unsigned 16 bit integer	[0, 2^16 - 1]
Int32	Signed 32 bit integer	[-2^31, 2^31 - 1]
UInt32	Unsigned 32 bit integer	[0, 2^32 - 1]
Int64	Signed 64 bit integer	[-2^63, 2^63 - 1]
UInt64	Unsigned 64 bit integer	[0, 2^64 - 1]
Floating point types		
Float	IEEE 754-1985 single precision floating point value	
Double	IEEE 754-1985 double precision data type	
String data types		
String	Unicode string	
ByteString	A sequence of byte values	
Time and date types		
DateTime	A date in the Gregorian calendar with time	
UtcTime	Time indication in coordinated universal time (UTC)	All time information between OPC UA servers and clients is in UTC. Clients should offer conversion to local time.
Duration	Specification of a time duration as a floating point value	*Double* value, that specifies a time interval in milliseconds. Sub-millisecond values can be represented by fractions.
TimeString	Represents a point in time as a string conforming to ISO 8601-2000	Uses 24 hour clock Simple format: [hh][mm][ss] Extended format: [hh]:[mm]:[ss]
DurationString	Represents a time duration as a string conforming to ISO 8601-2000	Format: P[n]Y[n]M[n]DT[n]H[n]M[n]S „P“ (period) introduces the specification. Example: „P3Y6M4DT12H30M5S“ = 3 years, 6 month, 4 days, 12 hours, 30 minutes, 5 seconds
User defined types		
Enumeration	Enumeration list	

Data type	Description	Comment
Structure	Structure, composition of fields	

Standard CODESYS data types

Table 11: Standard CODESYS data types

Data type	Description	Comment
Logic types		
BOOL	Truth value	TRUE, FALSE
BIT	1 bit logic value	1, 0
BYTE	8 bit logic value	[16#00, 16#FF]
WORD	16 bit logic value	[16#0000, 16#FFFF]
DWORD	32 bit logic value	[16#00000000, 16#FFFFFFFF]
LWORD	64 bit logic value	[16#0000000000000000, 16#FFFFFFFFFFFFFFFF]
Integral types		
SINT	Signed 8 bit integer	[-128, 127]
USINT	Unsigned 8 bit integer	[0, 255]
INT	Signed 16 bit integer	[-2^15, 2^15 - 1]
UINT	Unsigned 16 bit integer	[0, 2^16 - 1]
DINT	Signed 32 bit integer	[-2^31, 2^31 - 1]
UDINT	Unsigned 32 bit integer	[0, 2^32 - 1]
LINT	Signed 64 bit integer	[-2^63, 2^63 - 1]
ULINT	Unsigned 64 bit integer	[0, 2^64 - 1]
Floating point data types		
REAL	floating point value, 32 bit	IEC 60559 resp. IEEE 754-2008
LREAL	floating point value, 64 bit	IEC 60559 resp. IEEE 754-2008
String types		
STRING	Single byte character string	
WSTRING	Double byte character string	
Zeit- und Datumstypen		
DATE	Date	Prefix: DATE# or D# Example: DATE#1984-06-25 oder D#1984-06-25
TIME	Duration	Prefix: TIME# or T# Example: TIME#14ms T#14.7m T#5d_14h_12m_18s_3.5ms
LTIME	Long duration	
TIME_OF_DAY / TOD	Time of day	Prefix: TIME_OF_DAY# or TOD# Example: TIME_OF_DAY#15:36:30.123 oder TOD#15:36:30.123
DATE_AND_TIME / DT	Date and time	Prefix: DATE_AND_TIME# or DT# Example: DATE_AND_TIME#1996-05-06-15:36:30 or DT#1996-05-06-15: 36:30

Features

- A direct derivation of elementary types can be made..
  Example: `TYPE RU_REAL : REAL ; END_TYPE`
- Subsets of types (*Subranges*) can be used.
  Example: `TYPE ANALOG_DATA : INT (-4095..4095) ; END_TYPE`

Program Organization Units conforming to IEC 61131-3

- Functions
- Function blocks
- Programs

Mapping between OPC UA and CODESYS Data types

The following table shows the mapping between the data types used for CODESYS and OPC UA.

Table 12: Mapping zwischen OPC UA- und CODESYS Datentypen

CODESYS data type	OPC UA data type
BOOL	Boolean
BIT	Boolean
BYTE	Byte
WORD	UInt16
DWORD	UInt32
LWORD	UInt64
SINT	SByte
USINT	Byte
INT	Int16
UINT	UInt16
DINT	Int32
UDINT	UInt32
LINT	Int64
ULINT	UInt64
REAL	Float
LREAL	Double
STRING	String
WSTRING	String
DATE	DateTime
TIME	Int64
LTIME	Int64
TIME_OF_DAY / TOD	DateTime
DATE_AND_TIME / DT	DateTime
ENUM	Enumeration
STRUCT	Structure

8.2 Protected Rights

- Adobe® and Acrobat® are registered trademarks of Adobe Systems Inc.
- Android™ is a trademark of Google LLC.
- Apple, the Apple logo, iPhone, iPad and iPod touch are registered trademarks of Apple Inc. registered in the USA and other countries. “App Store” is a service mark of Apple Inc.
- AS-Interface® is a registered trademark of the AS-International Association e.V.
- BACnet® is a registered trademark of the American Society of Heating, Refrigerating and Air Conditioning Engineers, Inc. (ASHRAE).
- *Bluetooth*® is a registered trademark of Bluetooth SIG, Inc.
- CiA® and CANopen® are registered trademarks of CAN in AUTOMATION – International Users and Manufacturers Group e.V.
- CODESYS is a registered trademark of CODESYS Development GmbH.
- DALI is a registered trademark of the Digital Illumination Interface Alliance (DiiA).
- EtherCAT® is a registered trademark and patented technology licensed by Beckhoff Automation GmbH, Germany.
- ETHERNET/IP™ is a registered trademark of the Open DeviceNet Vendor Association, Inc (ODVA).
- EnOcean® is a registered trademark of EnOcean GmbH.
- Google Play™ is a registered trademark of Google Inc.
- IO-Link is a registered trademark of PROFIBUS Nutzerorganisation e.V.
- KNX® is a registered trademark of the KNX Association cvba.
- Linux® is a registered trademark of Linus Torvalds.
- LON® is a registered trademark of the Echelon Corporation.
- Modbus® is a registered trademark of Schneider Electric, licensed for Modbus Organization, Inc.
- OPC UA is a registered trademark of the OPC Foundation.
- PROFIBUS® is a registered trademark of the PROFIBUS Nutzerorganisation e.V. (PNO).
- PROFINET® is a registered trademark of the PROFIBUS Nutzerorganisation e.V. (PNO).
- QR Code is a registered trademark of DENSO WAVE INCORPORATED.
- Subversion® is a trademark of the Apache Software Foundation.
- Windows® is a registered trademark of Microsoft Corporation.

List of Figures

List of Tables

WAGO GmbH & Co. KG
Postfach 2880 · D - 32385 Minden
Hansastraße 27 · D - 32423 Minden
✉ **info@wago.com**
www.wago.com

Headquaters	+49 571/887 – 0
Sales	+49 (0) 571/887 – 44 222
Order Service	+49 (0) 571/887 – 44 333
Fax	+49 571/887 – 844169