Software manual | EN

OPC UA

Content

Content

About this documentation

Version	Date
2.0	2023-12-13

This documentation is part of the "Controller-based Automation" manual collection. It consists of the following sets of documentation:

Documentation type	Subject
Product catalogue	Controller-based Automation (system overview, sample topologies) Lenze controllers (product information, technical data)
System manuals	Visualisation (system overview/sample topologies)
Communication manuals Online helps	Bus systems • Controller-based Automation EtherCAT® • Controller-based Automation CANopen® • Controller-based Automation PROFIBUS® • Controller-based Automation PROFINET®
Reference manuals Online helps	Lenze controllers: • Controller 3200 C • Controller c300 • Controller p300 • Controller p500
Software manuals Online helps	Lenze engineering tools: • »PLC Designer« (programming) • »Engineer« (parameterisation, configuration, diagnostics) • »VisiWinNET® Smart« (visualisation) • »Backup & Restore« (backup, restore, update)

Current sets of documentation and software updates with regard to Lenze products can be found in the Download area at:
www.lenze.com

Target group

This documentation is intended for all persons who plan, program and commission a Lenze automation system on the basis of the Lenze FAST Application Software.

Conventions used

This documentation uses the following conventions to distinguish between different types of information:

Type of information	Highlighting	Examples/notes
Numeric notation		
Decimal separator	Point	The decimal point is always used. For example: 1234.56
Hexadecimal number	0x	For hexadecimal numbers, the "0x" prefix is used. Example: 0x60F4
Text		
Program name	» «	»PLC Designer« ...
Variable names	*italics*	By setting *bEnable* to TRUE...
Function blocks	**Bold**	The **L_MC1P_AxisBasicControl** function block ...
Function libraries		The **L_TT1P_TechnologyModules** function library...

Source code	Font
	"Courier new"
		...
		`dwNumerator := 1;`
		`dwDenominator := 1;`
		...

Hyperlink	underlined	Optically highlighted reference to another topic. It is activated with a mouse-click in this online documentation.
Symbols		
Step-by-step instructions		Step-by-step instructions are indicated by a pictograph.

Variable names

The conventions used by Lenze for the variable names of Lenze system blocks, function blocks, and functions are based on the "Hungarian Notation". This notation makes it possible to identify the most important properties (e.g. the data type) of the corresponding variable by means of its name, e.g. *xAxisEnabled*.

System

In this chapter you will learn basic information about

- the OPC UA standard,
- the use cases of OPC UA and
- the technical nature of OPC UA.

OPC UA Standard

OPC UA (Open Platform Communications Unified Architecture) is a globally recognized communication framework that is standardized by the IEC 62541 series of standards. It is currently the most promising standard for the implementation of Industry 4.0 communication in which machine data can be exchanged regardless of manufacturer and platform.

Of particular importance is the extensibility of OPC UA in the sense of a modeling framework, so that any industry-specific standards can be derived based on existing OPC UA standards. The architecture of OPC UA is service-oriented and thus modular and scalable.

With the help of profiles, OPC UA offers different combinations of services for establishing, securing and managing industrial communication from small embedded systems to PC and cloud platforms with virtually unlimited resources. OPC UA is represented as a standard by the OPC Foundation.

- https://opcfoundation.org/

The Lenze white paper on OPC UA also provides a further overview.

- Lenze Whitepaper

Use cases

OPC UA can be used at all levels of the automation pyramid. With the increasing influence of Industrie 4.0 approaches, these levels are becoming blurred, and OPC UA is a suitable communication standard to meet this need.

Significantly, there are the following use cases:

- Connection of a machine visualization
- Connection of the machine to MES, SCADA, Cloud and Edge
- Machines to machines communication
- Machines to field devices communication

Transport

OPC UA has two different communication modes.

Client/Server

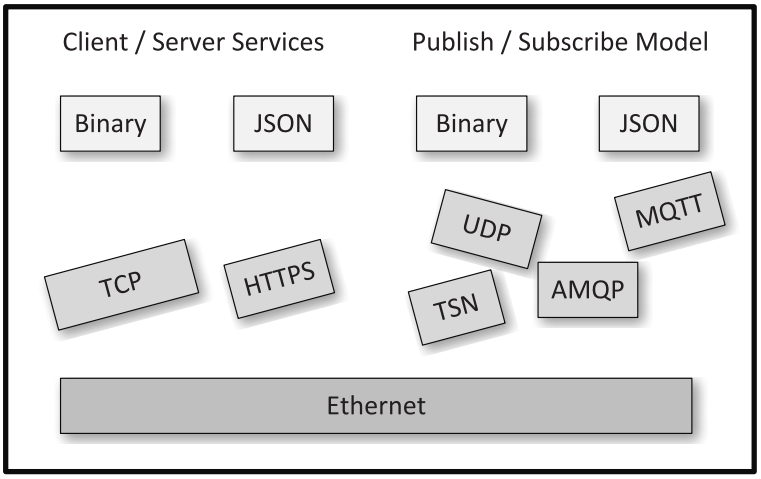
A client accesses information from the server over a fixed 1:1 connection.

Publisher/Subscriber

A publisher (1) sends information to any number (n) of subscribers without a fixed connection. This type of communication represents a 1:n connection.

Different codes and protocols exist for both types of communication. For better clarification, it is shown how the technologies basically build on each other or complement each other.

The frequently asked question of whether, MQTT is better suited for an application or OPC UA is superfluous. OPC UA and MQTT are complementary technologies and not mutually exclusive approaches.



Security

Industrial communication is subject to a wide variety of attack scenarios. The attacks occur on different communication levels. Accordingly, OPC UA by design takes into account several different security aspects to ensure full data and communication security - see also the OPC UA Specification Part 2.

- Confidentiality and integrity
- Application and user authentication
- Audit capability

Information access

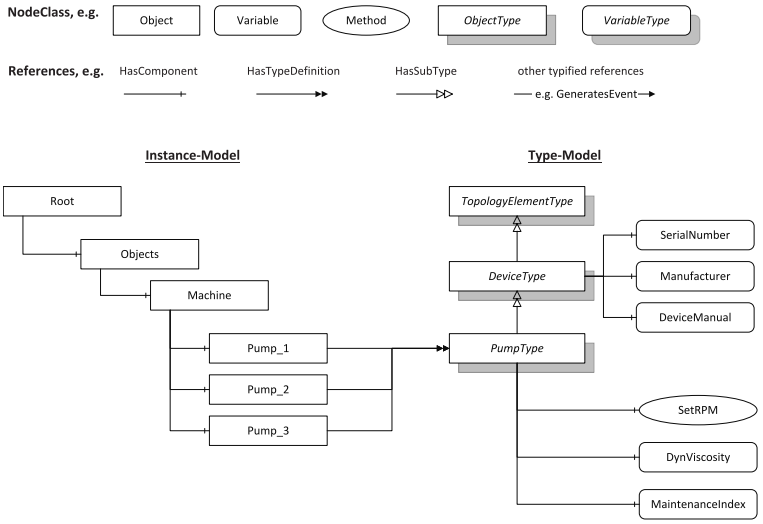
OPC UA makes it possible to represent the data and the behavior of systems, e.g. of a machine or plant, in information models, which in turn make this information available to other OPC UA applications via the following basic services:

- Browse/explore information model (browse)
- Read and write the variables depending on the access rights (read/write)
- Subscribe to changes in variable values (subscribe) and events (events), which eliminates the need to continuously query for new values, and
- Call methods and programs on the server side (methods/programs) to initiate actions directly.

Information Modeling and Companion Specifications

By means of object-oriented mechanisms, even complicated, multi-layered structures and their behavior can be clearly mapped and systematically extended.

A simple example of the definition and application of an information model that uses some essential descriptive elements is that of an intelligent pump. This not only generates a volume flow, but also measures inline the dynamic viscosity of the fluid in centipoise (cP). In addition, a maintenance index is used to indicate the need for maintenance. If the viscosity exceeds or falls below a certain limit, an alarm is automatically signaled.



The information models shown in the figure show the derivation of the own special pump type from the general OPC UA type "DeviceType" and the use, i.e. the location, of the pumps within the address space of the OPC UA Server of an exemplary machine or plant.

OPC UA information models basically consist of nodes and references. References represent the relationship between two nodes that describe something specific. A distinction is made between an instance model and a type model, whereby the object-oriented mechanisms such as inheritance can be used. The following constructs are available for modeling.

Node (NodeClass)

- Object & ObjectType
- Variable & VariableType
- MethodType

References (ReferenceType)

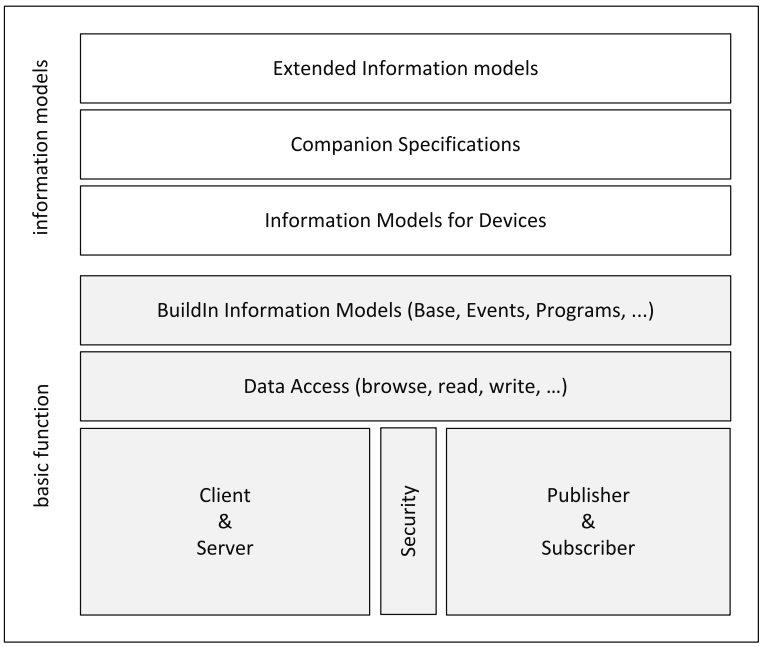
- HierarchicalReferences, e.g.
  - HasComponent
  - HasProperty
  - HasSubType
- NonHierarchicalReferences, e.g.
  - HasTypeDefinition

The nodes and references required to create the pump's information model in the very simple example above suggest, that it is neither easy nor efficient to create information models from scratch.

The following figure shows the logical layers of OPC UA modeling, which ¬favor a strong reuse of information models. The information models themselves can be built up in different ways. Either they are modeled from scratch, or they are built on top of each other.

The basis of many information models is the Information Model for Devices, which is described in Part100 of the OPC UA Specification. This information model is used by many Companion Specifications that standardize information models for a specific domain. Examples are the PLCopen Companion Standard for general IEC applications or the Euromap for injection molding machines.

Beyond the standard, the resulting information model in the OPC UA Server can still be enriched with manufacturer-specific models.



Lenze system overview

The following devices and engineering tools support OPC UA in the Lenze system. The platform to be selected depends on the application. Specific information on handling OPC UA can be found in the respective documentation for the Lenze components.

»EASY Starter«

- OPC UA Server
  - As a pure software solution for a Windows PC, this platform provides an OPC UA gateway to Lenze device parameters that can also be accessed with the »EASY Starter«. Especially for devices that do not have an integrated OPC UA server, this brownfield solution provides an OPC UA access option.

Controller of the c400 and c500 series

- OPC UA Server
  - Via an integrated OPC UA server on the controllers of the c-series, variables of the PLC can be published via OPC UA according to IEC 61131. Thus, for example, visualization connections or also MES, SCADA or cloud connections can be realized that require easy access to PLC variables.
  - It is also possible to map the PLC variables to user specific information models, which can be derived from Companion Specifications. Beside variables, also OPC UA methods and events can be mapped to the PLC application.
- OPC UA Client
  - To establish a connection from the PLC application to external entities via OPC UA, an OPC UA Client according to PLCopen is provided. With this, use cases like Controller to Controller or Controller to any external data source can be solved.
- OPC UA PubSub
  - For Controller to Controller or Controller to multiple receiver communication the OPC UA PubSub feature is provided. With this it is possible to communicate PLC data cyclic without a dedicated connection.

Controller of the c300 and 3200c series use the OPC UA Server from Codesys. The corresponding documentation can be found at the following web link:
▶ CODESYS Online Help

Lenze licenses

The Lenze OPC UA features are with following licence concept available.

Available features without an OPC UA license:

- One external Client and Server session on each target.
- An application specific information model on each target.

Additional features are available with a target specific license:

- Second and more external Client and Server sessions.
- PubSub feature.

The license configuration depends on the OPC UA target. In case of a Lenze Controller the licence can be configured by the Lenze License Manager in the »PLC Designer«.
Further information can be found in the OPC UA documentation for the target.

Mapping Editor

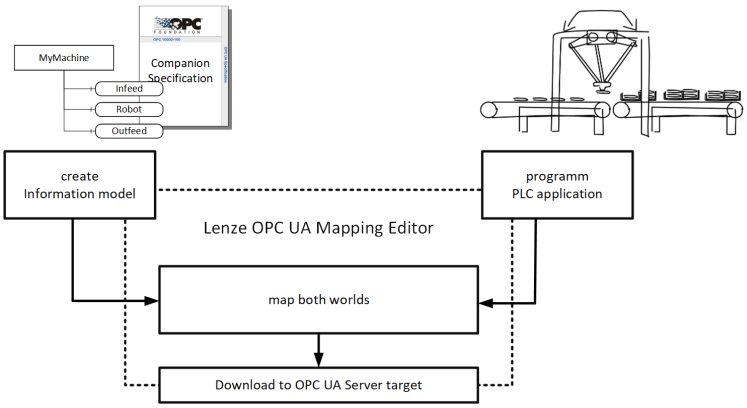
Overview

General information about the OPC UA technology can be found in the Lenze OPC UA system documentation.

Mapping Editor use cases and general workflow

With the Lenze Mapping Editor OPC UA information models based on companion specifications can be created. The created information model can be mapped to target specific data sources like the PLC application.

In general, the creation of an OPC UA information model with the Lenze Mapping Editor is independent of the machine application. Thus, it is possible to develop both worlds separatey and map them together in a final step.

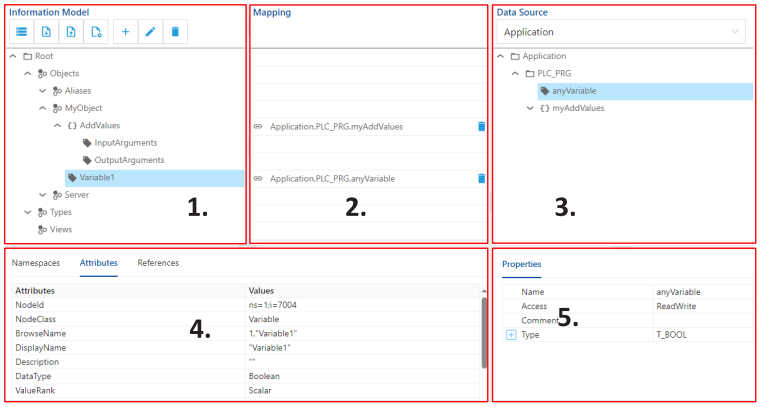


This brings advantages when connecting existing applications with a new user specific OPC UA information model. In this "brownfield scenario" for example the PLC application already exists and operates the functionality of the machine over years. Now, this machine shall get an OPC UA interface to become Industry 4.0 ready.

In addition, there are advantages if two different people develop the OPC UA information model and the PLC application. For example, an end customer needs a specific OPC UA information model for all the used machines in the plant. In this case the end customer provides the information model in form of an OPC UA Nodeset and in parallel the machine builder can setup the machine application.

Mapping Editor main screen overview

The Lenze Mapping Editor is separated into following areas.



1. OPC UA information model
   This area is used to create and edit the OPC UA information model.
2. Mapping
   This area is used to map data source items to OPC UA nodes of the information model.
3. Data source
   This area provides the mappable data source items of the target system.
4. OPC UA information model and mapping attributes
   This area provides OPC UA specific attributes and mapping information of a selected node.
5. Data source attributes
   This area provides data source specific attributes of a selected data source item.

Mapping Editor Engineering Access Rights

The Lenze Mapping Editor is embedded in the User and Access Rights Management of the host system.

Engineering Access Rights in PLC Designer

In case of the »PLC Designer« as host system, the User and Access Rights Management of the »PLC Designer« is used. In general, the »PLC Designer« provides a PLC project depending user management for the engineering tool itself and the integrated tool functionalities.

The designated administrator of the PLC project can configure the access rights via the context menu of the Mapping Editor Node in the device tree of the »PLC Designer«.

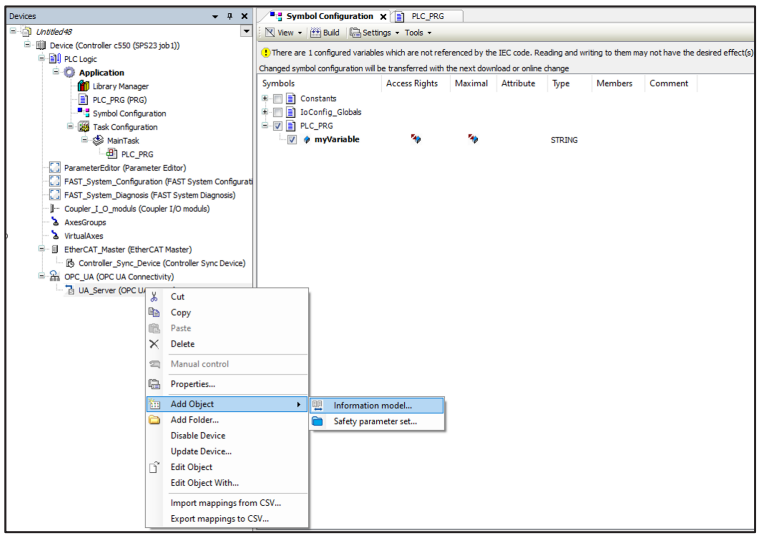
For following functions, the Access Right "granted explicitly", "not specified, but granted by default", "denied explicitly" or "not specified, but denied by default" are settable for User Groups via the Access Control menu. The "not specified, but…" Access right is an inherited right from the parent node.

Function	Access Right	Short Description
View	granted	• The Mapping Editor tab is visible in the »PLC Designer«. • The Information Model and the Data Sources are visible with their Attribute sections and the mapping of both sides. • It's possible to scroll through the trees and open subordinated Dialogs of the Attribute sections.
	denied	• It's not possible to open the Mapping Editor, the whole tab is not visible.
Modify	granted	• The "View" function is granted. • It's possible to use all Mapping Editor commands like base model selection, add & edit OPC UA Nodes and References, select Data Sources and editing the mapping.
	denied	• The "View" function is granted. • It's not possible to adjust any settings of the Information Model or the Mapping.
Remove	granted	• It's possible to delete the Mapping Editor Node in the device tree of the »PLC Designer«. In this case also the designed OPC UA information model and the mapping will be removed / deleted.
	denied	• It's not possible to remove / cut or delete the Mapping Editor Node in the device tree of the »PLC Designer«.

Setup Mapping Editor in the »PLC Designer«

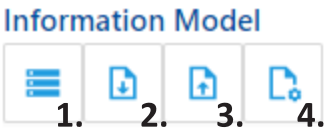
The following chapters describe the workflow to setup a user specific information model based on companion specifications.

Add Mapping Editor for user information model



Commands to setup the user information model

The Mapping Editor provides following commands to setup the user information model.



1. Select base models
2. Import existing user model
3. Export user model
4. User model settings

User model settings

The user information model will be created in an own namespace. In general, namespaces are used by OPC UA to create unique identifiers across different naming authorities.

The following information model settings define the user model namespace uniquely. These settings are visible in the user information model nodeset.xml as well as on the target OPC UA Server in the namespace meta data.

Model URI

The Model URI (Uniform Resource Identifier) [String]-parameter contains the unique namespace identifier. To ensure uniqueness, an internet domain is a good idea.

For example: "http://yourorganisation.org/user_model"

Version

The Version [String]-parameter defines version information for the model. Currently there is no known prescribed schema for the version number. Usually, a dot notation is used.

For example: "1.0.0"

Publication date

The Publication date [DateTime]-parameter provides the publication date of the information model version.

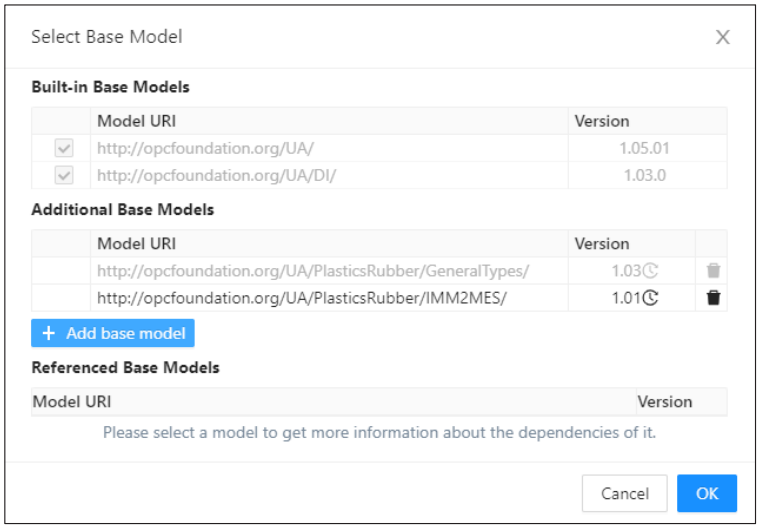
For example: "21.11.2022"

The settings dialog is available via a button in the information model area of the Mapping Editor.

If an existing user model is imported (Import existing user model (📖 20)) in the Mapping Editor, the settings of the existing model are used.

Select base models

OPC UA information models are based on each other. To use type definitions of existing models the corresponding nodesets.xml's have to be loaded as base models.



The Lenze Mapping Editor differentiates between already used models on the server target (build-in models) and external models (additional models).

The built-in models are used by the OPC UA Server to model Lenze specific services.

- These models are listed in the "Built-in Base Models"-table and can't be deleted.
- The versions of these models are fix with the host version.
- The built-in models can be selected as base models by checkbox.
- The http://opcfoundation.org/UA/ model is always activated.

The additional models can be external companion specifications which shall be used for the user specific information model.

- These models can be added via an operating system specific file dialog by click on the "Add base model" – button.
- These models can be deleted by click on the "garbage bin"-icon.
- These models are updateable via an operating system specific file dialog by click on the "update" – icon. It is only possible to select a new information model with the same Namespace URI and a sufficient version for other loaded base models.

Built-In and additional base models are grayed-out and not delete- or updateable, if they are referenced by other loaded base models.

Import existing user model

In some cases, an information model already exists. For example, an end customer has created an expected model. The task of the machine developer is now to map the machine application to this model.

Thus, it is optional possible to load an existing user information model with the Lenze Mapping Editor. It can be imported by a host specific "File"-Dialog. In chapter "Create OPC UA information model (📖 21)" it is described how to edit the loaded user specific information model if necessary.

Export user model

The Lenze Mapping Editor focus on the development of the OPC UA Server information model. For the development of a corresponding client application, the offline description of the expected Server information model is helpful.

Thus, it is possible to export the user information model.

Namespace overview

The used namespaces are listed in the namespace overview of the Mapping Editor. The built-in namespace http://opcfoundation.org/UA/ [0] and the user model namespace [1] have a fix namespace Index.

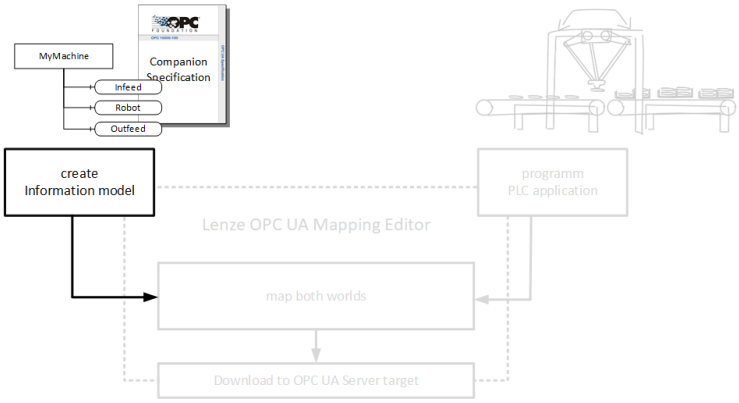
The additional loaded namespaces are listed in terms of their hierarchical dependencies.

Create OPC UA information model

An OPC UA information model is to provide data in a structured, standardized and semantic, also human understandable way.

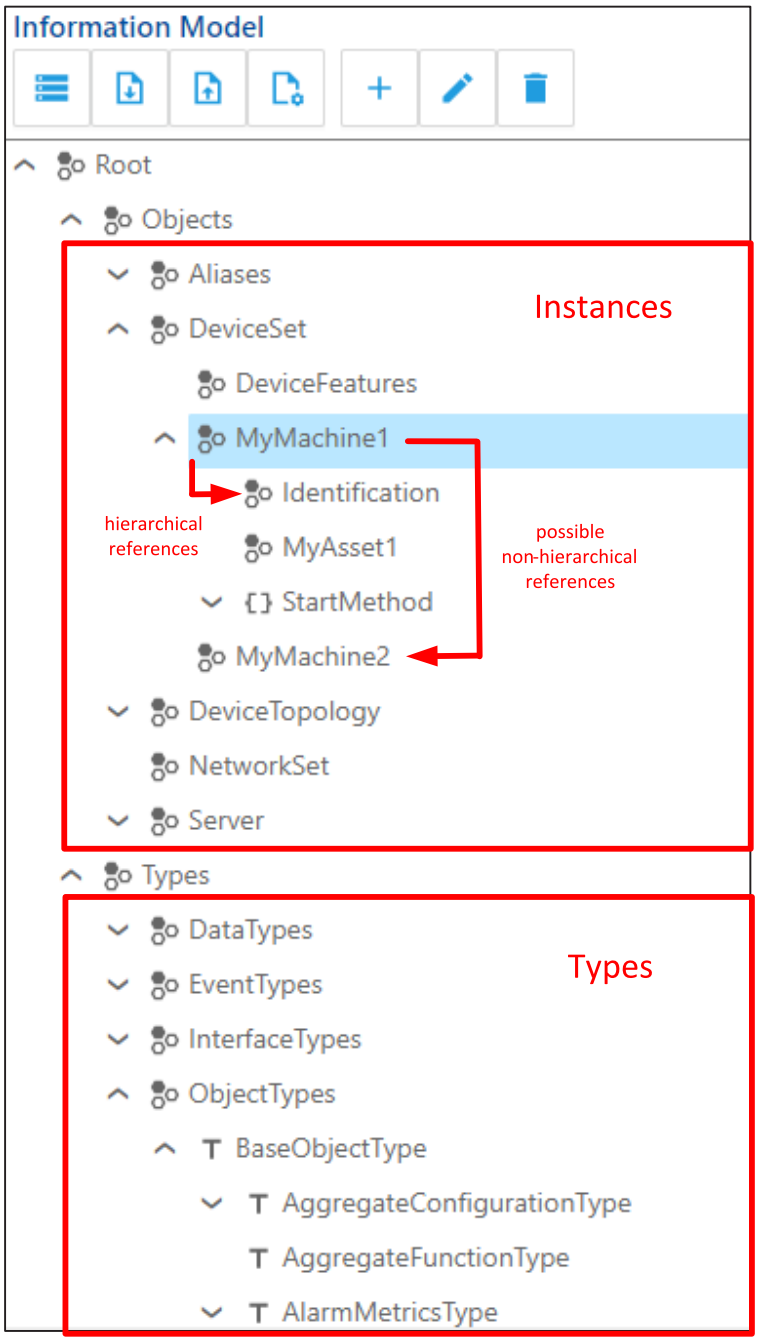
With the Lenze Mapping Editor it is possible to create own user specific information models based on Companion Specifications. It is also possible to import and edit an existing information model created by another tool, e. g. the UA Modeler from Unified Automation.

Due to a standardized description of these information models in form of an OPC UA Nodeset (.xml) it is possible to exchange them between engineering tools.



The OPC UA information model is a mesh of nodes and references.

- Node instances can be added under the "Object" folder in the information model.
- Node types are listed under the "Type" folder in the information model. To create types is currently not supported.
- References are divided into hierarchical and non-hierarchical references. They are used to expose different semantics on how the nodes are connected.



The OPC UA information model is a mesh of nodes and references. Even if a single node has two hierarchical references and is seeming twice in the information model, the node exists only once.

Commands

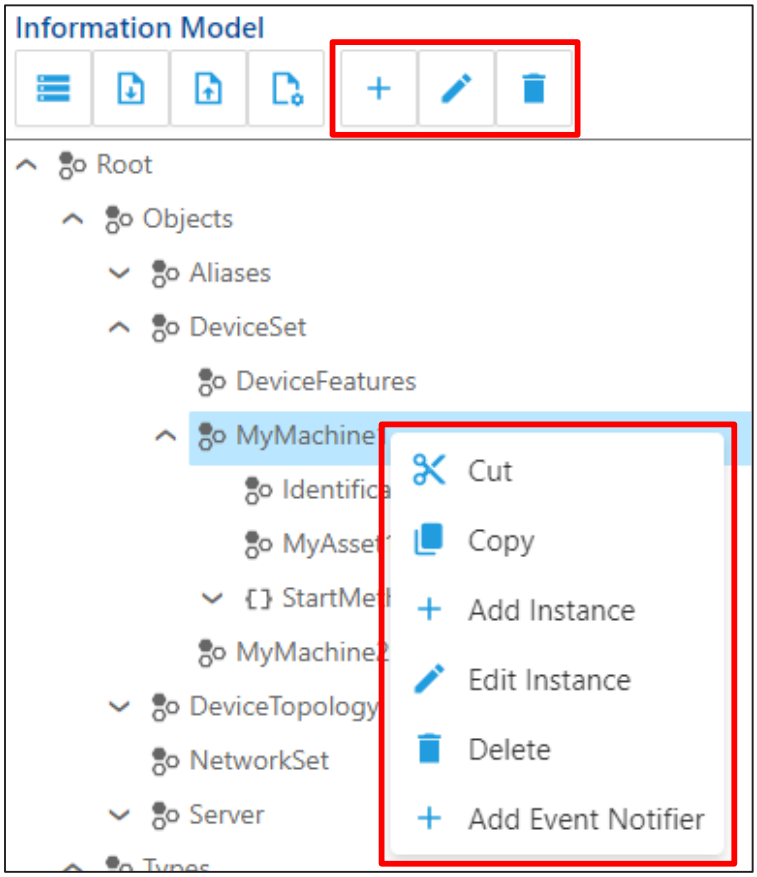
The Mapping Editor provides commands to create the user specific OPC UA information model via the menu bar of the information model section and the context menu of an individual node.

Commands of the menu bar:

- Add Instance
  Adds a new child node under the selected parent node
- Edit Instance
  Edits the parameter of a selected node
- Delete Node
  Deletes the selected node

Commands of the context menu of a selected node:

- Cut
  Copies the selected node and all child nodes and deletes those
- Copy
  Copies the selected node and all child nodes
- Paste
  Pastes the copied node and child nodes
  If the previously selected node is cut, the pasted node(s) has the same name(s) and NodeId(s) from this.
  If the previously selected node is copied, the pasted node(s) has new NodeId(s) and the parent node has an enhanced name e. g. "_1".
- + Commands of the menu bar



The Mapping Editor provides specific commands for the modeling of OPC UA methods. For further information how to model OPC UA methods with the Lenze Mapping Editor refer to chapter Methods (📖 80).

If a method node is selected, the context menu provides additional commands to create Input and OutputArguments property nodes.

- Add InputArguments
- Add OutputArguments
- \+ Above mentioned general commands.

If an InputArguments or OutputArguments node is selected, the context menu provides commands to define the arguments.

- Add Argument
- Delete Arguments

If an Argument is selected, the context menu provides commands to edit an individual argument.

- Edit Argument
- Delete Argument

On Nodes from base models only the Add Node command is usable. Nodes from base models are fix and not editable.
The first hierarchy of the OPC UA Information model is standardized. Thus, no commands are useable on the Root node.

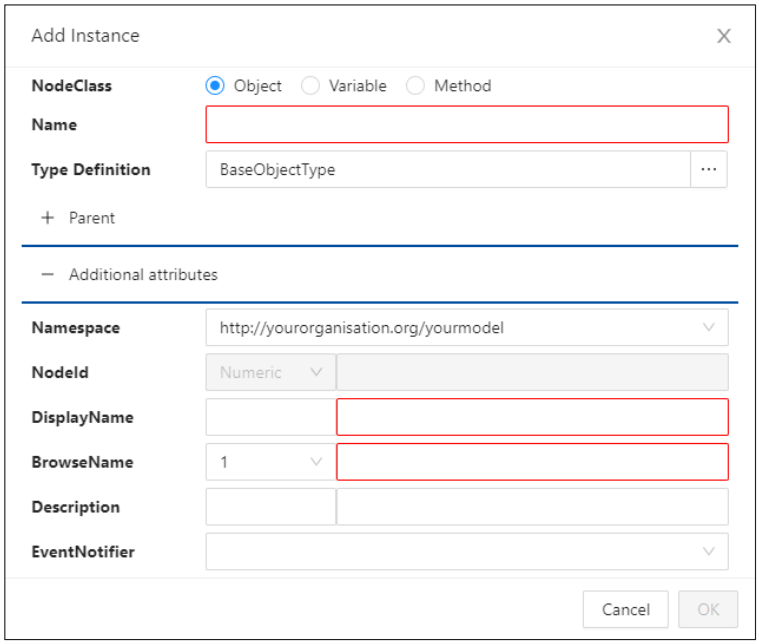
- Instances of Objects and Variables are addable and editable in the user specific information model under the Objects folder.
- Adding types is currently not supported.
- Views are currently not supported.

Add / Edit / Delete Instances

The Mapping Editor provides one generic dialog for adding and editing Instance nodes. For this purpose, only the NodeClass depending parameters appear. Following table gives an overview.

Parameter	NodeClass		
	Object	Variable	Method
Base instance parameter			
NodeClass	x	x	x
Name	x	x	x
Type definition	x	x	
DataType		x	
Relationship to the parent node			
Parent Name	x	x	x
Parent Reference	x	x	x
Additional attributes			
Namespace	x	x	x
NodeID	x	x	x
DisplayName	x	x	x
BrowseName	x	x	x
Description	x	x	x
EventNotifier	x		
Value		x	
ValueRank		x	
ArrayDimensions		x	
AccessLevel		x	
UserAccessLevel		x	
MinimumSamplingIntervall		x	
Historizing		x	

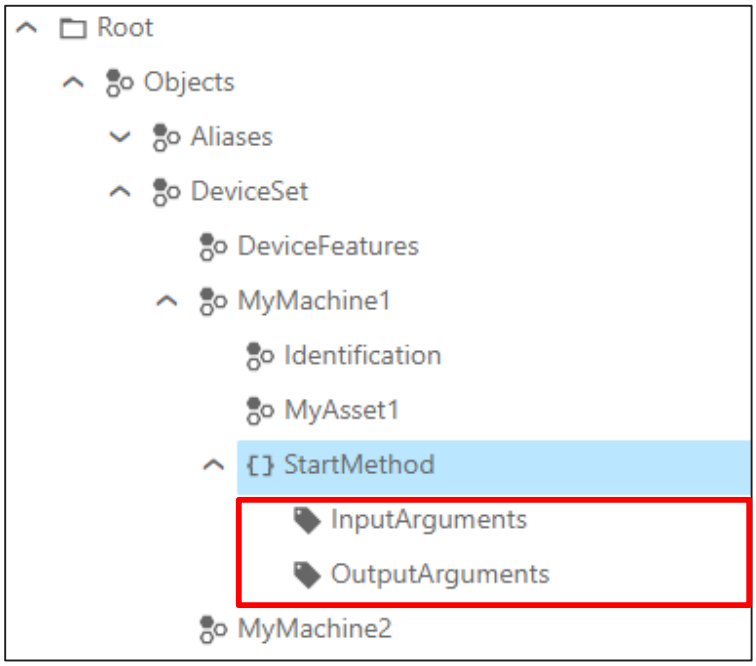
Object, Variable and Method instance nodes are add- and editable by the following dialog.



Add Instance: Methods

InputArguments and OutputArguments

If a new method is added, InputArguments and OutputArguments property nodes are created automatically by the Mapping Editor. These property nodes for methods are specified by the OPC UA core specification and used to represent the input and output arguments of the method in the information model.



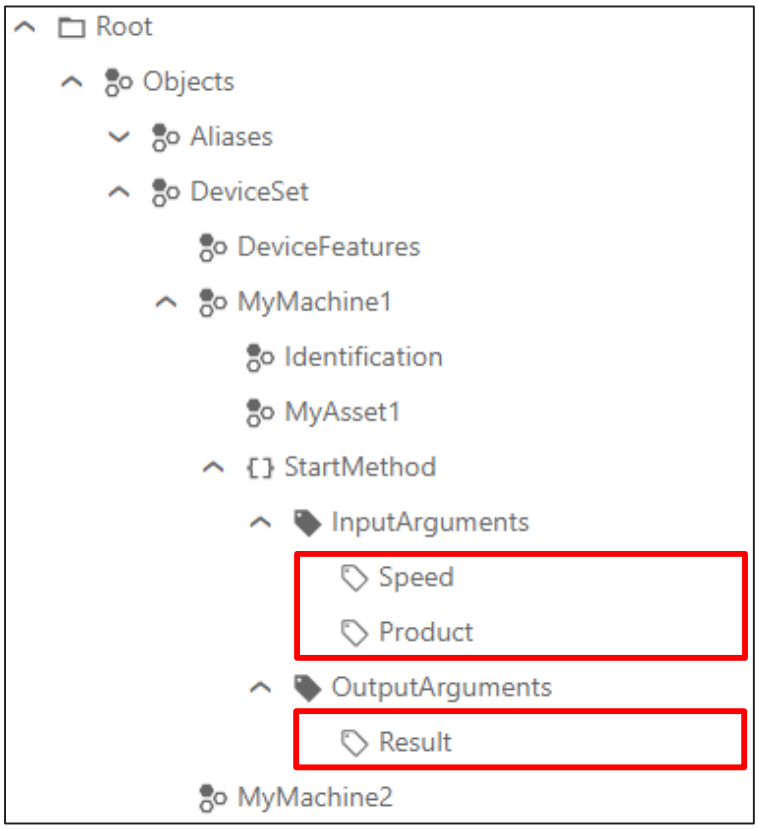
If the method node is selected, the context menu provides additional commands to create Input- and OutputArguments property nodes. This is useful, if the property nodes are deleted inadvertently.

- Add InputArguments
- Add OutputArguments

The NodeClass, the Name, the Type and the DataType as well as the reference to the parent (method) node are fix. Thus, no Add InputArguments or Add OutputArguments dialog is available.

Single input and output argument

The value of an Input- or OutputArguments property is a list of arguments. This list of arguments is represented in the information model tree of the Lenze Mapping Editor, although they are not OPC UA nodes and do not have a NodeId.



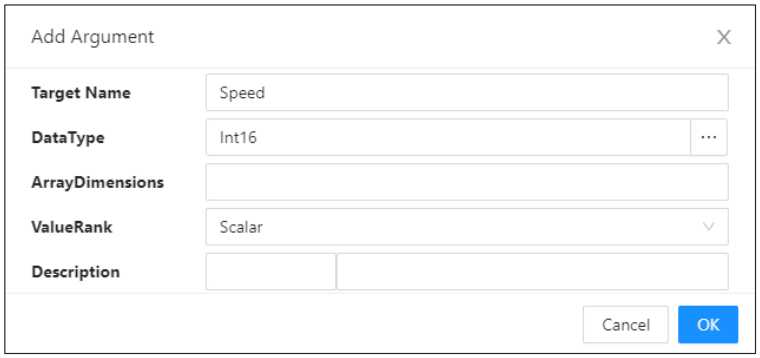
A single argument is not an instance in the OPC UA information model. The representation below an "InputArguments" or "OutputArguments" variable has only usability purposes.

Thus, a single argument does not have an OPC UA NodeID.

If a single argument is selected, the attributes of this argument are represented by the Attributes table.

If an InputArguments or OutputArguments property node is selected, the context menu provides commands to add individual arguments.

- Add Argument
- Delete Arguments



If a created Argument is selected, the context menu provides commands to edit it.

- Edit Argument
- Delete Argument

"Add & Edit Instance" dialog parameter

NodeClass

The NodeClass parameter of the Mapping Editor corresponds to the NodeClass definition of the OPC UA specification.

- https://reference.opcfoundation.org/v104/Core/docs/Part3/5.1/

The NodeClass parameter defines, which NodeClass shall be added as child to the selected parent Node in the Information Model. The additional attributes for the new instance are NodeClass depending.

- "Object", adds a new Object that are used to structure the information model.
- "Variable", adds a new Variable to represent a value.
- "Method", adds a new Method that can be called by a Client and returns in a result.
- There are some constraints regarding the possible NodeClass of the addable child nodes. Following table provides an overview with the hierarchical references used by default.

Parent / Source NodeClass	**Child / Target NodeClass**			
	Object	**Variable (DataVariable Type)**	**Variable (Property Type)**	**Method**
Object	Organizes	HasComponent	HasProperty	HasComponent
Variable (DataVariable Type)	Not allowed	HasComponent	HasProperty	Not allowed
Variable (Property Type)	Not allowed	Not allowed	Not allowed	Not allowed
Method	Not allowed	Not allowed	HasProperty	Not allowed

For more information about the differences between DataVariables and Properties Variables refer to chapter Annex (📖 56).

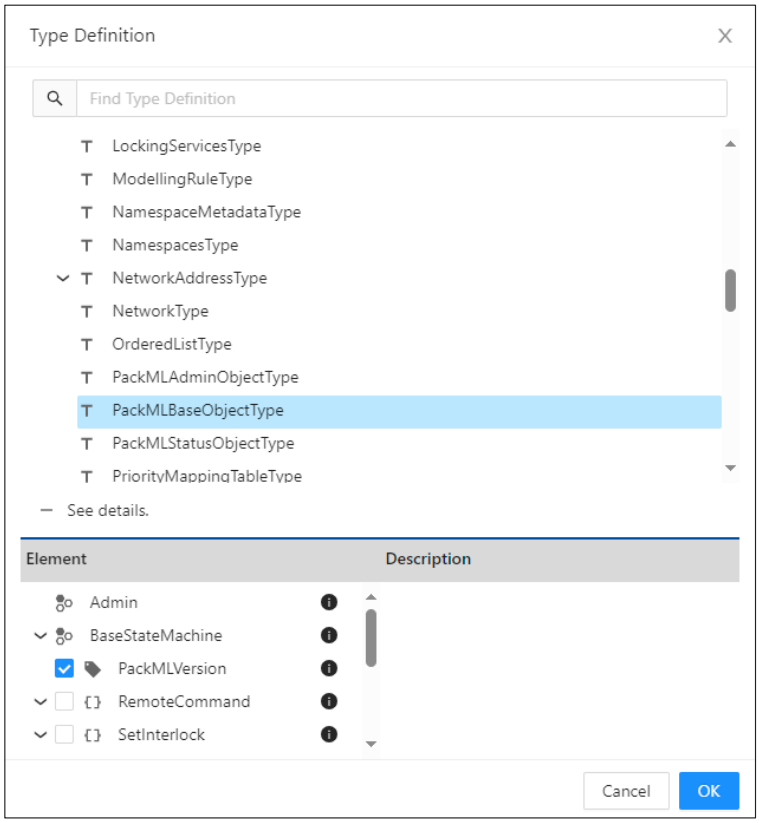
Name

The Name parameter is the *SymbolicName* for the new Node in the Nodeset xml-File. The *SymbolicName* does not appear in the Server AddressSpace and is intended for use by design tools only. Only letters, digits or the underscore ("_") are permitted.

- The *DisplayName* is used to represent the Node in the information model of the Mapping Editor. In general, the *DisplayName* shall be used by Clients to represent the Node to a user, e. g. via a visualization.
- The *BrowseName* is used when browsing the AddressSpace to create paths out of BrowseNames.
- The *DisplayName* and *BrowseName* parameters of the "Add Instance" dialog are automatically taken from this Name parameter, but they can also be overwritten by the user.

Typedefinition

The *Typedefinition* parameter defines the OPC UA Type definition of the new Instance. The desired type definition can be selected from a type definition hierarchy of all loaded base models.



In case of an Object instance the provided type definitions are from the BaseObjectType hierarchy.

In case of a Variable instance the provided type definitions are from the BaseVariableType hierarchy. The first hierarchy level is the decision between DataVariable type and Property type. For more information about the differences refer to chapter 7.2.

Element

If the suitable Type for the new instance is selected, the hierarchical elements of the selected one are listed in a details section. Also optional (markable) and mandatory (greyed out) child nodes of the listed types are presented. If necessary for the use case the customer is able to mark the optional ones. All marked child Nodes will be added to the new Instance.

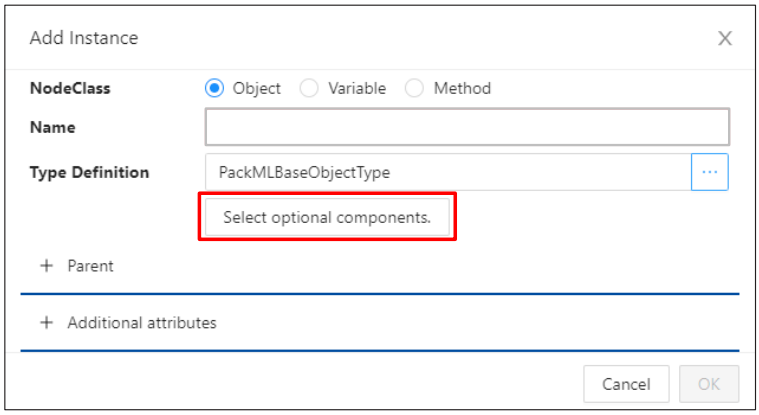
Description

For further information, the "Description" of the selected Type is presented, if it is provided by the base model.

OPC UA introduces abstract Type definitions. Types, which are abstract are only used to structure the Type hierarchy. It isn't allowed to create an instance from abstract Type definitions. Abstract Type definitions are marked in italic font.

Typedefinition – Select optional components

If the used Type definition provides optional child nodes, a button for a selection dialog appears.



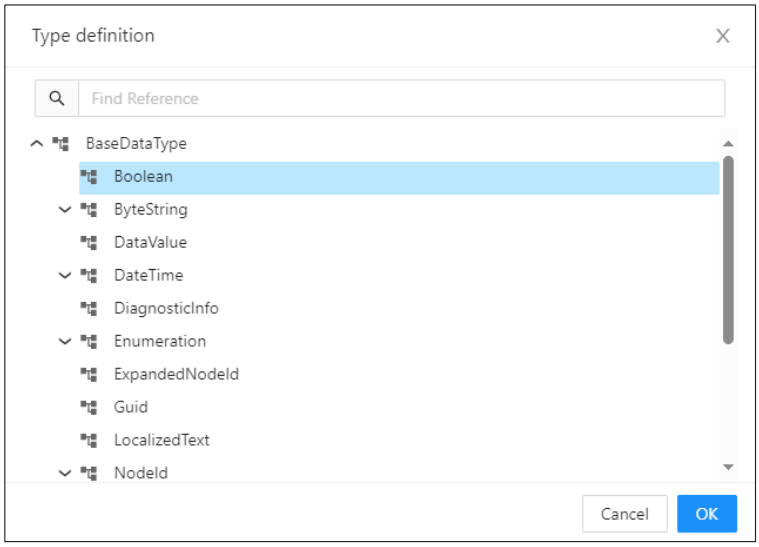
Via this dialog the optional child nodes can be selected to be added under the new instance node. Mandatory nodes are pre-selected and greyed out. The mandatory nodes have to be part of the new instance as specified by the used CompanionSpecification.

It is the same configuration as can be done by the Typedefinition dialog, just with easy access to the selection.

It is the same configuration as can be done by the Typedefinition dialog, just with easier access via the main "Add Instance" dialog.

DataType

The DataType parameter specifies the type definition of the value of a variable node. OPC UA introduces some BuiltIn DataTypes, structures and enumerations. Also additionally loaded base models bring derived DataTypes with them.



Parent Name

The "ParentName" represents the *DisplayName* of the selected Node in the information model of the Mapping Editor. Under this node the new instance will be added as child.

Parent Reference

Defines the hierarchical reference from the parent node to the new child node.

Depending on the NodeClass of the parent node following References and their subtypes are supported.

Parent NodeClass	Child NodeClass		
	Object	Variable	Method
Object	Organizes (default) HasComponent	Organizes HasComponent (default) HasProperty	HasComponent (default)
Variable	Not allowed	HasComponent (default) HasProperty	Not allowed
Method	Not allowed	Not allowed	Not allowed

Organizes ReferenceType

Used to organize Nodes in the AddressSpace. The parent node shall be an Object (mainly of FolderType or subtype definition). The Child node can be of any NodeClass.

HasComponent ReferenceType

The semantic of a HasComponent reference is a is -part-of relationship. It is mainly used between Objects as parent node and their containing (Sub-)Objects, DataVariables and Methods.

It is also usable to structure complex variable nodes, see chapter Structured DataType and a complex VariableType (📖 57).

HasProperty

The HasProperty reference is only used to identify the Properties of a node. The parent node can be of any NodeClass. The Child node is a Variable defined as Property.

Properties shall not have (Sub-) Properties. Thus, a Property node shall not be the Parent Node of this reference.

For further information about Properties see also chapter Mapping Editor (📖 14).

Namespace

The Namespace parameter defines the namespace URI in which the new Node shall be added.

Selected base models are fix and can't be modified by the Mapping Editor. Only the customer model can be enhanced or modified.

The namespace URI [String] is a global unique identifier of the namespace.

The namespace URI of the user model is editable as mentioned in chapter User model settings (📖 18).

If the information model is represented by an engineering tool (e.g. the Mapping Editor) or if the information model is represented "online" in a server AddressSpace, an namespaceIndex [UInt16] is additionally provided. It supports an easier handling, but it is only fix (and unique) on the current tool or server application.

NodeID

The NodeID is used to define a specific unique identifier for the new node. In general, OPC UA introduces four kinds of NodeIds. This is specified by an IdentifierType:

- NUMERIC (→ UInteger)
- STRING (→ String)
- GUID (→ Guid)
- OPAQUE (→ ByteString)

Currently the NodeID is provided with a NUMERIC IdentifierType by the Mapping Editor automatically. Thus, this parameter is not editable by the user.

The formalized NodeID type definition by the OPC UA specification is a structure of

- namespaceIndex [UInt16],
- identififierType [Enum: IdType],
- identifier [depends on the identifierType].

DisplayName

The *DisplayName* is used to represent the Node in the information model of the Mapping Editor. In general, the DisplayName is the localised name of the node and shall be used by Clients to represent the node to a user, e. g. via a visualization.

The *DisplayName* parameter of the "Add Instance" dialog is automatically taken from the Name parameter, but it can also be overwritten by the user.

The *DisplayName* type definition is a LocalizedText. Thus, a LocaleId and a String can be provided.

- The LocaleID (e.g. "en-US") can be left empty. In this case the corresponding string is the default Displayname, without localization.
- The Mapping Editor supports currently only one entry for the DisplayName.
- According to the OPC UA specification, each server shall provide the DisplayName identical to the BrowseName of the Node for the LocaleId "en" (or for empty LocaleIDs).

BrowseName

The *BrowseName* is used when browsing the AddressSpace to create paths out of BrowseNames.

Unlike NodeIds, the BrowseName cannot be used to unambiguously identify a Node. Different Nodes may have the same BrowseName.

The *BrowseName* parameter of the "Add Instance" dialog is automatically taken from the Name parameter, but it can also be overwritten by the user.

The *BrowseName* type definition is a structure. It contains a namespaceIndex and a string. The namespaceIndex is provided to make the *BrowseName* unique in some cases in the context of a Node, although not unique in the context of the Server.

If various CompanionSpecifications define (equal) *BrowseNames* for Variables, the namespace of the BrowseName provided by the CompanionSpecifications makes the BrowseName unique, although different CompanionSpecifications may use the same string having a slightly different meaning.

As an example, a Variable node "Manufacturer" shall be added, which is used in the context of the DI – Companion Specification. In this case, the BrowseName of the Node is "Manufacturer" and the namespaceIndex points to the DI – namespace URI.

Description

The Description parameter defines the localized description of the new node.

The *Description* type definition is a LocalizedText. Thus, a LocaleId and a String can be provided.

The LocaleID (e.g. "en-US") can be left empty. In this case the corresponding string is the default Description, without localization.
The Mapping Editor supports currently only one entry for the Description.

EventNotifier

The EventNotifier is used to indicate if the node is able to fire Events and / or to support reading and writing historic Events. Multiple selections are possible. The supported ability is OPCUA Server target depending.

The current Lenze OPC UA Server on c5xx controller supports "Subscribe to events" since Firmware Version 1.9. Historic Event reading and writing is not supported.
If an EventNotifier is added to an Object by the context menu, this parameter is set automatically.

Value

The Value parameter is used to provide a default value for the Variable node. This default value is part of the nodeset file, like other node parameters. The use case is to present static (default) values in the information model like a manufacturer name. In this case these static values don't need a special data source which has to be mapped.

If the variable node is mapped the value will always be overwritten by the data source.

If the Value "Null" checkbox is deactivated, a static (default) value can be inserted.

It is currently only possible to set a static value for primitive scalar DataTypes.

ValueRank

The ValueRank parameter specifies if the value is a scalar (a single value) or an array (a field of values) with one or more dimensions.

Following Table gives an overview.

ValueRank	Example		
	Notation	ArrayDimensions	Description
Scalar (-1)	Int32	omitted /empty	A scalar Int32.
1 Dimension (1)	Int32[]	omitted /empty	Single-dimensional array of Int32 with unknown size. The size will be taken over from the data source if mapped.
	Int32[3]	3;	Single-dimensional array of Int32 with fix size of 3.
2 Dimension (2)	Int32[][]	omitted /empty	Two-dimensional array of Int32 with unknown size. The size will be taken over from the data source if mapped.
	Int32[5][3]	5;3;	Two-dimensional array of Int32 with fix size of 5 for the first dimension and 3 for the second dimension.
3 Dimension (3)	Int32[][][]	omitted /empty	Three-dimensional array of Int32 with unknown size. The size will be taken over from the data source if mapped.
	Int32[5][3][3]	5;3;3;	Three-dimensional array of Int32 with fix size of 5 for the first dimension and 3 for the second and third dimension.

The OPC UA known ValueRanks Any(-2), ScalarOrOneDimension(-3) and ValueRanks with more than 3 dimensions are currently not supported by the Mapping Editor and Lenze OPC UA Server.

ArrayDimensions

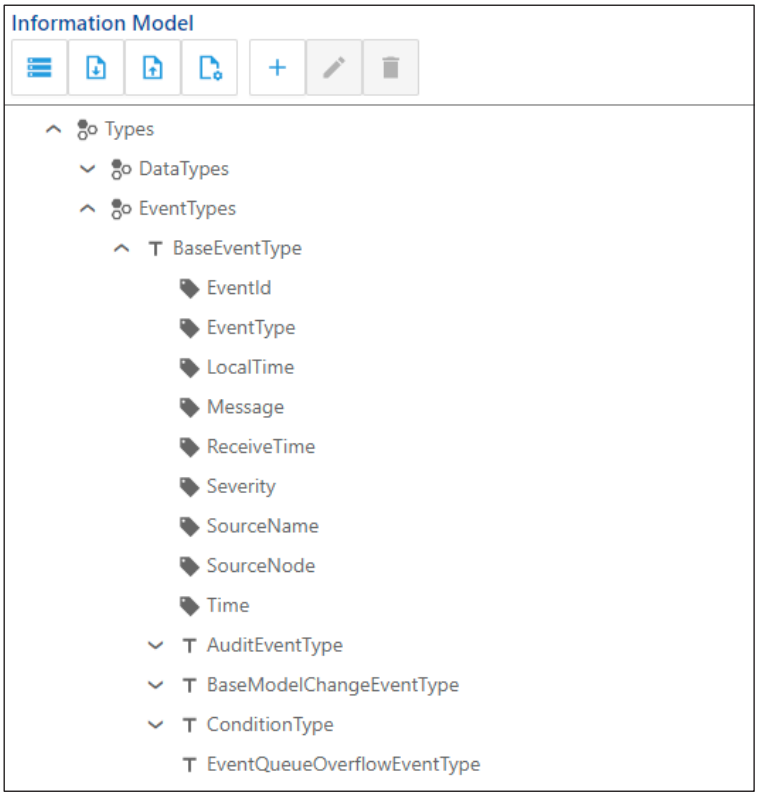
If the ValueRank indicates an array value, the ArrayDimensions parameter specifies the length of each array dimension. If the ValueRank indicates a scalar value the ArrayDimensions parameter can be omitted.

The length of the dimensions is entered as numbers separated by semicolons.

If the variable node will be mapped with a data source variable of the symbol configuration, the ArrayDimensions must not yet be known. The array size will be taken over from the mapped PLC variable by the Lenze OPC UA Server. In this case the value 0 must be entered at the ArrayDimensions parameter.

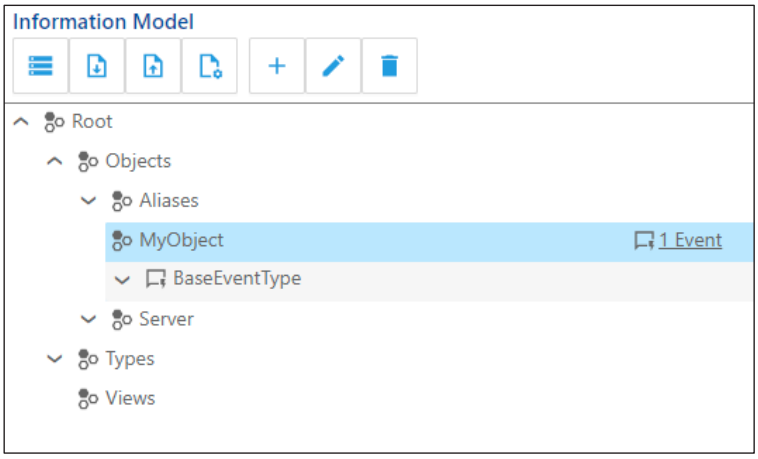
Add / Delete Event Notifier

OPC UA provides many types of events. The event type hierarchy can be extended by loaded base models (companion specifications).



In contrast to Objects, Variables and Methods, simple Events cannot be represented as an instance in the OPC UA information model. Simple events are generated dynamically when they occur and are sent from the OPC UA Server to a client.

In OPC UA these events can be sent not only by the Server Node but also by specific object instances.



With the Mapping Editor it is possible to configure object instances of the user information model as an event notifier. Multiple EventType-Notifier can be added via the context menu with the "Add Event Notifier" dialog.

The EventType–Notifier is not an instance in the OPC UA information model. The "EventType-Notifier" below an object has only usability purposes.

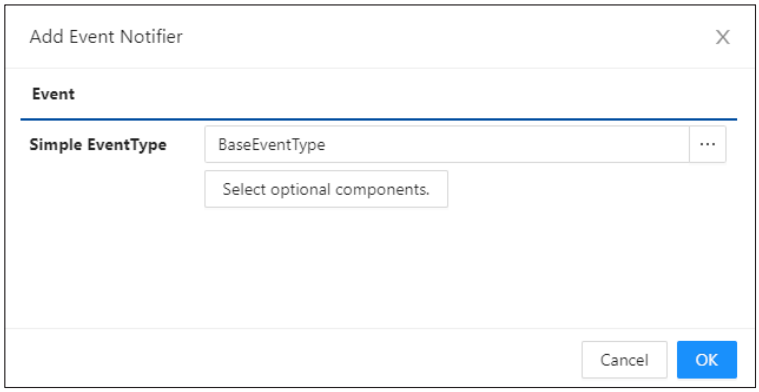
Thus, the EventType-Notifier does not have an OPC UA NodeID.

To distinguish EventTyp-Notifier from the instances, they are displayed directly below the associated object via an expandable section.

A single EventType-Notifier is expandable to disclose additional event fields (properties) which will be published when the event occurs. The event field provided by the BaseEventType are not displayed.

"Add Event Notifier" dialog

If an object instance of the user information model is selected, an EventType-Notifier can be added by the "Add Event Notifier" command.



Simple EventType-Parameter

The Simple EventType parameter defines the event type to be sent when the event occurs.

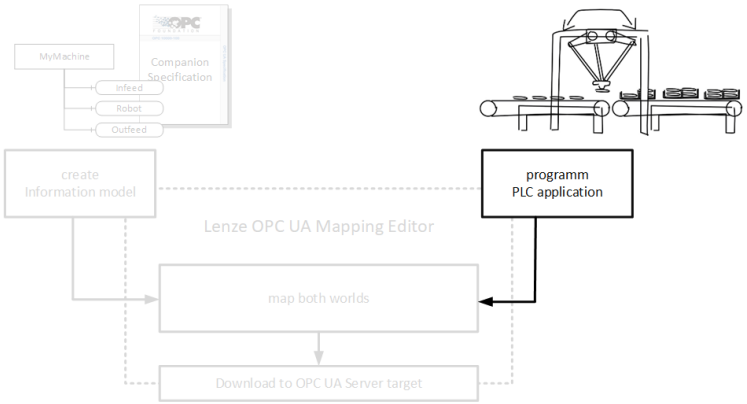
It is the same "Type definition" dialog as the "Add Instance".

As with other types, events can also have optional and mandatory event elements.

Due to the fact, that the AddEvent-Notifier command does not generate a static event instance in the OPC UA information model, also abstract (italic fond) event types can be selected.

It is possible to add multiple EventType-Notifier of different type definition to an object instance.
More than one EventType-Notifier of the same type definition are not allowed.
If an EventType-Notifier is selected, it can be deleted by the general "Delete" command of the context menu or the command bar.

Select data source



The Mapping Editor provides the selectable data sources by a drop-down list in the right Data Source area. The selectable data sources depends on the OPC UA Server target device and the loaded Lenze services.

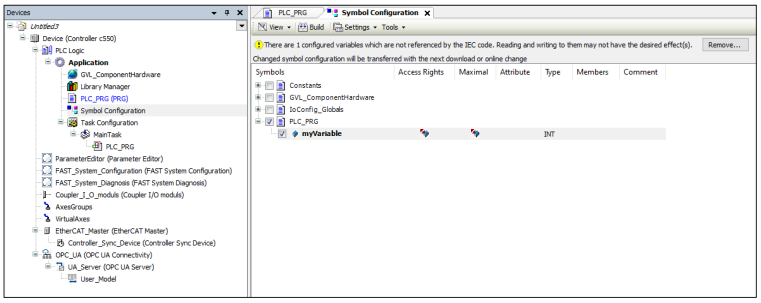
Data source "symbol configuration"

The data source of the "symbol configuration" provides access to the PLC content of the target device. To protect the PLC application against unauthorized external access, individual variables and object instances have to be selected for OPC UA communication by the PLC application developer. This selection is done with the symbol configuration.

Setup of symbol configuration

An PLC application holds various variables and object instances. The symbol configuration is used to publish individual instances via OPC UA.

1. Insert a Symbol configuration object below the application node in the device-tree of the »PLC Designer«.
2. Activate the option Support OPC UA functionalities in the dialog "Add symbol configuration".
3. Open the symbol configuration in the editor.
4. Select the "Build" command.
   The variables are displayed in a tree structure
5. Select the variables that you want to publish via OPC UA.



The handling of the symbol configuration is described in detail under the following link:
CODESYS Online Help

- Make sure that only the IEC variables are selected that are required for the use case and are therefore to be published. If a variable is selected in the symbol configuration, the subordinate member variables are also automatically activated for OPC UA access.
- If you have a user management active on the controller, IEC variables can be assigned to symbol sets and these in turn to user groups. Thus, a certain user has access only to the IEC variables intended for.
- It is possible to setup various symbol sets to grant specific user groups access to the data. In the Mapping Editor just the general (overall) symbol configuration is provided as a Data Source and not individual symbol sets.

Variables

Mappable variables are marked with a "tag"  icon.

This icon is used on the data source side as well as on the OPC UA information model side for variables. Both entities represent a data value of an automation component.

Variables are mappable according to chapter Mapping of Variables (📖 50).

Function blocks

Instances of PLC application function blocks are represented with a "three parts"  icon in the data source section.

This icon is used on the data source side for function blocks as well as on the OPC UA information model side for Objects. Both entities represent an automation component and are used to structure data (variables, methods, events).

Currently function blocks are not mappable.

Methods (UA specific function block)

Mappable methods are marked with a "brackets" {} icon.

This icon is used on the data source side as well as on the OPC UA information model side for methods.

In the PLC Application OPC UA methods are represented as function blocks. These function blocks are derived from a base function block **L_IOSP_MethodCall** provided by the L_IOSP_ OpcUaServer library. The base function block supports the generic method handling between OPC UA Server and the PLC application. The derived function block is enhanced by the Input and Output Arguments of the OPC UA method and the operation logic.

The IEC handling variables between the method function block and the OPC UA Server are not represented in the data source tree. (Method function block – OPC UA Server interaction (📖 44))

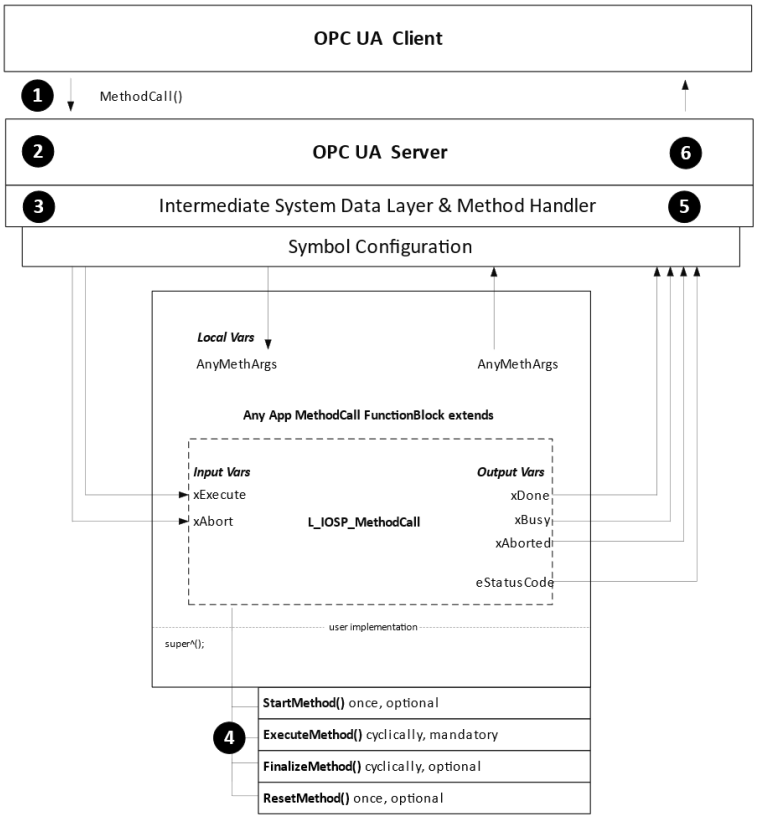
Method function block – OPC UA Server interaction

The following figure describes the general usage of the function block inside the PLC application and the interaction with the Lenze OPC UA Method Handler. To connect the function block with the OPC UA Method Handler, it is necessary to publish the function block via the symbol configuration and to map it to a specific OPC UA Method.

It is expected to extend an application specific function block with the **L_IOSP_MethodCall**. The application specific function block can be enhanced with local variables for the arguments of the OPC UA method, which must be of the same name and data type.

Sequence:

1. An OPC UA Client requests a specific Method call with any input arguments.
2. The OPC UA Server receives this call and performs a conversion and an alignment into a generic data format to distribute it further in the system.
3. A subsequent Method Handler stores and manages the method call, e. g. timeout handling. The Method Handler sets the function block variables, with which the method operation can be triggered (e.g. xExecute). The MethodCall InputArguments are assigned to the functions blocks local variables (Local Vars). If a positive edge occurs on *xExecute*, the arguments of the method call are provided by the Method Handler.
4. The method operation can be done in any suitable application specific IEC task. The operation may take several cycles. For the operation automatically called IEC methods are provided. Not needed methods can be deleted.
   - StartMethod()
   - ExecuteMethod()
   - FinalizeMethode()
   - ResetMethod()
5. After the Start-, Execute- and FinalizeMethod operation variables are provided for the OPC UA Method output arguments.
6. The OPC UA Server responds to the Method call of the client and the xExecute variable is set to false.


OPC UA Client
1
MethodCall()
2
OPC UA Server
6
3
Intermediate System Data Layer & Method Handler
5
Symbol Configuration
Local Vars
AnyMethArgs
AnyMethArgs
Any App MethodCall FunctionBlock extends
Input Vars
xExecute
xAbort
L_IOSP_MethodCall
Output Vars
xDone
xBusy
xAborted
eStatusCode
user implementation
super^();
4
StartMethod() once, optional
ExecuteMethod() cyclically, mandatory
FinalizeMethod() cyclically, optional
ResetMethod() once, optional

Events (UA specific function block)

Mappable events are marked with a "speech bubble" icon.

This icon is used on the data source side as well as on the OPC UA information model side for events.

In the PLC Application OPC UA events are represented as function blocks. These function blocks are derived from a base function block L_IOSP_BaseEventType provided by the L_IOSP_OpcUaServer library. The base function block supports the generic event handling between OPC UA Server and the PLC application. The derived function block is enhanced by the event fields of the OPC UA event and the operation logic.

The IEC handling variables between the event function block and the OPC UA Server and the event field variables for the BaseEventType are not represented in the data source tree.
(Events (UA specific function block) (📖 46))

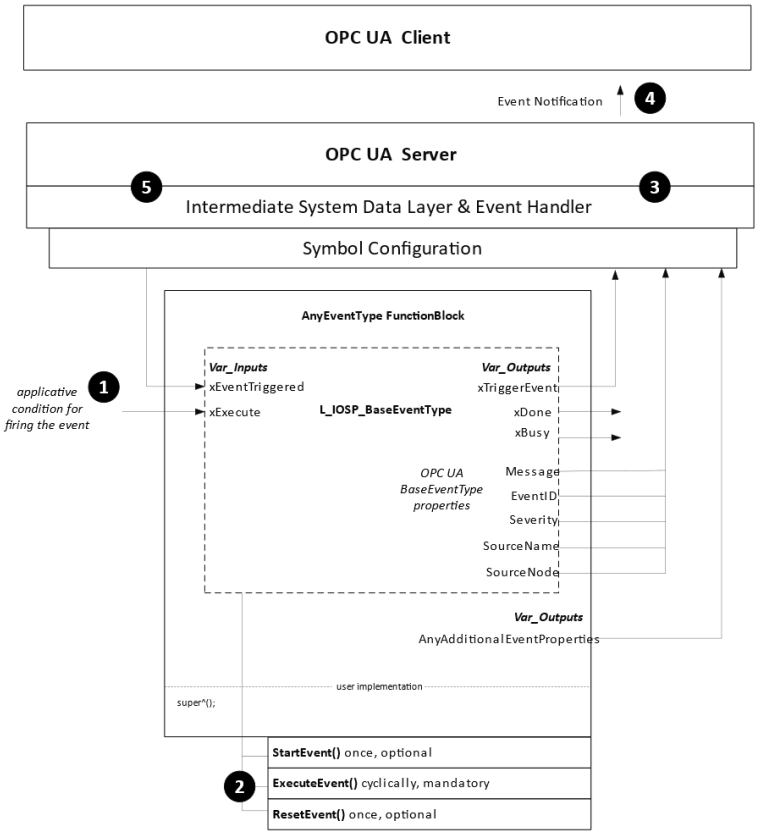
Event function block – OPC UA Server interaction

The following figure describes the general usage of the function block inside the PLC application and the interaction with the Lenze OPC UA Server. To connect the function block with the OPC UA Server it is necessary to publish the function block via the symbol configuration and to map it to an OPC UA EventNotifier in the user information model.

It is expected to extend an application specific function block with the L_IOSP_BaseEventType. The application specific function block can then be enhanced with output variables for additional Event properties.

Sequence:

1. The event trigger decision is done by the PLC application.
2. The event operation (preparing the event properties) can be done in any suitable application specific IEC task. The operation may take several cycles. For the event operation automatically called methods are provided. Not needed methods can be deleted.
   - StartMethod()
   - ExecuteMethod()
   - ResetMethod()
3. After the Start- and ExecuteMethod() operation the output variables are provided to the OPC UA Server.
4. The OPC UA Server sends the Event to the subscribing Client.
5. To signal the application that the Event has been fired, the OPC UA Server sets the corresponding input variable xEventTriggered on true of the function block.

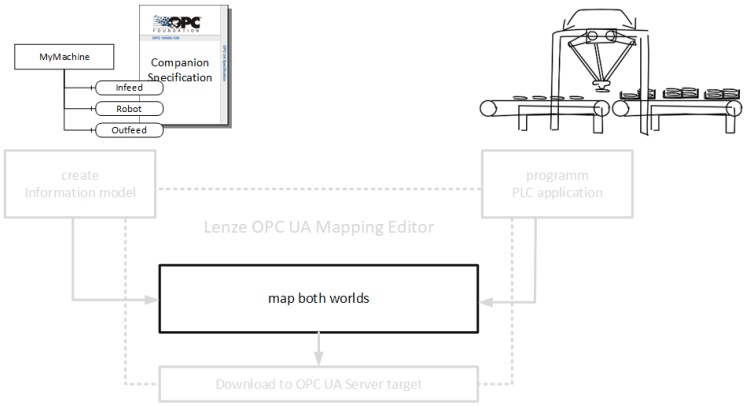


Applications, Programs and Co

Folder are used to structure the data source they are marked with a "folder" icon.

They aren't mappable. From IEC point of view folders are for example PLC applications or programs and do not have a type definition.

Map data source to OPC UA



General mapping

The Lenze Mapping Editor represents the current mapping for all visible OPC UA nodes in the middle of the screen in a mapping table. The possibility to scroll through the information model and the mapping synchronously provides a good overview which nodes are already mapped to a Data Source item.

To perform a mapping, navigate in the OPC UA information model to the Node to be mapped.

The mapping can be done by "drag & drop" of the Data Source item to the mapping table field of the desired OPC UA node.

Depending on the Data Source item following OPC UA NodeClasses are mappable. Only these nodes provide a link icon in the mapping table and the "Node Mapping Attributes".

- Variable (DataVariable and Property)
- Method
- EventType-Notifier

To check the mapping, a project compile of the PLC project is necessary in the »PLC Designer«.

The following chapters describe which items are mappable and what are the constraints.

Data source symbol configuration

The symbol configuration provides the following kinds of mappable items:

- Instances of variables (Mapping of Variables (📖 50))
- Instances of function blocks for OPC UA Methods (Mapping of Methods (📖 54))
- Instances of function blocks for OPC UA Events (Mapping of Events (📖 55))

Following items are currently not mappable:

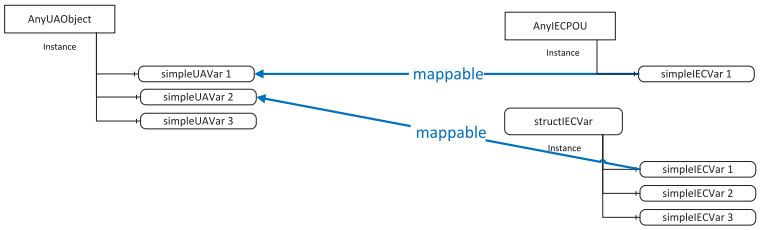
- Instances of function blocks
- Applications or Programs

Currently it is not possible to map whole PLC function blocks to the user specific OPC UA information model. Nevertheless, it is possible to map individual member variables of the function block.
If a PLC variable is mapped to an OPC UA variable node the access rights of the symbol configuration will be taken over automatically.

Mapping of Variables

It is possible to map a single variable Data Source item to multiple OPC UA variable nodes.

Simple UA Variable with Built-in data type



A simple PLC variable with primitive data type is mappable to any OPC UA variable node with a matching Build-in data type according to following table. The instance names of the OPC UA and IEC variables can be different.

It doesn't matter whether the IEC variable belongs to a POU or is located as a member variable in an IEC structure.

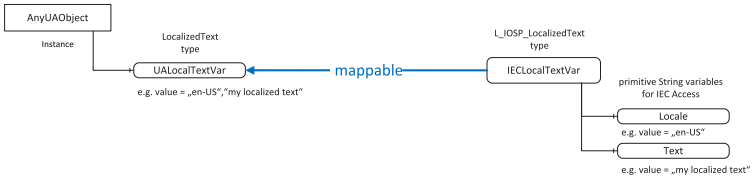
OPC UA Build-in Data type	Primitive IEC Data type	Comment
Boolean	BOOL	ns=0;i=1
SByte	SINT	ns=0;i=2
Byte	USINT	ns=0;i=3
Int16	INT	ns=0;i=4
UInt16	UINT	ns=0;i=5
Int32	DINT	ns=0;i=6
UInt32	UDINT	ns=0;i=7
Int64	LINT	ns=0;i=8
UInt64	ULINT	ns=0;i=9
Byte	BYTE	ns=0;i=3
UInt16	WORD	ns=0;i=5
UInt32	DWORD	ns=0;i=7
UInt64	LWORD	ns=0;i=9
Float	REAL	ns=0;i=10
Double	LREAL	ns=0;i=11
String	STRING	ns=0;i=12
Byte	CHAR	ns=0;i=3
String	WSTRING	ns=0;i=12
UInt16	WCHAR	ns=0;i=5
DateTime	DATE_AND_TIME	ns=0;i=13
DateTime	DATE	ns=0;i=13

OPC UA Build-in Data type	Primitive IEC Data type	Comment
UInt32	TIME_OF_DAY	ns=0;i=7
Int64	TIME	ns=0;i=8
Int64	LTIME	ns=0;i=8
Int64	LDATE	For this IEC data type is no mapping supported.
Int64	LTOD	For this IEC data type is no mapping supported.
Int64	LDT	For this IEC data type is no mapping supported.

UA specific built-in data types

Additional to the data types mentioned above OPC UA introduces OPC UA specific Build-in data types like the LocalizedText type definition.

For some of these OPC UA specific Build-in data type definitions Lenze provides corresponding IEC type definitions which can be used in the PLC application. These IEC type definitions are distributed by the L_IOSP_OpcUaServer library.

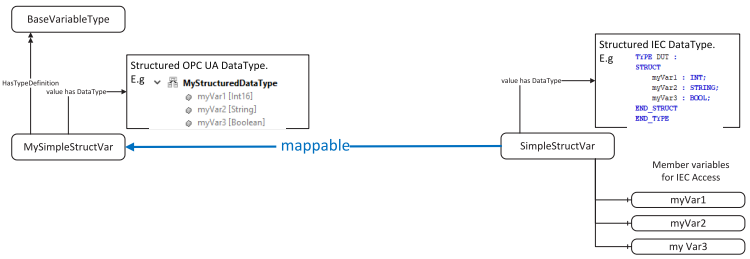


A PLC variable with an OPC UA specific IEC data type is mappable to any OPC UA variable node with a matching data type according to following table.

OPC UA Build-In Data type	OPC UA specific IEC Data type	Comment
NodeId	L_IOSP_NodeId	
QualifiedName	L_IOSP_QualifiedName	
LocalizedText	L_IOSP_LocalizedText	
ByteString	L_IOSP_ByteString	

Simple UA Variable with structured DataType

Like the symbol configuration, OPC UA introduces structured DataTypes. In contrast to the Symbol configuration, it's possible to model just one simple Variable Node with a structured DataType definition for the Value Attribute. In this case no Nodes for the "member-variables" of the structure are available in the OPC UA model. For further Information how to model structured data see Annex (📖 56).



A structured IEC variable is mappable to an OPC UA variable Node, if …

- … the structured data type definitions match and
- … the member variable (symbolic) names are the same.

This means that the member variables of the data type structure, down to the last leaf, are of the same name and primitive type according to chapter Mapping of Variables (📖 50).

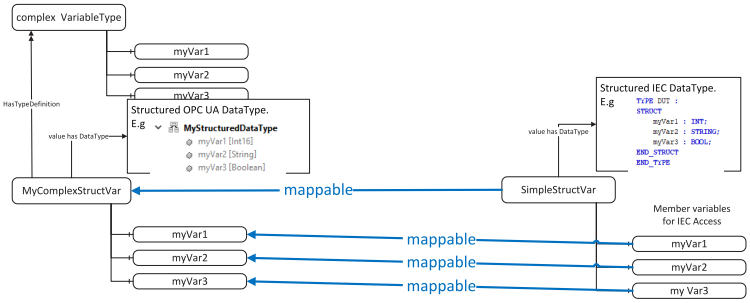
OPC UA Data type	IEC structured Data type	Comment
EUInformation	L_IOSP_EUInformation	
Range	L_IOSP_Range	

For some of these OPC UA structured data type definitions Lenze provides corresponding IEC type definitions which can be used in the PLC application. These IEC type definitions are distributed by the L_IOSP_OpcUaServer library.

Complex UA Variable with structured DataType

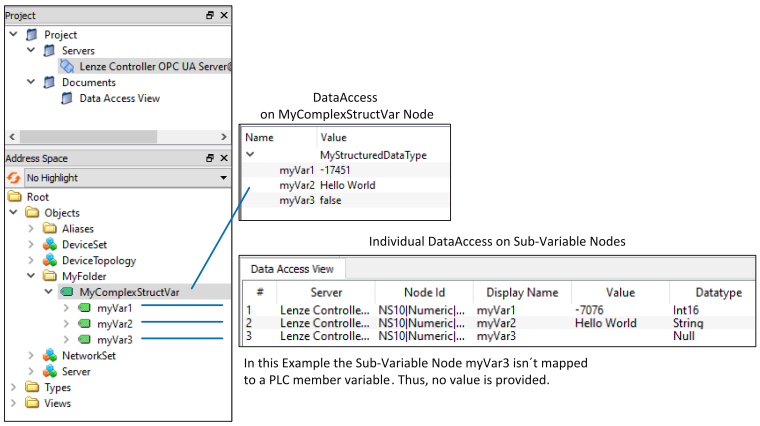
Like the symbol configuration, OPC UA introduces structured DataTypes. In OPC UA it is possible to present a structed Value in only one Variable Node (see Structured DataType and a simple Variable using the DataType (📖 57)).

But it is also possible to model it as a complex Variable Node with Sub-Variable Nodes for individual data access of the elements of the structure. For further Information how to model structured data see Structured DataType and a complex VariableType (📖 57).



As described in chapter Mapping of Variables (📖 50), a structured IEC variable is mappable to an OPC UA variable Node. Additional it is possible to map the individual member variables of the IEC structure to the corresponding OPC UA (Sub-) Variable Nodes. In this case the member variables, down to the last leaf, must be of the same primitive type according to chapter Mapping of Variables (📖 50).

If an IEC member variable isn't mapped to an OPC UA Sub-Variable Node, the value of the IEC member variable isn't available by the OPC UA Sub-Variable Node. This means that there isn't an OPC UA Server internal mapping from the structured value of the parent Node to the child nodes. This is shown in the following example with the UA Expert as an OPC UA Client connected to the Lenze OPC UA Server with mapped user specific information model.



Arrays

OPC UA as well as the symbol configuration are aware of Arrays. In OPC UA the ValueRank Attribute of a Node is defined to indicate if the DataType is used as scalar value (-1), as array (1) or as multidimensional array (>1). The ArrayDimensions Attribute specifies the maximum elements of each array dimension.

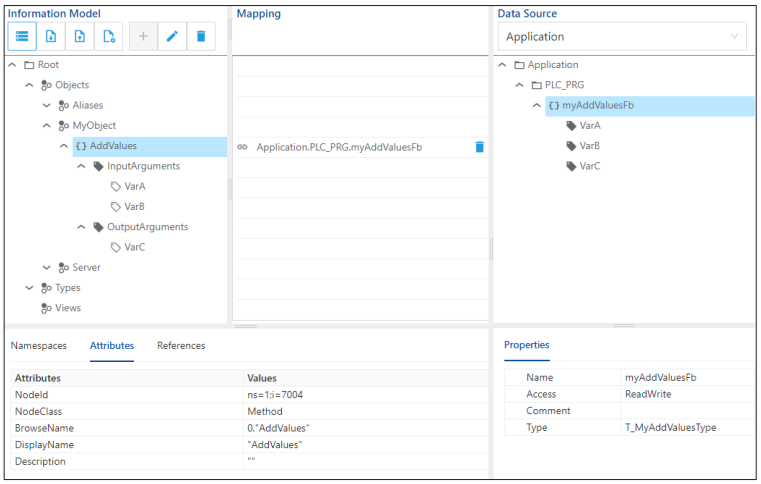
It is possible to map PLC array variables to OPC UA array variables with matching DataType definition and array dimensions and size.

The array sizes must not yet be known in the prepared OPC UA information model. The array size can be taken over automatically from the mapped PLC variable. In this case the value 0 must be entered at the ArrayDimensions Attribute of the OPC UA Node.
The symbol configuration as data source can publish an array variable with maximum 3 dimensions and 65536 elements.

Mapping of Methods

IEC method function blocks are mappable to OPC UA methods. Just the main method-nodes have to be mapped by the user. The corresponding arguments are mapped automatically by the Mapping Editor.

In contrast to variables, for methods it is just possible to map a single method Data Source item to a single OPC UA method node.



The mapping of the arguments can be done, if…

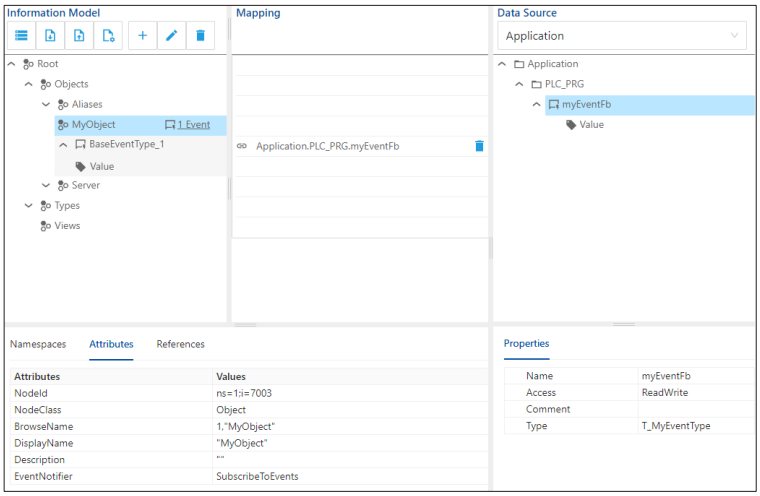
- the (symbolic) names of the arguments are the same and
- the data types fit according to chapter Mapping of Variables (📖 50),
- for all OPC UA Arguments an IEC variable is available (it is allowed to have more IEC variables than OPC UA Arguments).

Mapping of Events

In opposite to OPC UA objects, variables and methods, simple events have no static instances in the information model. Events are only represented as "EventType-Notifier" in the information model and instantiated dynamically (on demand) by the associated object, when the OPC UA Server is running.

IEC event function blocks are mappable to the "EventType-Notifier". Just the main event-nodes have to be mapped by the user. The corresponding properties are mapped automatically by the Mapping Editor.

In contrast to variables, for events it is just possible to map a single event Data Source item to a single EventType-Notifier.



The automatic proposal for the corresponding properties can be done, if…

- the names of the properties are the same and
- the data types fit according to chapter Mapping of Variables (📖 50),
- for all OPC UA properties an IEC variable is available (it is allowed to have more IEC variables than OPC UA properties).

The event fields provided by the BaseEventType are not displayed.

Annex

Complex Variables and stuctured DataTypes

There are basically three different approaches to model structured data in OPC UA.

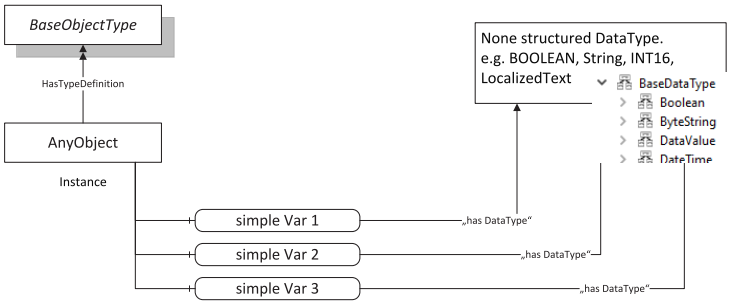
- Create several simple Variables using simple DataTypes always reflecting parts of the simple structure. Objects are used to group the Variables according to the structure of the data.
- Create a structured DataType and a simple Variable using this DataType.
- Create a structured DataType and a complex Variable using this DataType and also exposing the structured data structure as Variables of the complex Variable using simple DataTypes.

UA Part 3: Address Space Model - A.4.3 Many Variables and / or structured DataTypes (opcfoundation.org)

Object instance to group simple Variables

The advantages of the first approach are that the complex structure of the data is visible in the AddressSpace. A generic Client can easily access the data without knowledge of user-defined DataTypes and the Client can access individual parts of the structured data.

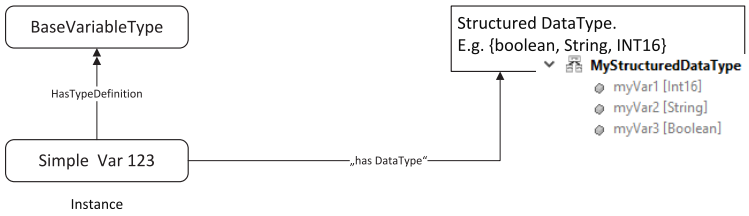
The disadvantages of the first approach are that accessing the individual data does not provide any transactional context and for a specific Client the Server first has to convert the data and the Client has to convert the data, again, to get the data structure the underlying system provides.



Structured DataType and a simple Variable using the DataType

The advantages of the second approach are, that the data is accessed in a transactional context and the structured DataType can be constructed in a way that the Server does not have to convert the data and can pass directly to the specific Client that can directly use them.

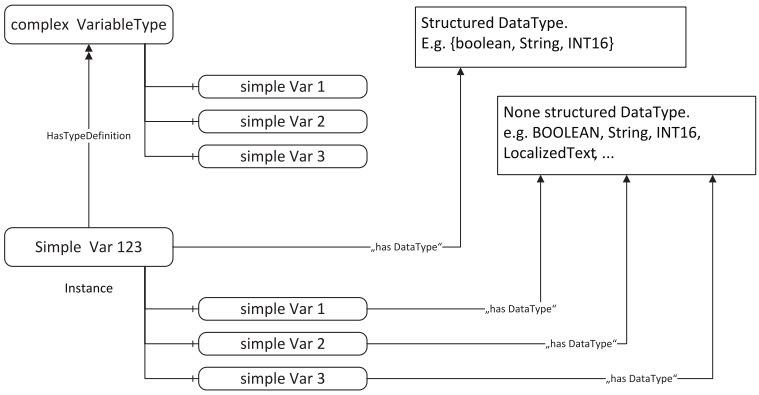
The disadvantages are that the generic Client might not be able to access and interpret the data or has at least the burden to read the DataType Definition to interpret the data. The structure of the data is not visible in the AddressSpace; additional Propertiesd escribing the data structure cannot be added to the adequate places since they do not exist in the AddressSpace. Individual parts of the data cannot be read without accessing the whole data structure.



Structured DataType and a complex VariableType

The third approach combines the other two approaches Object instance to group simple Variables (📖 56) and Structured DataType and a simple Variable using the DataType (📖 57). Therefore a specific Client can access data in its native format in a transactional context, whereas a generic Client can access simple DataTypes of the components of the complex Variable.

The disadvantage is that the Server must be able to provide the native format and also interpret it to be able to provide the information in simple DataTypes.



Controller: OPC UA Server

OPC UA Server on the Lenze Controller

This documentation describes the OPC UA Server functionality on the Lenze Controller.

A general introduction to OPC UA as well as the support of OPC UA in the Lenze system can be found in the following document:

- OPC UA System

The OPC UA Client is available as PLCopen function blocks for the PLC application.

Further information can be found for the L_IOCP_OpcUaClient library.

OPC UA Server profiles and functions

The Lenze OPC UA Server on the controller supports the following profiles, functions:

Profile / Function	Controller OPC UA Server
UA Server Profile	Embedded certified
Data Access - Lenze Parameters	no
Data Access - PLC	yes
Methods - PLC	yes (from Version V01.09.)
Events	yes (from Version V01.09.)
A&C	Planned
Historical Access	Planned
Transportation	UATCP UASC UA-Binary
User management	yes
Security	None Basic256Sha256 Sign Basic256Sha256 Sign&Enc. Aes256_Sha256_RsaPss (from Version V01.09.)

OPC UA Server Capabilities

The Lenze OPC UA Server on the controller supports the following service capabilities. The individually listed capabilities have the limit named here. Depending on the combination and utilization of these capabilities, as well as the utilization of the CPU due to the PLC application and other processes on the controller, the controller may reach its performance limit.

This must be considered during project engineering.

Service	c4xx Controller	c5xx Controller
Max. Number of instance nodes (in instance model per namespace)	No limit (in the PLCopen instance model)	

Service	c4xx Controller	c5xx Controller
Max. Number of nodes (in type model per namespace)	12010	
Max. Number of variables (in type model per namespace)	10008	
Max. Number of references (in type model per namespace)	20020	
Max. Number of dimensions (in type model per namespace)	10008	
Max. Number of User Sessions	4 [1)]	6 [1)]
SessionTimeOut	15 minutes	
Max. No. of subscriptions per session	10	16
Max. Number of Monitored Items per Subscription	1000	2000
Max. Sum Number of Monitored Items	4000 ±200 for Engineering Access	8000 ±400 for Engineering Access
Max. Sum Number of Subscriptions	10 +2 for Engineering Access	16 +4 for Engineering Access
Min. publishing interval	100	
Min. sampling interval	100	
Max. Que size	1	
Max. Monitored items per call	1000	2000
Max. Nodes per browse	100	
Max. Nodes per read	100	
Max. Nodes per write	100	
Max. Nodes per register nodes	100	
Max. Nodes per trans.Bro.PathToID	100	
Max. Array Length	65535	
Max Number of Nodesets	10	
Max. Event Queue	11	

1) One client connection (session) is included in the basic setup of the OPC UA Server. Additional sessions are subject to license

The Lenze OPC UA server on the controller can be reached under the Ethernet TCP port 4840 of the controller.

It can be identified in the system under the following ApplicationUri:

- Lenze Controller OPC UA Server@<host name of the controller>

Integrated information models and namespaces

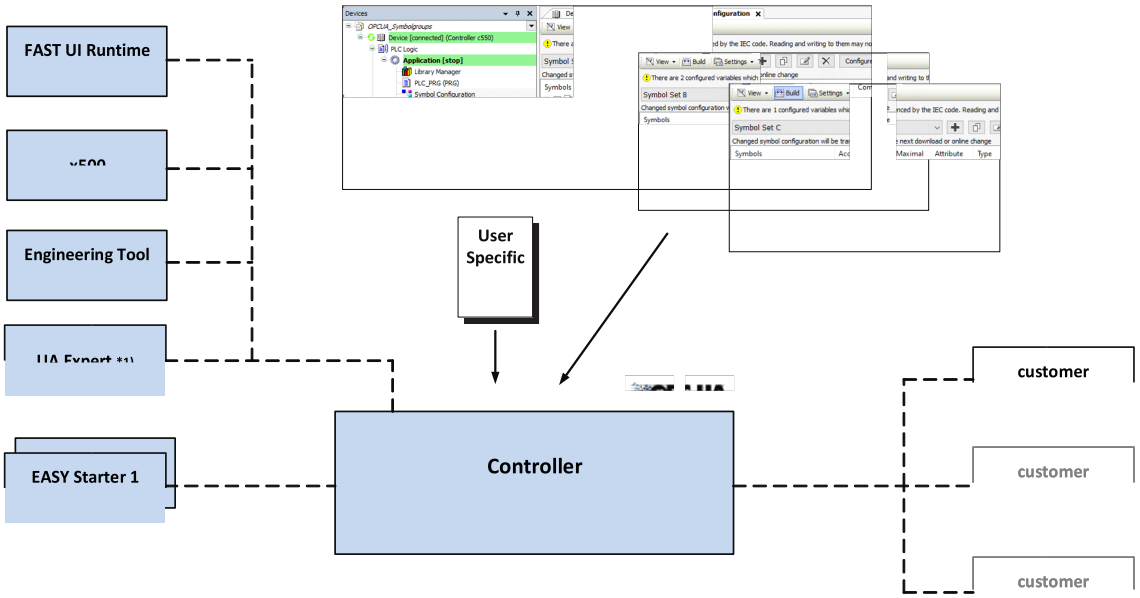
A client can access the PLC via the OPC UA server on the controller. It is also used for Lenze's internal engineering access from the EASY Starter. The following table provides an overview of the namespaces stored for this purpose.

Namespace	Description
http://opcfoundation.org/UA/	Base OPC UA information model with all standard types
urn:<hostname>:Lenze:< devicename>:OPCUA:Server	Lenze Base OPC UA Information Model
http://opcfoundation.org/UA/DI/	Information model for OPC UA for devices (Device Integration).
http://PLCopen.org/OpcUa/IEC61131-3/	Information model for PLCOpen. Contains all generic types from the specification
urn:Lenze:PLCOpen	Information model for mapping the PLC program to the PLCOpen specification. Contains all types and objects needed for the PLC program.
urn:Lenze:Enginering:Base	Internal basic information model for all engineering interfaces
urn:Lenze:Engineering:Parameter Manager	Internal information model for Parameter Manager Interface
urn:Lenze:Engineering:EtherCAT	Internal information model for EtherCAT
urn:Lenze:Engineering:Filesystem	Internal information model for file system access
urn:Lenze:extended_server_diagnostic	Internal information model for extended server disgnosis

Lenze OPC UA and customer-specific client connections

OPC UA is not only used for customer-specific client-server connections, such as the connection of the controller to an MES system. Lenze-specific client-server connections are also provided via OPC UA. These include:

- FAST UI Runtime as OPC UA Client to the Controller as OPC UA Server
- x500 as OPC UA client to the Controller as OPC UA server
- EASY Starter as OPC UA Client to the Controller as OPC UA Server



If a client mentioned above connects to the Lenze OPC UA server, it is identified as a Lenze client for internal system communication. This means that no separate OPC UA license is required for the Lenze clients, as is the case for the additional external clients.

*1) Also the UA Expert by Unified Automation has free access to the Server, as a recommended generic OPC UA Client for diagnosis purposes.

*2) From V01.09. it is possible to load user specific information models based on Companion Specifications.
https://www.lenze.com/de-de/service

The following table provides a detailed overview.

Session Category	Max. Sessions		Clients	Model Access
	c4xx	c5xx		
Lenze Engineering Clients	1	2	EASY Starter Firmware loader Application Loader	internal engineering / parameter - model
Lenze System Clients	3 + 1any	3 + 2any	1 x FAST UI Runtime 1 x x500 1 x UA Expert 2 x any other Lenze Client	PLCopen Model User specific Model
external Clients	4 [3)]	6 [3)]	external Cliente.g. other Controller, MES, …	PLCopen Model User specific Model

If a connection between client and server is lost, the server waits for a SessionTimeOut time until it cleans up the session and releases it for a new possible client connection. The advantage is, if the old client comes back during this time, the previously created settings like "registered nodes" and "subscriptions" are preserved for it. The disadvantage is that another client cannot reassign this session during the SessionTimeOut time.

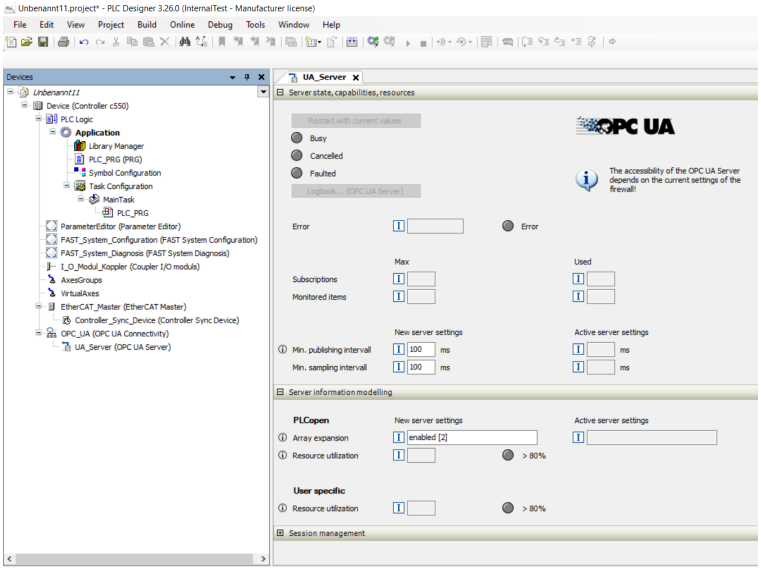
[3)] One client connection (session) is included in the basic setup of the OPC UA Server. Additional sessions are subject to license.

Configuration and Diagnosis

The Lenze OPC UA Server on the Controller can be configured and diagnosed via Lenze parameter. These parameters are available in the »PLC Designer«, which provides an entry point for OPC UA.

The configuration and diagnosis parameter are divided into three sections.

- Server state, capabilities, resources
- Server information modelling
- Session management



Additional to the »PLC Designer«, following engineering tools are recommended for diagnosis purposes.

- »EASY Starter«, for general Lenze parameter access.
- UA Expert by Unified Automation, for generic OPC UA Server interaction and diagnosis.
- Add the "Server Diagnostics View" Document.

Server state, capabilities, resources

Server State

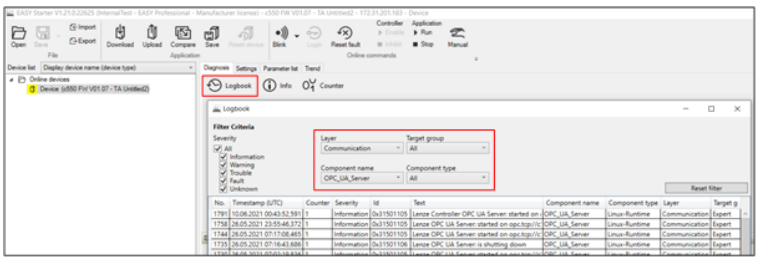
The Server State parameter [0x2473:001] provides the state of the Lenze OPC UA Server according to the OPC UA specification Part 5 – "ServerState".

Server Error

The Server Error parameter [0x2473:002] provides the current Lenze error code of the OPC UA Server. The error code and countermeasures are listed in chapter OPC UA Server error codes and countermeasures (📖 65).

Controller logbook

The Lenze OPC UA server reports its states to the controller logbook, which can be accessed via Lenze engineering tools such as EASY Starter or »PLC Designer«.



OPC UA Server error codes and countermeasures

The Lenze OPC UA Server can send the following messages to the controller logbook. Further notes on the error and corrective measures are described here.

ErrorCode Type	Description	
0x1301 Error	Out of memory	
	Cause	Not enough memory
	Remedy	Reduce the number of symbols in the symbol configuration in the PLC Designer. Further information about the handling of required resources for the PLCopen namespace can be found in chapter Resource requirements in the server address space (📖 83).
0x1302 Error	Failed to update address space	
	Cause	The address area "PLCopen" could not be updated
	Remedy	Reduce the number of symbols in the symbol configuration in the PLC Designer. Further information about the handling of required resources for the PLCopen namespace can be found in chapter Resource requirements in the server address space (📖 83).
0x1303 Error	Internal error	
	Cause	An internal operation has failed - Firmware / server configuration inconsistent - Hardware error - RAM/ internal flash memory not sufficient - Configuration or certificate data not valid - PLC project or runtime dependency, if applicable
	Remedy	Device restart / Update device firmware / Contact support and have trace created
0x1304 Error	Nodesetfile %s could not be found	
	Cause	Referenced Nodeset file can´t be found.
	Remedy	Nodeset file must be stored under /sdcard/plc/prg/pta/opc_ua_server/ Check Nodeset file name in Mapping file.
0x1305 Error	Mappingfile at line %s, column %s is erroneous	
	Cause	Content of Mapping file is invalid.
	Remedy	Check mapping information in mapping file.
0x1306 Error	Mappingfile includes more nodesets than supported	
	Cause	The mapping file configures more nodesets than are supported.
	Remedy	Reduce the number of nodesets according to chapter OPC UA Server Capabilities (📖 58).
0x1201 Warning	Update of address space is incomplete	
	Cause	The address area e.g. "PLCopen" is incomplete
	Remedy	Reduce the number of symbols in the symbol configuration in the »PLC Designer«. Further information about the handling of required resources for the PLCopen namespace can be found in chapter Resource requirements in the server address space (📖 83).
0x1202 Warning	Client user authentification failed	
	Cause	Client user authentication has failed.
	Remedy	Check user administration within PLC Designer (check existence of user and its password). Are the OPC UA client credentials correct?

ErrorCode Type	Description	
0x1203 Warning	Client (%s) operation %s failed with status code %s	
	Cause	The client (e.g. urn:CDE279763:UnifiedAutomation:UaExpert) operation "Read" failed with status code e.g. "BAD_NODEID_UNKNOWN".
	Remedy	Check existence of nodes and client rights. PLC project → Check symbol configuration and PLC status
0x1204 Warning	Too many Client requests on subsystem %s	
	Cause	The subsystem cannot handle all client requests.
	Remedy	Check client interaction (check session count or access frequency). DOS attack?
0x1205 Warning	Client access denied on subsystem %s	
	Cause	Unauthorized access to parts of the information model was detected and denied.
	Remedy	Check client rights. Match with error code "Client Operation Failed" 0x1203
0x1206 Warning	Client has been rejected because the certificate is not trusted	
	Cause	Client connection was rejected by the server due to untrusted client certificate.
	Remedy	Open PLC Designer security screen and check certificates. Trust or update the client certificate.
0x1207 Warning	Needed %s element for mapping of %s is missing or erroneous.	
	Cause	The mapping file contains incorrect entries (e.g. missing elements that are needed, wrong values like data type or NodeId).
	Remedy	Correct mapping file and add/correct missing element.
0x1208 Warning	Namespace-ID for namespace-uri %s could not be found	
	Cause	The namespace uri for a mapping element is incorrect and cannot be mapped to the server's namespaces.
	Remedy	Check and correct namespace uri
0x1209 Warning	Nodeid for csv name %s could not be found	
	Cause	The placeholder keyword for the nodeid is incorrect and could not be found.
	Remedy	Check placeholder keyword or update CSV file.
0x1210 Warning	Nodeid %s of element %s not valid	
	Cause	The NodeId is either empty or has an incorrect value that does not comply with the Opc Ua specification (e.g. wrong identifier type).
	Remedy	Check NodeId and align with the Nodeset.
0x1211 Warning	Element % is missing for mapping of %s. Skipping element	
	Cause	The mapping file contains a mapping element where the ParameterElement or VariableElement is missing.
	Remedy	Correct mapping file and add missing element.

ErrorCode Type	Description	
0x1212 Warning	CSV-File %s could not be found	
	Cause	Referenced csv file can´t be found.
	Remedy	csv file must be stored under /sdcard/plc/prg/pta/opc_ua_server/ Check csv file name in mapping file.
0x1213 Warning	CSV-File %s at line %s is errornous	
	Cause	The CSV format of the CSV file is incorrect.
	Remedy	Check CSV file again.
0x1214 Warning	Namecollision for %s in csv file %s	
	Cause	There are several entries with the same name in the CSV file.
	Remedy	Fix name collision.
0x1215 Warning	The arguments of the node %s of the Mappingfile do not match the arguments of the nodeset file. Skipping element	
	Cause	Replace first argument with placeholder if necessary (methodargument, eventargument). The mapping file does not match the nodesets. Maybe different versions were used where the arguments do not match.
	Remedy	Align mapping file and nodeset file and check methods/events
0x1216 Warning	The notifier %s for the node %s could not be found. Setting notifier to server node	
	Cause	The NodeId of the notifier could not be found in the nodeset.
	Remedy	Synchronize mapping file and nodeset files and check OptionalPlaceholder/Events
0x1217 Warning	The method %s was called again, before the current execution was finalised	
	Cause	A mapped method was called again before the first call was completed.
	Remedy	Wait until method has been finalized. Increase client timeout if necessary.
0x1218 Warning	Out of memory (%s)	
	Cause	Not enough memory
	Remedy	Reduce number of sessions, monitored items, …
0x1219 Warning	Internal warning (%s)	
	Cause	An internal operation has failed.
	Remedy	–
0x1401 Info	Started on %s	
	Cause	Server was started successfully.
	Remedy	–
0x1402 Info	Is shutting down	
	Cause	Server is shut down.
	Remedy	• Device restart. • Update device firmware. • Contact support. Create trace.

ErrorCode Type	Description	
0x1403 Info	Certificate store changed	
	Cause	The certificate store has been updated.
	Remedy	–
0x1404 Info	User database changed	
	Cause	The user database has been updated.
	Remedy	–
0x1405 Info	Client %s connected	
	Cause	Client (e.g. urn:CDE279763:UnifiedAutomation:UaExpert) has connected to server.
	Remedy	–
0x1406 Info	Client %s disconnected	
	Cause	Client (e.g. urn:CDE279763:UnifiedAutomation:UaExpert) has disconnected from the server.
	Remedy	–
0x1407 Info	Client %s rejected	
	Cause	Client (e.g. urn:CDE279763:UnifiedAutomation:UaExpert) was rejected by the server.
	Remedy	The client connection was rejected, due to the occupied session count (all reserved sessions are in use).
0x1408 Info	Address space %s changed	
	Cause	The address area, e. g. "PLCopen" has been changed.
	Remedy	–
0x1409 Info	The mapping was loaded. %s elements have been skipped due to mapping errors.	
	Cause	The mapping was loaded. X Elements were skipped due to mapping errors
	Remedy	–
0x1410 Info	No nodeset files are specified in mapping file. Disabling provider.	
	Cause	No nodeset files were specified in the mapping file. The provider is disabled.
	Remedy	–

Subscriptions and monitored items

As a server utilization indication, Lenze parameters are available for…

- maximum [0x2472:011] and used [0x2473:011] number of subscriptions and
- maximum [0x2472:012] and used [0x2473:011] number of monitored items.

Publishing and sampling intervall

In case of performance problems, the minimum possible publishing and sampling interval can be increased. For this purpose, following Lenze parameter setting parameters are available…

- minimum publishing interval [0x2471:013]
- minimum sampling interval [0x2471:014]

To get the settings active, a Server restart is necessary via the parameter [0x2470:001]. After the restart the active values are represented in the corresponding parameters [0x2472:013] and [0x2472:014].

Server information modeling

PLCopen

The OPC UA Server can be configured via the »PLC Designer« with a standardized information model according to PLCopen. With this configuration PLC objects from the IEC 61131 application are published via OPC UA.

The configuration steps for the PLCopen information model can be found under PLC application with OPC UA access (📖 73).

Array Expansion

The PLCopen information model represents IEC array elements as individual nodes. This array expansion is enabled [1] by default. It supports older Clients that don´t offer the feature of array complete access. If the Clients supports array complete access, the array expansion in setting parameter [0x2471:051] can be disabled [0] to reduce the complexity of the UA namespace and the memory consumption.

To get the setting active, a Server restart is necessary via the parameter [0x2470:001]. After the restart the active value is represented in the corresponding parameters [0x2472:051].

For further information how arrays are modeled, see Array IEC type mapping (📖 76).

Resource utilization

The number of various IEC types, which should be represented in the automatically generated PLCopen namespace utilizes the resource.

When the percentage value of the resource utilization parameter [0x2473:052] approaches 100% the resource limit is reached. Use the controller logbook for detailed resource utilization information.

Counter measure is to reduce various variable types in the »PLC Designer« via the "Symbol Configuration". Make sure to publish only necessary variables for the use case and not complete function blocs with all their member variables.

For detailed information see Resource requirements in the server address space (📖 83).

User specific

From Controller Version V01.09. it is possible to load user specific information models.

The number of nodes and used companion specifications of the user specific information model utilizes the resource of the Controller.

When the percentage value of the resource utilization parameter [0x2473:053] approaches 100 % the resource limit is reached. Use the controller logbook for detailed resource utilization information.

Counter measure is to reduce the number of nodes and complexity of the user specific information model.

Session Management

The maximum number of sessions can be found in Lenze OPC UA and customer-specific client connections (📖 61)

The current number of active OPC UA sessions is available by the Server address space as well as by Controller parameters. Browsepath in the address space:

- Root.Objects.Server.VendorServerInfo.ExtendedServerDiagnostics

External sessions

The maximum number of external sessions can be reduced or increased due to security, performance or availability issues via the setting parameter [0x2471:103].

To get the setting active, a Server restart is necessary via the parameter [0x2470:001]. After the restart the active value is represented in the corresponding parameters [0x2472:103].

The number of used external sessions are available in the parameter [0x2473:103]. It can be an indicator of the Server utilization.

To know "who" is connected to the Server, the parameters [0x2473:130], [0x2473:131] and [0x2473:132] provide the connected client Application URI.

Lenze engineering and system sessions

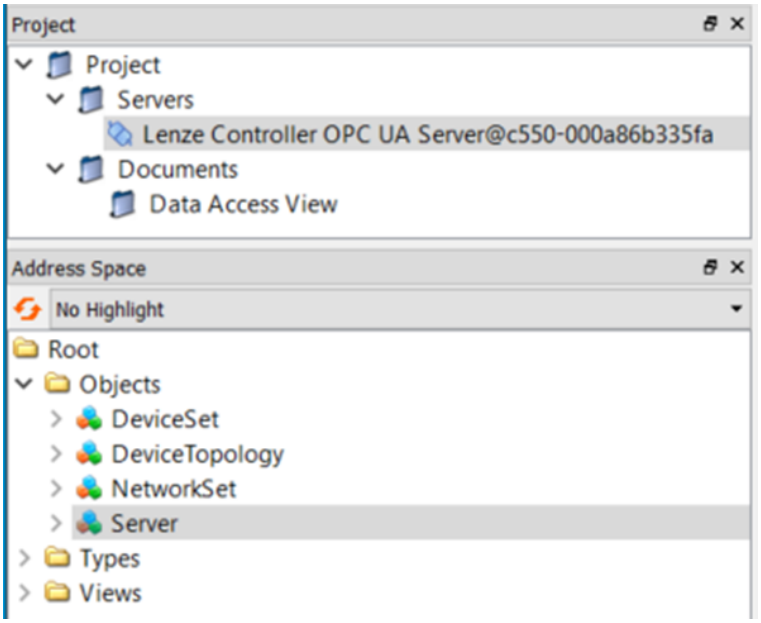
The number of used Lenze engineering sessions are available in the parameter [0x2473:101].

The number of used Lenze system sessions are available in the parameter [0x2473:102].

Both can be an indicator of the Server utilization.

Generic diagnosis of the Lenze OPC UA Server

Diagnostics can also be performed via the address space of Lenze OPC UA server, for example with a generic client like the UA Expert by Unified Automation. Standardized information is available in the server address space under the server object in accordance with OPC UA Specification Part 5.



The object *Server* is defined in the OPC UA specification Part 5:

- https://reference.opcfoundation.org/v104/Core/docs/Part5/6.3.1/

The most important diagnostic information is briefly explained below.

Server status

Under the variable *ServerStatus* information about the server status, the current time or also build information can be queried.

The variable *ServerStatus* is defined in the OPC UA specification Part 5:

- https://reference.opcfoundation.org/v104/Core/docs/Part5/12.10/

Server Diagnostics

Under the object *ServerDiagnostics* and the subordinate nodes, detailed diagnostic information of the server and the current utilization of the server capabilities can be queried. This includes, for example:

- Listing and identification of connected clients
- Number of current sessions
- Number of aborted sessions
- Number of rejected sessions
- Number of established subscriptions
- Number of read or write requests

The object *ServerDiagnostics* is defined in the OPC UA specification Part 5:

- https://reference.opcfoundation.org/v104/Core/docs/Part5/6.3.3/

Server Capabilities

Under the object *ServerCapabilities* the current properties and limitations of the server can be viewed.

For an overview of the platform dependent server capabilities and limitations available in the Lenze system please refer to "Server Capabilities".

The object *ServerCapabilities* is defined in the OPC UA specification Part 5:

- https://reference.opcfoundation.org/v104/Core/docs/Part5/6.3.2/

Namespace Array

Under the variable *NamespaceArray* you can see under which index the OPC UA Server has loaded the current namespaces. All nodes in the address space of the server are assigned to a namespace. Via the displayed NamespaceUri you can also see who is the originator of the corresponding namespace.

The variable *NamespaceArray* is defined in the OPC UA specification Part 5:

- https://reference.opcfoundation.org/v104/Core/docs/Part5/6.3.2/

In addition to the basic standard namespaces of the OPC UA Foundation, there are embedded namespaces for mapping the PLC application according to PLCopen and the non-public Lenze internal engineering access.

PLC application with OPC UA access

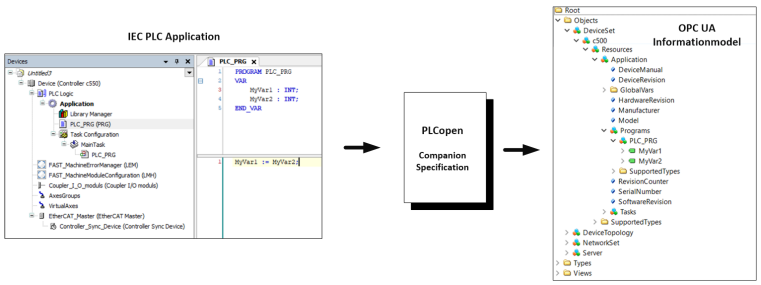
The Lenze OPC UA server on the controller offers the possibility to publish variables of the PLC application according to the PLCopen Companion Specification. This serves various use cases such as the connection of the application to a visualization or to an MES system.

This chapter describes the configuration of a standardized PLCopen information model for publishing PLC variables.

The Lenze OPC UA Server is also able to import user specific information models based on Companion Specifications. This approach enables the representation of OPC UA Methods and Events. For this see Import and map information models (📖 95).

PLCopen Companion Specification

The PLCopen Companion Specification describes in a generic way how a PLC application is represented in an OPC UA information model.



Here, the specification not only addresses the pure representation of IEC variables, but also describes the embedding of the application in the controller system, such as task assignment.

The following figure from the specification shows an example for OPC UA object and variable instances of a controller with PLC application and application specific types.

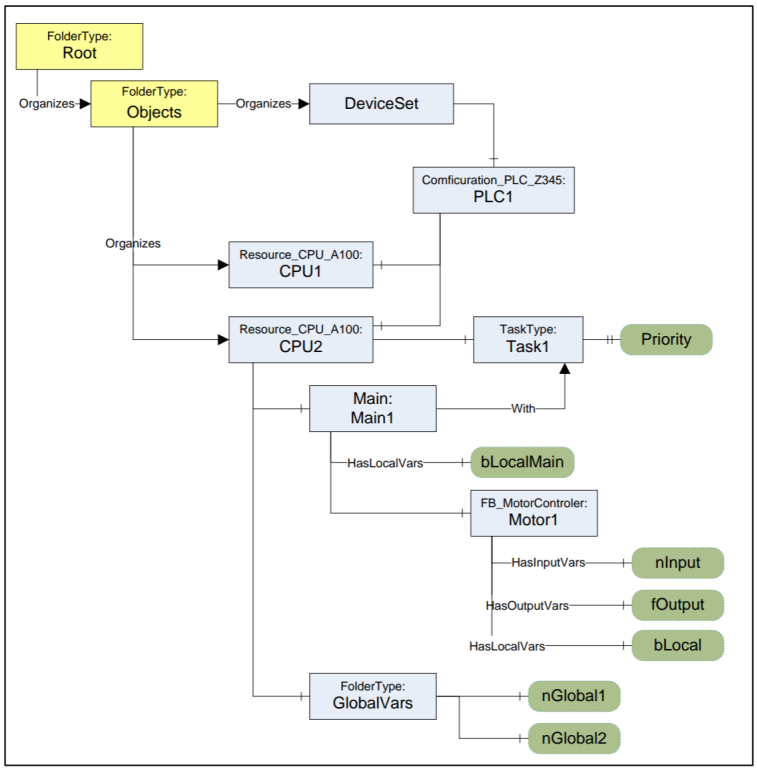


Figure from PLCopen Companion Specification

Detailed information can be found in the PLCopen Companion Specification at the following link:

- https://www.plcopen.org/system/files/downloads/plcopen_opcua_information_model_for_iec_61131-3_version_1.00.pdf

The following chapters describe the most common IEC types and their mapping in OPC UA.

Mapping of primitive IEC types

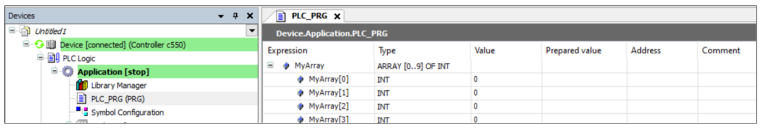
The following tables show how IEC types are represented as OPC UA types according to the PLCopen Companion Specification.

IEC Data type	OPC UA Data type	PLCopen Companion Specification Data type	Comment
BOOL	Boolean	–	
SINT	SByte	–	
USINT	Byte	–	
INT	Int16	–	
UINT	UInt16	–	
DINT	Int32	–	
UDINT	UInt32	–	
LINT	Int64	–	
ULINT	UInt64	–	
BYTE	Byte	BYTE	
WORD	UInt16	WORD	
DWORD	UInt32	DWORD	
LWORD	UInt64	LWORD	
REAL	Float	–	
LREAL	Double	–	
STRING	String	STRING	
CHAR	Byte	CHAR	
WSTRING	String	–	
WCHAR	UInt16	WCHAR	
DATE_AND_TIME	DateTime	DT	
DATE	DateTime	DATE	
TIME_OF_DAY	UInt32	TOD	
TIME	Int64	TIME	
LDATE	Int64	LDATE	Not supported
LTIME	Int64	LTIME	
LTOD	Int64	LTOD	Not supported
LDT	Int64	LDT	Not supported

To give the user a clearer representation of the data types, new OPC UA data types have been added to the PLC Open specification. These are each derived from the original OPC UA data type and only serve for a better overview.

Array IEC type mapping

In IEC programs arrays can be defined as variables.



In principle, these IEC arrays can also be mapped directly to OPC UA variables, whereby the OPC UA variable reveals the structure of the array via additional properties.

According to the PLCopen Companion Specification, an OPC UA array node has the following additional properties:

- Dimensions
- IndexMax
- IndexMin

The following describes how IEC arrays are represented and accessed in the Lenze OPC UA information model.

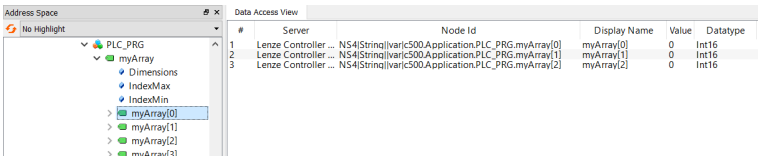
The option can be selected via the parameter [0x2471:051]. For this, see also in Server information modeling (📖 69).

Array Expansion enabled [1] (default)

Each IEC array element is represented as a separate node in the OPC UA information model. Especially older OPC UA clients only support this access option.

The advantage of this option is that especially older OPC UA clients only support this access option. In addition, a targeted reading and writing to an array element is possible.

The disadvantage of this option is that the information model becomes very complex and, depending on the array size, a corresponding number of nodes is required.



IEC variable = myArray : ARRAY [0..9] OF INT;

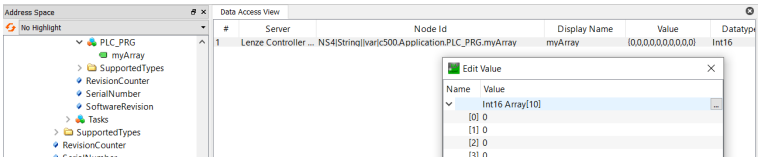
Required number of nodes = 14 (1 for the array itself and 3 for the properties and 10 for the array elements)

Array Expansion disabled [0]

Here the entire array is represented only as a central node with its properties. The individual array elements [0], [1], ... [n] are not available in the OPC UA information model as separate nodes.

The advantage of this option is a minimization of the required OPC UA nodes in the information model and thus a lower complexity.

The disadvantage of this option is the handling and interpretation of the central array node on client side. This is defined in Part 4 Services of the OPC UA specification.

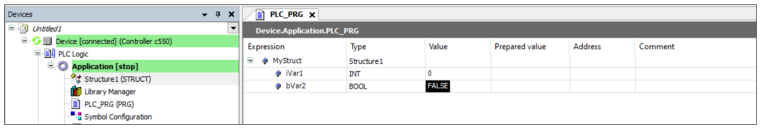


IEC-Variable = myArray : ARRAY [0..9] OF INT;

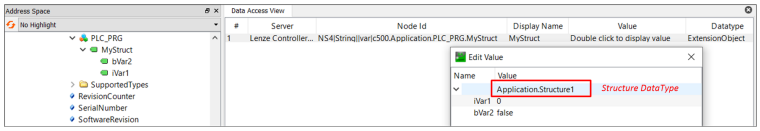
Required number of nodes = 4 (1 for the array itself and 3 for the properties)

Structure IEC type mapping

In IEC programs structures can be defined as types (DUT). Instances formed from these are mapped as individual variables with their member variables.



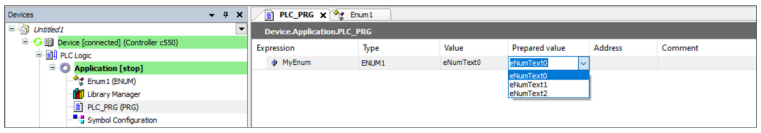
These IEC structures are also represented in the OPC UA information model as individual variables with their subordinate member variables.



The IEC type information is also available in the OPC UA information model. The corresponding OPC UA structure type is a derivative of the OPC UA DataType Structure and is available in binary format on the server.

Enum IEC type mapping

In IEC programs enums can be defined as types (DUT). Instances formed from these are mapped as individual variables.

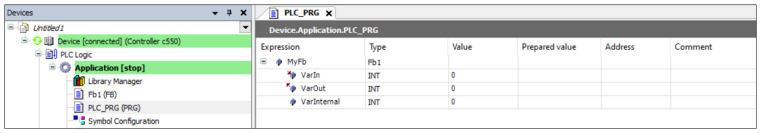


These IEC enums are also represented in the OPC UA information model as individual variables.

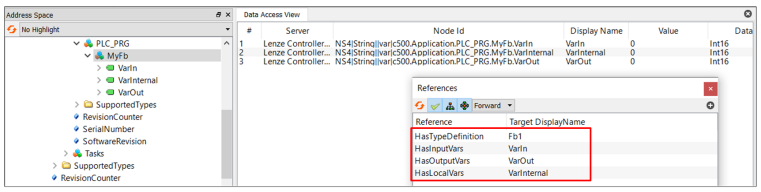
The IEC type information is also available in the OPC UA information model. The corresponding OPC UA Enum type is a derivative of the OPC UA DataType *Enumeration*. These types have a property with listing the possible values and their string equivalents.

Function block IEC type mapping

In IEC programs function blocks can be defined as types (POU). Instances formed from these are mapped as individual variables with their member variables.



These IEC function blocks are also represented in the OPC UA information model as individual variables with their subordinate member variables. Unlike structures, the member variables of function blocks are referenced in the OPC UA information model after their declaration, for example as input or output variables.



The IEC type information is also available in the OPC UA information model. The corresponding OPC UA type is a derivation of the OPC UA *CtrlFunctionBlockType*, which is defined in the PLCopen Companion Specification.

Configure PLCopen information model

The PLCopen information model can be configured and activated in a simple way. It is explained in the following chapters.

Create PLC application

First of all, a PLC application must be created with corresponding variables that are to be published via OPC UA.

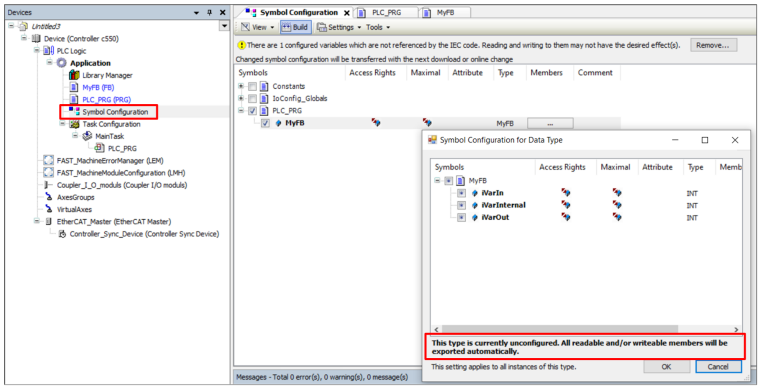
1. Create a new controller project with the PLC Designer.
2. Create a PLC application for your use case.
3. Declare variables of different types within the PLC application that you want to publish via OPC UA.

For complex PLC applications it is helpful to declare separate variables for OPC UA communication. The advantage is that it is easier for the programmer to see what is to be released for OPC UA communication and that internal application variables are not mistakenly accessible via OPC UA.
This can be done, for example, via a separate global variable list or via a structure prepared specifically for the use case.
In some situation it is also helpful to copy internal application variables to variables to be published within the PLC application. For example, if only one element from an array is to be published, this can be copied to a separate variable especially for communication. Thus, the complete array is not accessible via OPC UA.

Configure variables for OPC UA access

In a next step the variables to be published via OPC UA have to be selected from the PLC application. The symbol configuration is available for this purpose.

1. Insert a Symbol configuration object below the application.
2. Activate the option Support OPC UA functionalities in the dialog "Add symbol configuration".
3. Open the symbol configuration in the editor.
4. Select the Create command.
   The variables are displayed in a tree structure
5. Activate the variables that you want to publish via OPC UA. Set the access rights.
6. Transfer the project to the controller.



The handling of the symbol configuration is described in detail at the following link:

- CODESYS Online Help

Make sure that only the IEC variables are selected that are required for the use case and are therefore also to be published. If a variable is selected in the symbol configuration, the subordinate member variables are automatically activated for OPC UA access.
If you have a user administration active on the controller, IEC variables can be assigned to symbol groups and these in turn to user groups. Thus, a certain user has access only to the IEC variables intended for him. For the configuration of symbol groups see CODESYS Online Help.

Methods

OPC UA Methods are represented in the PLC application as Function Blocks with specific input and output variables for handshaking with the OPC UA Server.

If a client calls the OPC UA Method, the Server will set the Function Block input variables, with which the method operation can be triggered. The Function Block output variables signals the OPC UA Server, that the operation has been finished and the calculated output values.

The Lenze IEC library L_IOSP_OpcUaServer provides for this purpose the generic Function Block L_IOSP_MethodCall. This Function Block is used to extend an application specific Function Block with method specific input and output arguments.

The application specific Function Block (with all the input and output variables) has to be published via the symbol configuration to grant access by the OPC UA Server.

For further information see the L_IOSP_OpcUaServer documentation. To realize projects with methods please contact your local Lenze sales representative.

Events

OPC UA Events are represented in the PLC application as Function Blocks with specific input and output variables for handshaking with the OPC UA Server.

The Function Block itself "fires" the Event and sets the Event properties via the Function Block output variables. The OPC UA Server sends the Event notification to an OPC UA Client and gives the "done" feedback to the function Block via the input variables.

The Lenze IEC library L_IOSP_OpcUaServer provides for this purpose the generic Function Block L_IOSP_BaseEventType. This Function Block is used to represent an OPC UA BaseEventType. The Function Block is also used to extend other application specific Event types with additional properties.

For further information see the L_IOSP_OpcUaServer documentation. To realize projects with events please contact your local Lenze sales representative.

Resource requirements in the server address space

In the PLC Designer, variables are published via the symbol configuration and can thus be exchanged via various interfaces such as OPC UA or also the so-called network variables.

Here the variables and the associated types of the symbol configuration are counted in the PLC-Designer. The result is displayed in a compiler log output.

Since an associated PLCopen information model is only generated dynamically in the OPC UA Server, a concrete determination of the required resources is not possible during offline engineering.

The final resources required and any limit exceeded are currently displayed in the controller logbook. Further information about server diagnostics can be found in chapter ???.

The hard limiting values refer to the type model of the Lenze OPC UA Server. Exceeding these will result in an error message and a reset of the information model. There is no hard limitation regarding the required number of nodes and references in the instance model of the server. This means that theoretically many instances of a PLC variable type can be published without hitting a hard limit.

However, a distinction must be made here between Struct and FB`s data types in the type model. The address area is created in two stages.

- In phase 1, all Struct data types and also enumerations are first created in the OPC UA address space.
- In phase 2 the FB types are created in the OPC UA address range.

If an "out of memory" error occurs in phase 1, this is a serious error as it can result in severe functional restrictions in Data Access. For example, complete access (the complete reading of an entire data structure via a node value attribute) may then no longer be possible.

Therefore, if an error occurs in phase 1, we abort the address space creation and delete the previously created address space completely. This results in an error.

If an "Out of Memory" error occurs in phase 2, this is unpleasant but not a serious error, since this normally does not result in any function cuts in the Data Access. Not all FB types can be created, but the missing type information is often not needed in the PLCopen context.

Therefore, if an error occurs in phase 2, we abort the address space creation without deleting the previous created address space. This results in a warning.

Resource limitations in the PLCopen type model

The controller reserves certain resources of the system for the type model. These can be divided into different pools.

- Maximum number of nodes pool = 12010
  Each element in a UA address space is a node. There are different classes of nodes (objects, variables, methods, data types). Each node class has a defined number of attributes. The maximum number of these nodes in the type model is limited.
- Maximum number of variables pool = 10008
  Variables are OPC UA nodes that contain a value. Variables can map primitive (bool, int, ...) and complex types (structures). Variable types can contain further (nested) variable nodes. The type definition of these variables is the limiting factor in the type model.
- Maximum number of references pool = 20020
  The basic modelling concepts of OPC UA are nodes and references between nodes. A reference is an explicit relationship (a pointer) from one node to another. The number of used references in the type model is the limiting factor here.
- Maximum number of dimensions = 10008
  Variables can not only hold scalar values, but also arrays in different dimensions. As soon as an array is present, array dimensions (ArrayDimensions) are specified (one dimension is required per array dimension). The sum of the dimensions in the type model is the limiting size here.

Example of required resources

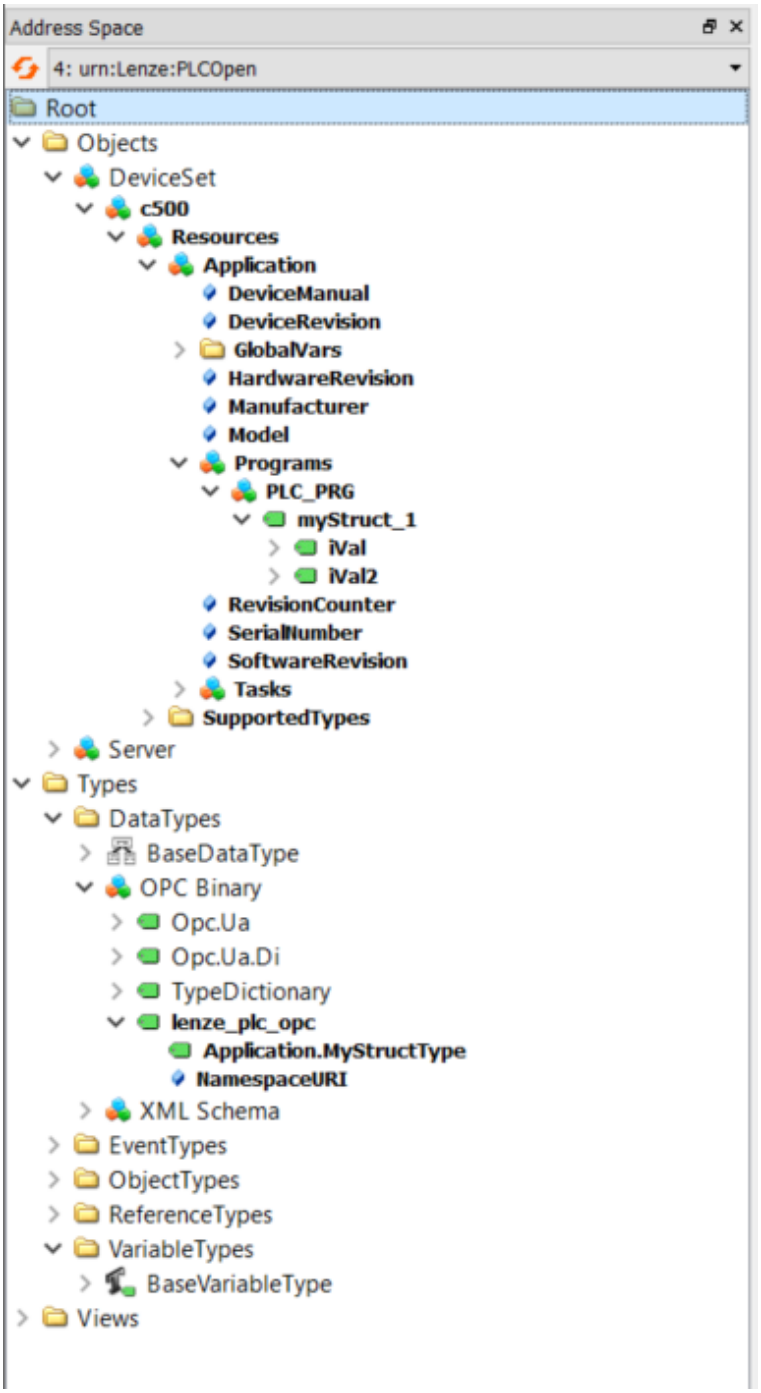
Theoretical limitation for pure IEC structure types

Independent of the number of member variables.

	Required per structure	Pool limit	Max. 2994 structures
Node	3	12010	12009
Variables	1	10008	2994
References	5	20020	14970
Dimensions	0[1)]	10008	0

1) Depending on number of member variables with arrays, without arrays = 0

This results in a **maximum of 4003 different structure types** until the above pools are exhausted.



Theoretical limitation for FB-only types

Example with 5 primitive member variables.

	Required per FB [1]	Pool limit	Max. 1331 structures
Node	6	12010	8004
Variables	5	10008	6670
References	15	20020	20010
Dimensions	0[2]	10008	0

1) Exception are FB in FB, there are no VariableTypes but ObjectTypes needed

2) Depending on number of member variables with arrays, without arrays = 0

This results in a **maximum of different 1334 FB types with 5 member variables** until the above pools are exhausted.

Example with 10 primitive member variables.

	Required per FB [1]	Pool limit	Max. 665 structures
Node	11	12010	7315
Variables	10	10008	6650
References	30	20020	19950
Dimensions	0[2]	10008	0

1) Exception are FB in FB, there are no VariableTypes but ObjectTypes needed

2) Depending on number of member variables with arrays, without arrays = 0

This results in a **maximum of different 665FB types with 10 member variables** until the above pools are exhausted.

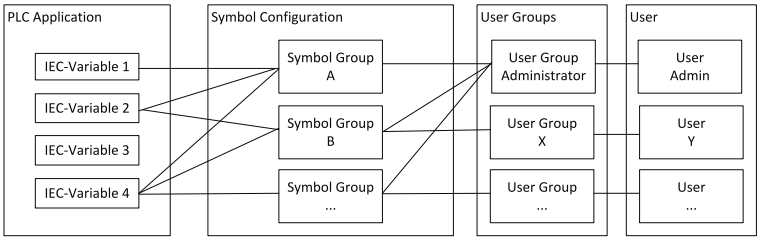
Remedial action when resource limits are exceeded

- Only release the variables via the symbol configuration that you need for your use case. Regardless of the resources required, this should be considered for security reasons.
- Avoid the activation of top-level variables in the symbol configuration and thus an implicit activation of all subordinate variables. Especially the variable budget of the "IOConfigGlobals" quickly reaches the resource limits.
- Avoid the activation of Functionblocks of the FAST IEC Libraries. Technology Modules and Motion FB´s are internally very complex and have therefore a complex type definition with several needed subtypes of structures. Even if you want to publish a single boolean variable like "xError" from a functionblock, the whole functionblock type definition with all the subtypes is needed in the UA addressspace.
- For complex PLC applications it is helpful to declare separate variables for OPC UA communication. The advantage is also that it is more manageable for the programmer what should be released for OPC UA communication and not mistakenly internal application variables are accessible via OPC UA.
- This can be done, for example, via a separate global variable list or via a structure prepared specifically for the use case.
- In some situations it is also helpful to copy internal application variables to variables to be published within the PLC application. For example, if only one element from an array or an axis data structure is to be published, this can be copied to a separate variable especially for communication. This means that the complete array or axis data structure is not accessible via OPC UA.

Symbol groups

The variable household that is to be published via OPC UA can be divided into groups, which in turn can be assigned to user groups. Thus, a certain user has access only to the IEC variables intended for him.

A schematic representation of this relationship can be found in the following figure as an example.



The activation of the device user management with users and user groups is a prerequisite for the use of symbol groups. The configuration of the device user management is explained under the following link:

- CODESYS Online Help

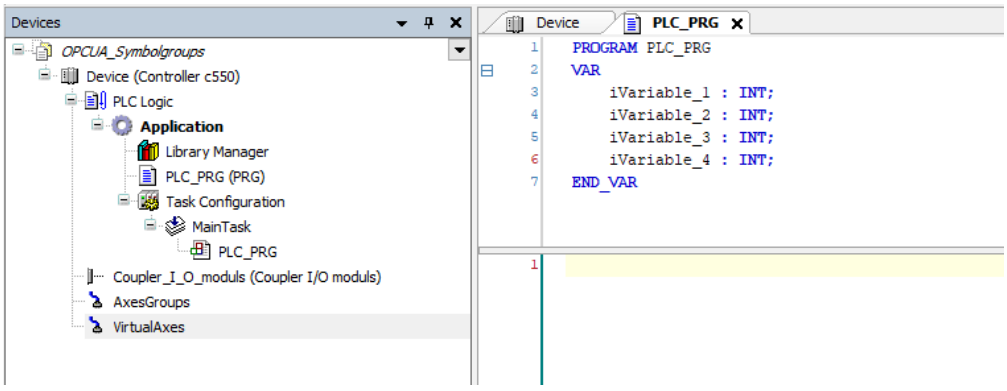
The creation of symbol groups with IEC variables as well as the assignment of rights with regard to device user management is explained under the following link:

- CODESYS Online Help

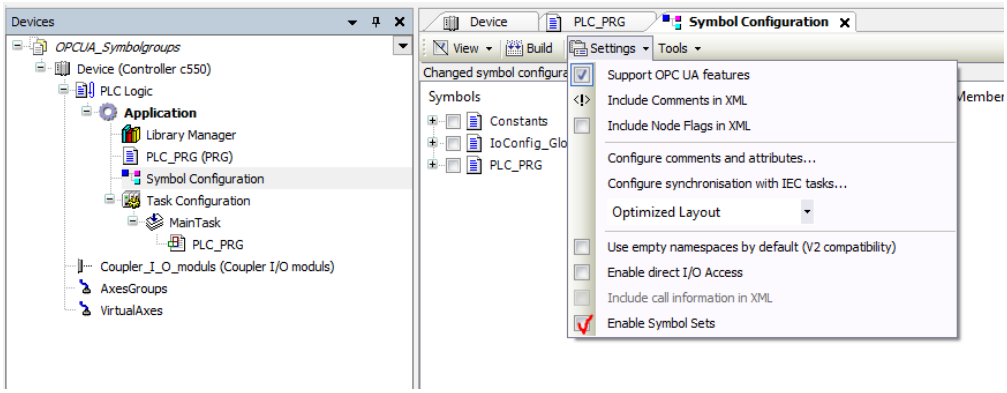
When changing existing symbol groups and assignments to user groups, make sure that this is also properly transferred to the controller and activated. This is at least the case after an "online change" or project download.

The following is an example of how to configure the user administration and symbol groups:

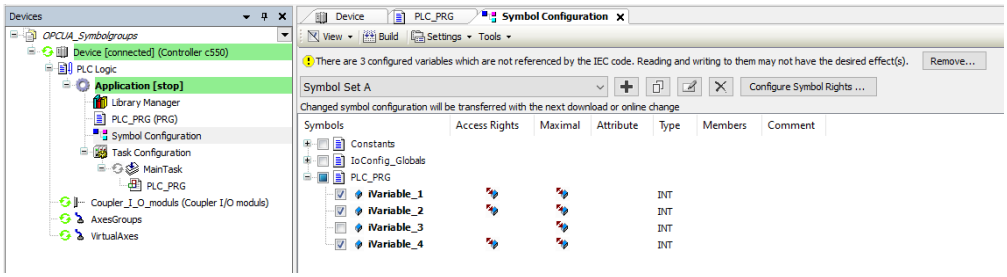
1. Create a PLC application on a c550 with various variables that are to be published via symbol groups.

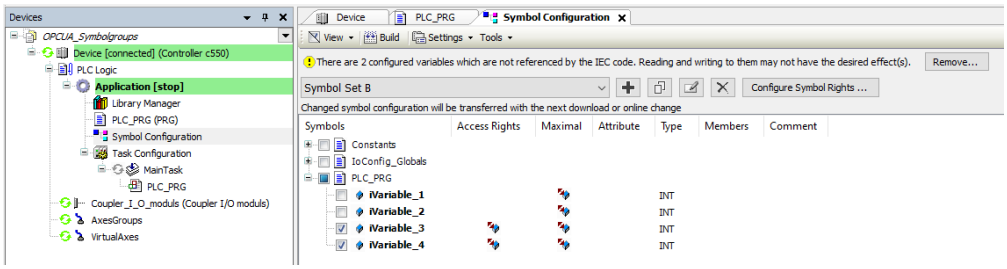


2. Add a symbol configuration and activate "Support OPC UA Features" and "Enable Symbol Sets".

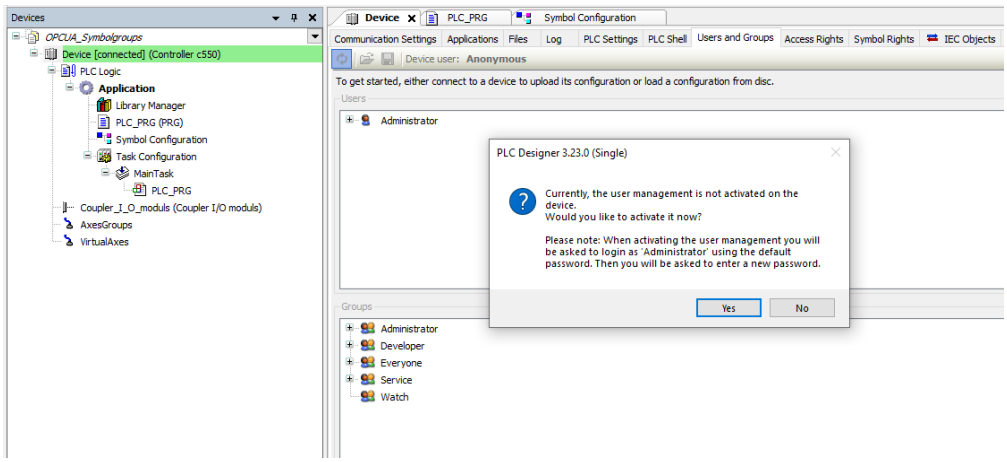


3. Create symbol groups and assign different variables to them.

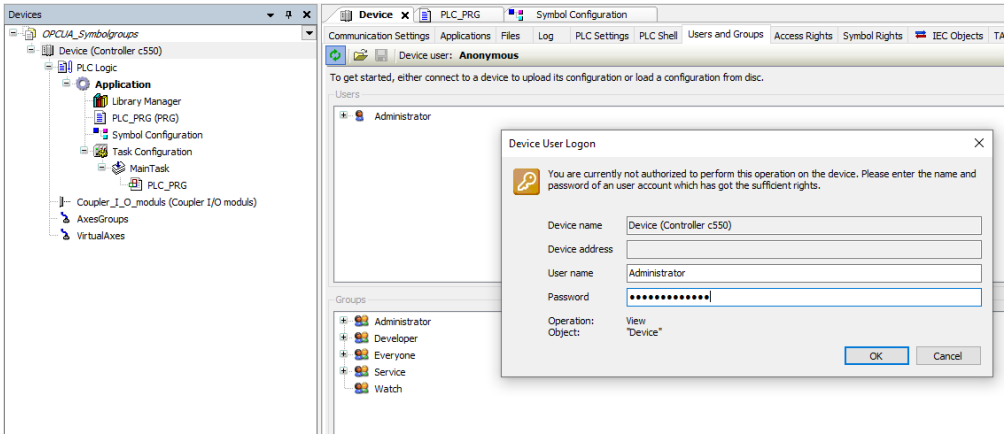




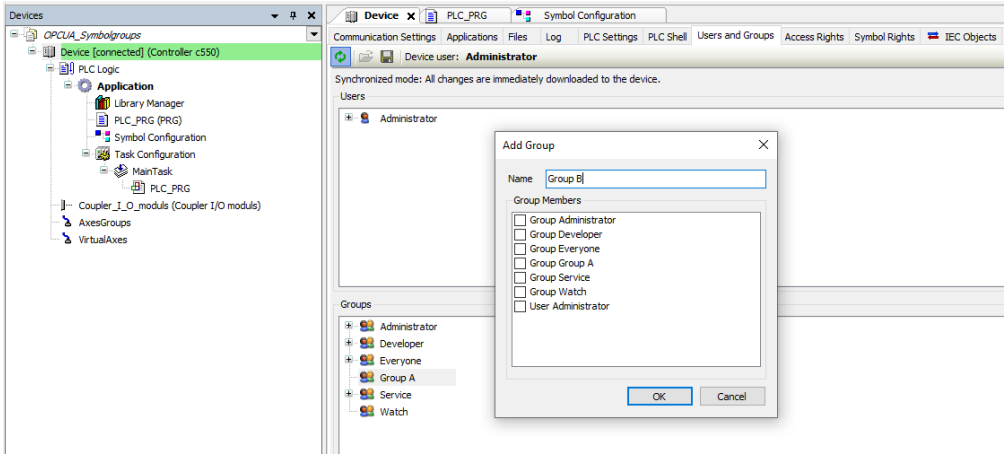
4. Activate the device user management.



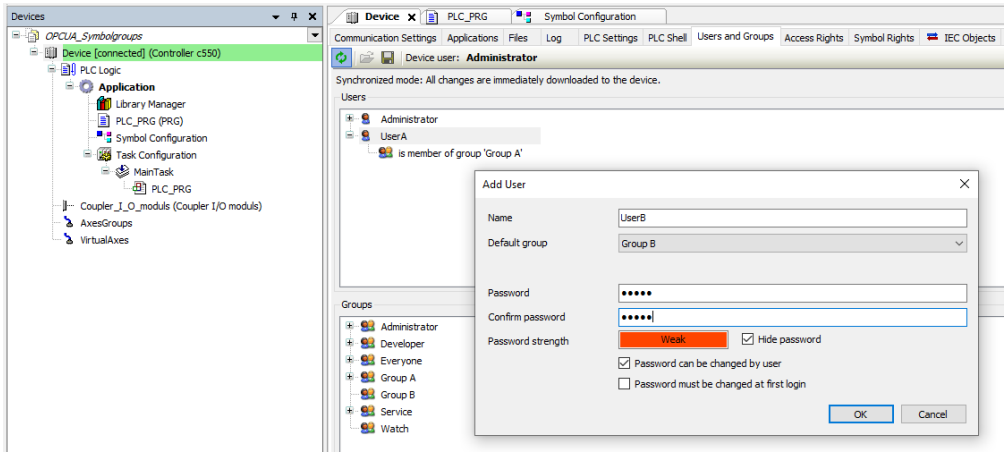
5. Log in as User = *Administrator* (PW = *Administrator*, at first login).
   Note: You will be prompted to change the password after the first login.



6. Create user groups.

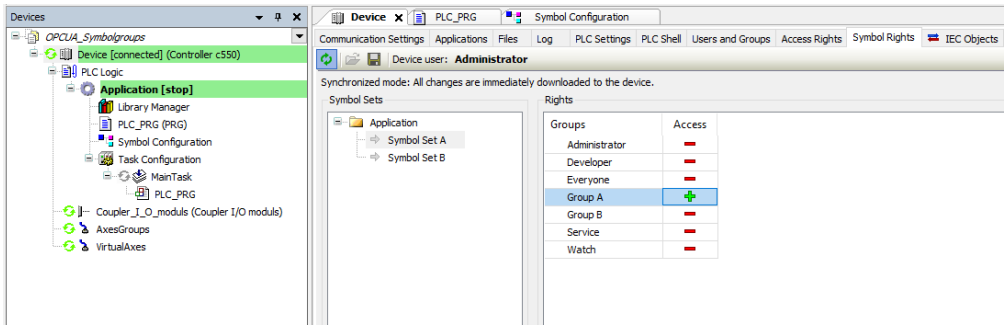


7. Create users and assign them to user groups.



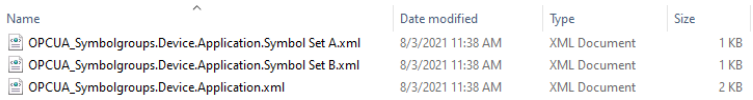
8. Load the project onto the controller.

9. Assign the access rights to the symbol groups.
   The rights can be changed with a double click on the -, + symbol.



You have configured the device user management and the symbol groups on your controller.

For further offline engineering, e.g. with regard to the FAST UI Runtime, a separate and an all-encompassing .xml file with the assigned symbols is available for each symbol group. These are located in the project folder on your engineering PC.



Access to symbol groups with the UaExpert client

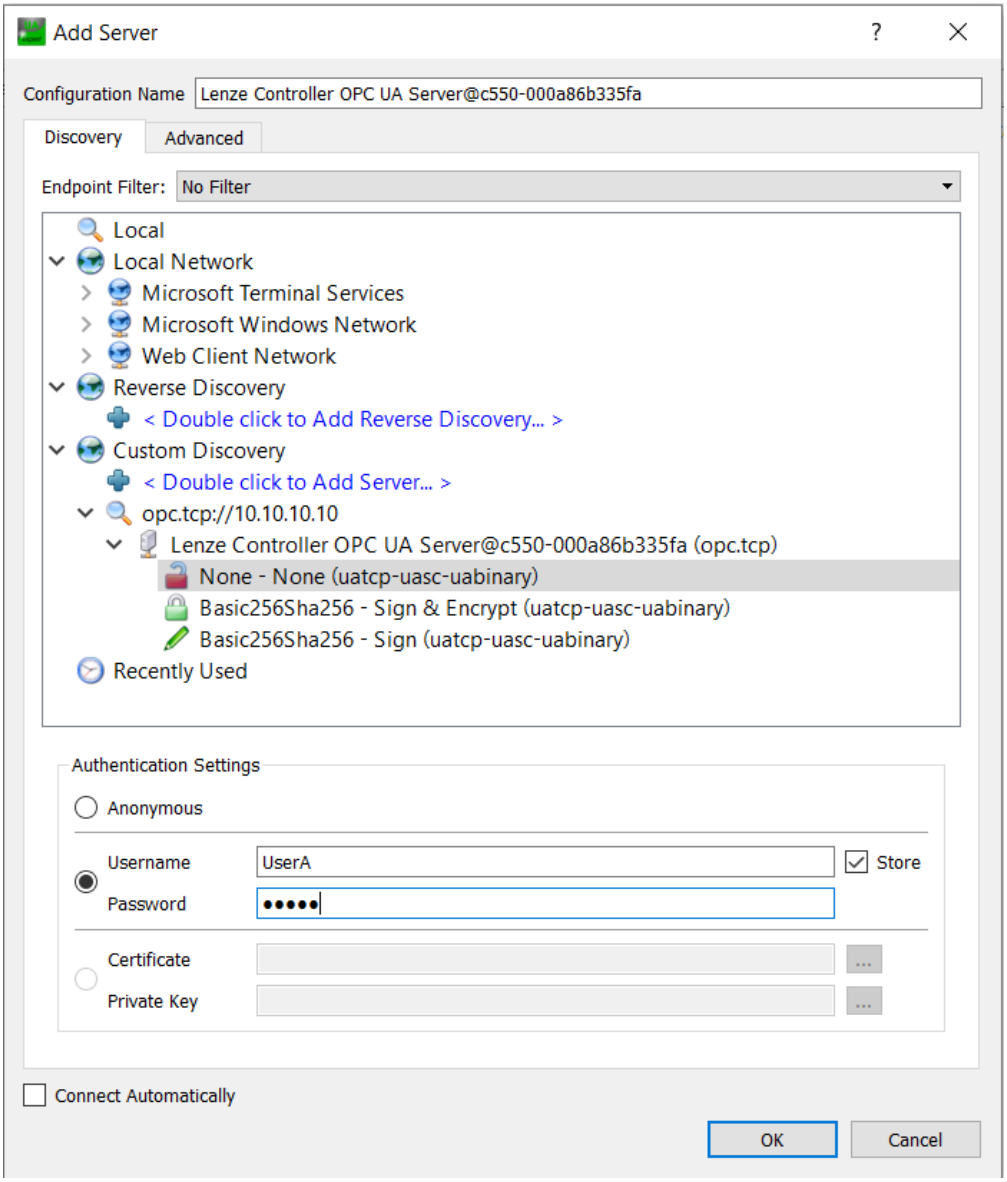
The generic OPC client "UaExpert" is with an evaluation license a freely available software of the company Unified Automation, which you can download from the Internet.

- https://www.unified-automation.com/products/development-tools/uaexpert.html

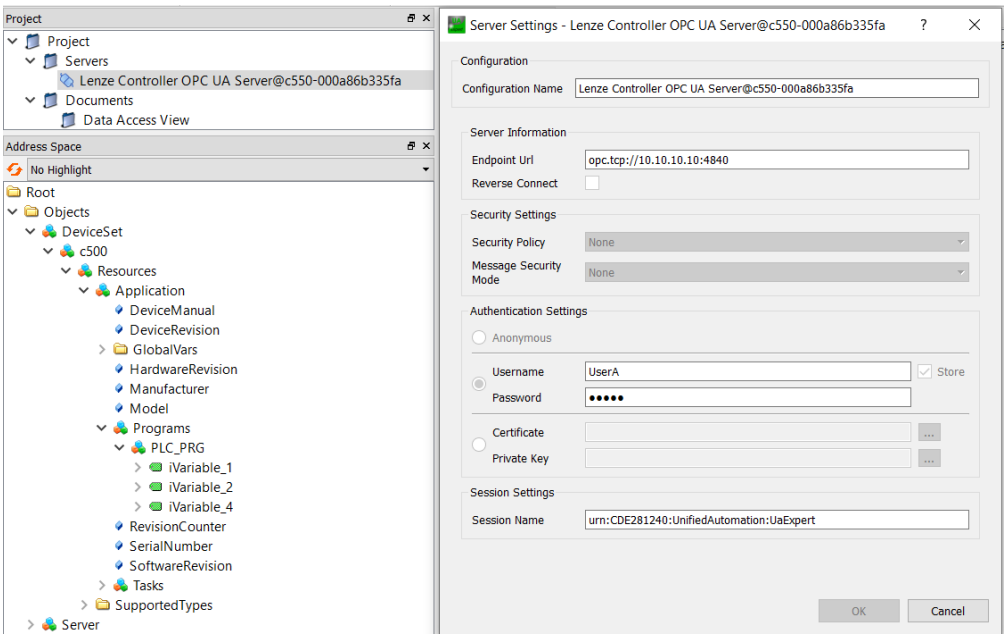
With this client you can connect to the Lenze OPC UA server. The following description refers to this program. Other OPC UA clients work similarly.

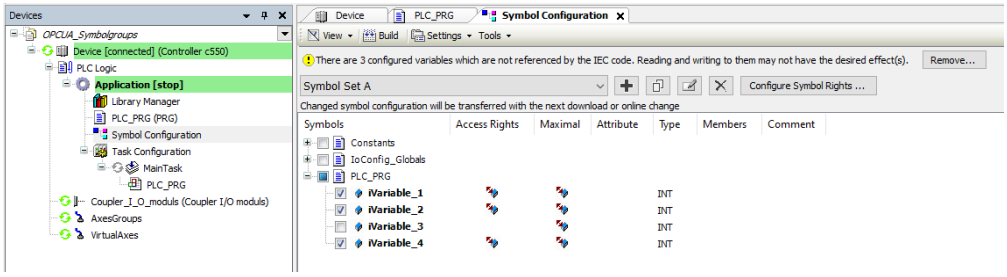
How to connect the client to the Lenze OPC UA server:

1. Start the UaExpert program.
2. Select the command "Server -> Add...".
   - The "Add Server" dialog opens.
3. Add a new OPC-Server by double-clicking on it.
4. Enter the endpoint URL: opc.tcp://<IP address of the controller>:4840.
   - So e.g. opc.tcp://10.10.10.10:4840.
5. Select the Security Settings "None-None" and enter an appropriate user name and password. Close the dialog with **OK**.



6. Only the symbols of the symbol group assigned to the user or user group can be reached.





Data consistency

The Lenze OPC UA Server accesses the PLC variables internally and displays them in its information model depending on the configuration.

Data consistency is only guaranteed for elementary data types up to 8 bytes in size. This does not apply to strings. With structures and arrays, the individual underlying variables are accessed so that these are also consistent. However, it is not guaranteed that the complete structure or array is consistent in itself.

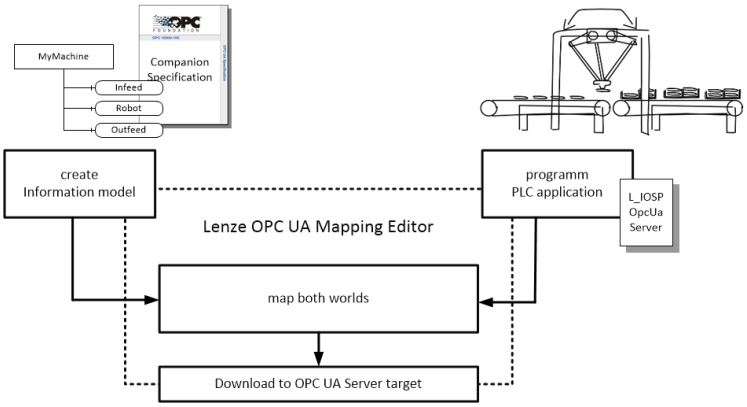
This data consistency on elementary data types up to 8 bytes is sufficient for the vast majority of use cases.

- In a visualization it could happen that a value is inconsistent for a short moment. In the next moment, however, this value is displayed correctly.
- With clock-based machines, synchronization can be guaranteed applicatively via an additional variable that is used by the server and client application for a handshake. This means that there are no problems in this case either.

Import and map information models

With the Lenze Mapping Editor OPC UA information models based on companion specifications can be created. The created information model can be mapped to target specific data sources like the PLC application.

For more information on how to use the Mapping Editor, see the related documentation.

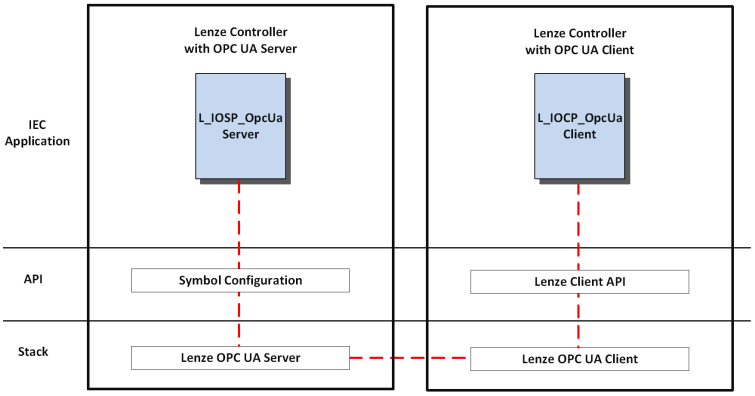


The IEC library L_IOSP_OpcUaServer is available for mapping special OPC UA data such as Methods, Events and Built-In data types to the PLC application.

L_IOSP_OpcUaServer

The following chapters describes all function blocks that are included in the "L_IOSP_ OpcUaServer" function library. The function blocks can be used to create an OPC UA Server application that performs interactions with an OPC UA Client. The OPC UA Server application can be mapped to an user specific OPC UA Information model based on Companion Specifications.

The following picture represents the main layers and used IEC libraries for an OPC UA Server and Client application hosted on a Lenze Controller.



Function blocks

In order to interact with the OPC UA Server and to map the PLC application to a user specific information model, e. g. with OPC UA methods and events, following function blocks can be used.

Function block	Info
L_IOSP_MethodCall	This generic Function Block is used to handle an OPC UA method in the PLC application for a Server.
L_IOSP_ BaseEventType	This generic Function Block is used to handle (fire) an OPC UA event in the PLC application for a Server.

Data Types

The following tables provides an overview with Data Types.

Data Type	Info
L_IOSP_LocalizedText	Used to represent an OPC UA LocalizedText in IEC. OPC 10000-3 Unified Architecture Part 3 Address Space Model Localized Text | OPC UA Online Reference (opcfoundation.org)
L_IOSP_NodeId	Used to represent an OPC UA NodeId in IEC. OPC 10000-3 Unified Architecture Part 3 Address Space Model General | OPC UA Online Reference (opcfoundation.org)
L_IOSP_ByteString	Used to represent an OPC UA ByteString in IEC. Core Data Types Byte String | OPC UA Online Reference (opcfoundation.org)

Enumerations

The following tables provides an overview with the used Enumerations

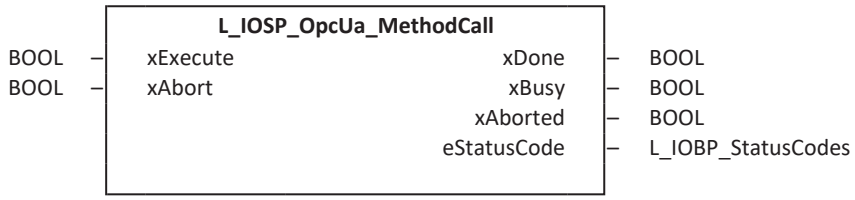
Enumeration	Info
L_IOSP_IdentifierType	Provides the OPC UA NodeID identifier types.
L_IOSP_StatusCodes	Provides the OPC UA Status Codes provided by the Server (-Application) for a Client request.

L_IOSP_MethodCall

For the execution of Methods via OPC UA the corresponding objects have to be created in the OPC UA Nodeset as well as in the PLC project. In addition to this, both instances of the methods must then be mapped.

This generic function block represents an OPC UA Method in the IEC application, which can be mapped to an OPC UA Server Method.

This basic function block contains all the variables necessary for the handling of the method. In addition to this, it provides a few methods and properties that simplify its use.



Inputs

If the function block is mapped, this variable is internally connected to the OPC UA Server.

Name Data type	Information/possible settings	
xExecute BOOL	FALSE↗TRUE	The Method is called by an OPC UA Client. Request to execute the method logic in IEC application by OPC UA Server.
xAbort BOOL	FALSE ↗TRUE	The Method is aborted by an OPC UA Client. Request to abort the method logic in IEC application by OPC UA Server.

Outputs

If the function block is mapped, this variable is internally connected to the OPC UA Server.

Name Data type	Value/meaning	
xDone BOOL	TRUE	Signals the OPC UA Server that the method logic in IEC application is done and the output variables are calculated.
xBusy BOOL	TRUE	Signals the OPC UA Server that the method logic in IEC application is not finished.
xAborted BOOL	TRUE	Signals the OPC UA Server that the method logic in IEC application is aborted.
eStatusCode L_IOBP_StatusCodes L_IE1P_Error	Signals the OPC UA Server the StatusCode for the method call. This StatusCode is send to the Client.Here you can find information on error or warning messages.	

Sub-Methods

This function block provides following methods for the method operation in the PLC application.

Name Data type	Description
StartMethod()	This method is called automatically once, if the OPC UA Method is requested (*xExecute* FALSE↗TRUE). It is used to prepare the method operation, e. g. to initialize additional function blocks or to prepare a consistent data set. This method is optional.
ExecuteMethod() Output variable: xComplete BOOL	This method is called automatically cyclically, after the **StartMethod()** call. It is used to calculate the method operation, e. g. calculate a simple addition, start an internal process, or set device parameter via an additional function block. The cyclic call of this method is terminated if *xComplete* is set to TRUE (normal method operation finished by the appliaction) or *xAbort* is true (abort request by the OPC UA Server).
FinalizeMethod() Output variable: xComplete BOOL	This method is called automatically cyclically, after the **ExecuteMethod()** calls. It is used to finalize the method operation, e. g. to finalize additional used function blocks. It can also be used for the abort handling, if requested by the OPC UA Server (*xAbort* == true). The cyclic call of this method is terminated if *xComplete* is set to TRUE. This method is optional.
ResetMethod()	This method is called automatically once, after the **FinalizeMethod()** calls. It is used to reset the method operation, e. g. to reset additional variables. This method is optional.

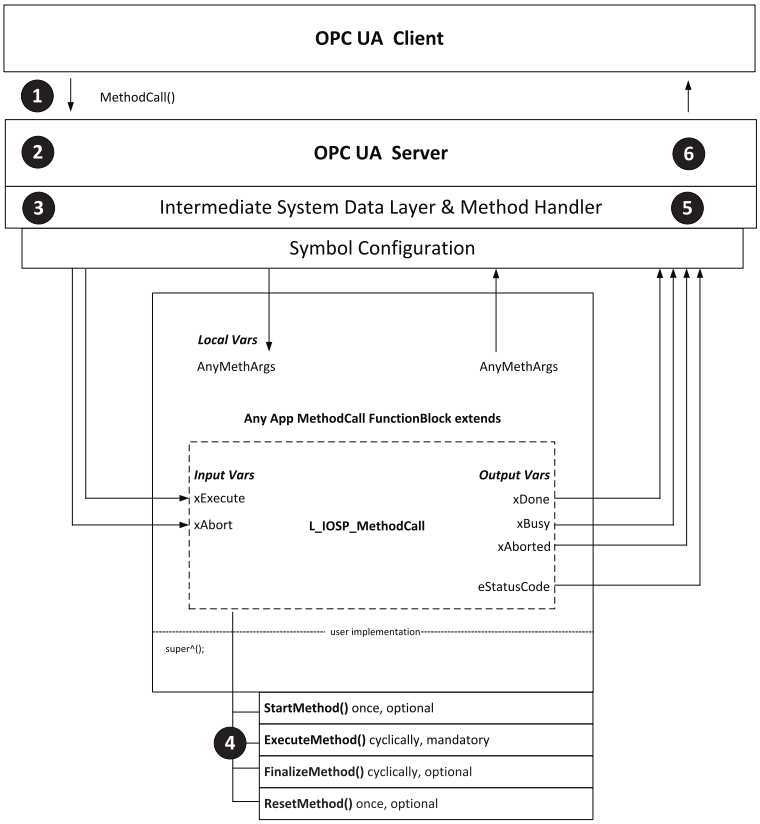
FB – OPC UA Method Handler interaction

The following figure describes the general usage of the function block inside the PLC application and the interaction with the Lenze OPC UA Method Handler. To connect the function block with the OPC UA Method Handler, it is necessary to publish the function block via the symbol configuration and to map it to a specific OPC UA Method.

It is expected to extend an application specific function block with the L_IOSP_MethodCall. The application specific function block can be enhanced with local variables for the arguments of the OPC UA method, which must be of the same name and data type.

Sequence:

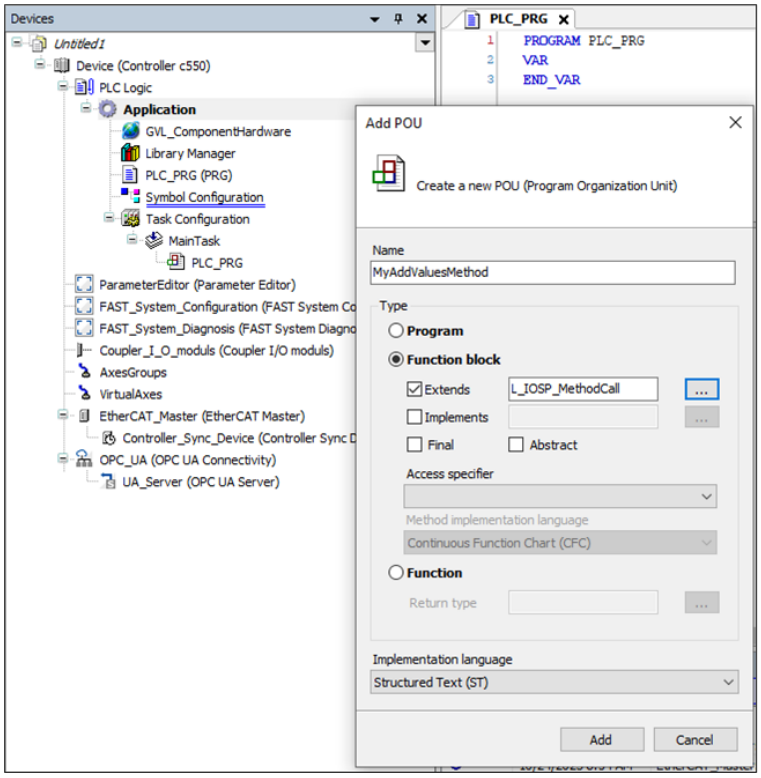
1. An OPC UA Client requests a specific Method call with any input arguments.
2. The OPC UA Server receives this call and performs a conversion and an alignment into a generic data format to distribute it further in the system.
3. A subsequent Method Handler stores and manages the method call, e. g. timeout handling. The Method Handler sets the function block variables, with which the method operation can be triggered (e.g. xExecute). The MethodCall InputArguments are assigned to the functions blocks local variables (Local Vars). If a positive edge occurs on xExecute, the arguments of the method call are provided by the Method Handler.
4. The method operation can be done in any suitable application specific IEC task. The operation may take several cycles. For the operation automatically called IEC methods are provided. Not needed methods can be deleted.
   - StartMethod()
   - ExecuteMethod()
   - FinalizeMethode()
   - ResetMethod()
5. After the Start-, Execute- and FinalizeMethod operation variables are provided for the OPC UA Method output arguments.
6. The OPC UA Server responds to the Method call of the client and the xExecute variable is set to false.


OPC UA Client
1
MethodCall()
2
OPC UA Server
6
3
Intermediate System Data Layer & Method Handler
5
Symbol Configuration
Local Vars
AnyMethArgs
AnyMethArgs
Any App MethodCall FunctionBlock extends
Input Vars
xExecute
xAbort
L_IOSP_MethodCall
Output Vars
xDone
xBusy
xAborted
eStatusCode
user implementation
super^();
4
StartMethod() once, optional
ExecuteMethod() cyclically, mandatory
FinalizeMethod() cyclically, optional
ResetMethod() once, optional

Method example

Add a new POU for the method

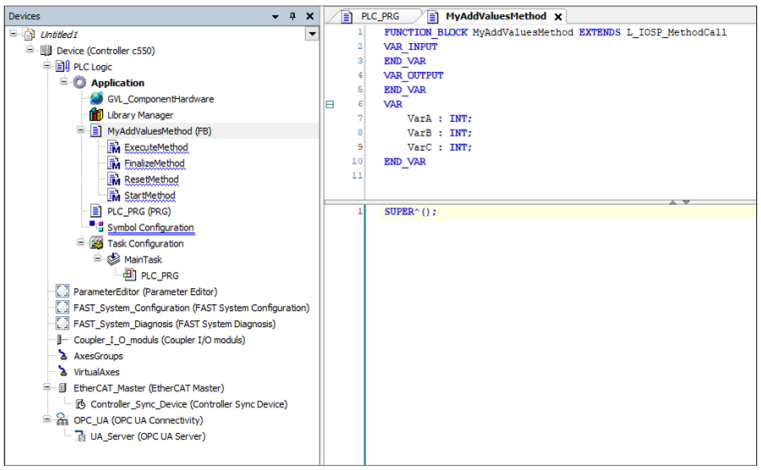
Add a new POU for the Method operation which extends the L_IOSP_MethodCall.



Define local variables for the Method arguments

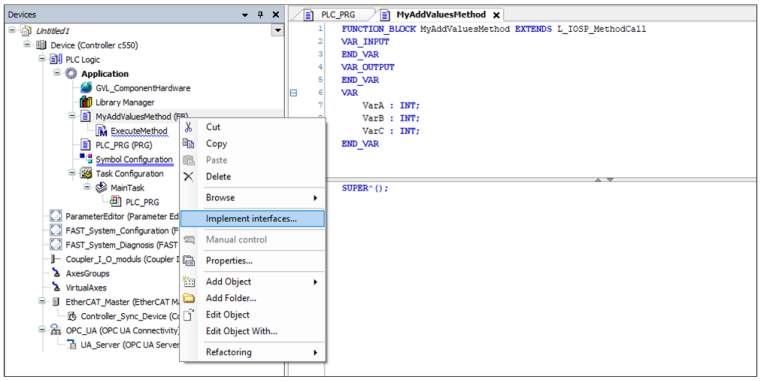
Define local variables for the Method arguments. The individual variable Name and Daty Type must fit to the argument.

Add *super^();* to the implementation to invoke the base function block during operation.



Delete optional operation methods

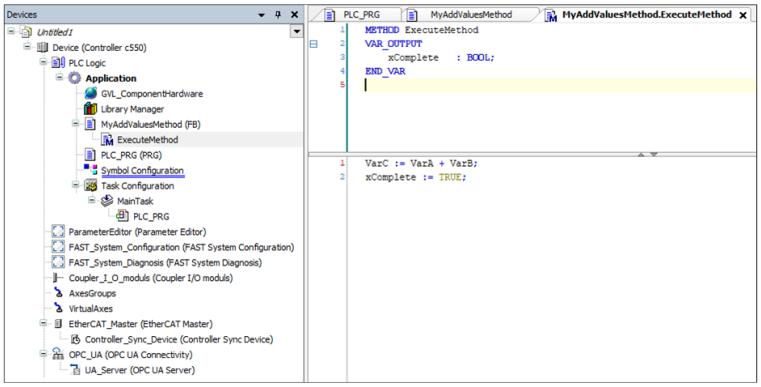
The Start-, Finalize- and ResetMethod() are optional. They can be deleted if they are not needed. They can be added again with the command "Implements interfaces...".



Implement the method operation

Implement the method operation in the ExecuteMethod() and other optional operation methods.

Finalize the cyclically called methods with *xComplete* := TRUE;



Instantiate and call method function block in a task.

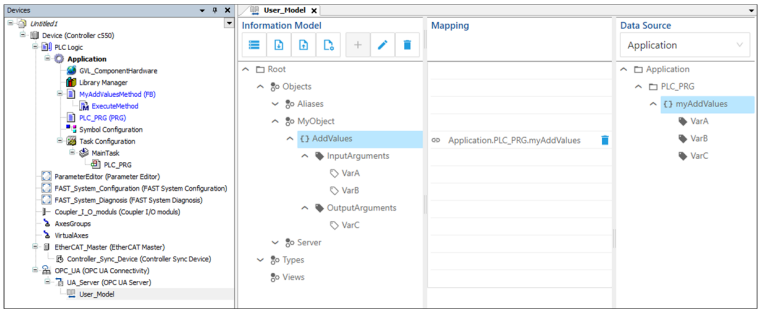
The created method function block must be instantiated and called in an IEC task suitable for the PLC application.

Publish method for OPC UA access via symbol configuration

The created instance of the method function block must be published for OPC UA access via the symbol configuration. After the selection of the instance a "generate code" of the PLC application is necessary.

Create and map an OPC UA Method with the Mapping Editor

Add an OPC UA user information model below the OPC UA Server device. Create an OPC UA Method with matching Arguments in the user information model with the Mapping Editor. Map the previously created method function block (published via the symbol configuration) to the OPC UA Method.



For further information how to use the Mapping Editor see the corresponding documentation. Mapping Editor (📖 14)

Go online and call the OPC UA Method

Download the PLC application (including the OPC UA user information model and the mapping to the PLC) on the controller device.

Start the PLC application.

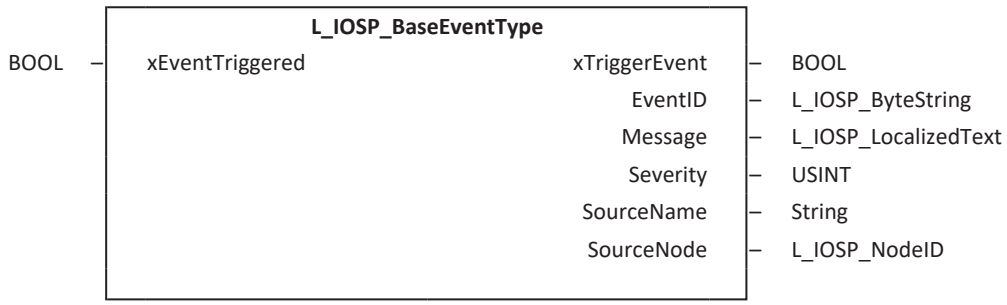
Afterwards the user information model is accessible by an OPC UA Client (e. g. UA Expert) and it is possible to call the OPC UA Method.

L_IOSP_BaseEventType

For the execution of Methods via OPC UA corresponding objects must be existing in the OPC UA information model as well as in the PLC application. In addition to this, both entities must then be mapped. This can be done with the Lenze Mapping Editor.

This generic function block L_IOSP_BaseEventType represents an OPC UA BaseEventType in the IEC application, which can be mapped to an BaseEventType in the OPC UA Nodeset.

This basic function block contains all the variables necessary for the mapping of the event to the OPC UA Server. In addition to this, it provides a few sub-methods for the event operation in the PLC application.



Inputs

If the function block is mapped, this variable is internally connected to the OPC UA Server.

Name Data type	Information/possible settings	
xEventTriggered BOOL	FALSE↗TRUE	The event has been fired by the Server. If the function block is mapped, this variable is internally connected to the OPC UA Server.

Outputs

If the function block is mapped, this variable is internally connected to the OPC UA Server.

Name Data type	Value/meaning
xTriggerEvent BOOL	Signals the OPC UA Server that the event shall be fired. If the function block is mapped, this variable is internally connected to the OPC UA Server.
EventID L_IOSP_ByteString	OPC UA specific value.[1] Shall be generated by the Server application to uniquely identify a particular Event Notification.
Message L_IOSP_LocalizedText	OPC UA specific value.[1] Shall provide a human-readable and localizable text description of the Event. If not set the BrowseName will be used.
Severity USINT L_IE1P_Error	OPC UA specific value.[1] It is an indication of the urgency of the Event. Values will range from 1 to 1 000, with 1 being the lowest severity and 1 000 being the highest. Here you can find information on error or warning messages.
SourceName String	OPC UA specific value.[1] SourceName provides a description of the source of the Event. If the Event is not specific to a source the SourceName can be empty.
SourceNode L_IOSP_NodeID	OPC UA specific value.[1] The SourceNode Property identifies the Node that the Event originated from. If the Event is not specific to a Node the SourceNode can be empty.

1) For more information, see:
OPC 10000-5 Unified Architecture Part 5 Information Model Base Event Type | OPC UA Online Reference (opcfoundation.org)

Sub-Methods

This function block provides following methods for the method operation in the PLC application.

Name	Description
StartEvent()	This method is called automatically once, if the OPC UA Event is executed by the PLC application (*xExecute* FALSE↗TRUE). It is used to prepare the event operation, e. g. to determine the EventId, to initialize additional function blocks or to prepare a consistent data input set. This method is optional.
ExecuteEvent() Output variable: xComplete BOOL	This method is called automatically cyclically, after the **StartEvent()** call. It is used to provide the event properties, e. g. getting device parameter via an additional function block for an application specific property. The cyclic call of this method is terminated if *xComplete* is set to TRUE by the application. The OPC UA Server is triggered to send the event.
ResetEvent()	This method is called automatically once, after the **ExecuteEvent ()** calls and the OPC UA Server signals, that the event is sent. It is used to reset the event operation, e. g. to reset additional output variables. This method is optional.

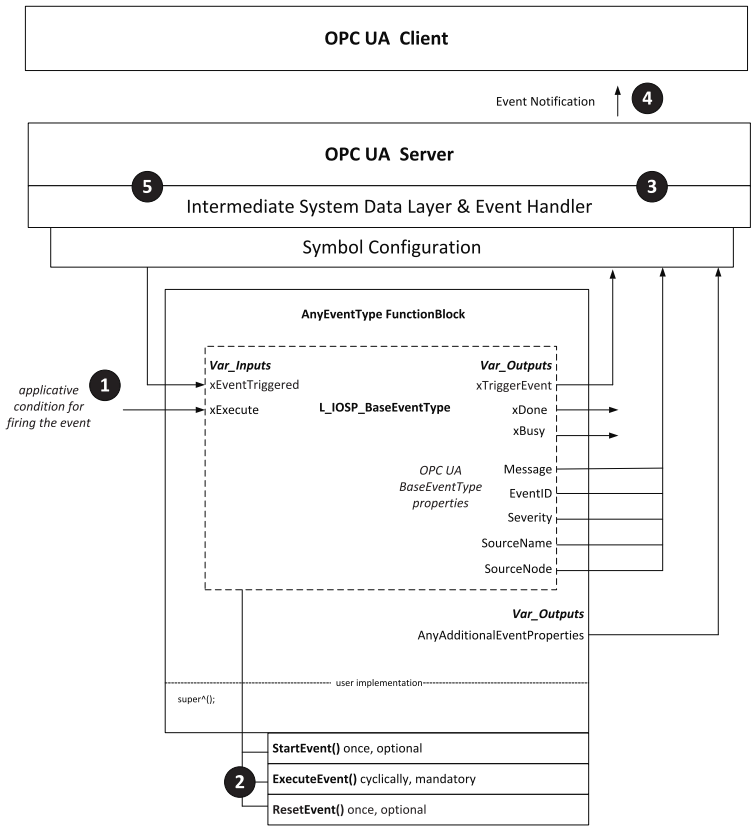
FB – OPC UA Server interaction

The following figure describes the general usage of the function block inside the PLC application and the interaction with the Lenze OPC UA Server. To connect the function block with the OPC UA Server it is necessary to publish the function block via the symbol configuration and to map it to an OPC UA EventNotifier in the user information model.

It is expected to extend an application specific function block with the L_IOSP_BaseEventType. The application specific function block can then be enhanced with output variables for additional Event properties.

Sequence:

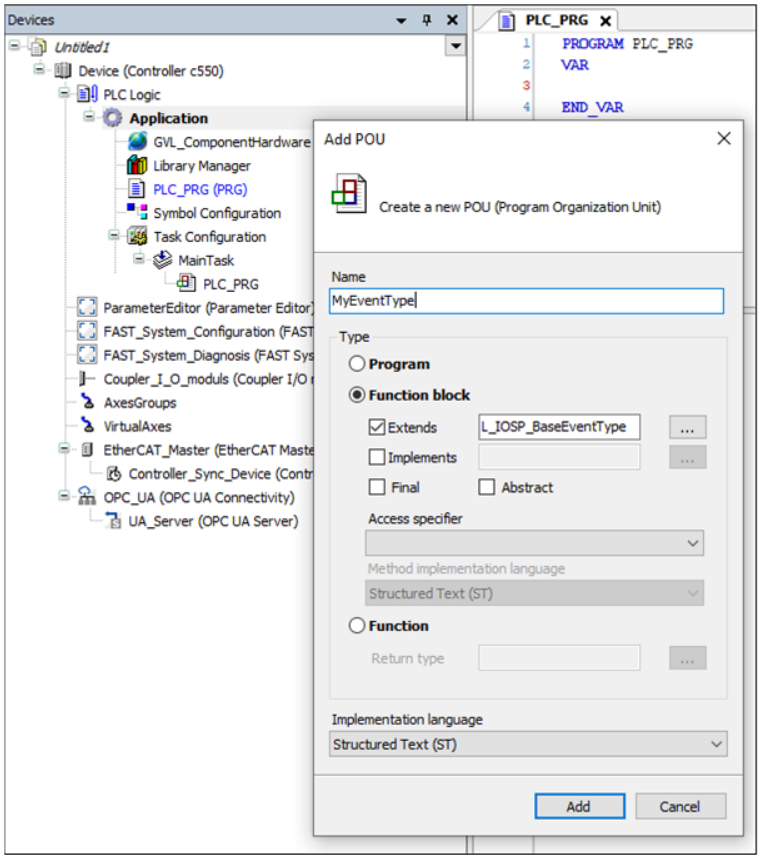
1. The event trigger decision is done by the PLC application.
2. The event operation (preparing the event properties) can be done in any suitable application specific IEC task. The operation may take several cycles. For the event operation automatically called methods are provided. Not needed methods can be deleted.
   - StartMethod()
   - ExecuteMethod()
   - ResetMethod()
3. After the Start- and ExecuteMethod() operation the output variables are provided to the OPC UA Server.
4. The OPC UA Server sends the Event to the subscribing Client.
5. To signal the application that the Event has been fired, the OPC UA Server sets the corresponding input variable xEventTriggered on true of the function block.



Event example

Add a new POU for the event

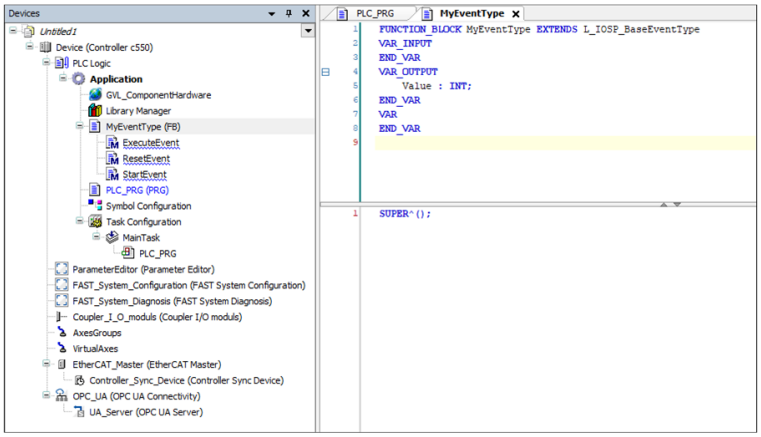
Add a new POU for the event operation which extends the L_IOSP_BaseEventType.



Define the additional event properties

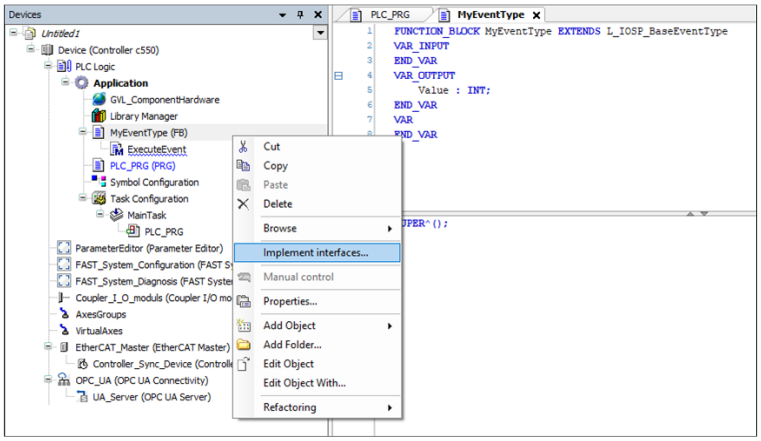
Define the additional needed event properties as output variables.

Add *super^();* to the implementation to invoke the base function block during operation.



Delete optional operation sub-methods

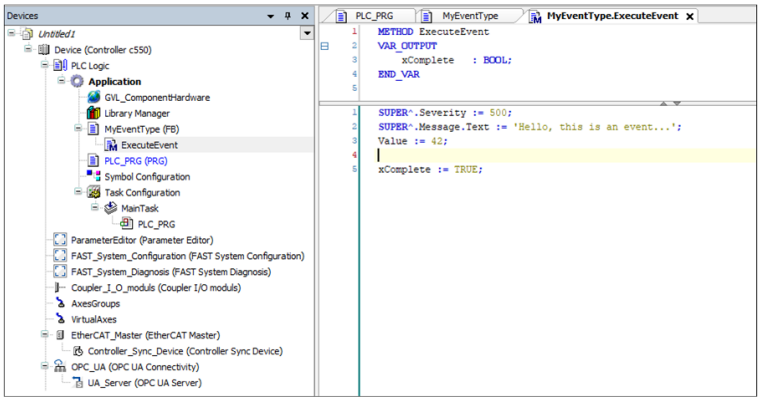
The Start- and ResetEvent() are optional. They can be deleted if they are not needed. They can be added again with the command "Implements interfaces...".



Implement the event operation

Implement the event operation in the ExecuteEvent() and other optional operation methods.

Finalize the cyclically called methods with *xComplete* := TRUE;



Instantiate and call event function block in a task.

The created event function block must be instantiated and called in an IEC task suitable for the PLC application.

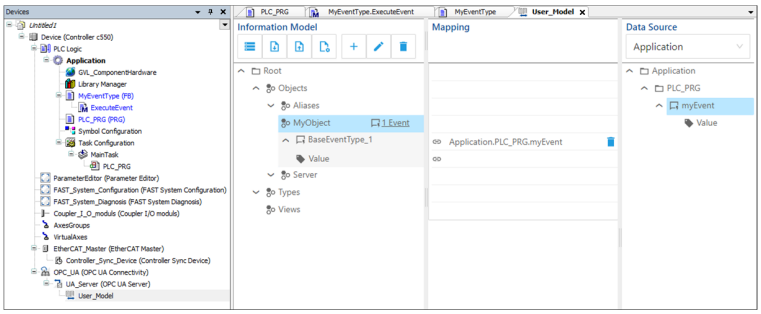
Publish event for OPC UA access via symbol configuration

The created instance of the event function block must be published for OPC UA access via the symbol configuration. After the selection of the instance a "generate code" of the PLC application is necessary.

Create and map an OPC UA EventNotifier with the Mapping Editor

Add an OPC UA user information model below the OPC UA Server device. Create an OPC UA EventNotifier with matching event properties in the user information model with the Mapping Editor. Map the previously created event function block (published via the symbol configuration) to the OPC UA EventNotifier.

For further information how to use the Mapping Editor see the corresponding documentation. Mapping Editor (📖 14)



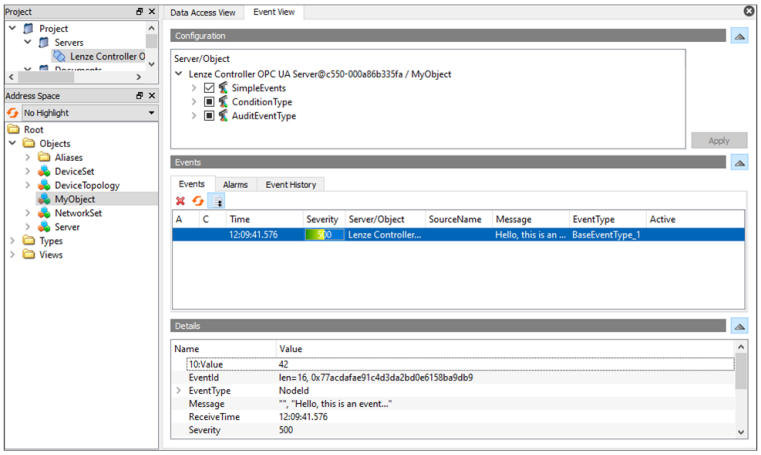
Go online and subscribe to the OPC UA Event

Download the PLC application (including the OPC UA user information model and the mapping to the PLC) on the controller device.

Start the PLC application.

Afterwards the user information model is accessible by an OPC UA Client (e. g. UA Expert) and it is possible to subscribe to the event.

If the PLC application fires the event with *xExecute* = TRUE, it will be sent in a notification to the subscribing client.



OPC UA Security

"Safety" in the context of automation is divided into two areas

- Functional safety→Operational safety (Keep crazy people from doing stupid things)
- Security→Information security (Keep smart people from doing clever things)

The project user management, the device user management and the encrypted data exchange by means of certificates belong to the security topic.

In the following, only the security mechanisms relevant for communication with the Lenze OPC UA Server on the controller are explained.

- Device user management
- Certificate handling

Device user management

Device user management is used to manage a wide range of access rights to the controller:

- Go online with the PLC Designer
- Access rights to individual control objects / functions
- OPC UA communication

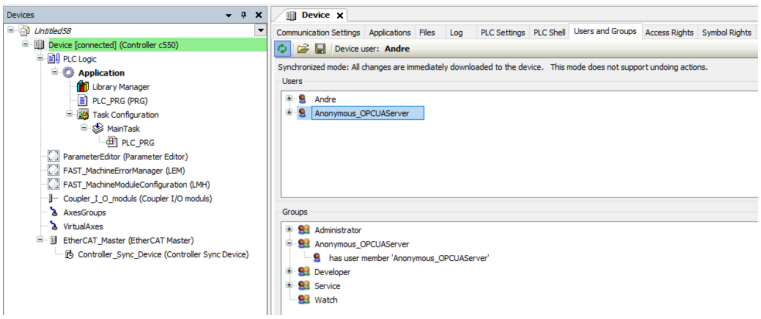
The activation and handling of the device user management with users and user groups is explained in general under the following link:

▶ CODESYS Online Help

Anonymous access to the OPC UA Server

If the device user management is activated, an "Anonymous_OPCUAServer" user is added to the user management by default. Thus, it is possible to connect to the OPC UA Server anonymously (without specifying user name and password). This "user" can be assigned to user groups like any other user.

If no anonymous access to the OPC UA Server shall be allowed, the "AnonymousOPCUAServer" user can be deleted from the user administration.



Device user management in the context of the PLCopen information model

Via the »PLC Designer« the entire variable household to be published can be sorted into symbol groups, which in turn are accessible to defined user groups. The configuration and an example are explained under the following link:

▶ Symbol groups

Deleting the device user management

When you try to remove the last device user in the »PLC Designer« user management, you receive a message that this user cannot be removed. To delete the device user management completely, you have to delete it on the SD card.

The device user management is stored on the SD card in two "hidden" files. These can be deleted manually to clean the controller again from a previously activated device user management.

- /sdcard/plc/prg/pta1/.UserMgmtRightsDB.lcsv
- /sdcard/plc/prg/pta1/.UserDatabase.csv
- /sdcard/plc/prg/pta1/.GroupDatabase.csv

Afterwards a restart of the controller (and the PLC Designer or the project) is necessary.

Device user management is used to prevent unauthorized access (e.g. sabotage). It can also be used to prevent an application from being accidentally transferred to a protected controller. However, this does not guarantee absolute protection, since the user administration can be bypassed, for example, via the path described.

Encryption

In addition to authentication, OPC UA communication can be encrypted. This is explained in the following chapter:

▶ Certificate handling (📖 112)

Certificate handling

In order to exchange data with an OPC UA client in an encrypted and secure manner, the Lenze OPC UA server requires a certificate that must be classified as "trustworthy" by the client when the connection is first established.

The certificate handling for encrypted communication described below can be combined with device user management (Message Security Mode = Sign & Encrypt).

For the configuration of the device user management see the following link:

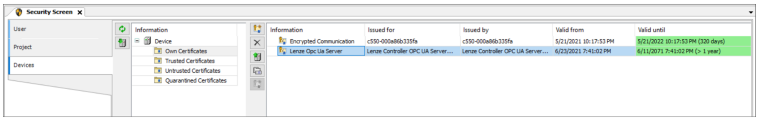
- Device user management (📖 110)

The OPC UA certificate handling on the controller side is done via the "Security Screen" in the PLC Designer. Via the integrated "Security Agent" general certificates of the controller can be managed. A general description of the "Security Agent" can be found under the following link:

- CODESYS Online Help

The following is an example of how to configure an encrypted OPC UA communication:

1. Select the "View➞Security Screen" command.
2. Select the control in the left view.
   - In the right view, all services of the controller that require a certificate are displayed.
3. Select "OPC UA Server".
4. Generate a new certificate for the device.
5. Define the parameters of the certificate and close the dialog with "Ok".

- The certificate is generated on the controller.
- 

6. Perform a restart of the PLC control system.

Creating an encrypted connection with the UaExpert client

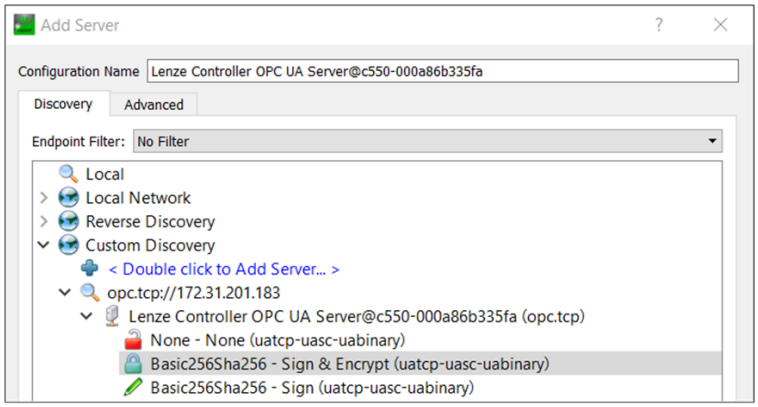
The generic OPC client "UaExpert" is with an evaluation license a freely available software of the company Unified Automation, which you can download from the Internet.

- https://www.unified-automation.com/products/development-tools/uaexpert.html

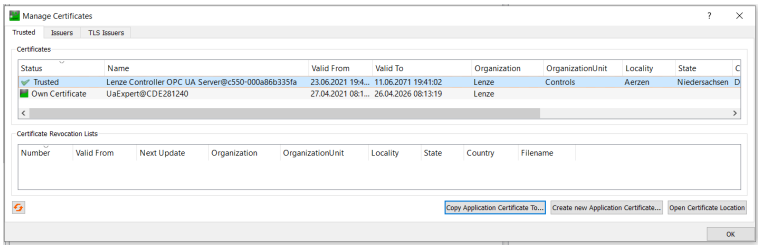
With this client you can connect to the Lenze OPC UA server. The following description refers to this program. Other OPC UA clients work similarly.

How to connect the client to the Lenze OPC UA server:

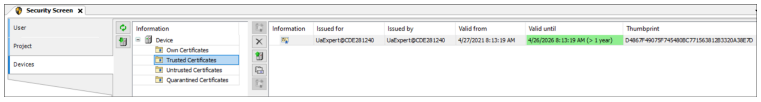
1. Start the UaExpert program.
2. Select the command "Server➞Add...". The "Add Server" dialog opens.
3. Add a new OPC-Server by double-clicking on it.
4. Enter the endpoint URL: opc.tcp://<IP address of the controller>:4840 (e.g. opc.tcp://172.31.201.183:4840).
5. Select the connection type "Basic256Sha256 - Sign & Encrypt (uatcp-uasc-uabinary)" and close the dialog with **OK**.



6. Select the "Server -> Connect" command. The server and client certificates are exchanged when the session is established. First, the "Certificate Validation" dialog opens with an error message.
7. Check the option "Accept the server certificate temporarily for this session" and click **Continue**.
8. After the connection has been established, make sure that the certificate of the Lenze OPC UA Server is located in the "Trusted" folder under the "Manage Certificates" menu.



9. In the PLC Designer Security Screen, refresh the view (click on ).
10. Select the "Quarantined Certificates" certificate folder. In the right view the client certificate "UaExpert@..." is displayed.
11. Drag the certificate with the mouse to the Trusted Certificates folder. The client certificate has now been assessed as "trusted" by the Lenze OPC UA Server.



12. After the certificate exchange between client and server application and the classification as "trusted", encrypted communication can take place.

Commissioning and device exchange with user management and certificates

When commissioning new and replacement devices, a number of setup steps must be completed for use. The measures regarding user administration and certificate handling are described below.

General protection of the device during commissioning

The following must be observed during initial commissioning and device replacement.

For secure communication of the device with other devices, the device to be used requires a certificate key pair that is used to secure the OPC UA communication. The procedure for generating the key pair is described in section Certificate handling.

To prevent misuse of the device by third parties, the SFTP access of the device should be individualized. In the factory settings, the device can be changed and viewed via an access key published by Lenze. Since this access can be used to influence secure communication via OPC UA, this is recommended for OPC UA users.

Under the following "Application Knowledgebase" (AKB) entry we describe how to establish a connection to a device via SFTP, as well as how to realize an initialization of the individualized access key. This entry also contains the published key for SFTP access ex works.

- AKB 201800294 - i950 / c5x0 / c750 : Software : Accessing the file system

Device replacement: Additional points to note

Since information is to be transferred from a replacement device to a replacement device during a device exchange, information relevant for OPC UA user management is stored on the SD card of the device. To transfer the information, the SD card of the old device must be transferred to the new device or the corresponding files must be copied from the SD card of the replacement device to the SD card of the replacement device.

User management relies on information from the following sources.

User database

The user database stores the information about the accesses of each user who has access to the device (Device user management).

The user administration files are located on the SD card in the path

- */plc/prg/pta1*

with the names

- *.UserMgmtRightsDB.lcsv*
- *. UserDatabase.csv*
- *. GroupDatabase.csv*

These must be copied to the same path of the new SD card for a transfer of user rights.

Certificates of the communication partners

To avoid having to reconfigure the associated communication partners when replacing devices, the public keys of the certificates of the communication partners already marked as trusted (see section 4.2) are provided on the SD card. The individual keys can be found in the following folder on the SD card:

- */plc/.pki/trusted*

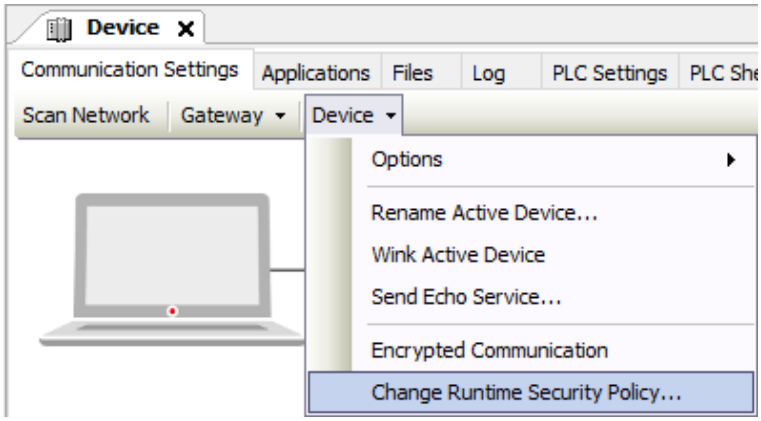
The individual files in this folder represent the respective public keys of the communication partners' certificates. To transfer the keys of the communication partners, the entire folder must be copied to the same path on the SD card.

With this procedure, it should be noted that the own certificate required for communication is not included (neither the public nor the private key). The private key is bound to the device and must not be distributed further. The key combination must be recreated as described in section Certificate handling.

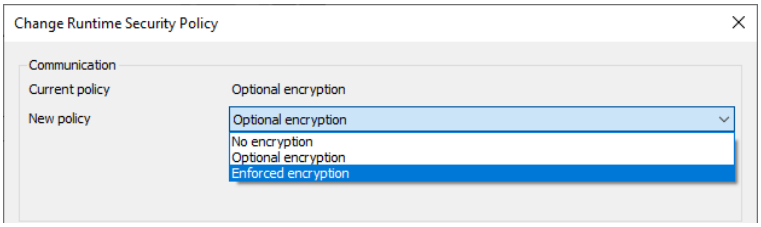
- This means that the Lenze OPC Server must also be classified as confidential again when a device is replaced on the clients.

Forced Security

Starting with the firmware releases of V01.09.00.00, certificate-based secure OPC UA communication security measures can be forced in the "Device" tab via "Device"→"Change Runtime Security Policy" of the »PLC Designer«.



Within the Communication group the OPC UA Connection policy can be forced by selecting "Enforced encryption" in the "New Policy" drop down menu.



By selecting the "Enforced encryption" the OPC UA Server does only provide communication endpoints with certificate-based secure communication enabled. With this particular security measurement enabled, limitations for other tools arise. The limitations and possible mitigations are described in the following sub-sections.

Limitations for Easy UI Designer

For the V01.09.00.00release the »EASY UI Designer« is not able to use certificate-based secure OPC UA communication.

Limitations for x4Remote (x500)

As of the V01.09.00.00 release, certificate-based secure communication with a OPC UA server is not possible. If the x500 communication is necessary, the certificate-based secure communication must be disabled until this feature is available via a x500 firmware upgrade.

OPC UA licenses

The Lenze OPC UA Server and Client on the controller is part of the controller firmware and does not have to be installed separately.

The following basic functionality can be used without additional licenses.

- Basic server and client function and OPC UA services.
- Information model according to PLCopen.
- User specific model and mapping to the PLC application (variables, methods and events).
- User management and certificates.
- Lenze internal OPC UA communication to FAST UI Runtime and x500.
  - See the license conditions at the corresponding Lenze product.
- Basic client function as PLCopen function blocks 1× external session to the Lenze server.
- 1× external session from the Lenze client.

The following licence package can be additionally activated via "Application Credit".
"OPC UA extended", provides…

- additional external sessions for server and client communication. The maximum number of additional sessions is limited according to chapter OPC UA Server Capabilities (📖 58) by the FW.
- the PubSub Feature.

- To activate the "OPC UA extended" license on c4xx Controller 100 "Application Credit" are needed.
- To activate the "OPC UA extended" license on c5xx Controller 150 "Application Credit" are needed.
- The activation of OPC UA licence packages can be done via the Lenze Licence Manager.

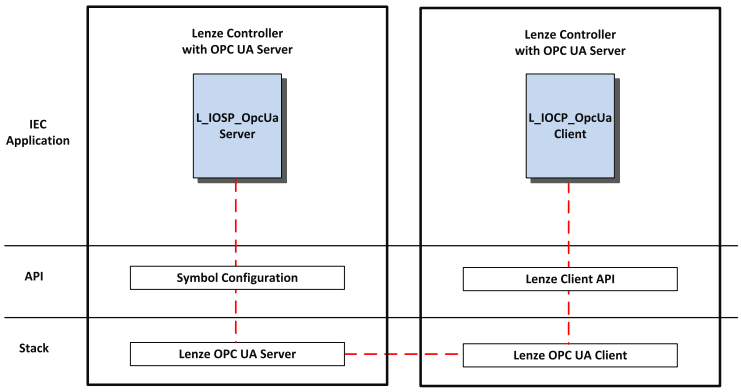
Controller: OPC UA Client

L_IOCP_OpcUaClient for OPC UA Client communication

This chapter describes all function blocks that are included in the "L_IOCP_OpcUaClient" function library. The function blocks can be used to create an OPC UA client application that performs interactions with an OPC UA server. These are ...

- PLCopen modules ("UA_" prefix),
- Lenze-specific modules ("L_IOCP _" prefix).

The following picture represents the main layers and used IEC libraries for an OPC UA Server and Client application hosted on a Lenze Controller.



PLC Task for the Client application

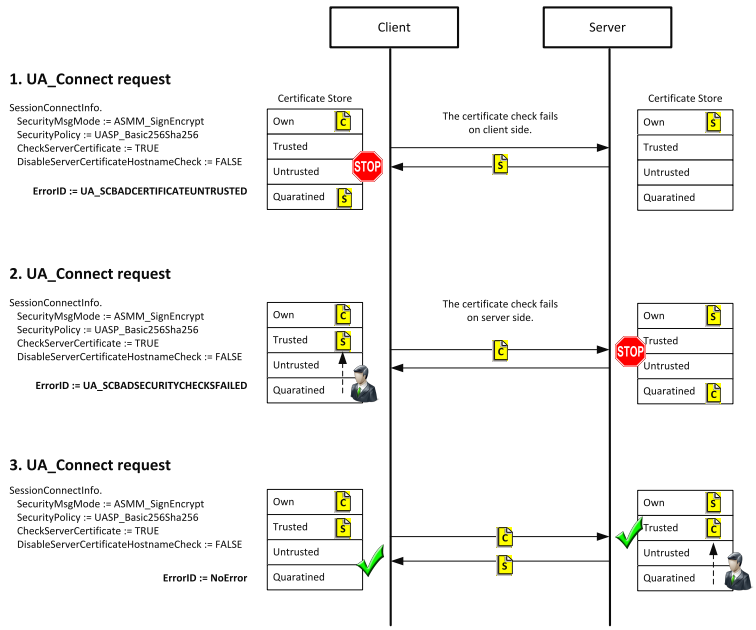
The OPC UA Client Server connection is a none realtime communication. Thus, the PLC application is not synchronized with the communication.

In order not to disturb the PLC realtime task, the Client PLC program should be executed in a low prio task without watchdog or free-wheeling task.

Certificate handling

The following figure describes the certificate handling between the OPC UA Server and Client application using the UA_Connect function block.

The used SessionConnectInfo input variables in this figure are examples for an establishment of a secure session.



The certificate check on client side includes a check of the hostnames in the used server certificate and the ServerEndpointUrl of the **UA_Connect** function block. Normally this should match, because otherwise an attacker could use any trusted certificate to fake a server's identity.
Nevertheless, the check can be disabled by the *DisableServerCertificateHostnameCheck* variable. It´s useful if the IT infrastructure is not able to resolve hostnames via DNS server and an IP address has to be used for the ServerEndpointUrl.

Controller Firewall

To address security concerns, the Lenze c5xx controller series includes a firewall to restrict communication with the surrounding network. Per default the firewall is disabled, yet can be enabled using the »PLC Designer«.

The firewall status and settings can be configured within the »PLC Designer« using the Tab Device and the options "Settings➞Communication➞Firewall".
To change the settings the »PLC Designer« must be connected to the PLC.

Within the Firewall setting, rules for ports (UDP & TCP) can be configured to either allow, reject or drop packages. As an OPC UA client creates an outbound connection the following must be configured.

Handling of outbound connections

The controller firewall focuses on incoming and not outgoing connections. However, if an outgoing connection is established, in most cases a response is made, which in turn must be taken into account in the firewall. As an example, the OPC-UA connection is considered here. If a connection is established from controller A with an OPC UA client to controller B with an OPC UA server, four steps in the respective firewalls of the controllers are relevant:

1. Controller A with OPC UA Client and an outgoing connection
   - The OPC UA client communicates via a randomly selected dynamic port (from the range 32,768 – 60,999). Outgoing ports are generally not blocked by the firewall.
2. Controller B with OPC UA Server and an incoming connection
   - In this example, the incoming connection takes place on the OPC UA server port 4840, which must be enabled in the firewall.
3. Controller B with OPC UA Server and an outgoing connection
   - The OPC UA server responds via port 4840 and creates an outgoing connection. Outgoing ports are generally not blocked by the firewall.
4. Controller A with OPC UA Client and an incoming connection
   - The OPC UA Client receives the response on the same randomly selected dynamic port. This must be opened in the firewall. Since the port is chosen randomly, the dynamic range in the controller firewall must be specified as the application port range.

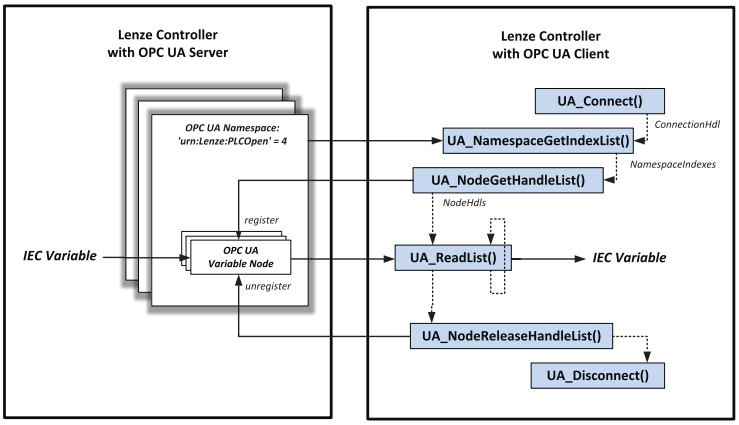
A controller firewall rule for the Controller a might look like this:

- Protocol name: e.g. ‘Open dynamic ports’
- Port Range Start: 32768
- Port Range End: 60999
- Protocol-Type : TCP
- Activation: Allow
- Client IP Range: Any

Read and Write of multiple items

The application of the data access use case can be divided into three phases:

1. Preperation
   - The function block UA_Connect is used to create a transport connection of an OPC UA session between the Client PLC application and any OPC UA Server. The UA_NamespaceGetIndexList is to be performed once to get the namespace index, where the nodes to be accessed are hosted. Also the NodeHdl for a specific node is to be retrieved once. It can be done with the UA_NodeGetHandleList functionblock for several nodes concurrent.
2. Data access operation
   - Read and write can be performed as frequent as necessary with the UA_ReadList and UA_WriteList function block.
3. Clean up
   - Once the communication is done, the node handle is not required anymore and shall be released via the use of UA_NodeReleaseHandleList for all relevant handles. The connection handle shall be released using UA_Disconnect.
     It´s possible to manage multiple sessions (max. 6) with the same function blocks. For this purpose, a handle for the connection, NodeID, and so on is used.

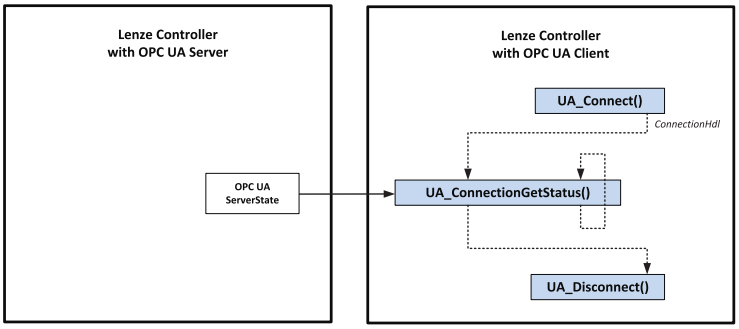


For the first steps, to programm an OPC UA Client application, it is possible to connect the Lenze OPC UA Server on the same Controller. For this purpose, the ServerEndpointUrl "opc.tcp://localhost:4840" can be used.

Diagnostics

The UA_ConnectionGetStatus function block is used to check if a connection is still alive. The function block UA_Connect will deliver the ConnectionHdl, which is also used for the "read and write" use case. The functionblock UA_ConnectionGetStatus requires this ConnectionHdl as input to deliver the connection status. In case the connection is lost the ServerState Unknown will be returned.

It is recommended to call UA_ConnectionGetStatus periodically but for performance reasons not in every control program cycle.



Overview Function blocks and Data Types

Function blocks

For general communication establishment and in order to perform operations like UA_ ReadList and UA_WriteList following function blocks are used.

Function block	Info
Function blocks for connection establishment and administrative tasks	
UA_Connect	This Function Block is used to create a (optional secure) transport connection and an OPC-UA session. The connection shall be terminated by calling the UA_Disconnect after establishing the connection.
UA_Disconnect	This Function Block is used to close a transport connection of an OPC-UA session.
UA_NamespaceGetIndexList	This Function Block is used to get the namespace-indexes of numerus namespace-URIs.
UA_ConnectionGetStatus	This Function Block is used to get the connection Status.
UA_NodeGetHandleList	This Function Block is used to get node handles for multiple nodes.
UA_ NodeReleaseHandleList	This Function Block is used to release a set of node handles.
Function blocks for Data Access	
UA_ReadList	This Function Block is used to read values of multiple nodes using a list of node handles.
UA_WriteList	This Function Block is used to write values to multiple nodes using a list of node handles.

Data Types

The following tables provides an overview with the used Data Types.

Data Type	Where used	Info
UAUserIdentityToken	UASessionConnectInfo	Specifies the identity token to authenticate a user
UASessionConnectInfo	UA_Connect	Specifies the necessary information for OPC UA connection establishment.
UANodeAdditionalInfo	UA_ReadList UA_WriteList	Specifies additional information for the node access.
UAIndexRange	UANodeAdditionalInfo	Describes the index range of an array.
UALocalizedText		L_IOBP_LocalizedText. Used to represent an OPC UA LocalizedText in IEC.
UANodeID	UANodeInformation UA_ NodeGetHandleList UA_MethodGetHandleList	Used to represent an OPC UA NodeId in IEC.
L_IOCP_Variable	UA_ReadList UA_WriteList	Used to reference an IEC variable for OPC UA data access.

Enumerations

The following tables provides an overview with the used Enumerations.

Enumeration	Where used	Info
UAUserIdentityTokenType	UAUserIdentityToken	
UASecurityMsgMode	UASessionConnectInfo	
UASecurityPolicy	UASessionConnectInfo	
UATransportProfile	UASessionConnectInfo	
UAConnectionStatus	UA_ConnectionGetStatus	
L_IOCP_ErrorID	UA_Connect UA_Disconnect UA_NamespaceGetIndexList UA_ConnectionGetStatus UA_NodeGetHandleList UA_NodeReleaseHandleList UA_ReadList UA_WriteList	
UAIdentifierType	UANodeID	Provides the OPC UA NodeID identifier types.
UAServerState	UA_ConnectionGetStatus	Provides the OPC UA Server states according to OPC UA Part 5 - Information Model
L_IOCP_NodeDataType	L_IOCP_Variable	Provides the data types of an OPC UA variable node.

Constants

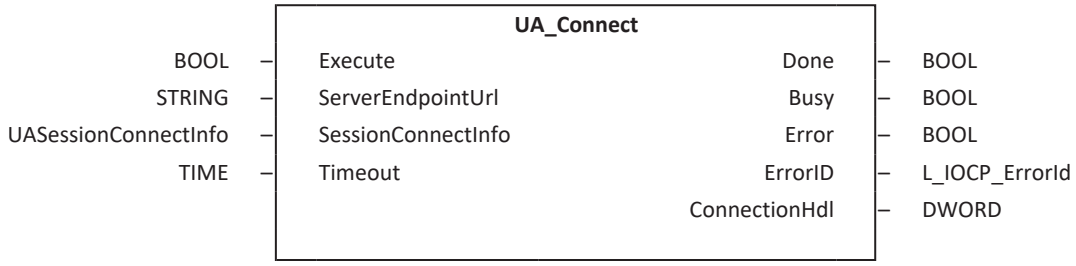
The following tables provides an overview with the used Constants.

Enumeration	Value	Info
c_Max_Elements_Namespaces	10	Max number of namespaces per GetNamespaceIndex call.
c_Max_Elements_NodeList	50	Max number of Nodes per Read, Write ,… call.
c_Max_Elements_IndexRange	10	Max number of Array-Elements per Node.

Function blocks

UA_Connect

This function block is used to create a transport connection and an OPC UA session. The connection handle offered is valid for the duration of the connection and is used to determine the session for the subsequent FBs. The connection shall be terminated by calling the UA_Disconnect.



Inputs

Name Data type	Information/possible settings	
Execute BOOL	FALSE↗TRUE	On rising edge connection is started. • All outputs of the FB are reset to the initial values.
ServerEndpointUrl STRING	The URL of the Server. For example: • "opc.tcp://10.10.10.10". • It is also possible to use a hostname instead of an IP address. In that case, an external Ethernet network entity like DNS (domain name service) has to be added for communication.	
SessionConnectInfo UASessionConnectInfo	Optional, further parameters for connection establishment. See UASessionConnectInfo	
Timeout TIME	Maximum time to establish the connection.	

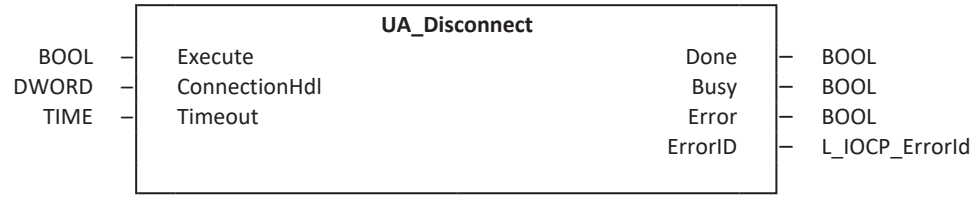
Outputs

Name Data type	Value/meaning	
Done BOOL	TRUE	Signals a connection has been initially established successful.
Busy BOOL	TRUE	The FB is not finished and new output values are to be expected.
Error BOOL	TRUE	An error has occurred in the service request. For details see *ErrorID*.
ErrorID L_IOCP_ErrorId L_IE1P_Error	If *Error* = TRUE: Error number for the error that has occurred. "FAST technology modules" reference manual: Here you can find information on error or warning messages.	
ConnectResultInfo L_IOCP_UASessionConnectResultInfo	*ConnectResultInfo* provides further information about the established session, e. g. used security mode.	

- Security / certificate handling is supported by default with Controller FW versions V1.11 and higher. It´s possible to adjust the security settings with the *SessionConnectInfo* input variable of the function block.
- It is an automatic re-connect implemented. In case of a session interruption, the client tries to reconnect to the server automatically. If the connection could be re-established, all handles for the session and nodes are still valid. Check the connection state via the UA_ConnectionGetStatus FB.
- It is possible to establish 1 session for free.
- It is possible to establish multiple sessions in parallel with appropriate OPC UA licence package. For this purpose, see the documentation of the Lenze Licence Manager in the »PLC Designer«.
  - c4xx Controller supports max. 4 sessions.
  - c5xx Controller supports max. 6 sessions.

UA_Disconnect

This function block is used to close a transport connection of an OPC UA session. After calling this FB the connection handle is no longer valid.



Name Data type	Information/possible settings	
Execute BOOL	FALSE↗TRUE	On rising edge connection is terminated.
ConnectionHdl DWORD	Connection handle – of the connection to be closed.	
Timeout TIME	Maximum time to close the connection.	

Outputs

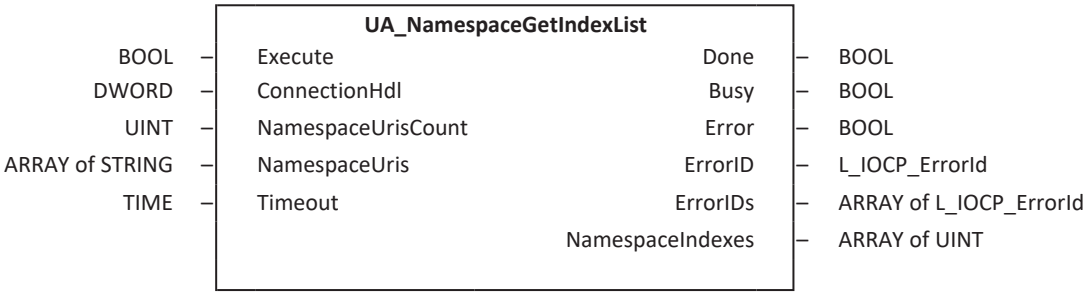
Name Data type	Value/meaning	
Done BOOL	TRUE	Signals a connection has been closed successful.
Busy BOOL	TRUE	The FB is not finished and new output values are to be expected.
Error BOOL	TRUE	An error has occurred in the service request. For details see *ErrorID*.
ErrorID L_IOCP_ErrorId L_IE1P_Error	If *Error* = TRUE: Error number for the error that has occurred. "FAST technology modules" reference manual: Here you can find information on error or warning messages.	

A_NamespaceGetIndexList

This function block is used to get the namespace-indexes of several namespace-URIs. In case the requested NamespaceUri is not found the error PLCopenUA_Bad_NSNotFound will be returned in the corresponding ErrorIDs.

Note: This FB is optional in a Client application. If the related namespace-index is already known, this FB don't has to be used. With a Lenze OPC UA server on a Lenze controller, the PLCopen information model with the URI "urn:Lenze:PLCopen" is always located on NamespaceIndex 4.

However, fixed NamespaceIndexes, beyond a restart of the server, are not guaranteed with other (user-specific) information models and other 3rd party OPC UA servers.



Inputs

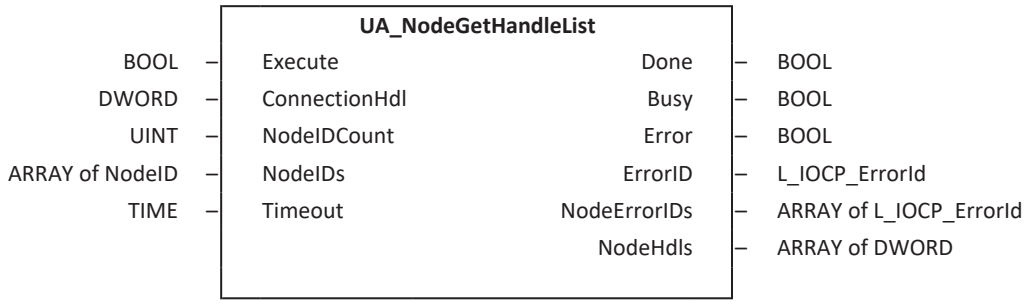
Name Data type	Information/possible settings	
Execute BOOL	FALSE↗TRUE	On rising edge FB performs its task.
ConnectionHdl DWORD	Connection handle.	
NamespaceUrisCount UINT	Number of Namespace Uris in Array of NamespaceUris.	
NamespaceUris ARRAY of STRING	Array of String. Each array element contains the namespace URI for which the index is to be requested.	
Timeout TIME	Maximum time to response.	

Outputs

Name Data type	Value/meaning	
Done BOOL	TRUE	Signals has completed its task.
Busy BOOL	TRUE	The FB is not finished and new output values are to be expected.
Error BOOL	TRUE	An error has occurred in the service request or in a specific namespace access. For details see *ErrorID* and / or *ErrorIDs*.
ErrorID L_IOCP_ErrorId L_IE1P_Error	If *Error* = TRUE: Error code for a failed service request. "FAST technology modules" reference manual: Here you can find information on error or warning messages.	
ErrorIDs ARRAY of L_IOCP_ErrorId	If *Error* = TRUE and *ErrorID* = LUA_EADNODERESULTERROR: Error number for individual Namespace elements.	
NamespaceIndexes ARRAY of UINT	Namespace Indexes.	

UA_NodeGetHandleList

This Function Block is used to get node handles for multiple OPC UA nodes. The NodeHdl is a reference to the internal management object for the node in the client. The nodes are registered at the server, which makes communication more performant.



Inputs

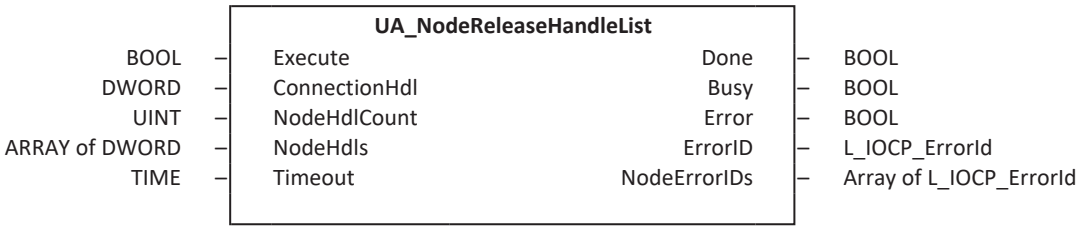
Name Data type	Information/possible settings	
Execute BOOL	FALSE↗TRUE	On rising edge FB performs its task.
ConnectionHdl DWORD	Connection handle.	
NodeIDCount UINT	Number of NodeIDs in Array of NodeIDs.	
NodeIDs ARRAY of NodeID	See UANodeID. Length of Array is MAX_ELEMENTS_NODELIST. Array length of NodeIDs and NodeHdls must be same.	
Timeout TIME	Maximum time to response.	

Outputs

Name Data type	Value/meaning	
Done BOOL	TRUE	Signals has completed its task.
Busy BOOL	TRUE	The FB is not finished and new output values are to be expected.
Error BOOL	TRUE	An error has occurred in the service request or in a specific node access. For details see *ErrorID* and / or *NodeErrorIDs*.
ErrorID L_IOCP_ErrorId L_IE1P_Error	If *Error* = TRUE: Error code for a failed service request. "FAST technology modules" reference manual: Here you can find information on error or warning messages.	
NodeErrorIDs ARRAY of L_IOCP_ErrorId	If *Error* = TRUE and *ErrorID* = LUA_EADNODERESULTERROR: Error code for each valid element of the NodeIDs array. Length of array is MAX_ELEMENTS_NODELIST. Array length of NodeIDs and NodeHdls must be same.	
NodeHdls ARRAY of DWORD	Array of Node Handles. Length of array is MAX_ELEMENTS_NODELIST. Array length of NodeIDs and NodeHdls must be same.	

UA_NodeReleaseHandleList

This Function Block is used to release a set of node handles. After calling UA_ NodeReleaseHandleList the NodeHdls will be invalid.



Inputs

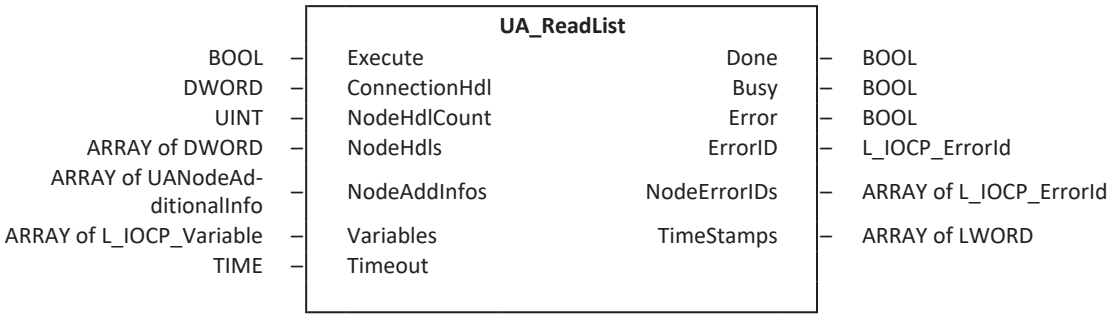
Name Data type	Information/possible settings	
Execute BOOL	FALSE↗TRUE	On rising edge FB performs its task.
ConnectionHdl DWORD	Connection handle.	
NodeHdlCount UINT	Number of Nodes in NodeHdls Array.	
NodeHdls ARRAY of DWORD	Array of Node handles to be released. Length of Array is MAX_ELEMENTS_NODELIST.	
Timeout TIME	Maximum time to response.	

Outputs

Name Data type	Value/meaning	
Done BOOL	TRUE	Signals has completed its task.
Busy BOOL	TRUE	The FB is not finished and new output values are to be expected.
Error BOOL	TRUE	An error has occurred in the service request or in a specific node access. For details see ErrorID and / or NodeErrorIDs.
ErrorID L_IOCP_ErrorId L_IE1P_Error	If *Error* = TRUE: Error code for a failed service request. Here you can find information on error or warning messages.	
NodeErrorIDs ARRAY of L_IOCP_ErrorId	If *Error* = TRUE and *ErrorID* = LUA_EADNODERESULTERROR: Error code for each valid element of the NodeIDs array. Length of array is MAX_ELEMENTS_ NODELIST. Array length of NodeIDs and NodeHdls must be same.	

UA_ReadList

This Function Block is used to read values of multiple nodes using a list of node handles.



Inputs/outputs

Name Data type	Information/possible settings
Variables ARRAY of L_IOCP_Variable	Array of structure to address an IEC variable by the OPC UA Client. A UA Server node attribute (determined by the node handle and UANodeAdditionalInfo) will be written to the IEC variable in case of this function block. See: **L_IOCP_Variable**

Inputs

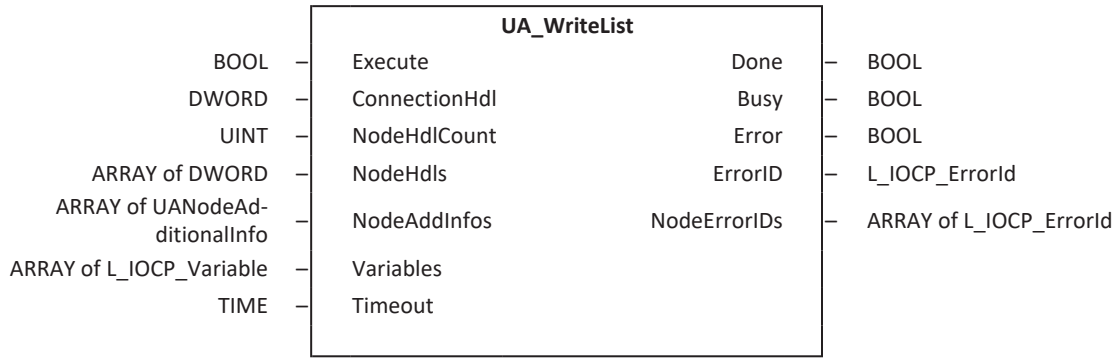
Name Data type	Information/possible settings	
Execute BOOL	FALSE↗TRUE	On rising edge FB performs its task.
ConnectionHdl DWORD	Connection handle.	
NodeHdlCount UINT	Number of Nodes in NodeHdls Array.	
NodeHdls ARRAY of DWORD	Array of Node handles. Length of Array is MAX_ELEMENTS_NODELIST.	
NodeAddInfos ARRAY of UANodeAdditionalInfo	Specifies additional information for the node access, e. g. which node attribute or what index range for arrays to be accessed. • Non scalar values: if no index range is set, the client will read the complete variable (e.g. the whole array).	
Timeout TIME	Maximum time to response.	

Outputs

Name Data type	Value/meaning	
Done BOOL	TRUE	Signals has completed its task.
Busy BOOL	TRUE	The FB is not finished and new output values are to be expected.
Error BOOL	TRUE	An error has occurred in the service request or in a specific node access. For details see *ErrorID* and / or *NodeErrorIDs*.
ErrorID L_IOCP_ErrorId L_IE1P_Error	If *Error* = TRUE: Error code for a failed service request. "FAST technology modules" reference manual: Here you can find information on error or warning messages.	
NodeErrorIDs ARRAY of L_IOCP_ErrorId	If *Error* = TRUE and *ErrorID* = LUA_EADNODERESULTERROR: Error code for each valid element of the NodeIDs array. Length of array is MAX_ELEMENTS_NODELIST. Array length of NodeIDs and NodeHdls must be same.	
Timestamps ARRAY of LWORD	Contains a TimeStamp for each valid element of the Variables array, if provided by the Server. Length of array is MAX_ELEMENTS_NODELIST.	

UA_WriteList

This Function Block is used to write values to multiple nodes using a list of node handles.



Inputs/outputs

Name Data type	Information/possible settings
Variables ARRAY of L_IOCP_Variable	Array of structure to address an IEC variable by the OPC UA Client. The value of this IEC Variable will be written to an OPC UA Server variable node (determined by the node handle and UANodeAdditionalInfo) in case of this function block. See: **L_IOCP_Variable**

Inputs

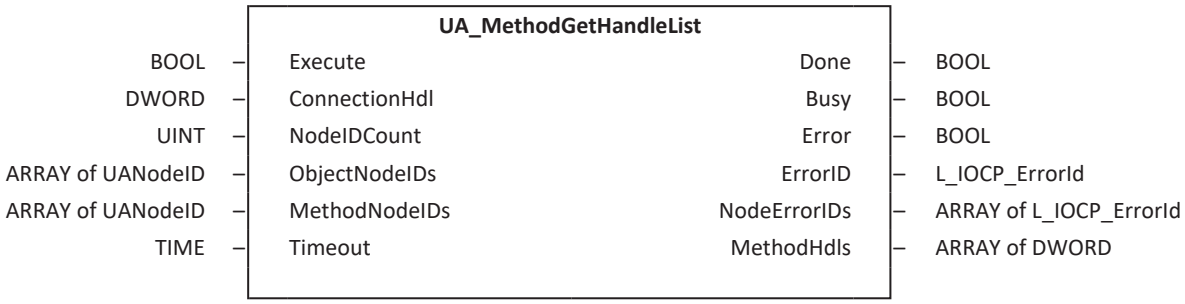
Name Data type	Information/possible settings	
Execute BOOL	FALSE↗TRUE	On rising edge FB performs its task.
ConnectionHdl DWORD	Connection handle.	
NodeHdlCount UINT	Number of Nodes in NodeHdls Array.	
NodeHdls ARRAY of DWORD	Array of Node handles. Length of Array is MAX_ELEMENTS_NODELIST.	
NodeAddInfos ARRAY of UANodeAdditionalInfo	Specifies additional information for the node access, e. g. which node attribute or what index range for arrays to be accessed	
Timeout TIME	Maximum time to response.	

Outputs

Name Data type	Value/meaning	
Done BOOL	TRUE	Signals has completed its task.
Busy BOOL	TRUE	The FB is not finished and new output values are to be expected.
Error BOOL	TRUE	An error has occurred in the service request or in a specific node access. For details see ErrorID and / or NodeErrorIDs.
ErrorID L_IOCP_ErrorId L_IE1P_Error	If *Error* = TRUE: Error code for a failed service request.	
NodeErrorIDs ARRAY of L_IOCP_ErrorId	If *Error* = TRUE and *ErrorID* = LUA_EADNODERESULTERROR: Error code for each valid element of the NodeIDs array. Length of array is MAX_ELEMENTS_NODELIST. Array length of NodeIDs and NodeHdls must be same.	

UA_MethodGetHandleList

This Function Block is used to get handles for multiple OPC UA methods. The handle is a reference to the internal management object for the Method call in the client.



Inputs

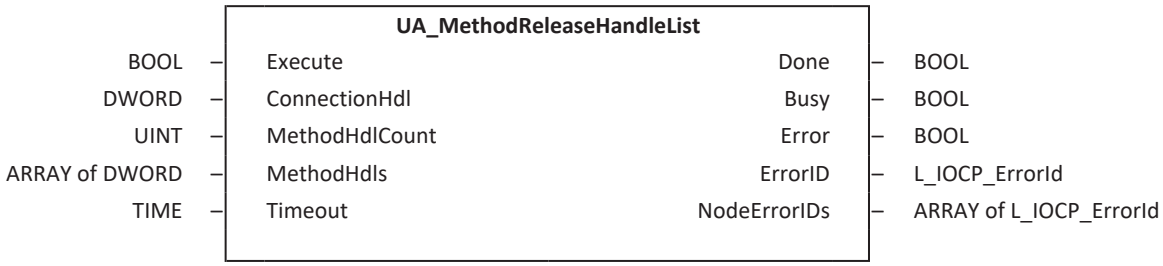
Name Data type	Information/possible settings	
Execute BOOL	FALSE↗TRUE	On rising edge FB performs its task.
ConnectionHdl DWORD	Connection handle.	
NodeIDCount UINT	Number of NodeIDs in Array of NodeIDs.	
ObjectNodeIDs ARRAY of NodeID	See *UANodeID*. Length of Array is MAX_ELEMENTS_NODELIST. Array length of NodeIDs and NodeHdls must be same.	
MethodNodeIDs ARRAY of NodeID	See *UANodeID*. Length of Array is MAX_ELEMENTS_NODELIST. Array length of NodeIDs and NodeHdls must be same.	
Timeout TIME	Maximum time to response.	

Outputs

Name Data type	Value/meaning	
Done BOOL	TRUE	Signals has completed its task.
Busy BOOL	TRUE	The FB is not finished and new output values are to be expected.
Error BOOL	TRUE	An error has occurred in the service request or in a specific node access. For details see *ErrorID* and / or *NodeErrorIDs*.
ErrorID L_IOCP_ErrorId L_IE1P_Error	If Error = TRUE: Error code for a failed service request. "FAST technology modules" reference manual: Here you can find information on error or warning messages.	
NodeErrorIDs ARRAY of L_IOCP_ErrorId	If Error = TRUE and *ErrorID* = LUA_EADNODERESULTERROR: Error code for each valid element of the NodeIDs array. Length of array is MAX_ELEMENTS_NODELIST. Array length of NodeIDs and NodeHdls must be same.	
MethodHdls ARRAY of DWORD	Array of Method Handles. Length of array is MAX_ELEMENTS_NODELIST. Array length of NodeIDs and NodeHdls must be same.	

UA_MethodReleaseHandleList

This Function Block is used to release a set of method handles. After calling UA_MethodReleaseHandleList the Method handles will be invalid.



Inputs

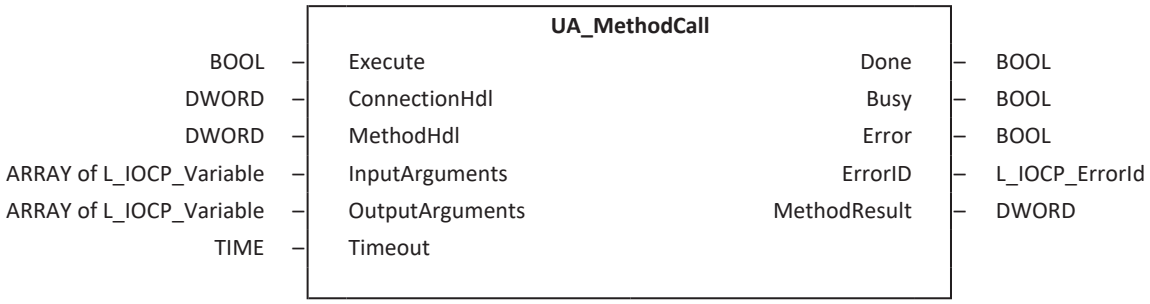
Name Data type	Information/possible settings	
Execute BOOL	FALSE↗TRUE	On rising edge FB performs its task.
ConnectionHdl DWORD	Connection handle.	
MethodHdlCount UINT	Number of Methods in MethodHdls Array.	
MethodHdls ARRAY of DWORD	Array of Method handles to be released. Length of Array is MAX_ELEMENTS_Methods.	
Timeout TIME	Maximum time to response.	

Outputs

Name Data type	Value/meaning	
Done BOOL	TRUE	Signals has completed its task.
Busy BOOL	TRUE	The FB is not finished and new output values are to be expected.
Error BOOL	TRUE	An error has occurred in the service request or in a specific node access. For details see *ErrorID* and / or *NodeErrorIDs*.
ErrorID L_IOCP_ErrorId L_IE1P_Error	If Error = TRUE: Error code for a failed service request. "FAST technology modules" reference manual: Here you can find information on error or warning messages.	
NodeErrorIDs ARRAY of L_IOCP_ErrorId	If Error = TRUE and *ErrorID* = LUA_EADNODERESULTERROR: Error code for each valid element of the NodeIDs array. Length of array is MAX_ELEMENTS_NODELIST. Array length of NodeIDs and NodeHdls must be same.	

UA_MethodCall

This Function Block is used to call one method routine.



Inputs/outputs

Name Data type	Information/possible settings
InputArguments ARRAY of L_IOCP_Variable	Array of structure to address the IEC variables related to the method input arguments by the OPC UA Client. The array of structure must match the InputArguments of the Server Method. See: **L_IOCP_Variable**
OutputArguments ARRAY of L_IOCP_Variable	Array of structure to address the IEC variables related to the method output arguments by the OPC UA Client. The array of structure must match the OutputArguments of the Server Method. See: **L_IOCP_Variable**

Inputs

Name Data type	Information/possible settings	
Execute BOOL	FALSE↗TRUE	On rising edge FB performs its task.
ConnectionHdl DWORD	Connection handle.	
MethodHdl DWORD	Method handle.	
Timeout TIME	Maximum time to response.	

Outputs

Name Data type	Value/meaning	
Done BOOL	TRUE	FB has completed its task.
Busy BOOL	TRUE	The FB is not finished and new output values are to be expected.
Error BOOL	TRUE	An error has occurred in the service request. For details see ErrorID.
ErrorID L_IOCP_ErrorId L_IE1P_Error	If Error = TRUE: Error number for the error that has occurred.	

Data Types

UAUserIdentityTokenType

UAUserIdentityTokenType describes the supported user token.

Designator	Value	Info
UAUITT_Anonymous	0	Performs anonymous connection establishment to an OPC UA Server. http://opcfoundation.org/UA-Profile/Security/UserToken/Anonymous
UAUITT_Username	1	Performs connection establishment by user name and password to an OPC UA Server. http://opcfoundation.org/UA-Profile/Security/UserToken/Client/UserNamePassword

UASessionConnectInfo

UASessionConnectInfo specifies the necessary information for OPC UA connection establishment.

Name Data type	Information/possible settings
SessionName STRING	Defines the name of the session assigned by the client. The name is shown in the diagnostics information of the server. If the String is empty, following default value is used "urn:<hostname>:Lenze:Controller:OPCUAClient"
ApplicationName STRING	Defines the readable name of the OPC UA client application. If the String is empty, following default value is used "Lenze Controller OPC UA Client@<hostname>"
SecurityMsgMode UASecurityMsgMode	Defines the OPC UA message security mode to be used.
SecurityPolicy UASecurityPolicy	Defines the OPC UA security policy to be used.
ServerUri STRING	Currently not supported This parameter can be empty.
CheckServerCertificate BOOL	Flag indicating if the server certificate should be checked with the trust list of the client application. **Note**: • The subject or alternative name in the server certificate must match to the used ServerEndpointUri of the UA_Connect function block. • The Lenze OPC UA Server supports the device hostname (not IP address) as subject name.
DisableServerCertificateHostnameCheck BOOL	Flag indicating if the hostname in the used server certificate should be checked with the hostname of the ServerEndpointUrl of the UA_Connect. Normally this should match, because otherwise an attacker could use any trusted certificate to fake a server's identity.
TransportProfile UATransportProfile	Defines the transport profile. TCP is currently supported.

Name Data type	Information/possible settings
UserIdentityToken UAUserIdentityToken	Defines the user identity token. Username and password is currently supported. The transmission of the username and password to the OPC UA Server is encrypted.
VendorSpecificParameter	Currently not supported.
SessionTimeout TIME	Defines how long the session will survive when there is no connection. The Client limits are min: 10 sec. and max: 15 min The default is, if parameter is not set 60 sec. **Note**: The Server can limit the requested value by its own.
MonitorConnection TIME	Currently not supported. Defines the interval time to check the connection. Note; An automatic reconnect is implemented.
LocaleIDs ARRAY [1..5] OF STRING[6]	Currently not supported. According to OPC 10000-3. With a two letter ISO639 code for language and a three letter ISO3166 code for the country/region. Sample: en-US, zh-CHS

UAIndexRange

UAIndexRange specifies the index range for OPC UA arrays to be accessed.

Name Data type	Information/possible settings
StartIndex UINT	Start index of array range.
EndIndex UINT	End index of array range.

Arrays are not supported with Controller c5xx FW V01.09.

UANodeAdditionalInfo

UANodeAdditionalInfo specifies additional information for the node access, e. g. which node attribute or what index range for arrays to be accessed.

Name Data type	Information/possible settings
AttributeID UAAttributeID	Selects the attribute to be accessed. The default AttributeID is UAAI_Value (13).
IndexRangeCount UINT	Count of valid IndexRange specified.
IndexRange ARRAY OF UAIndexRange	The max elements of UAIndexRange is defined by the constant MAX_ELEMENTS_INDEXRANGE.

UALocalizedText

UALocalizedText is used to represent an OPC UA LocalizedText in IEC.

- OPC 10000-3 Unified Architecture Part 3 Address Space Model Localized Text | OPC UA Online Reference (opcfoundation.org)

Name Data type	Information/possible settings
Locale STRING[6]	According to OPC 10000-3. With a two letter ISO639 code for language and a three letter ISO3166 code for the country/region. Sample: en-US, zh-CHS
Text STRING	Contains localized text as string.

UANodeID

UANodeID describes an OPC UA NodeId according to OPC UA Part 3 – Address Space Model.

- OPC 10000-3 Unified Architecture Part 3 Address Space Model General | OPC UA Online Reference (opcfoundation.org)

Name Data type	Information/possible settings
NamespaceIndex UINT	The index for a namespace URI.
Identifier STRING	The identifier for a Node in the Address Space of an OPC UA Server. Example: • for UAIT_Numeric = `12345’ in decimal notation • for UAIT_String = `|var|c500.App.PLC_Prg.iVar’
IdentifierType UAIdentifierType	The format and data type of the identifier.

L_IOCP_Variable

Structure to address an IEC variable by the OPC UA Client. The value of this IEC Variable will be written to an OPC UA Server variable node in case of the function block UA_WriteList. A UA Server node attribute will be written to the IEC variable in case of the function block UA_ReadList. The IEC and OPC UA variable type definition must fit together, see L_IOCP_VariableDataType.

Name Data type	**Information/possible settings**
NodeDataType L_IOCP_VariableDataType	Data type of the IEC variable to be accessed.
pData POINTER to BYTE	Address of the IEC variable
udiDataSize UDINT	Size of the IEC variable

Currently simple variables of *L_IOCP_VariableDataType* type definitions are supported.

L_IOCP UASessionConnectResultInfo

L_IOCP_UASessionConnectResultInfo provides information about the current established OPC UA connection with the **UA_Connect** function block.

Name Data type	**Information/possible settings**
SecurityMsgMode UASecurityMsgMode	The actual used security message mode.
SecurityPolicy UASecurityPolicy	The actual used security policy.
RevisedSessionTimeout LREAL	The actual used session timeout. The requested timeout can be revised by the server.
RevisedSecureChannelLifetime DWORD	The actual used secure channel timeout. The requested timeout can be revised by the server.

L_IOCP_MethodArguments

Structure to address the Input or Output Arguments of an OPC UA Server method by the OPC UA Client.

Name Data type	Information/possible settings
usiArgumentCount USINT	Number of Arguments
pArgument POINTER to L_IOCP_Variable	Address of an IEC variable with type of L_IOCP_Variable as a method argument. In case of multiple arguments, an IEC array of L_IOCP_Variable is required. The pArgument must address the first element (argument variable) in this array.

Code example

Client_MethodCall_PRG

```
(* Declarations for UA_MethodCall *)
fbUA_MethodCall         : UA_MethodCall;
stInputArguments        : L_SOCP_MethodArguments;
stOutputArguments       : L_SOCP_MethodArguments;
InputArguments          : ARRAY [1..2] OF L_IOCP_Variable;
OutputArguments         : ARRAY [1..1] OF L_IOCP_Variable;
MethodResult            : DWORD;

// InputArguments
VarA : INT := 3;
VarB : INT := 5;
// OutputArguments
VarC : INT;
```

```
// InputArguments
InputArguments[1].NodeDataType := L_IOCP_VariableDataType.VarType_INT;
InputArguments[1].pData := ADR(VarA);
InputArguments[1].udiDataSize := SIZEOF(VarA);

InputArguments[2].NodeDataType := L_IOCP_VariableDataType.VarType_INT;
InputArguments[2].pData := ADR(VarB);
InputArguments[2].udiDataSize := SIZEOF(VarB);

stInputArguments.usiArgumentCount := 2;
stInputArguments.pArgument := ADR(InputArguments[1]);

// OutputArguments
OutputArguments[1].NodeDataType := L_IOCP_VariableDataType.VarType_INT;
OutputArguments[1].pData := ADR(VarC);
OutputArguments[1].udiDataSize := SIZEOF(VarC);

stOutputArguments.usiArgumentCount := 1;
stOutputArguments.pArgument := ADR(OutputArguments[1]);

fbUA_MethodCall(
    Execute         := TRUE,
    ConnectionHdl   := nConnectionHdl,
    MethodHdl       := MethodHdls[1],
    InputArguments  := stInputArguments,
    OutputArguments := stOutputArguments,
    MethodResult    := MethodResult
    );
```

Enumerations

UAUserIdentityToken

UAUserIdentityToken specifies the identity token to authenticate a user.

The information is used for the connection establishment.

Name	Data type	Information/possible settings
UserIdentityTokenType	UAUserIdentityTokenType	Defines the identity Token to authenticate a user during the creation of a Session.
TokenParam1	STRING	In case of TokenType "Username" the Param1 contains the user name.
TokenParam2	STRING	In case of TokenType "Username" the Param2 contains the user password.

UASecurityMsgMode

UASecurityMsgMode describes the supported OPC UA message security modes.

Designator	Value	Info
UASMM_BestAvailable	0	Best available message security mode to the UA server. The client receives the available message security from the server and selects the best. This could also result in level "none security".
UASMM_None	1	No security is applied.
UASMM_Sign	2	All messages are signed but not encrypted.
UASMM_SignEncrypt	3	All messages are signed and encrypted.

UASecurityPolicy

UASecurityPolicy describes the supported OPC UA security policies.

Designator	Value	Info
UASP_BestAvailable	0	Provides the best available security connection to the UA server. The client receives the available policies from the server and selects the best. This can also result in level "none security".
UASP_None	1	http://opcfoundation.org/UA/SecurityPolicy#None
UASP_Basic256Sha256	4	http://opcfoundation.org/UA/SecurityPolicy#Basic256Sha256
UASP_Aes256Sha256RsaPss	6	http://opcfoundation.org/UA/SecurityPolicy#Aes256_Sha256_RsaPss

UATransportProfile

UATransportProfile describes the supported OPC UA transport profiles as a combination of network protocol, message encoding and security protocol.

Designator	Value	Info
UATP_UATcp	1	http://opcfoundation.org/UA-Profile/Transport/uatcp-uasc-uabinary

UAConnectionStatus

UAConnectionStatus describes the state of an OPC UA connection between Client and Server.

Designator	Value	Info
UACS_Connected	0	UA client is connected to UA server.
UACS_ConnectionError	1	The connection from UA client to UA server has an error.
UACS_Shutdown	2	The UA client has been disconnected from the UA server.

UAAttributeID

UAAttributeID provides the supported OPC UA node attributes that can be accessed.

Designator	Value	Info
UAAI_Value	13	The value of a variable.

UAServerState

UAServerState describes the OPC UA Server states according to OPC UA Part 5-Information Model ServerState type definition.

Designator	Value	Info
UASS_Running	0	The server is running normally. This is the usual state for a server.
UASS_Failed	1	A vendor-specific fatal error has occurred within the server. The server is no longer functioning.
UASS_NoConfiguration	2	The server is running but has no configuration information loaded and therefore does not transfer data.
UASS_Suspended	3	The server has been temporarily suspended by some vendor-specific method and is not receiving or sending data.
UASS_Shutdown	4	The server has shut down or is in the process of shutting down. Depending on the implementation, this might or might not be visible to clients.
UASS_Test	5	The server is in Test Mode. The outputs are disconnected from the real hardware, but the server will otherwise behave normally. Inputs may be real or may be simulated depending on the vendor implementation. StatusCode will generally be returned normally.
UASS_CommunicationFault	6	The server is running properly, but is having difficulty accessing data from its data sources. This may be due to communication problems or some other problems preventing the underlying device, control system, etc. from returning valid data. It may be a complete failure, meaning that no data is available, or a partial failure, meaning that some data is still available. It is expected that items affected by the fault will individually return with a BAD FAILURE status code indication for the items.
UASS_Unknown	7	This state is used only to indicate that the OPC UA server does not know the state of underlying servers.

UAIdentifierType

UAIdentifierType describes the format and data type of an OPC UA NodeId identifier. GUID and Opaque are currently not supported.

Designator	Value	Info
UAIT_Numeric	0	See OPC 10000-3 or OPC 10000-6
UAIT_String	1	See OPC 10000-3 or OPC 10000-6

L_IOCP_ErrorID

L_IOCP_ErrorID provides the ErrorIDs according to the PLCopen Client specification.

Following bit-mask is used to signal the severity and the error type.

Field	Bit Range	Description
Severity	30:31	Indicates whether the ErrorID represents a good, bad or uncertain condition. These bits have following meaning:
		00 = good Indicates that the operation was successful and the associated results may b used.
		01 = warning Indicates that the operation was partially successful and that associated results might not be suitable for some purposes.
		10 = failure Indicates that the operation failed and any associated results cannot be used.
		11 = reserved Reserved for future use.
ErrorType	28:29	00 = OPC UA error Indicates an OPC UA specific error. These codes are provided by the server and forwarded by the client to the FB. OPC 10000-4 Unified Architecture Part 4 Services General | OPC UA Online Reference (opcfoundation.org) A list of all codes is provided by the OPC Foundation is under following link: OPC 10000-6 Unified Architecture Part 6 Mappings Status Codes | OPC UA Online Reference (opcfoundation.org)
		10 = PLCopen error Indicates an PLCopen specific error. A list of all codes is provided by the PLCopen companion specification, see table below.
		11 = Lenze specific error Indicates an Lenze specific error. A list of all codes is provided by Lenze, see table below.

PLCopen specific error codes

Designator	Value	Info
General – PLCopen specific		
PLCopenUA_Bad_FW_PermanentError	16#A000_0001	Internal, permanent error.
PLCopenUA_Bad_FW_TempError	16#A000_0002	Temp. error; FB could retry to reach FW.
Connection – PLCopen specific		
PLCopenUA_Bad_ConnectionError	16#A000_0100	Connection could not be established.
PLCopenUA_Bad_HostNotFound	16#A000_0101	The requested hostname could not be found.
PLCopenUA_Bad_AlreadyConnected	16#A000_0102	Connection was already established.
PLCopenUA_Bad_SecurityFailed	16#A000_0103	Connection failed due to security setup.
PLCopenUA_Bad_Suspended	16#A000_0104	Connection is suspended.
PLCopenUA_Bad_ConnectionInvalidHdl	16#A000_0105	Provided ConnectionHdl is not known.
Namespace – PLCopen specific		
PLCopenUA_Bad_NSNotFound	16#A000_0200	A namespace with the requested name cannot be found on server.
Node – PLCopen specific		
PLCopenUA_Bad_ResultTooLong	16#A000_0300	Target PLC variable is too short for retrieved data.
PLCopenUA_Bad_InvalidType	16#A000_0301	Invalid or unsupported Type.
PLCopenUA_Bad_NodeInvalidHdl	16#A000_0302	Provided NodeHdl is not known.
PLCopenUA_Bad_MethodInvalidHdl	16#A000_0303	Provided MethodHdl is not known
PLCopenUA_Bad_ReadFailed	16#A000_0304	Read failed for unknown reason.
PLCopenUA_Bad_WriteFailed	16#A000_0305	Write failed for unknown reason.
PLCopenUA_Bad_CallFailed	16#A000_0306	Method Call failed for unknown reason.
PLCopenUA_Bad_InParamFailed	16#A000_0307	Method Call Input parameter conversion failed.
PLCopenUA_Bad_OutParamFailed	16#A000_0308	Method Call Output parameter conversion failed. ATTENTION: this means the MethodCall was executed successfully but the returned values could not be converted.
Attribute – PLCopen specific		
PLCopenUA_Bad_AttributeIdUnknown	16#A000_0400	Used in UA_NodeGetInformation for elements, which are not in this NodeClass existing.
PLCopenUA_Bad_AttributeIdInvalid	16#A000_0401	Used in UA_NodeGetInformation for elements, which should exist but don’t.
Monitoring – PLCopen specific		
PLCopenUA_Bad_SubscriptionInvalidHdl	16#A000_0500	Provided SubscriptionHdl is not known.
PLCopenUA_Bad_MonitoredItemInvalidHdl	16#A000_0501	Provided MonitoredItemHdl is not known.
PLCopenUA_Bad_MonitoredItemSyncMismatch	16#A000_0502	Mixed controller sync and firmware sync in same list
PLCopenUA_Bad_SyncModeInvalid	16#A000_0503	Sync mode invalid

Lenze specific error codes

Designator	Value	Info
UA_EBAD	16#B000_0001	General fault
UA_EBADDISCONNECT	16#B000_0002	The connection was closed by the peer.
UA_EBADSHUTDOWN	16#B000_0003	The connection was shut down by the peer
UA_EBADSOCKETCLOSED	16#B000_0004	The connection was closed locally
UA_EBADINVALIDARGUMENT	16#B000_0005	One or more of the supplied parameters were not valid
UA_EBADNOBUF	16#B000_0006	Allocating a buffer failed
UA_EBADDECODING	16#B000_0007	An decoding error occured
UA_EBADENCODING	16#B000_0008	An encoding error occured.
UA_EBADNOMEM	16#B000_0009	Allocating memory failed
UA_EBADNOTIMPL	16#B000_000A	The called function is not implemented fully or parially
UA_EBADHOSTUNKNOWN	16#B000_000B	The referenced host name could not be resolved
UA_EBADSYSCALL	16#B000_000C	An underlying system call failed
UA_EBADOPCANCELLED	16#B000_000D	The operation has been canceled
UA_EBADOPTIMEOUT	16#B000_000E	The operation timed out
UA_EBADIOPENDING	16#B000_000F	There is already an operation pending. Call again after completion
UA_EBADINVALIDSTATE	16#B000_0010	An object is not in the required state for the current operation
UA_EBADTRUNCATED	16#B000_0011	The result has been truncated
UA_EBADBUFTOOSMALL	16#B000_0012	The given destination buffer is too small
UA_EBADTOOMANYNESTEDCALLS	16#B000_0013	The operation would require too many iterations to succeed
UA_EBADNOTSUPPORTED	16#B000_0014	The requested operation is not supported
UA_EBADCERTIFICATINVALID	16#B000_0015	The certificate passed to the operation was not valid
UA_EBADNOTFOUND	16#B000_0016	A searched object could not be found
UA_EBADSIGNATURE	16#B000_0017	The operation failed because of a bad data signature
UA_EBADMISMATCH	16#B000_0018	The actual value didn't match the expected value
UA_EBADOUTOFRESOURCE	16#B000_0019	No more resources available
UA_EBADINUSE	16#B000_001A	Resource is still in use
UA_EBADAGAIN	16#B000_001B	Operation needs at last one more call to succeed
UA_EBADOUTOFRANGE	16#B000_001C	parameter is out of range
UA_EBADINTERNALERROR	16#B000_001D	Internal error
UA_ EBADENCODINGLIMITSEXCEEDED	16#B000_001E	Same as UA_SCBADENCODINGLIMITSEXCEEDED
UA_EBADENCODINGTOOLARGE	16#B000_001F	Request or response would be too large if encoded
UA_EBADACCESSDENIED	16#B000_0020	Access denied
UA_EBADLOOP	16#B000_0021	A loop exists in symbolic links encountered during resolution of the path argument
UA_EBADFILEEXISTS	16#B000_0022	The file or directory already exists

Designator	Value	Info
UA_EBADFILENOTFOUND	16#B000_0023	The specified file was not found
UA_EBADPATHNOTFOUND	16#B000_0024	The specified directory was not found
UA_EBADUNEXPECTED	16#B000_0025	For unexpected and unhandled errors to trigger general error handling
UA_EBADCOMMUNICATIONERROR	16#B000_0026	General error regarding the communication layer
UA_EBADSERVICERESPONSE	16#B000_0027	The server returned a bad status code in its response
UA_EBADLICENSELIMITSEXCEEDED	16#B000_0028	An operation failed because of limitations of the active license
UA_EBADREFEXISTS	16#B000_0029	The reference you tried to add already exists.
UA_EBADREFISCONST	16#B000_002A	The reference you tried to modify is const.
UA_EBADNOTHINGTODO	16#B000_002B	
UA_EBADSECURITYCHECKSFAILED	16#B000_002C	The security checks for an certificate are failing
LUA_EBADNODERESULTERROR	16#B000_002E	Invalid configuration parameters
LUA_EBADTYPEMISMATCH	16#B000_1388	The value supplied for the attribute is not of the same type as the attribute's value
LUA_EBADINDEXRANGEINVALID	16#B000_1389	The syntax of the index range parameter is invalid
LUA_EBADINITIALVALUE	16#B000_1770	The value is initialized with default values
LUA_EBADNODERESULTERROR	16#B000_1771	One or more of the nodes got an error. Please check the list of the ErrorIDs.
LUA_EINVALIDINPUTDATAFB	16#B000_1B58	Invalid Input data of **PLC_Open** FB
LUA_ EBADUSERTOKENNOTSUPPORTED	16#B000_1B59	The configured user token could not be found
LUA_FB_SERVERURL_INVALID	16#B0001F40	Invalid Server URL
LUA_FB_CONNECTIONHDL_INVALID	16#B0001F41	Invalid ConnectionHandle
LUA_FB_NAMESPACEURISCOUNT_ INVALID	16#B0001F42	Invalid NamespaceUrisCount
LUA_FB_NODEHANDLECOUNT_ INVALID	16#B0001F43	Invalid NodeHdlCount
LUA_FB_NODEIDCOUNT_INVALID	16#B0001F44	Invalid NodeIDCount
LUA_FB_METHODHDL_INVALID	16#B0001F45	Invalid MethodHdl
LUA_FB_INPUTARGUMENTS_ INVALID	16#B0001F46	Invalid Pointer Input Arguments
LUA_FB_OUTPUTARGUMENTS_ INVALID	16#B0001F47	Invalid Pointer Output Arguments
LUA_FB_VARIABLESINVALID	16#B0001F48	Invalid Variables
LUA_FB_ INDEXRANGECOUNTEXCEEDED	16#B0001F49	IndexRangeCount > 1

Controller: OPC UA PubSub

Overview

OPC UA introduction

OPC UA (Open Platform Communications Unified Architecture) is a globally recognized communication framework that is standardized by the IEC 62541 series of standards. It is currently the most promising standard for the implementation of Industry 4.0 communication in which machine data can be exchanged regardless of manufacturer and platform.

Of particular importance is the extensibility of OPC UA in the sense of a modeling framework, so that any industry-specific standards can be derived based on existing OPC UA standards. The architecture of OPC UA is service-oriented and thus modular and scalable.

The Lenze white paper on OPC UA also provides a further overview.

- Lenze Whitepaper

PubSub

PubSub enables decoupled communication between publishers and subscribers by communicating via middleware. Publishers publish their data in the form of DataSets to the middleware and subscribers fetch the DataSets they are interested in. As a result, a publisher does not need to know whether and how many subscribers are using the data it sends.

A typical use case for PubSub is publishing data that is of interest to a large number of recipients. Since no one to one connection setup is necessary between publisher and subscriber, the load on the publisher remains independent of the number of interested subscribers.

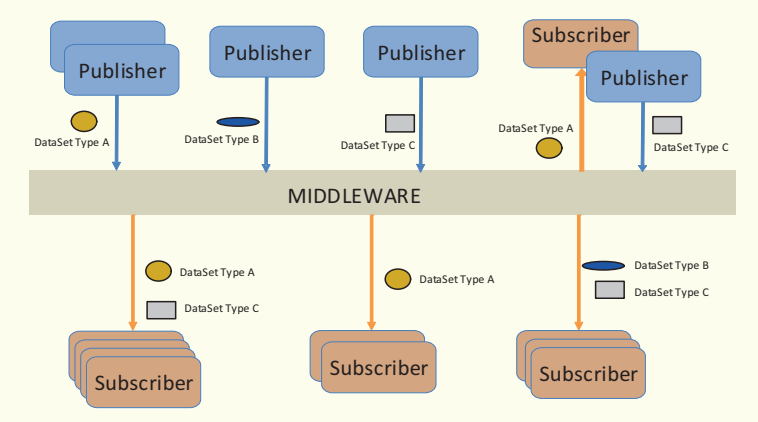


Figure from OPC Foundation – Part 14

PubSub configuration components

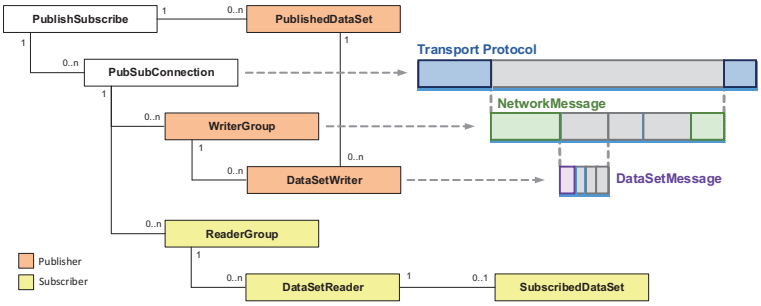


Figure from OPC Foundation – Part 14

The PublishSubscribe component can contain several PubSubConnections, on which any number of publishers and subscribers can be configured.

A Publisher includes the components WriterGroup, DataSetWriter and PublishedDataSet.

PublishedDataSet contain the metadata (DataSetMetaData) that define the content and data retrieval for a DataSet.

DataSetWriter are grouped into WriterGroups and the DataSetMessages of the DataSetWriters in a WriterGroup are written together into a NetworkMessage.

Groups are used to set common parameters, such as parameters for creating DataSet messages that are identical within a NetworkMessage.

A DataSetMessage in turn contains header information and the encoded data for a DataSet.

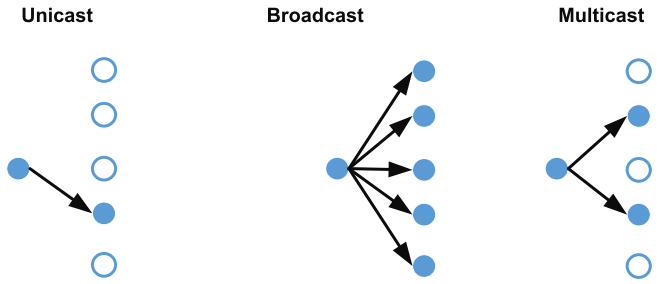
A Subscriber analogously comprises the components ReaderGroup, DataSetReader and SubscribedDataSet.

DataSetReader define filters to pick out from received messages those that are of importance to the subscriber.

OPC UA UDP and multicast communication

UADP is based on UDP-based transport layer mapping.

Furthermore, IP-based multicast communication is used to realize a targeted M:N communication relationship.



In contrast to broadcasting, where only one broadcast address is used to reach all subscribers of a local network, multicast communication has a number of multicast addresses.

In principle, a special IP range is provided for this, which is between 224.0.0.0 and 239.255.255.255.

The IP range 224.0.0.0 to 224.0.0.255 is reserved for routing and maintenance protocols.

For the OPC UA PubSub Discovery, the IP address 224.0.2.14 is specified.

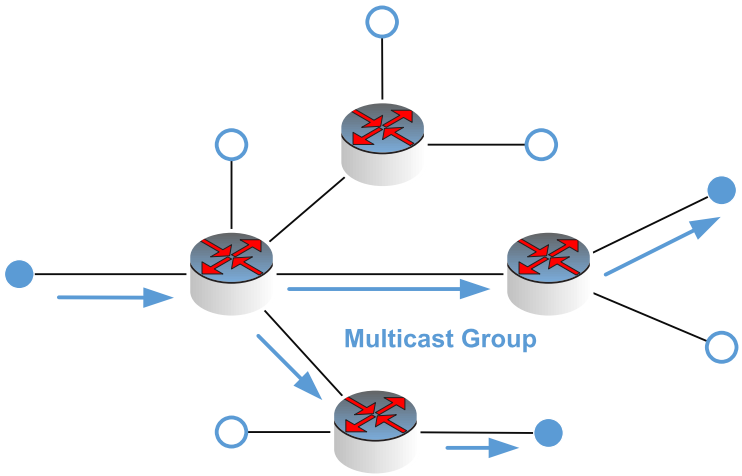
The IP range 239.0.0.0 - 239.255.255.255 is provided for the actual application-specific data communication. The default port for the UADP is 4840 (analogous to the TCP/IP based OPC UA client/server communication).

Broadcast addresses (refers to IPv4; with IPv6, broadcasting is also realized via multicast addresses) are not forwarded by routers, whereas multicast addresses can be forwarded, but this depends on the router and its routing configuration.

Broadcast as well as multicast is mainly operated via UDP protocol.

With broadcast communication, all hosts receive the broadcast messages by default; with multicast communication, on the other hand, a configuration is necessary so that multicast messages can be received.

All hosts configured for a specific multicast address are part of a multicast group for that address. The number of hosts within a group is not limited. A host can communicate on multiple multicast groups. A host can send multicast datagrams even if there are no receivers in the specific group.



In the context of multicast routing, multicast receivers must make themselves known so that the multicast routing protocol can identify the path.

In multicast routing, receivers must register with their local router by sending what is called an IGMP membership message. This message is sent to the router, which then uses it to create and maintain the multicast distribution structure.

The IGMP membership message contains the multicast address that the recipient wants to receive, as well as its own IP address. Based on this information, the router can then identify the optimal path to the sender and send multicast data to the receiver.

When a receiver unsubscribes or is no longer interested in the multicast group, it sends an IGMP leave message to signal its departure. The router can then update the multicast distribution structure to ensure that data is sent only to those receivers that are still interested in the group.

Specifically, for a switch/router to support multicast UDP efficiently, it must have several features and properties:

- multicast support: the switch must support multicast and be able to receive, process, and distribute multicast data.
- IGMP snooping: The switch must support IGMP snooping to process IGMP membership messages. IGMP snooping allows the switch to forward multicast data only to the ports that have receivers connected for a specific multicast group.
- Multicast routing protocols: The switch must be able to support multicast routing protocols such as PIM-SM, PIM-DM, or DVMRP to control and optimize multicast distribution in the network.
- VLAN support: The switch must provide VLAN support to isolate and control traffic from different multicast groups.
- Bandwidth management: the switch must be able to effectively manage the bandwidth for multicast data to ensure optimal network performance.
- Multicast filtering: The switch must be able to filter multicast data based on IP addresses and port numbers to optimize and control network traffic.

Overall, the switch must be able to effectively process and distribute multicast data to ensure that multicast transmission on the network is effective and smooth.
Without IGMP support, a switch/router would distribute all multicast traffic to all ports, which can result in a high traffic load. Depending on the context, this can be problematic for heavy traffic and large networks, or it can be unproblematic in a small local network (same subnet).
In general, it can be stated that in large networks any unwanted traffic should be avoided, as it would otherwise unnecessarily burden the participants.
In addition, without IGMP the receiver is responsible for filtering the received data independently.

Lenze Controller OPC UA PubSub

Features and capabilities

The Lenze OPC UA PubSub on the controller supports the following service capabilities. The individually listed capabilities have the limit named here. Depending on the combination and utilization of these capabilities, as well as the utilization of the CPU due to the PLC application and other processes on the controller, the controller may reach its performance limit.

This must be considered during project engineering.

PubSub Feature / Capabilitiy[1)]	**c430**	**c520, c550**
Data Access	Simple / primitive Server Variable Nodes from PLCopen and User Information model	
Quality	None Realtime	
protocols	UADP	
Routing schemes	Unicast and Multicast	
Security	Not supported	
Sum of Maximum number of connections	2	4
Sum of Maximum number of WriterGroups	4	8
Sum of Maximum number of ReaderGroups	4	8
Sum of Maximum number of Writers	8	16
Sum of Maximum number of Readers	8	16
Sum of Maximum number of datasets[2)]	8	16
Minimum publishing interval	100 ms	50 ms
Maximum NetworkMessage size[3)]	1400 byte	1400 byte

1) For the OPC UA PubSub functionality a Lenze Licence is needed.The activation of OPC UA licence packages can be done via the Lenze Licence Manager.

2) There is no limitation of the number of nodes in a DataSet. Just the NetworkMessage size limits the payload.

3) Maximum NetworkMessage size for all variable values of a WriterGroup or ReaderGroup

PubSub function activation

The OPC UA PubSub function must first be switched on via parameter 0x247B:001 "PubSub activation" = 1:Enabled. For a permanent activation after power on the parameter set on the device has to be stored.

For the PubSub function to become active, the OPC UA Server must be restarted via parameter 0x2470:001, or a reboot of the Controller is needed. After this the PubSub function gets access to the OPC UA Server on the device and can be configured via the Server information model.

For the PubSub configuration a configuration file is needed on the SD card of the controller. The file format is standardized by the OPC Foundation.

The configuration (and the generation of the configuration file) can be done for instance with the UAExpert V1.7 or higher.

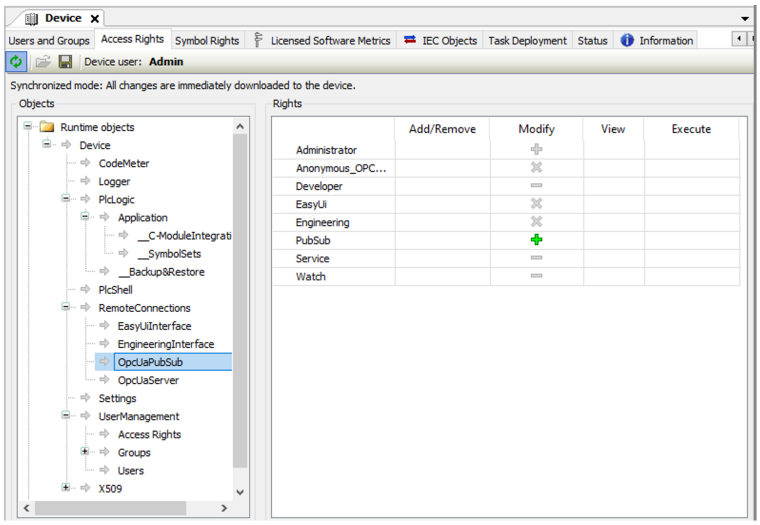
For the OPC UA PubSub functionality a Lenze Licence is needed. The activation of OPC UA licence packages can be done via the Lenze Licence Manager.

In some cases, a restart of the PubSub function for proper communication is required via Parameter 0x2470:001. This is necessary to ensure conditions for correct operation.

- After an Ethernet network initialization at the later runtime of the controller (e.g. if the Ethernet-Link is not available after reboot).

Configuration Authorization

If the User Management of the device is activated, a permission to configure OPC UA PubSub is needed. This permission can be granted by the “Access Rights” and “Users and Groups” of the device. A predefined PubSub group is authorized to configure the functionality by default. Users can be added to this group.



Data Access Authorization

The OPC UA PubSub functionality requires data access to the variables, which should be communicated.

In case of data access to the PLC, the IEC variables must be enabled for OPC UA communication via the symbol configuration.

If the User Management of the device is activated, a permission for this data access is needed. The OPC UA PubSub functionality itself connects to this data source as an "anonymous user".

Thus, only IEC variables can be accessed which are granted to an "anonymous user".

In case of a Publisher read access rights for the "anonymous user" are required for the IEC variable.

In case of a Subscriber write access rights for the "anonymous user" are required for the target IEC variable.

Configuration Example with UA Expert

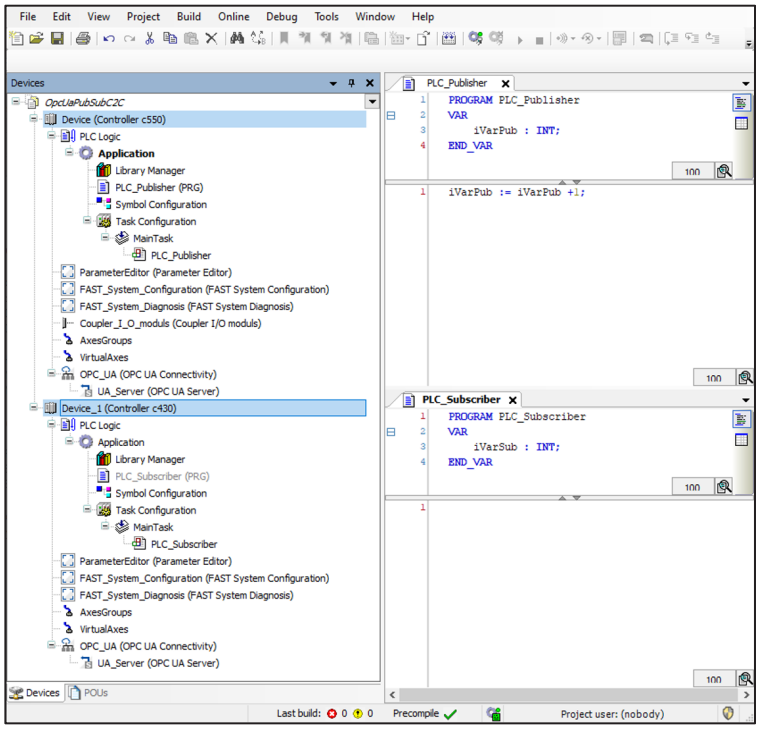
The configuration can be done with the UAExpert V1.7 or higher.

The Version V1.7 of the UA Expert supports the data set configuration of simple data types (Bool, SByte, Byte, Int16, UInt16, Int32, UInt32, Int64, UInt64, Float, Double, String, DateTime, ByteString). All other data types (enums, complex data types and structures) are currently not supported.

The use case in this example is to setup one Lenze Controller as an OPC UA Publisher and another Lenze Controller as an OPC UA Subscriber. Both Controller are connected to an Ethernet switch.

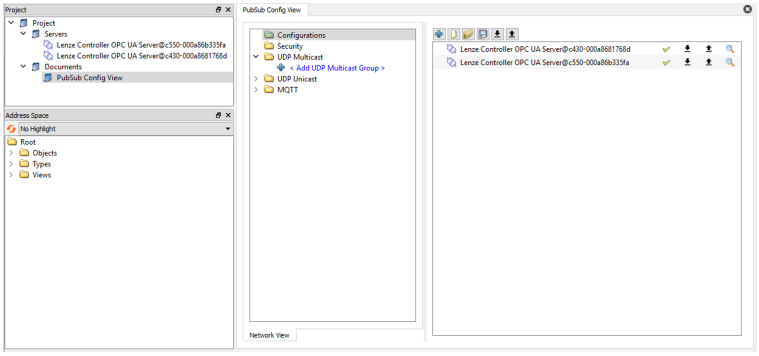
For this use case the PLC applications for the Publisher and Subscriber are presented in the following figure. The Publisher and Subscriber IEC variables are enabled for OPC UA communication via the symbol configuration.

The PubSub functionality is activated and a OPC UA Lenze Licence is configured (PubSub function activation).

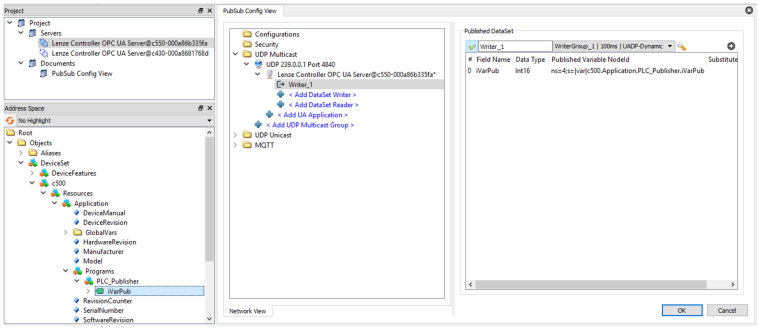


The following configuration steps are necessary for the PubSub example in UAExpert:

1. Connect to both OPC UA Servers (Publisher device and Subscriber device) with an authorized user token.
2. Add the "PubSub Config View" via the "Document" menu.

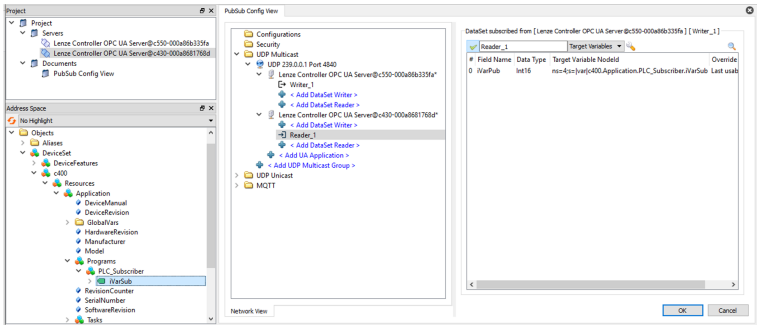


3. Add an "UDP Multicast Group".
   This is the PubSub Connection e.g. "UDP 239.0.0.1 Port 4840".
4. Add an "UA Application" to the "UDP Multicast Group" and select the Lenze OPC UA Server which should publish data (Device c550).
5. Add a "DataSet Writer".
6. The variable to be published can be added to the "Published DataSet" from the Server information model by drag and drop.

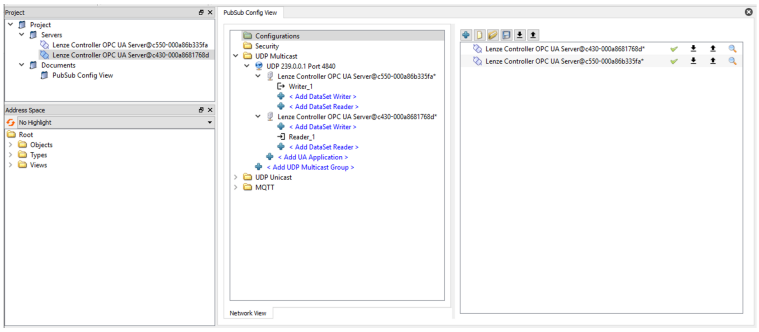


7. Apply the configuration.
8. Add an additional "UA Application" and select the Lenze OPC UA Server which should subscribe data (Device_1 c430).
9. Add a "DataSet Reader" and select the template from the applied DataSet Writer.

10. The target variable can be added to the "Subscribed DataSet" from the Server information model by drag and drop.



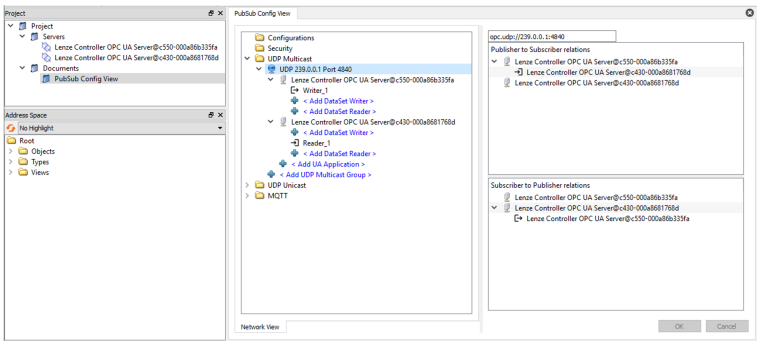
11. Apply the configuration.
12. Download the configurations to both OPC UA Server on the devices.



13. The publisher starts publishing the variable directly and the subscriber receives it. The configuration is stored persistent on the SD Card of the devices.

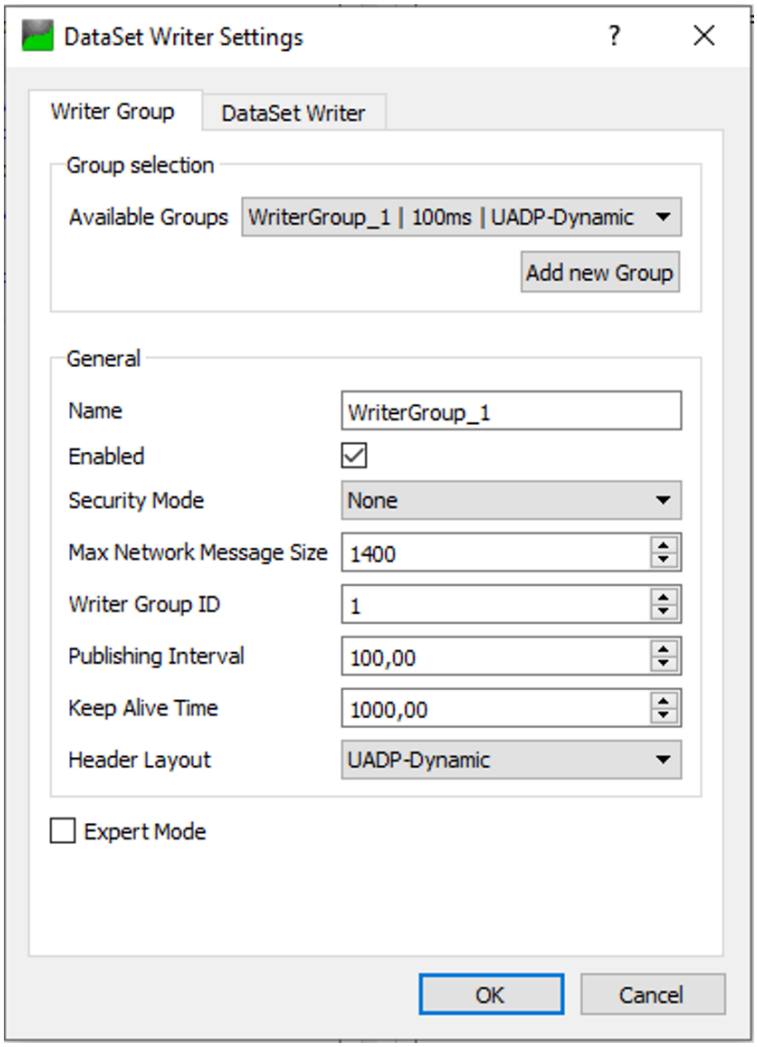
PubSubConnection

- A PubSubConnection is a combination of protocol selection, protocol settings and addressing information. It describes for example a UDP Multicast Group, e.g. "UDP 239.0.0.1 Port 4840"
- The most interesting parameters are the "Publisher Id", "Transport Facet" and the "Network Address" used by the connection.
- The Publisher to Subscriber relations and Subscriber to Publisher relations are displayed here in an overview.



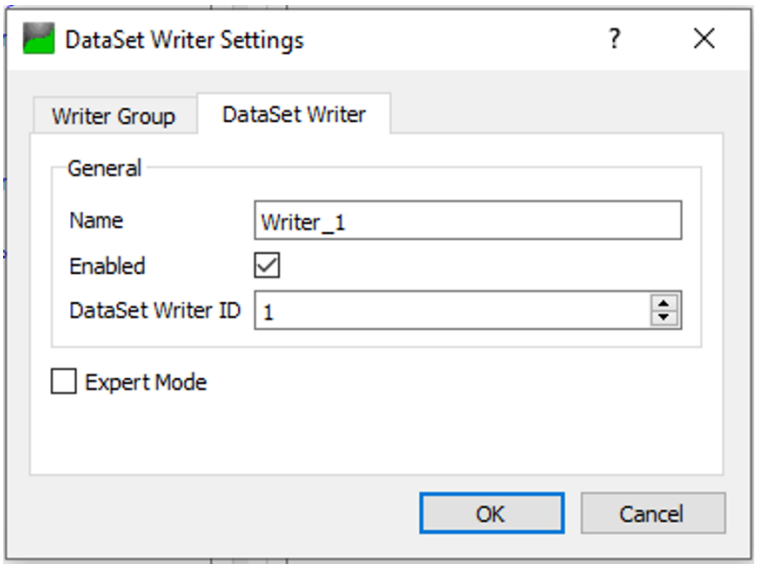
WriterGroup

- In a PubSubConnection multiple WriterGroups can exists.
- WriterGroup parameters are necessary for creating a NetworkMessage.
- The "Writer Group ID" parameter identifies the WriterGroup uniquely in the PubSub Connection.
- The "Publishing Interval" parameter defines the cycle time in milliseconds to publish the configured DataSets.


DataSet Writer Settings
Writer Group
DataSet Writer
Group selection
Available Groups
WriterGroup_1 | 100ms | UADP-Dynamic
Add new Group
General
Name
WriterGroup_1
Enabled
Security Mode
None
Max Network Message Size
1400
Writer Group ID
1
Publishing Interval
100,00
Keep Alive Time
1000,00
Header Layout
UADP-Dynamic
Expert Mode
OK
Cancel

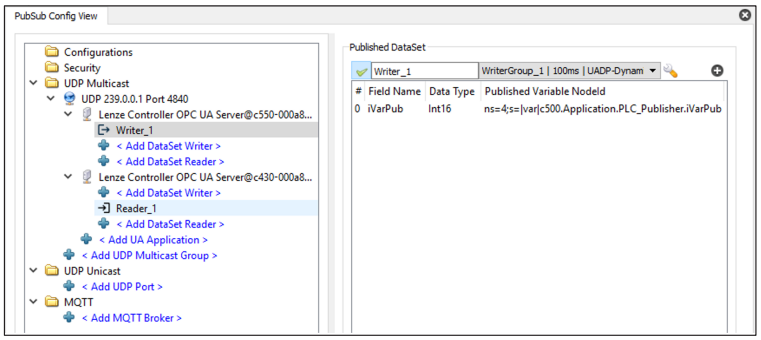
DataSetWriter

- The DataSetWriter is linked to one PublishedDataSet.
- The "DataSet Writer ID" is used to select DataSetMessages for a PublishedDataSet on the Subscriber side.



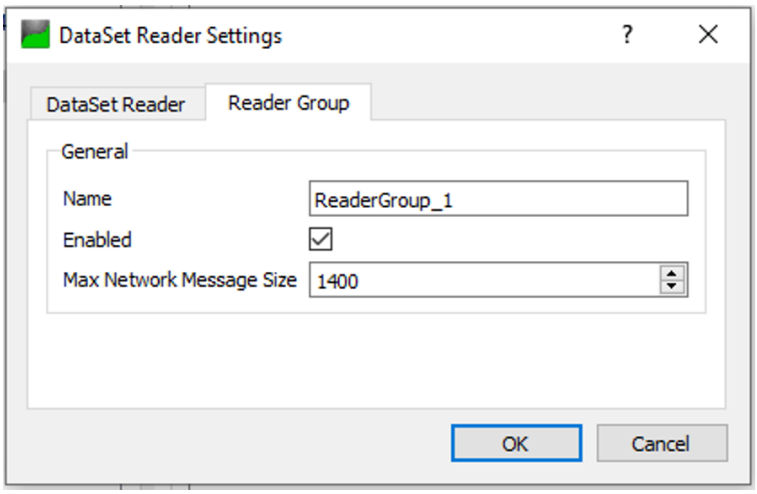
PublishedDataSet

- A PublishedDataSet defines the variables and metadata to be published.
- It can be setup by dragging and dropping variables from the AddressSpace View of the corresponding OPC UA Server.



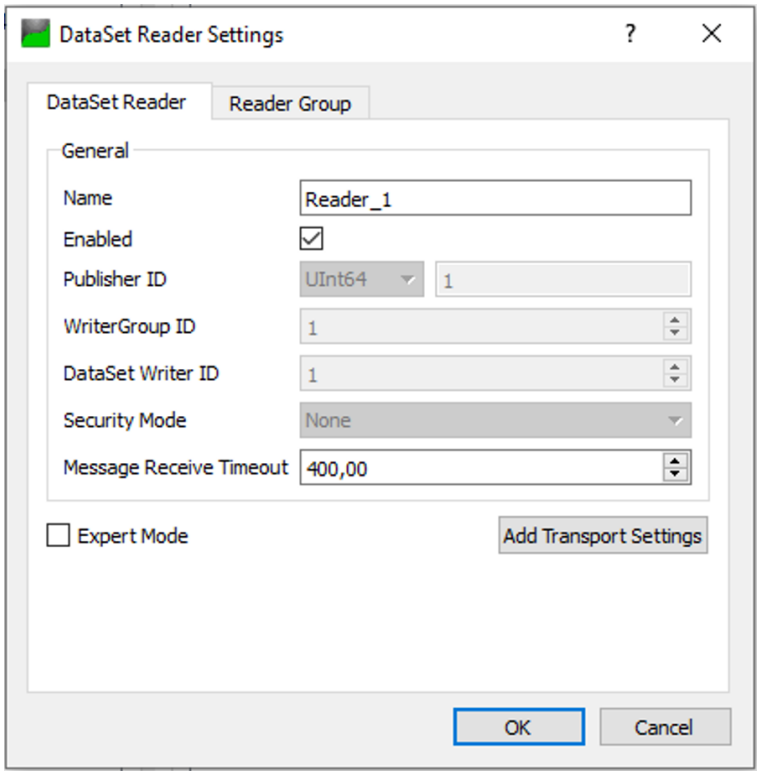
ReaderGroup

- In a PubSubConnection multiple ReaderGroups can exists.
- The ReaderGroup is used to group a list of DataSetReaders. It is not symmetric to a Writer-Group and it is not related to a particular NetworkMessage. The NetworkMessage related filter settings are on the DataSetReaders.



DataSetReader

- DataSetReader parameters represent settings for filtering of received NetworkMessages and DataSetMessages.
- A DataSetReader can be assigned directly to an existing DataSetWriter at creation (sets correct WriterGroup ID and Publisher ID).



SubscribedDataSet

- SubscribedDataSet parameters define the processing of the decoded DataSet in the Subscriber for one DataSetReader.
- It can be setup by drag & drop or manually target variables from the AddressSpace View of the corresponding OPC UA Server.
- An override method can be selected for each field (Last usable value or defined override value).