Manual

TC3 OPC UA

TwinCAT

Version: 2.6
Date: 2019-02-08
Order No.: TF6100

BECKHOFF

Table of contents

1 Foreword ... 7
1.1 Notes on the documentation ... 7
1.2 Safety instructions ... 8

2 Overview ... 9
2.1 Scenarios ... 9
2.2 OPC UA Server revision overview ... 12
2.3 Version overview ... 12
2.4 Application examples ... 14
2.4.1 Post-processing in the Cloud ... 14

3 Installation ... 20
3.1 System requirements ... 20
3.2 Installation (TC3) ... 22
3.3 Installation (TC2) ... 25
3.4 Installation Windows CE (TC3) ... 28
3.5 Installation Windows CE (TC2) ... 30
3.6 Licensing (TC3) ... 35
3.7 Licensing (TC2) ... 39

4 Technical introduction ... 40
4.1 Server ... 40
4.1.1 Overview ... 40
4.1.2 Quick start ... 41
4.1.3 PLC ... 43
4.1.4 C++ ... 75
4.1.5 I/O ... 94
4.1.6 Matlab/Simulink ... 106
4.1.7 File Transfer ... 119
4.1.8 Security ... 121
4.1.9 Symbol Caching ... 124
4.1.10 Miscellaneous ... 125
4.2 Client I/O ... 134
4.2.1 Overview ... 134
4.2.2 Quick start ... 135
4.2.3 Supported data types ... 138
4.2.4 Adding nodes of a Server ... 139
4.2.5 Node attributes ... 140
4.2.6 Method call ... 141
4.2.7 StructuredTypes ... 144
4.2.8 Data recording ... 146
4.2.9 Security ... 147
4.3 Client PLCopen ... 149
4.3.1 Overview ... 149
4.3.2 Supported data types ... 150
4.3.3 Best practice ... 150

4.3.4 Security 164
4.4 Gateway 166
4.4.1 Overview 166
4.4.2 Quick start 167
4.4.3 Licensing 169
4.4.4 Scenarios 169
4.4.5 Configurator 171
4.4.6 Migrating from Tx6120 176
4.4.7 Security 178
4.5 Sample Client 180
4.5.1 Overview 180
4.5.2 Establishing a secure connection to OPC UA Server 181
4.5.3 Browsing the UA namespace 184
4.5.4 Using the Watchlist 185

5 Configuration 187
5.1 Setup Version 3.x.x 187
5.1.1 Overview 187
5.1.2 Data Access 188
5.1.3 Historical Access 189
5.1.4 Server Security 189
5.1.5 Alarms and Conditions 190
5.1.6 Online Panel 191
5.2 Setup Version 4.x.x 192
5.2.1 Overview 192
5.2.2 Creating a new project 193
5.2.3 Select target device 194
5.2.4 Adding ADS devices 195
5.2.5 Downloading and uploading configurations 198
5.2.6 Importing and exporting configuration files 199
5.2.7 Configuring historical access 200
5.2.8 Configuring Alarms and Conditions 202
5.2.9 Configuring alarm texts 205
5.2.10 Configuring endpoints 206
5.2.11 Configuring certificate trust settings 207
5.2.12 Configuring security settings 208
5.2.13 Restarting the server 216
5.2.14 Troubleshooting and logging 216

6 PLC API 218
6.1 Function blocks 218
6.1.1 UA_Connect 218
6.1.2 UA_Disconnect 219
6.1.3 UA_GetNamespaceIndex 220
6.1.4 UA_HistoryUpdate 221
6.1.5 UA_MethodCall 223
6.1.6 UA_MethodGetHandle 224

6.1.7 UA_MethodReleaseHandle 225
6.1.8 UA_NodeGetHandle 226
6.1.9 UA_NodeGetHandleList 227
6.1.10 UA_NodeReleaseHandle 228
6.1.11 UA_NodeReleaseHandleList 229
6.1.12 UA_Read 230
6.1.13 UA_ReadList 231
6.1.14 UA_Write 232
6.2 Data types 233
6.2.1 E_UAAttributeID 233
6.2.2 E_UADataType 234
6.2.3 E_UAIdentifierType 234
6.2.4 E_UASecurityMsgMode 235
6.2.5 E_UASecurityPolicy 235
6.2.6 E_UATransportProfile 235
6.2.7 ST_UASessionConnectInfo 235
6.2.8 ST_UAIndexRange 236
6.2.9 ST_UAMethodArgInfo 237
6.2.10 ST_UANodeID 237
6.2.11 ST_UANodeAdditionalInfo 237
6.2.12 UAHADataValue 238
6.2.13 UAHAUpdateStatusCode 239

7 Samples 240

8 Appendix 241
8.1 Error diagnosis 241
8.2 OPC UA Client error codes 242
8.3 ADS Return Codes 243
8.4 Support and Service 246

1 Foreword

1.1 Notes on the documentation

This description is only intended for the use of trained specialists in control and automation engineering who are familiar with the applicable national standards.
It is essential that the documentation and the following notes and explanations are followed when installing and commissioning the components.
It is the duty of the technical personnel to use the documentation published at the respective time of each installation and commissioning.

The responsible staff must ensure that the application or use of the products described satisfy all the requirements for safety, including all the relevant laws, regulations, guidelines and standards.

Disclaimer

The documentation has been prepared with care. The products described are, however, constantly under development.
We reserve the right to revise and change the documentation at any time and without prior announcement.
No claims for the modification of products that have already been supplied may be made on the basis of the data, diagrams and descriptions in this documentation.

Trademarks

Beckhoff®, TwinCAT®, EtherCAT®, Safety over EtherCAT®, TwinSAFE®, XFC® and XTS® are registered trademarks of and licensed by Beckhoff Automation GmbH.
Other designations used in this publication may be trademarks whose use by third parties for their own purposes could violate the rights of the owners.

Patent Pending

The EtherCAT Technology is covered, including but not limited to the following patent applications and patents:
EP1590927, EP1789857, DE102004044764, DE102007017835
with corresponding applications or registrations in various other countries.

The TwinCAT Technology is covered, including but not limited to the following patent applications and patents:
EP0851348, US6167425 with corresponding applications or registrations in various other countries.

EtherCAT® is registered trademark and patented technology, licensed by Beckhoff Automation GmbH, Germany

Copyright

© Beckhoff Automation GmbH & Co. KG, Germany.
The reproduction, distribution and utilization of this document as well as the communication of its contents to others without express authorization are prohibited.
Offenders will be held liable for the payment of damages. All rights reserved in the event of the grant of a patent, utility model or design.

1.2 Safety instructions

Safety regulations

Please note the following safety instructions and explanations!
Product-specific safety instructions can be found on following pages or in the areas mounting, wiring, commissioning etc.

Exclusion of liability

All the components are supplied in particular hardware and software configurations appropriate for the application. Modifications to hardware or software configurations other than those described in the documentation are not permitted, and nullify the liability of Beckhoff Automation GmbH & Co. KG.

Personnel qualification

This description is only intended for trained specialists in control, automation and drive engineering who are familiar with the applicable national standards.

Description of symbols

In this documentation the following symbols are used with an accompanying safety instruction or note. The safety instructions must be read carefully and followed without fail!

⚠ DANGER

Serious risk of injury!

Failure to follow the safety instructions associated with this symbol directly endangers the life and health of persons.

⚠ WARNING

Risk of injury!

Failure to follow the safety instructions associated with this symbol endangers the life and health of persons.

⚠ CAUTION

Personal injuries!

Failure to follow the safety instructions associated with this symbol can lead to injuries to persons.

NOTE

Damage to the environment or devices

Failure to follow the instructions associated with this symbol can lead to damage to the environment or equipment.

Tip or pointer

This symbol indicates information that contributes to better understanding.

2 Overview

OPC Unified Architecture (OPC UA) is the next generation of the familiar OPC standard. This is a globally standardized communication protocol via which machine data can be exchanged irrespective of the manufacturer and platform. OPC UA already integrates common security standards directly in the protocol. Another major advantage of OPC UA over the conventional OPC standard is its independence from the COM/DCOM system.

Detailed information on OPC UA can be found on the webpages of the OPC Foundation.

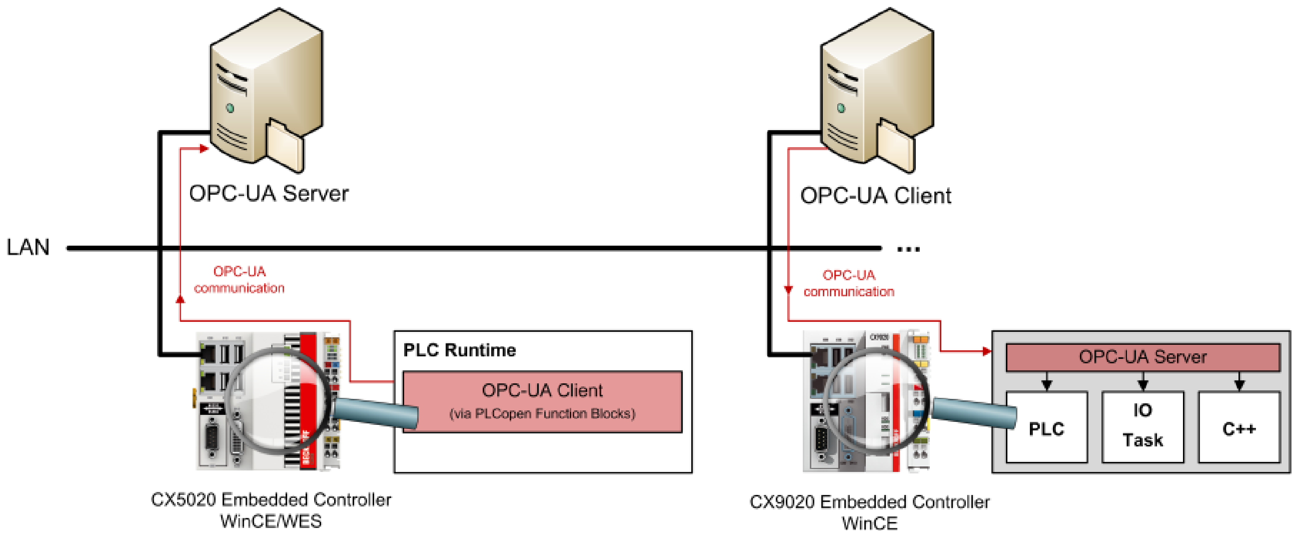
Components

The following software components have been integrated for Win32/64 and Windows CE-based systems:

Software component	Description
OPC UA Server [▶ 40]	Provides an OPC UA Server interface so that UA clients can access the TwinCAT runtime.
OPC UA Client [▶ 149]	Provides OPC UA Client functionality to enable communication with other OPC UA Servers based on PLCopen-standardized function blocks and an easy-to-configure I/O device.

The following software components have been integrated for Win32/64-based systems:

Software component (Windows only)	Description
OPC UA Configurator [▶ 187]	Graphical user interface for configuring the TwinCAT OPC UA Server
OPC UA Sample Client [▶ 180]	Graphical sample implementation of an OPC UA Client in order to carry out a first connection test with the TwinCAT OPC UA Server.
OPC UA Gateway [▶ 166]	Wrapper technology that provides both an OPC COM DA Server interface and OPC UA Server aggregation capabilities.


OPC-UA Server
OPC-UA Client
LAN
...
OPC-UA communication
OPC-UA communication
PLC Runtime
OPC-UA Client (via PLCopen Function Blocks)
OPC-UA Server
PLC
IO Task
C++
CX5020 Embedded Controller WinCE/WES
CX9020 Embedded Controller WinCE

2.1 Scenarios

In the following, some scenarios will be illustrated in accordance with which the components can be implemented and used, depending on the application case and infrastructure:

- OPC UA Server: Integrated in Industrial PC or Embedded PC [▶ 10]
- OPC UA Server: Runs on a central computer with connection to remote TwinCAT runtime(s). [▶ 11]

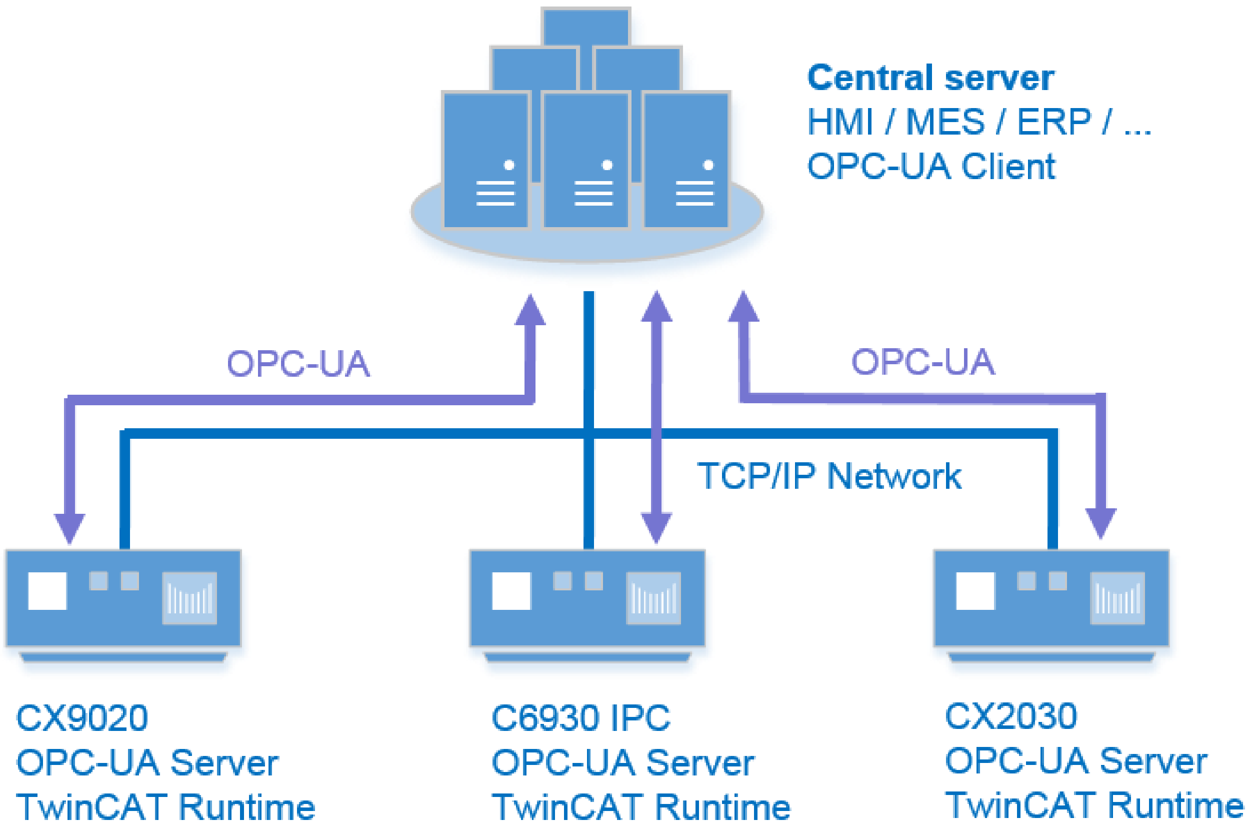
- OPC UA Server: Access to BC Controller [⏵ 11]

OPC UA Server: Integrated in Industrial PC or Embedded PC

Recommended scenario

This scenario describes how the TwinCAT OPC UA Server should be used under normal circumstances.

One of the biggest advantages of the TwinCAT OPC UA Server is that it can be integrated into even the smallest embedded platform, e.g. the CX8000 series. Thanks to this integration, general handling is very simple and convenient. OPC UA Clients, e.g. HMI or MES/ERP systems, can connect to the OPC UA Server and read or write symbol information from the TwinCAT runtime.



The following software components and configurations run on the central server:

- Third party OPC UA Client, which may be an HMI, MES or ERP system, for example.

The following software components and configurations run on the Industrial PC or Embedded PC:

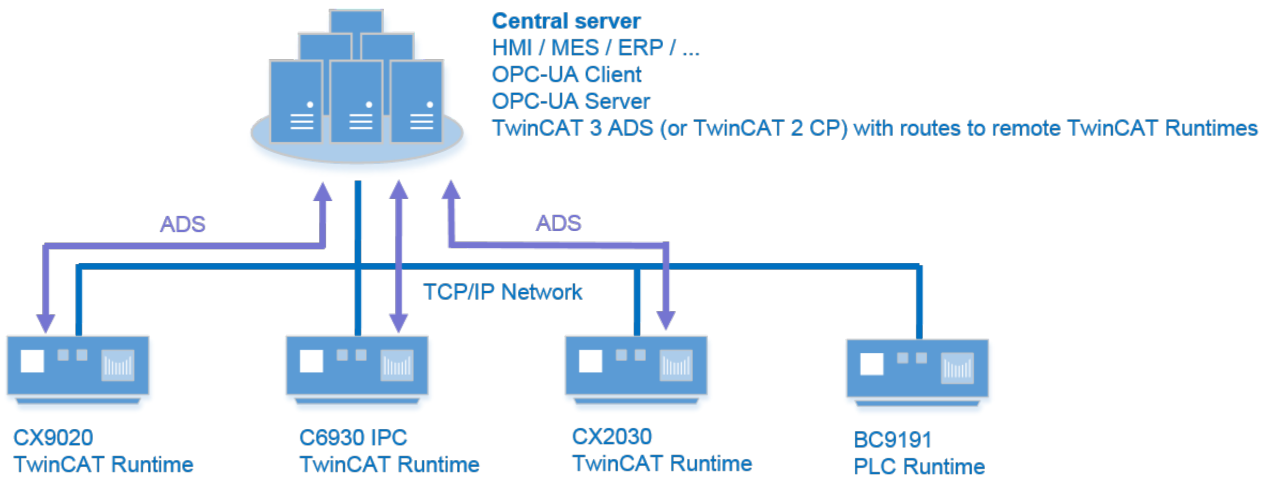
- The OPC UA Server automatically establishes a connection with the first local TwinCAT PLC runtime.
- TwinCAT runtime

This scenario has the following advantages:

- Network usage is optimized because it is based on OPC UA communication technologies, such as OPC UA registrations.
- Memory usage is decentralized. Each device is only responsible for its own memory requirements.
- OPC UA features security mechanisms that are directly integrated in the protocol. This is very useful if one of the Industrial PCs or Embedded PCs is connected via the internet, for example.

OPC UA Server: Runs on a central computer with connection to remote TwinCAT runtime(s)

This scenario describes the conventional implementation of OPC Servers. Servers using the old OPC A technology were often implemented on a central server instead of the Industrial PC on which TwinCAT runtime was executed, in order to avoid DCOM configurations. (Remember: in contrast to OPC UA, OPC DA is based on COM/DCOM technologies)



The following software components and configurations run on the central server:

- TwinCAT 3 ADS (or TwinCAT 2 CP) for the required ADS connectivity
- OPC UA Server with data access devices configured for remote TwinCAT runtimes
- ADS routes to remote TwinCAT runtimes
- Symbol files from any remote TwinCAT runtime
- OPC UA Client, which can be an HMI, MES or ERP system, for example.

The following software components and configurations run on the Industrial PC or Embedded PC:

- TwinCAT runtime
- ADS route to the central server

The more remote TwinCAT runtimes are connected to the central OPC UA Server, the higher the network usage will be.

The OPC UA Server uses advanced ADS recording techniques to query the symbol values of the TwinCAT runtime. The more symbols there are, the more cyclical queries are in progress. As a result, optimized OPC UA communication only takes place locally on the central server (between OPC UA Client and OPC UA Server).

This scenario has the following disadvantages compared to scenario 1:

- Network usage can be very high depending on the number of devices and symbols present.
- The memory requirement on the central server is very high because the OPC UA Server has to import symbol information from every TwinCAT runtime.
- No security between the central server and TwinCAT runtime based on data encryption - ADS was designed for high performance.
- The need to exchange symbol files between TwinCAT runtime and central server after each PLC program modification.

OPC UA Server: Access to BC Controller

Although small BC controllers are not able to run their own OPC UA Server, they may nevertheless have access to the PLC runtime executed on a remote BC Controller, such as a BC9191, and publish their symbol values via OPC UA. In this scenario the OPC UA Server must run together with a TwinCAT 3 ADS (or TwinCAT 2 CP) on another computer, and the scenario must be configured in the same way as scenario 2.

2.2 OPC UA Server revision overview

The following revision overview shows important differences between the product versions. This information is important when the software product undergoes a significant software update.

Important changes in the version	Component	Comments
All versions	Arrays in a PLC program	By default, array elements of a PLC array are not displayed as separate nodes in the UA namespace. Instead, only one node is used for the entire array, and UA Clients can access each item through its IndexRange. (See also: Use of arrays [▶ 71])
2.1	Enumerations in a PLC program	By default, enumerations in a PLC program are not displayed as Enum types in the UA namespace. In the previous version these were simply Int16.
2.1	NamespaceName	Since version 2.1, the NamespaceName is generated based on the URI concept and always contains the host name of the computer on which the UA Server is running. In older versions the NamespaceName is the same as the name defined in the configurator, e.g. "PLC1". This information is particularly important if a UA Client evaluates the NamespaceName in a NamespaceIndex before accessing a node. For compatibility purposes and to simplify the upgrade without interfering with the UA Client, version 2.2 of TwinCAT OPC UA Server provides a mechanism to activate the old NamespaceName concept. (See also: Server [▶ 40])
2.2	Structures in PLC	Since version 2.2.x structures in a PLC program can be accessed using a StructuredType in the UA namespace. This allows UA Clients to interpret element types in STRUCT. (See also: StructuredTypes [▶ 68])
2.2	Namespace configuration of a C++ module instance	Compared to older versions (up to 2.1.x), the UA namespace of a C++ module was redesigned in version 2.2.x to allow advanced functionality such as method calls for a C++ module instance.

2.3 Version overview

The TwinCAT OPC UA supplement/function product is supplied in various setup versions, which reflect different development states. For new projects it is recommended to change to the latest setup version in order to be able to use all new product functions. Older setup versions are still available to download for existing projects.

Two versions are currently available:

- Setup Version 3.x.x: Long-standing existing version, which already offers a variety of functions. This version is still maintained with regard to optimizations.
- Setup Version 4.x.x: New version. Future developments and new product functionalities will only be implemented in this version.

The following table provides an overview of the different versions with regard to their product functionalities.

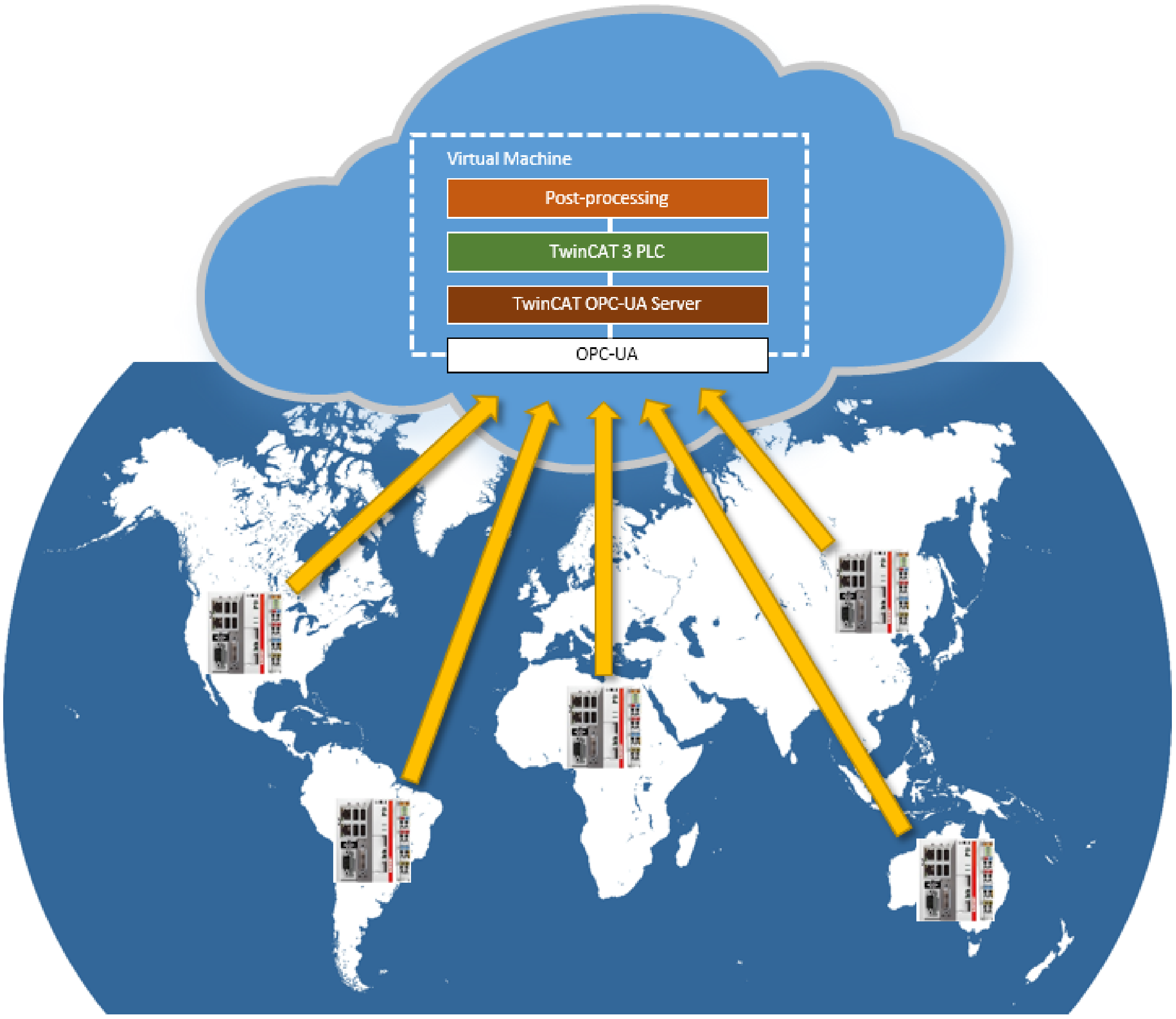
Functionality	3.x.x	4.x.x	Comments
Data Access PLC	yes	yes	Enables read/write access to the TwinCAT 2/3 PLC
Data Access TcCOM	yes	yes	Enables read/write access to TwinCAT 3 TcCOM modules

Functionality	3.x.x	4.x.x	Comments
Data Access I/O task	yes	yes	Enables read/write access to the TwinCAT 2/3 I/O task
Data Access BC controllers	yes	yes	Enables read/write access to Beckhoff BC controllers
Historical Access PLC	yes	yes	Enables activation of variables from the TwinCAT 2/3 PLC for historical access
Historical access TcCOM	yes (only TC3 C++)	yes	Enables activation of variables from a TwinCAT 3 TcCOM module for historical access
Historical Access I/O task	no	yes	Enables activation of variables from a TwinCAT 2/3 I/O task for historical access
Historical Access BC controllers	no	yes	Enables activation of variables from a Beckhoff BC controller for historical access
Historical Access for existing projects	no	yes	Enables activation of variables for historical access without having to adapt the runtime project
Alarms and conditions PLC	yes	yes	Enables activation of variables from the TwinCAT 2/3 PLC for alarms and conditions
Alarms and conditions TcCOM	no	yes	Enables activation of variables from a TwinCAT 3 TcCOM module for alarms and conditions
Alarms and conditions I/O Task	no	yes	Enables activation of variables from a TwinCAT 2/3 I/O task for alarms and conditions
Alarms and conditions BC controllers	no	yes	Enables activation of variables from a Beckhoff BC controller for alarms and conditions
Alarms and conditions for existing projects	no	yes	Enables activation of variables for alarms and conditions without having to adapt the runtime project
PLC method calls	yes	yes	OPC UA Server enables TwinCAT 3 PLC methods to be activated and displayed as OPC UA methods
TcCOM method calls	yes (only TC3 C++)	yes	OPC UA Server enables TwinCAT 3 C++ methods to be activated and displayed as OPC UA methods
Security X509 certificates, self-signed	yes	yes, configurable	Generation of self-signed certificates
Security X509 certificates, custom	yes	yes	Enables the use of custom certificate authorizations
Security user authentication	yes	yes	OPC UA Server enables authentication of local or domain users
Security access level namespace level	no	yes	OPC UA Server allows definition of different access levels at namespace level
Security access level node level	no	yes	OPC UA Server allows definition of different access levels at node level
Configurator in Visual Studio	no	yes	-
Configurator remote access	no	yes	Configurator allows remote configuration of TwinCAT OPC UA Servers independent of the operating system used on the target device.

2.4 Application examples

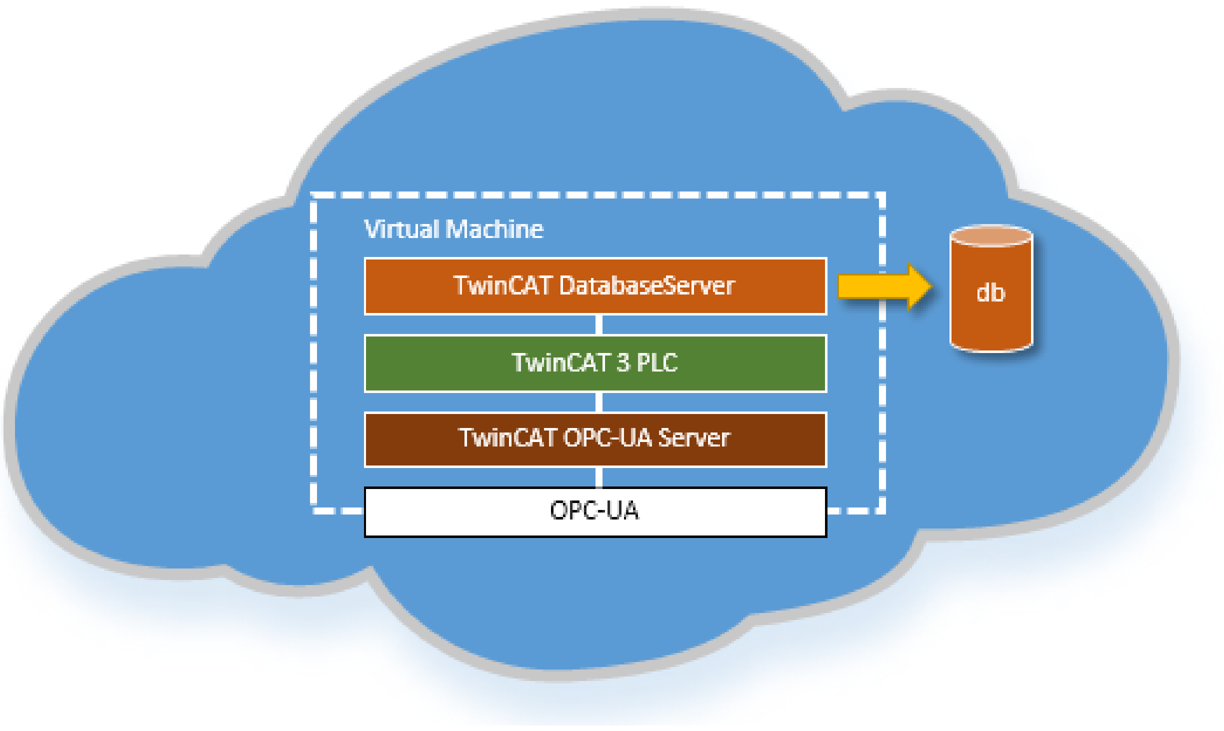
2.4.1 Post-processing in the Cloud

The overall concept of providing a generic, standardized and callable interface in the cloud that accepts incoming data for further analysis is based on an endpoint in the cloud, either a Windows-based virtual machine or a real hardware running a Windows operating system, in addition to the decentralized clients that call this service. In this documentation, this device is generally referred to as OPC UA Server (in the cloud).



Data logger

The documentation shows how computing resources in the Cloud can be accessed and how local TwinCAT PLC devices can be connected in order to use these resources. As an initial example, this application case can be applied to a typical data storage scenario in which decentralized TwinCAT PLC devices send data to the cloud, in which the incoming data is stored in an SQL database via the TwinCAT Database Server. Although the general scenario also involves sending data back and forth (i.e. sending parameters to the cloud, performing calculations in the cloud, and then returning the computed results), this specific scenario does not require the return of results from the cloud.



System requirements: OPC UA Server (the cloud)

The cloud must be able to host one of the following operating system variants. Note that using virtualization technology requires a more costly solution for 64-bit operating systems.

Device is a virtual machine

- 32-bit Windows operating system
- 64-bit Windows operating system with CPU core isolation (two cores required as a minimum)

Device is a dedicated hardware

- 32-bit Windows operating system
- 64-bit Windows operating system

In addition, the following software components must be installed on the device:

- TwinCAT 3 XAR (Runtime only) or XAE (Runtime and Engineering)
- TwinCAT 3 Function TF6100 OPC UA

Note that when installing a complete TwinCAT 3 XAE, additional configuration settings such as handling licenses or the option of debugging the device directly in the cloud may be useful. The following software components must be installed to store data in a central database received from a client:

- TwinCAT 3 Function TF6420 Database

System requirements: Decentralized OPC UA Client

The decentralized OPC UA Client is based on a TwinCAT 3 PLC runtime and can therefore be hosted on any hardware configuration that supports the operation of a TwinCAT 3 PLC. In addition, TwinCAT 3 Function TF6100 OPC UA must be installed on the client device in order to be able to use the TwinCAT OPC UA Client to execute UA methods on the server in the cloud.

Technical description

The implementation of an OPC UA Server in the cloud is very application-specific. Beckhoff provides a general architecture description, which is independent of the individual application. As an application example, this article frequently refers a data aggregation scenario in which process data is transferred from decentralized clients via OPC UA to a central server, where the data is collected in a central database. The advantage of OPC UA in this case is not only the standardized interface (and therefore to use this concept for any type of OPC UA Client), but also the ability to secure the communication channel by using the integrated security mechanism of OPC UA.

The TwinCAT OPC UA Server is used to disclose the methods defined in IEC61131-3 PLC to the OPC UA namespace. These methods can be called by any TwinCAT OPC UA Client (based on the PLCopen function blocks) or by third party OPC UA Clients.

Decentralized OPC UA Client

The main purpose of the OPC UA Client is to call methods on the remote server. The client is most probably a TwinCAT-3-embedded device that fulfils the decentralized part of the application. For the data aggregation example, this would mean querying process data and sending it to the server by calling a method.

Interface description (to the cloud)

As an interface to a centralized cloud system, there are only two methods provided by the server:

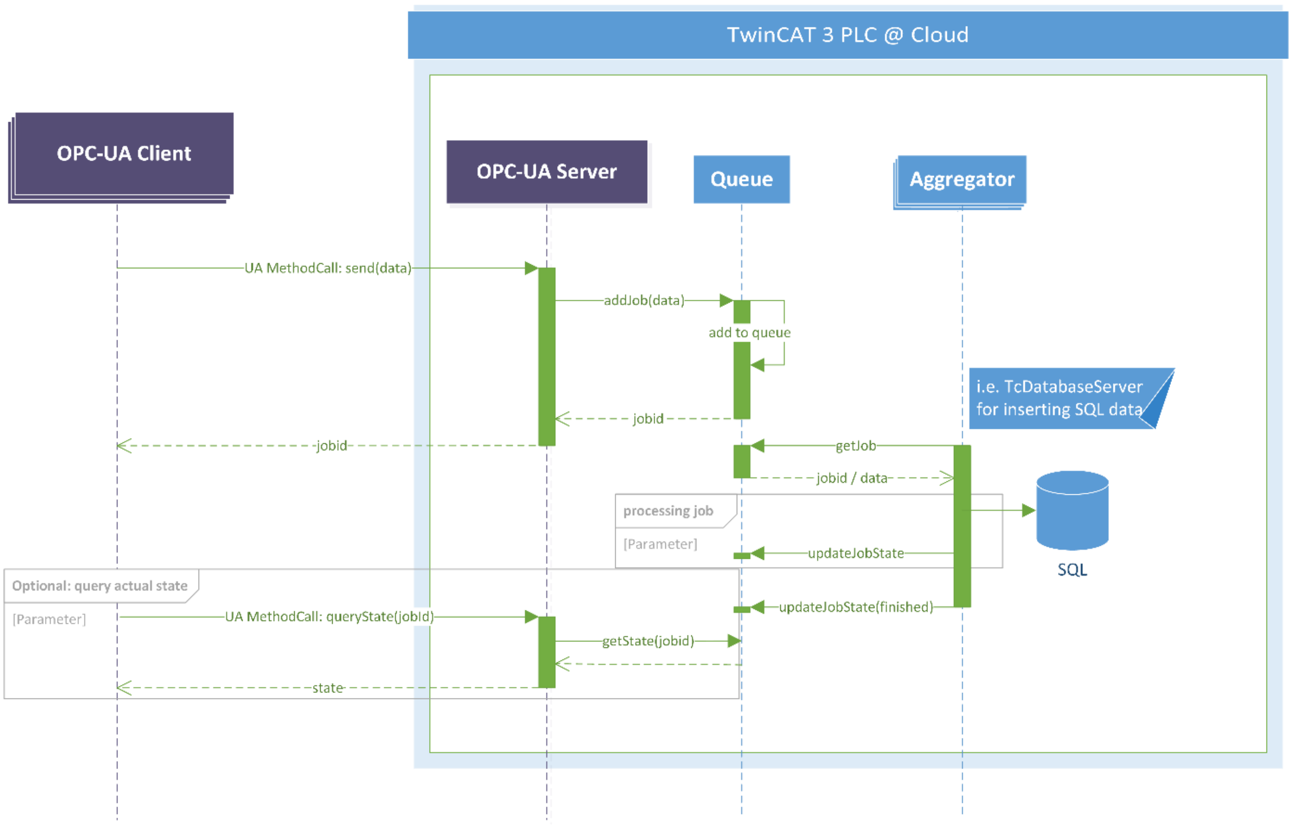
- int Send(data): This method forwards the data using an OPC UA method call and returns a JobID.
- int QueryState(jobID): Using the previously retrieved JobID, the client can cyclically query the current status of the job. This could be omitted if the server in the cloud can handle all situations, or if the client cannot help out in case of problems.

Communication between clients and the cloud can be secured by OPC UA's built-in security concepts, e.g. by using client and server certificates to encrypt data communication.

The cloud

The OPC UA Server in the cloud is based on TwinCAT 3 PLC methods that are disclosed to the OPC UA namespace. The PLC project contains the following three components:

- MethodCall provider: This component provides the above-mentioned send method, which collects data, creates a JobID and places the data in the queue as different jobs. There may also be a QueryState method to allow OPC UA Clients to query the current job status.
- Queue: The queue stores a list of jobs to be executed. Each job in the queue contains the following information: {ID, Status, Data}. New elements (jobs) are added by the MethodCall provider and can be deleted by it. The entire queue can be based on a hash table within the PLC program.
- Aggregator: The application that processes the jobs. It cyclically looks in the queue to see whether there are any jobs to be processed. If this is the case, it takes over the job (sets the status to "processed") and starts processing the job, e.g. connecting to a database via the function blocks of the TwinCAT Database Server. Note that there can be more than one aggregator to enable parallel processing of the queue.



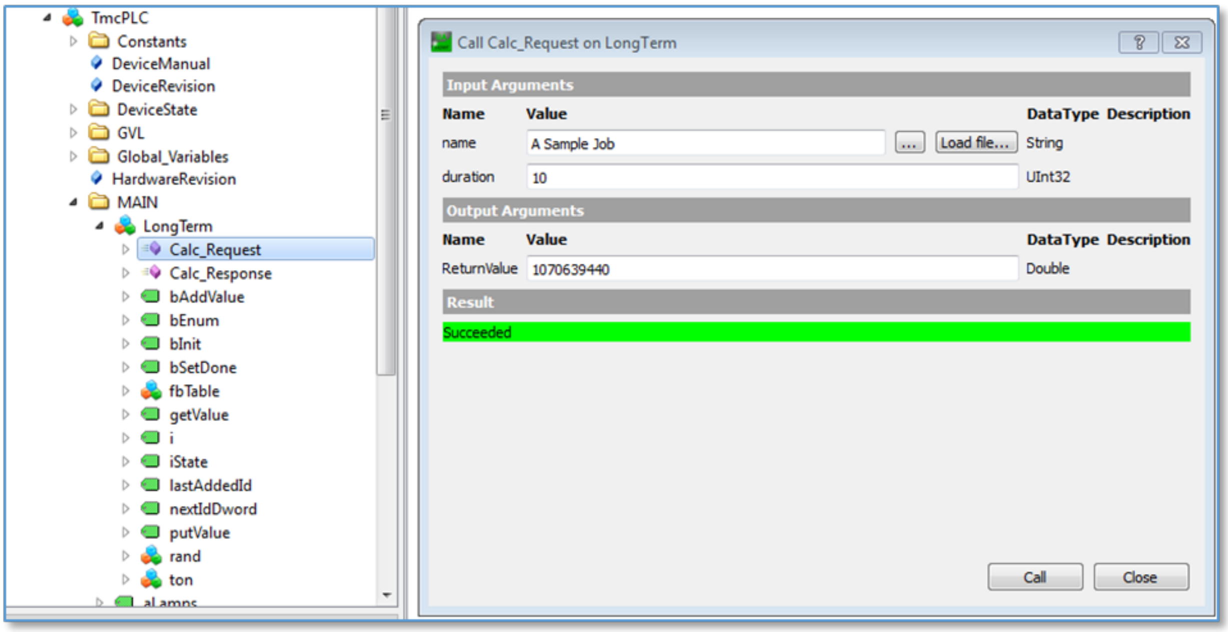
Sample

The following sample illustrates the scenario by executing simple jobs: Jobs can be presented to the server by an OPC UA Client. In this sample, the job data consists of a definable time interval that is recorded by the aggregator to complete the job. The sample consists of the following PLC components on the OPC UA Server:

- ST_JobEntry: Represents a job in the queue. In terms of the data, the job only consists of a name and a duration. A duration defines the length of time a job takes.
- E_JobState: Represents the status of a job. The sample implementation defines the following values: QUEUED, PROCESSING, READY, FAILED and INVALID.
- LongTerm: Represents the MethodCall server (consisting of the methods Calc_Request and Calc_Response) and the aggregator that processes the job (which is implemented in the function block)
- FB_SpecialHashTableCtrl: Represents the queue in the form of a hash table, as reflected by PLC examples from the Beckhoff Information System. Here, different methods for handling the queue are provided (Add, Count, GetFirst, GetNext, Lookup, Remove, Reset).

Using the example

UA Expert could be used as a sample client to call the methods provided by the OPC UA Server in the cloud. By calling the Calc_Request method, the client receives a JobID (or 0 for displaying an error):



- TmcPLC
  - Constants
  - DeviceManual
  - DeviceRevision
  - DeviceState
  - GVL
  - Global_Variables
  - HardwareRevision
  - MAIN
    - LongTerm
      - Calc_Request
      - Calc_Response
      - bAddValue
      - bEnum
      - bInit
      - bSetDone
      - fbTable
      - getValue
      - i
      - iState
      - lastAddedId
      - nextIdDword
      - putValue
      - rand
      - ton

Call Calc_Request on LongTerm

Input Arguments

Name	Value	DataType	Description
name	A Sample Job	String	
duration	10	UInt32	

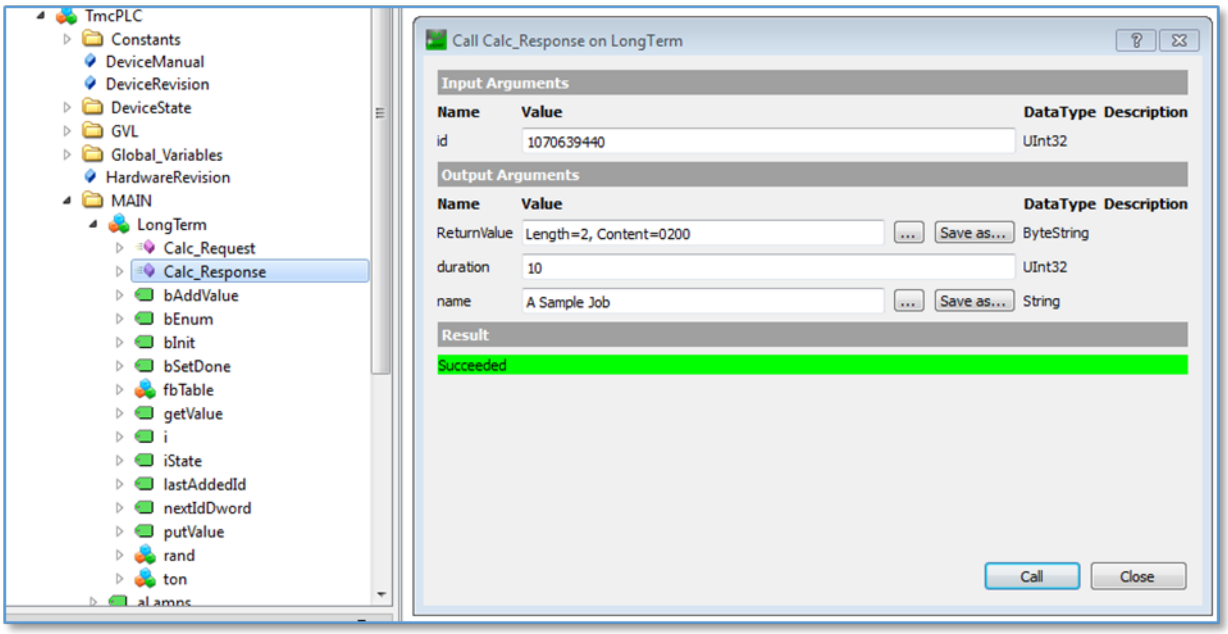
Output Arguments

Name	Value	DataType	Description
ReturnValue	1070639440	Double	

Result

Succeeded

The client can query the JobState by calling the method Calc_Response with the JobID. In addition, the previously set duration and the job name are returned for subsequent reference.



- TmcPLC
  - Constants
  - DeviceManual
  - DeviceRevision
  - DeviceState
  - GVL
  - Global_Variables
  - HardwareRevision
  - MAIN
    - LongTerm
      - Calc_Request
      - Calc_Response
      - bAddValue
      - bEnum
      - bInit
      - bSetDone
      - fbTable
      - getValue
      - i
      - iState
      - lastAddedId
      - nextIdDword
      - putValue
      - rand
      - ton

Call Calc_Response on LongTerm

Input Arguments

Name	Value	DataType	Description
id	1070639440	UInt32	

Output Arguments

Name	Value	DataType	Description
ReturnValue	Length=2, Content=0200	ByteString	
duration	10	UInt32	
name	A Sample Job	String	

Result

Succeeded

Installation

The following chapter describes the installation and configuration of each software component required on the server in the cloud.

Runtime only

If the server does not host the engineering environment of TwinCAT 3, the TwinCAT 3 XAR installation must be used. Note that in this case the entire handling for proper function involves several steps, and future maintenance may be more difficult because the XAR installation lacks the programming and debugging environment. In this scenario, the user must ensure that maintenance of the ADS routes between the actual engineering computer (on which the PLC program for the server is developed) and the device that hosts the server in the cloud must allow not only downloading and debugging of the PLC program, but also activation of licenses for the TwinCAT 3 runtime. Note that you must open the firewall ports to enable ADS traffic. See the ADS documentation for more information. The TC3 License Request Generator tool was developed for easier handling of TwinCAT 3 runtime licenses. It can be downloaded via the Beckhoff FTP server and enables the creation of License Request Files and the import of License Response Files:

ftp.beckhoff.com/Software/TwinCAT/Unsupported_Utilities/TC3-LicenseGen/

Runtime and Engineering

This is the recommended setup scenario. The server in the cloud hosts a TwinCAT 3 runtime and the corresponding engineering components to enable debugging and easier handling of the runtime system. In this case it is necessary to install TwinCAT 3 XAE on the system.

Additional software

After successful installation of TwinCAT XAE or XAR, the following software components must be installed and configured on the device:

TF6100 OPC UA

The TF6100 function installs an OPC UA Server (and Client) on the device. The server is required to provide the OPC UA Clients with the PLC methods. When the PLC program is downloaded into the runtime, the OPC UA Server automatically imports the first PLC runtime into its namespace. All variables and methods of the PLC marked as available via OPC UA are imported into the OPC UA namespace and become available to clients. Refer to the TF6100 documentation for more information.

TF6420 Database

If the server is to store the data received from the clients in a database, the TF6420 function must be used. It provides generic function blocks for TwinCAT 3 PLCs for accessing the database, e.g. to insert values into a database table. Install the TF6420 setup and consult the TF6420 documentation for more information on using the corresponding function blocks.

Licensing

All TwinCAT 3 software products require a license to be available on the server. The license can be either a 7-day trial or an unlimited license. Licenses can be activated using the TwinCAT 3 XAE License Manager. If a TwinCAT 3 XAR installation is used on the server, use the TC3 License Generator.

3 Installation

3.1 System requirements

OPC UA Server

Technical data	Description
Operating system	Windows 7, 10 Windows CE 6/7 Windows Embedded Standard 2009 Windows Embedded 7 Windows Server 2008 R2 Windows Server 2012
Target platforms	PC architecture (x86, x64)
.NET Framework	4.5.2 (only required for configurator)
TwinCAT Version	TwinCAT 2, TwinCAT 3
TwinCAT installation level	TwinCAT 2 CP, PLC, NC-PTP TwinCAT 3 XAE, XAR, ADS
Required TwinCAT license	TS6100 TwinCAT OPC UA (for TwinCAT 2) TF6100 TC3 OPC UA (for TwinCAT 3)

OPC UA Client

Technical data	Description
Operating system	Windows 7, 10 Windows CE 6/7 Windows Embedded Standard 2009 Windows Embedded 7 Windows Server 2008 R2 Windows Server 2012
Target platforms	PC architecture (x86, x64)
.NET Framework	---
TwinCAT version	TwinCAT 2, TwinCAT 3
Minimum TwinCAT installation level	TwinCAT 2 PLC, NC-PTP TwinCAT 3 XAE, XAR
Required TwinCAT license	TS6100 TwinCAT OPC UA (for TwinCAT 2) TF6100 TC3 OPC UA (for TwinCAT 3)

OPC UA I/O Client

Technical data	Description
Operating system	Windows 7, 10 Windows CE 7 Windows Embedded 7 Windows Server 2008 R2 Windows Server 2012
Target platforms	PC architecture (x86, x64)
.NET Framework	4.6 (only for namespace browser)

Technical data	Description
TwinCAT version	TwinCAT 3.1 Build 4022.4
Minimum TwinCAT installation level	TwinCAT 3 XAE, XAR
Required TwinCAT license	TF6100 TC3 OPC UA

OPC UA Gateway

Technical data	Description
Operating system	Windows 7, 10 Windows Embedded Standard 2009 Windows Embedded 7 Windows Server 2008 R2 Windows Server 2012
Target platforms	PC architecture (x86, x64)
.NET Framework	---
TwinCAT version	---
Minimum TwinCAT installation level	---
Required TwinCAT license	---

OPC UA Configurator

OPC UA Configurator	Description
Operating system	Windows 7, 10 Windows Embedded Standard 2009 Windows Embedded 7 Windows Server 2008 R2 Windows Server 2012
Target platforms	PC architecture (x86, x64)
.NET Framework	4.5.2
TwinCAT version	TwinCAT 2 (standalone tool, up to setup version 3) TwinCAT 3 (Visual Studio integrated, from setup version 4)
Minimum TwinCAT installation level	TwinCAT 2 CP, PLC, NC-PTP (standalone tool, up to setup version 3) TwinCAT 3 XAE, XAR, ADS (standalone tool, up to setup version 3) TwinCAT 3 XAE (Visual Studio integrated, from setup version 4)
Required TwinCAT license	---

UPC UA Sample Client

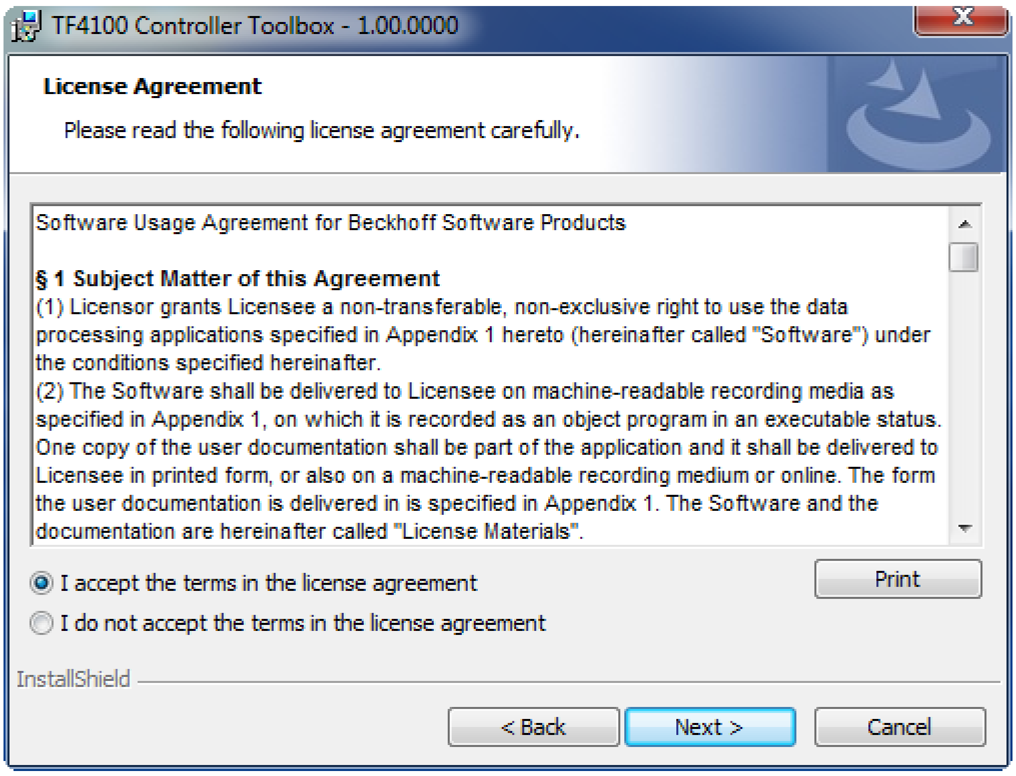
OPC UA Sample Client	Description
Operating system	Windows 7, 10 Windows Embedded 7 Windows Server 2008 R2 Windows Server 2012
Target platforms	PC architecture (x86, x64)
.NET Framework	4.6
TwinCAT version	---
Minimum TwinCAT installation level	---
Required TwinCAT license	---

3.2 Installation (TC3)

The following section describes how to install the TwinCAT 3 function TF6100 OPC UA for Windows-based operating systems.

✓ The TwinCAT 3 function setup file was downloaded from the Beckhoff website.

1. Run the setup file as administrator. To do this, select the command **Run As Admin** in the context menu of the file.
   ⇨ The installation dialog opens.
2. Accept the end user licensing agreement and click **Next**.



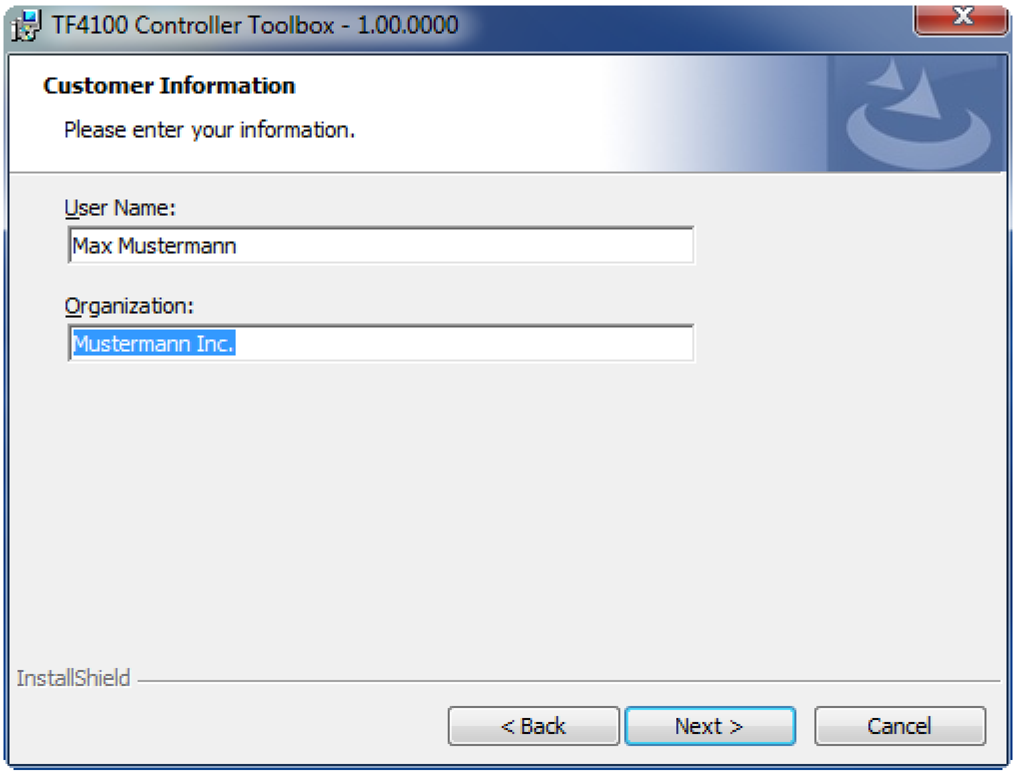
TF4100 Controller Toolbox - 1.00.0000
License Agreement
Please read the following license agreement carefully.

Software Usage Agreement for Beckhoff Software Products

§ 1 Subject Matter of this Agreement
(1) Licensor grants Licensee a non-transferable, non-exclusive right to use the data processing applications specified in Appendix 1 hereto (hereinafter called "Software") under the conditions specified hereinafter.
(2) The Software shall be delivered to Licensee on machine-readable recording media as specified in Appendix 1, on which it is recorded as an object program in an executable status. One copy of the user documentation shall be part of the application and it shall be delivered to Licensee in printed form, or also on a machine-readable recording medium or online. The form the user documentation is delivered in is specified in Appendix 1. The Software and the documentation are hereinafter called "License Materials".

[✓ I accept the terms in the license agreement] I do not accept the terms in the license agreement

3. Enter your user data.


TF4100 Controller Toolbox - 1.00.0000

Customer Information

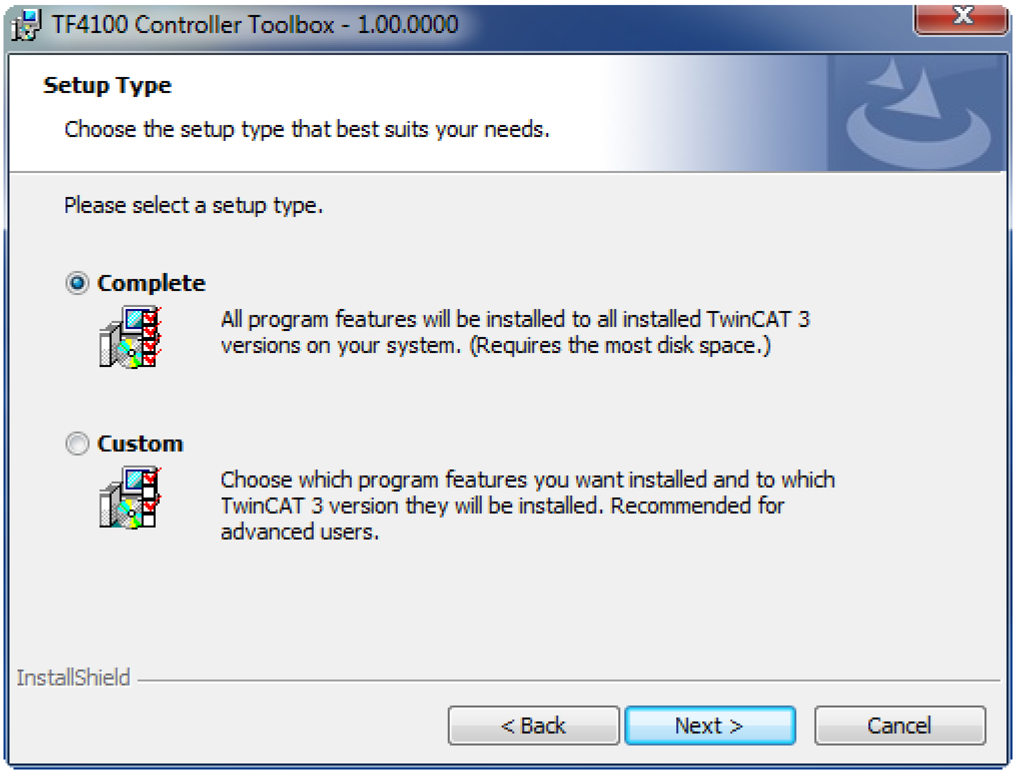
Please enter your information.

User Name:
Max Mustermann

Organization:
Mustermann Inc.

InstallShield

4. If you want to install the full version of the TwinCAT 3 function, select **Complete** as installation type. If you want to install the TwinCAT 3 function components separately, select **Custom**.


TF4100 Controller Toolbox - 1.00.0000

Setup Type

Choose the setup type that best suits your needs.

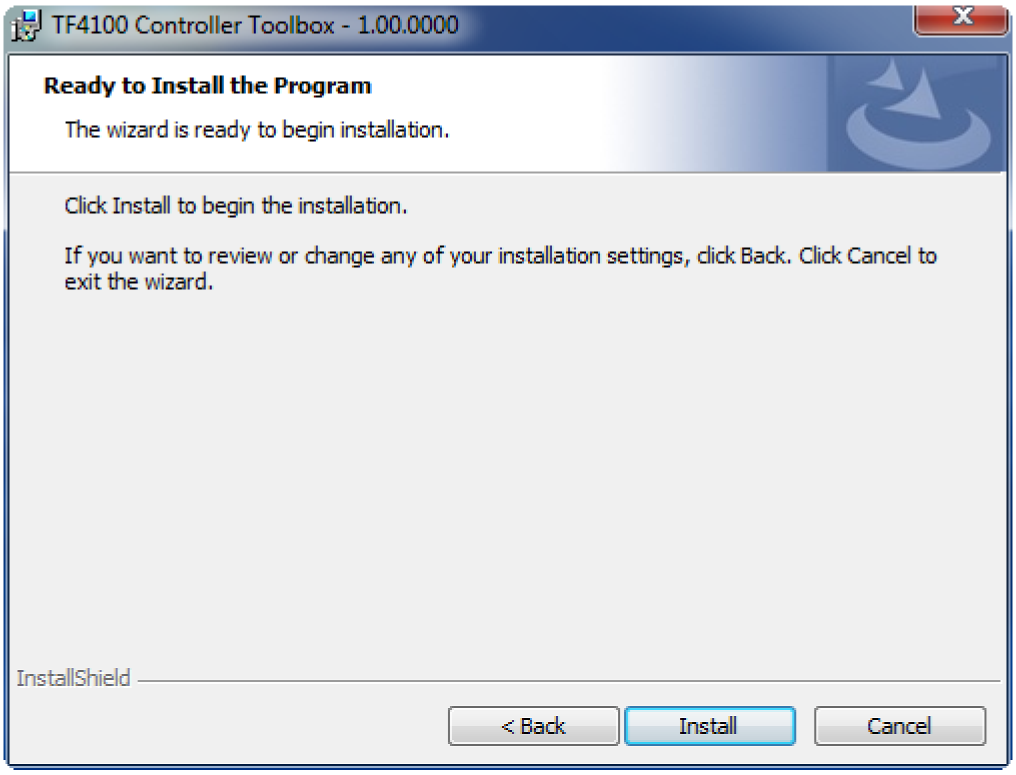
Please select a setup type.

[✓ Complete]
All program features will be installed to all installed TwinCAT 3 versions on your system. (Requires the most disk space.)

Custom
Choose which program features you want installed and to which TwinCAT 3 version they will be installed. Recommended for advanced users.

InstallShield

5. Select **Next**, then **Install** to start the installation.


TF4100 Controller Toolbox - 1.00.0000

Ready to Install the Program

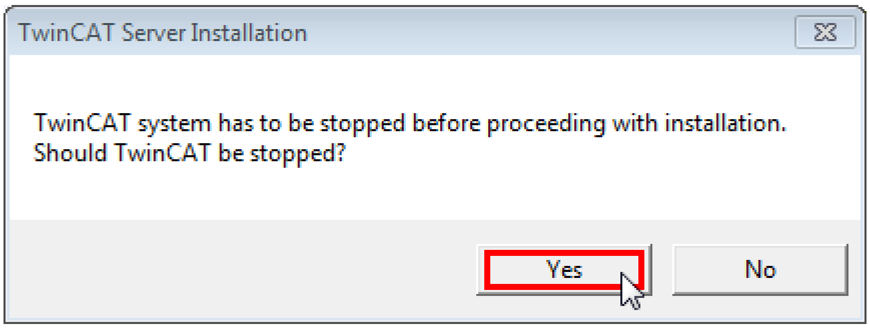
The wizard is ready to begin installation.

Click Install to begin the installation.

If you want to review or change any of your installation settings, click Back. Click Cancel to exit the wizard.

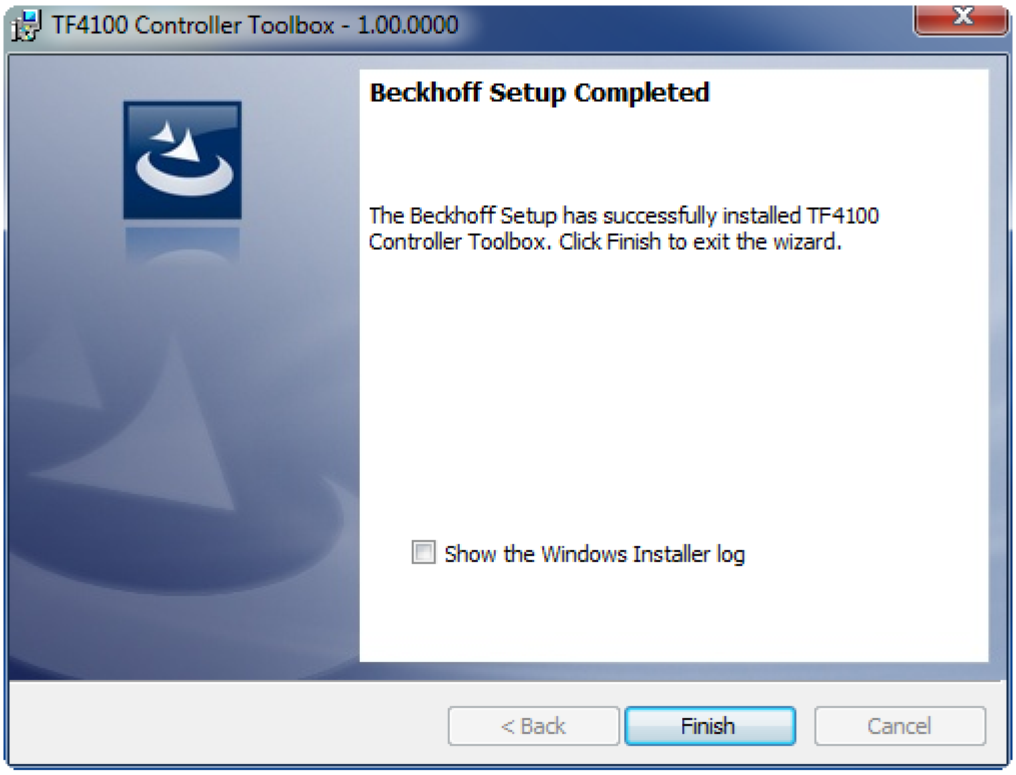
⇨ A dialog box informs you that the TwinCAT system must be stopped to proceed with the installation.

6. Confirm the dialog with **Yes**.


TwinCAT Server Installation

TwinCAT system has to be stopped before proceeding with installation.
Should TwinCAT be stopped?

7. Select **Finish** to exit the setup.


TF4100 Controller Toolbox - 1.00.0000
Show the Windows Installer log
< Back
Finish
Cancel

Beckhoff Setup Completed

The Beckhoff Setup has successfully installed TF4100 Controller Toolbox. Click Finish to exit the wizard.

⇨ The TwinCAT 3 function has been successfully installed and can be licensed (see Licensing (TC3) [▶ 35]

3.3 Installation (TC2)

The following section describes how to install the TwinCAT 2 Supplement TwinCAT OPC UA for Windows-based operating systems.

- Download and install the setup file [▶ 25]
- After the installation [▶ 28]

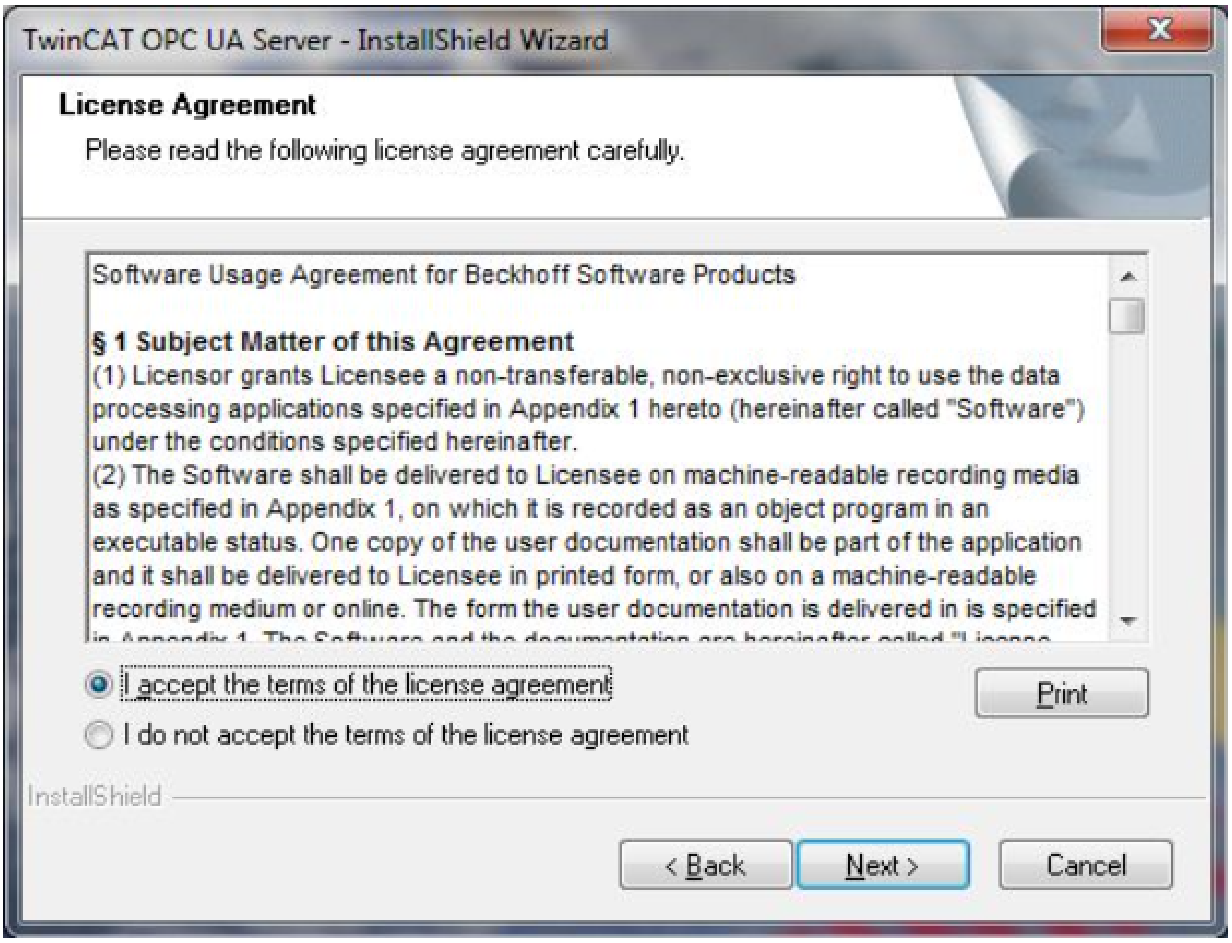
Download and install the setup file

Like many other TwinCAT add-on products, OPC UA is available as a 30-day demo version or full version for download from the Beckhoff website www.beckhoff.com.

✓ The setup file for the TwinCAT 2 Supplement has been downloaded from the Beckhoff homepage.

1. Run the downloaded setup file *TcOpcUaSvr.exe* as an administrator. To do this, select the command **Run As Admin** in the context menu of the file.
   ⇨ The installation dialog opens.
2. Select an installation language.

3. Click **Next** and accept the license agreement.



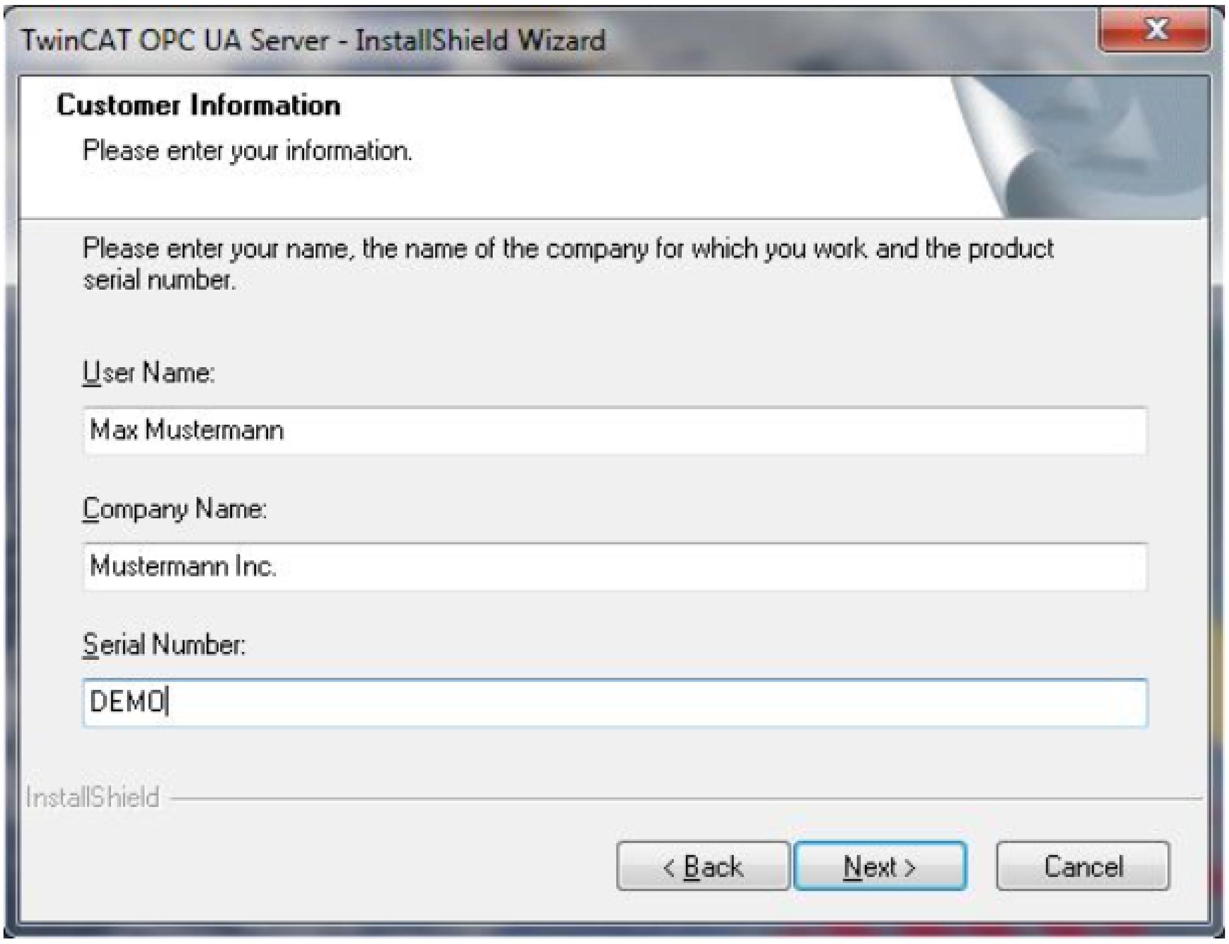
TwinCAT OPC UA Server - InstallShield Wizard
License Agreement
Please read the following license agreement carefully.

Software Usage Agreement for Beckhoff Software Products

§ 1 Subject Matter of this Agreement
(1) Licensor grants Licensee a non-transferable, non-exclusive right to use the data processing applications specified in Appendix 1 hereto (hereinafter called "Software") under the conditions specified hereinafter.
(2) The Software shall be delivered to Licensee on machine-readable recording media as specified in Appendix 1, on which it is recorded as an object program in an executable status. One copy of the user documentation shall be part of the application and it shall be delivered to Licensee in printed form, or also on a machine-readable recording medium or online. The form the user documentation is delivered in is specified

[✓ I accept the terms of the license agreement] I do not accept the terms of the license agreement

4. Enter your information. All fields are mandatory. If you want to install a 30-day demo, please enter "DEMO" as license key.



TwinCAT OPC UA Server - InstallShield Wizard

Customer Information

Please enter your information.

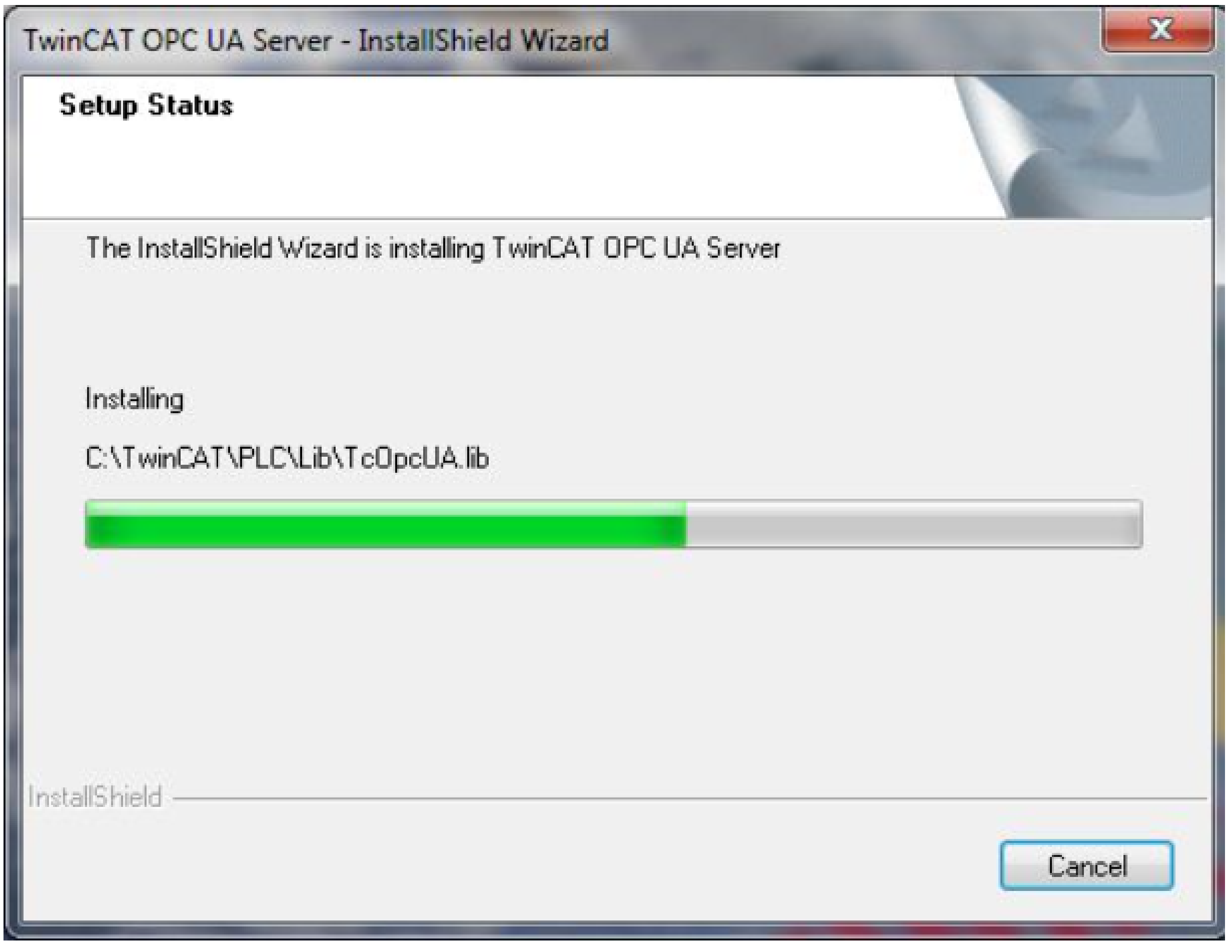
Please enter your name, the name of the company for which you work and the product serial number.

User Name: Max Mustermann

Company Name: Mustermann Inc.

Serial Number: DEMO

5. Click **Install** to start the installation.



The InstallShield Wizard is installing TwinCAT OPC UA Server

Installing

C:\TwinCAT\PLC\Lib\TcOpcUA.lib

6. Restart the computer to complete the installation.

After the installation

The TwinCAT add-on OPC UA is automatically configured during installation, and no further settings are required for using this product. Further steps may include:

- Establish an initial connection to the installed UA server and test its configuration with the OPC UA Sample Client. (See Sample Client [▶ 180])
- Personalize the UA server setup using the OPC UA Configurator. (See Configuration [▶ 187])
- Also, make sure your firewall opens TCP port 4840 because the OPC UC Server requires this port.

3.4 Installation Windows CE (TC3)

The following section describes how to install the TwinCAT 3 function TF6100 OPC UA on a Beckhoff Embedded PC with Windows CE.

- Download and install the setup file [▶ 29]
- Transfer the CAB file to the Windows CE device [▶ 29]
- Run the CAB file on the Windows CE device [▶ 29]

If an older version of TF6100 is already installed on the Windows CE device, it can be updated:

- Software upgrade [▶ 30]

Download and install the setup file

The executable file for Windows CE is part of the TF6100 OPC UA setup. This is made available on the Beckhoff website www.beckhoff.com and automatically contains all versions for Windows XP, Windows 7 and Windows CE (x86 and ARM).

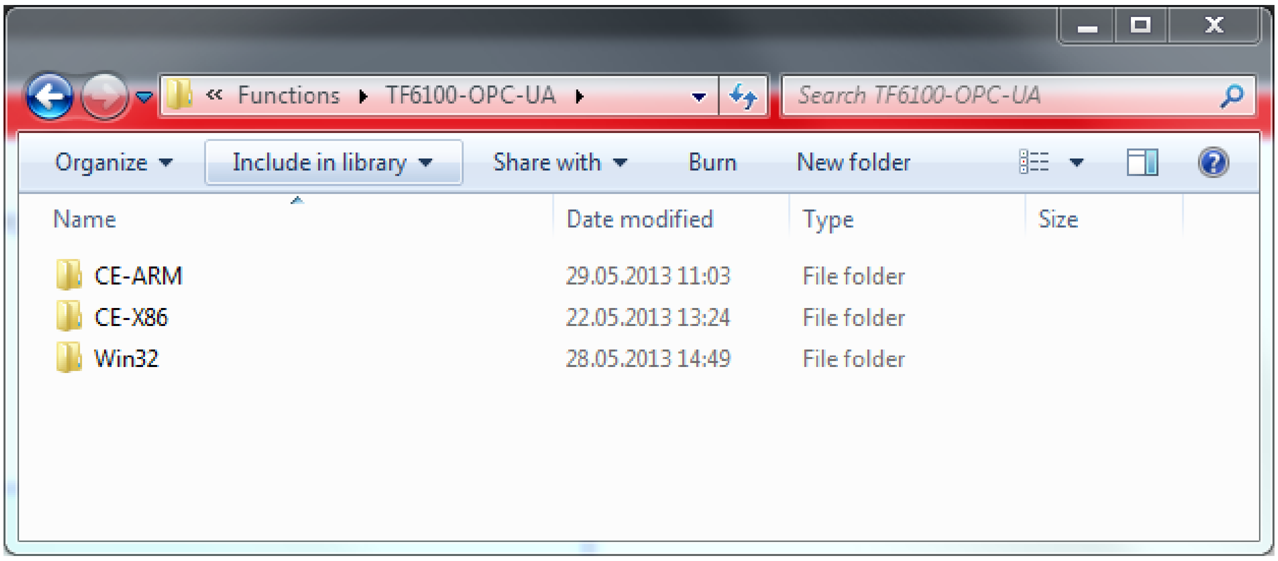
Download the setup and install the function as described in the section Installation.

After the installation, the installation folder contains three directories (one directory per hardware platform)

- **CE-ARM:** ARM-based Embedded PCs running Windows CE, e.g. CX8090, CX9020
- **CE-X86:** X86-based Embedded PCs running Windows CE, e.g. CX50xx, CX20x0
- **Win32:** Embedded PCs running Windows XP, Windows 7 or Windows Embedded Standard

The CE-ARM and CE-X86 directories contain the CAB files of the TwinCAT 3 function for Windows CE in relation to the respective hardware platform of the Windows CE device.

Sample: "TF6310" installation folder



Name	Date modified	Type	Size
CE-ARM	29.05.2013 11:03	File folder	
CE-X86	22.05.2013 13:24	File folder	
Win32	28.05.2013 14:49	File folder	

Transfer the CAB file to the Windows CE device

Transfer the corresponding executable file to the Windows CE device.

There are various options for transferring the executable file:

- via network shares
- via the integrated FTP server
- via ActiveSync
- via CF/SD cards

Further information can be found in the Beckhoff Information System in the "Operating Systems" documentation (Embedded PC > Operating Systems > CE).

Run the CAB file on the Windows CE device

After transferring the CAB file to the Windows CE device, double-click the file there. Confirm the installation dialog with **OK**. Then restart the Windows CE device.

After restarting the device, the files (Client and Server) are automatically loaded in the background and are available.

The software is installed in the following directory on the Windows CE device:
\Hard Disk\TwinCAT\Functions\TF6310-TCP-IP

Software upgrade

If an older TF6100 version is already installed on the Windows CE device, carry out the following steps on the Windows CE device to upgrade to a new version:

1. Open the CE Explorer by clicking **Start > Run** and entering "Explorer".
2. Navigate to *\Hard Disk\TwinCAT\Functions\Tf6100-OPC-UA\Server* or (in a second step) *\Hard Disk \TwinCAT\Functions\Tf6100-OPC-UA\Client*.
3. Rename the file*TcOpcUaServer.exe* or *TcOpcUaClient.exe*.
4. Restart the Windows CE device.
5. Transfer the new CAB file to the Windows CE device.
6. Run the CAB file on the Windows CE device and install the new version.
7. Delete the old (re-named) files.
8. Restart the Windows CE device.

⇨ The new version is active after the restart.

3.5 Installation Windows CE (TC2)

The following section describes how to install the TwinCAT 2 Supplement on a Beckhoff Embedded PC with Windows CE, e.g. CX1000, CX1020, CX9000, CX9001, CX9010, CX8090, CP62xx or CP69xx.

- Download and install the setup file [▶ 30]
- Transfer the setup file to a Windows CE device [▶ 33]
- Run the setup file on the Windows CE device [▶ 33]

Installation on small embedded platforms (CX9001, CX9010)

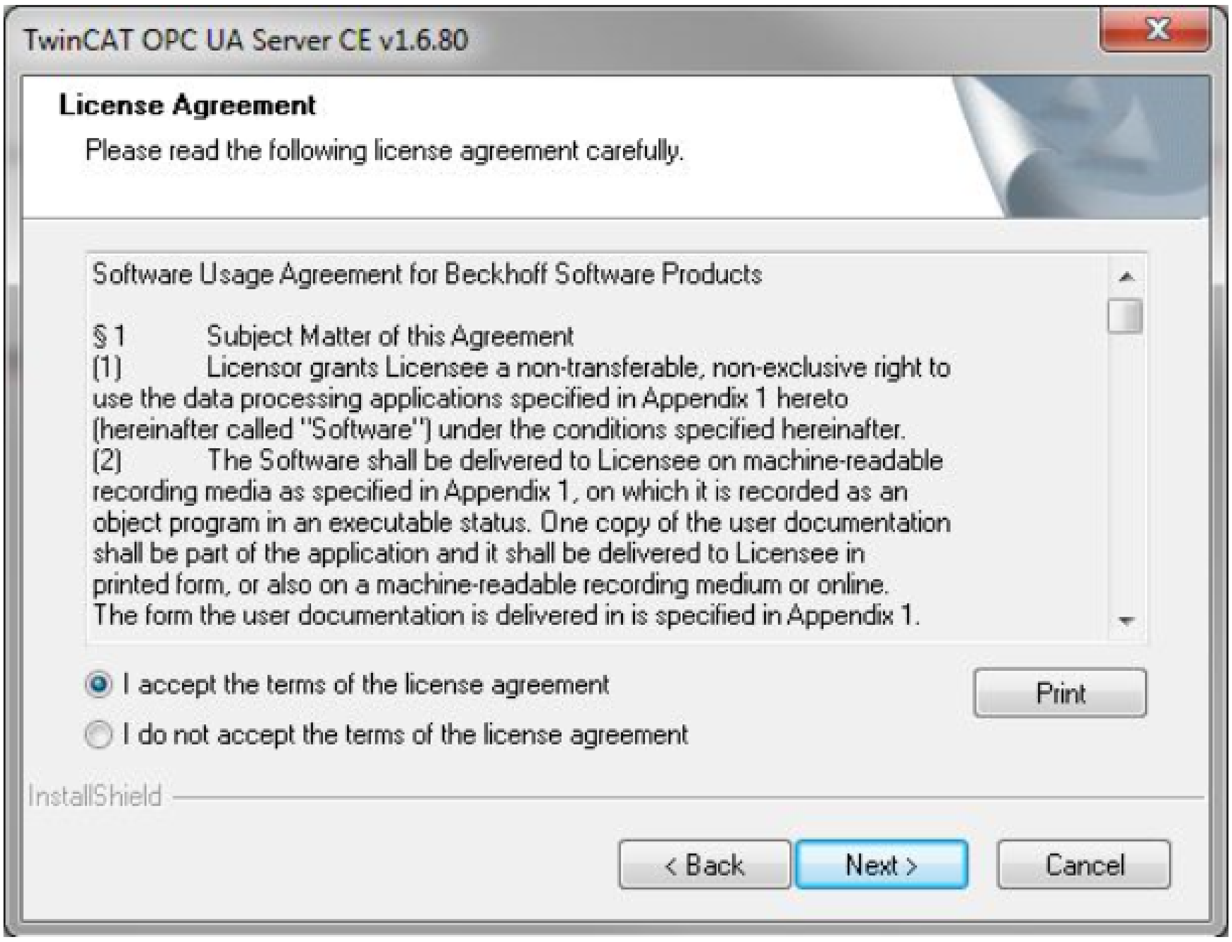
Very small embedded devices may require some additional manual steps to install the CAB file. These steps may include, for example, the deleting of unnecessary files from the memories of the devices so that there is sufficient space to install all the files of the application.

Download and install the setup file

Just like many other TwinCAT supplementary products, OPC UA for CE is available as a download on the Beckhoff homepage. To obtain the installation files for Windows CE, you must first install the downloaded setup file on a host computer. This can be any Windows CE-based system.

1. Double-click the downloaded file *TcOpcUaSvrCE.exe*.
2. Select an installation language.

3. Click **Next** and accept the license agreement.


TwinCAT OPC UA Server CE v1.6.80

License Agreement

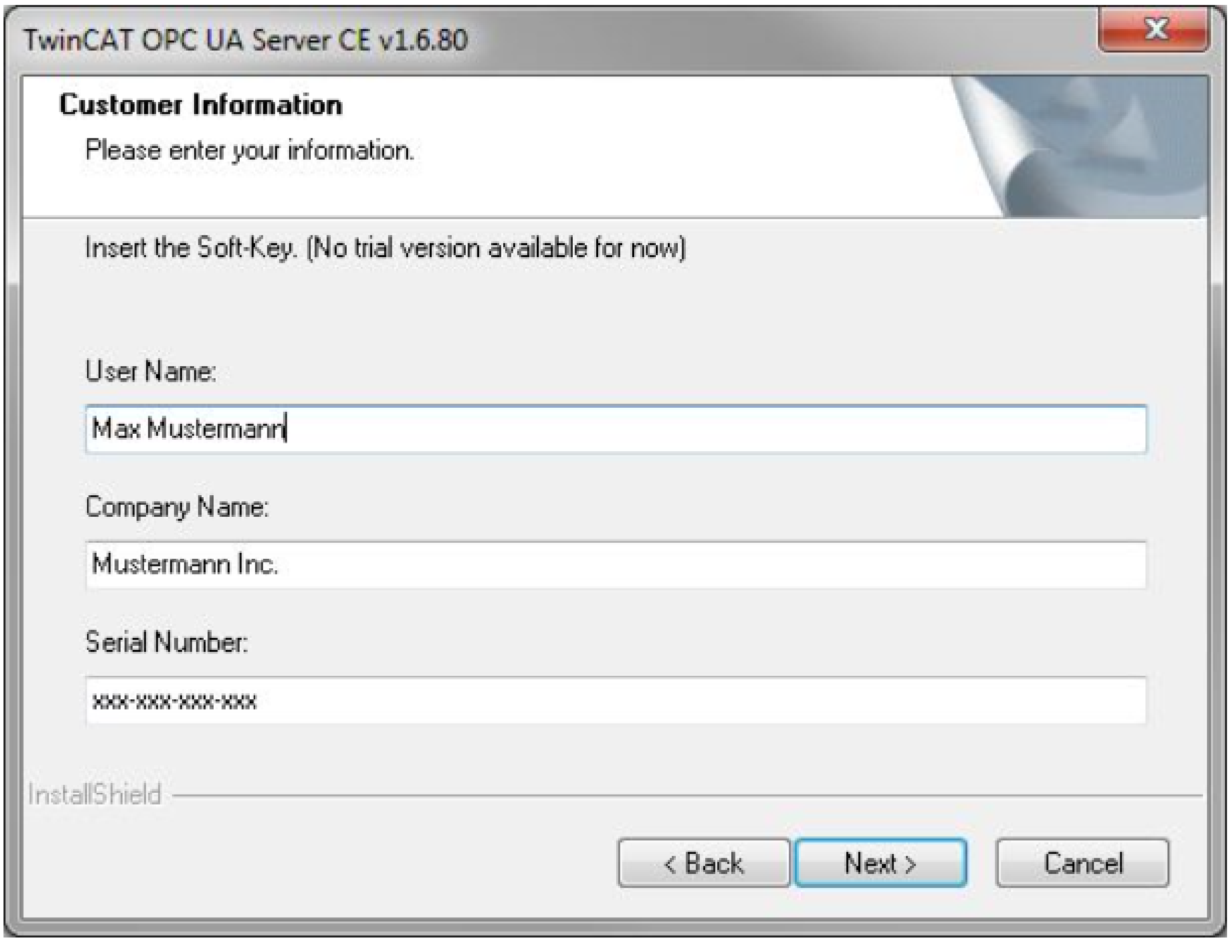
Please read the following license agreement carefully.

Software Usage Agreement for Beckhoff Software Products

§ 1 Subject Matter of this Agreement
(1) Licensor grants Licensee a non-transferable, non-exclusive right to use the data processing applications specified in Appendix 1 hereto (hereinafter called "Software") under the conditions specified hereinafter.
(2) The Software shall be delivered to Licensee on machine-readable recording media as specified in Appendix 1, on which it is recorded as an object program in an executable status. One copy of the user documentation shall be part of the application and it shall be delivered to Licensee in printed form, or also on a machine-readable recording medium or online. The form the user documentation is delivered in is specified in Appendix 1.

[✓ I accept the terms of the license agreement]
I do not accept the terms of the license agreement

4. Enter your information. All fields are mandatory. Note that OPC UA for CE is not currently available as a demo version. Therefore, you will need a valid license key to continue with the installation.



TwinCAT OPC UA Server CE v1.6.80

Customer Information

Please enter your information.

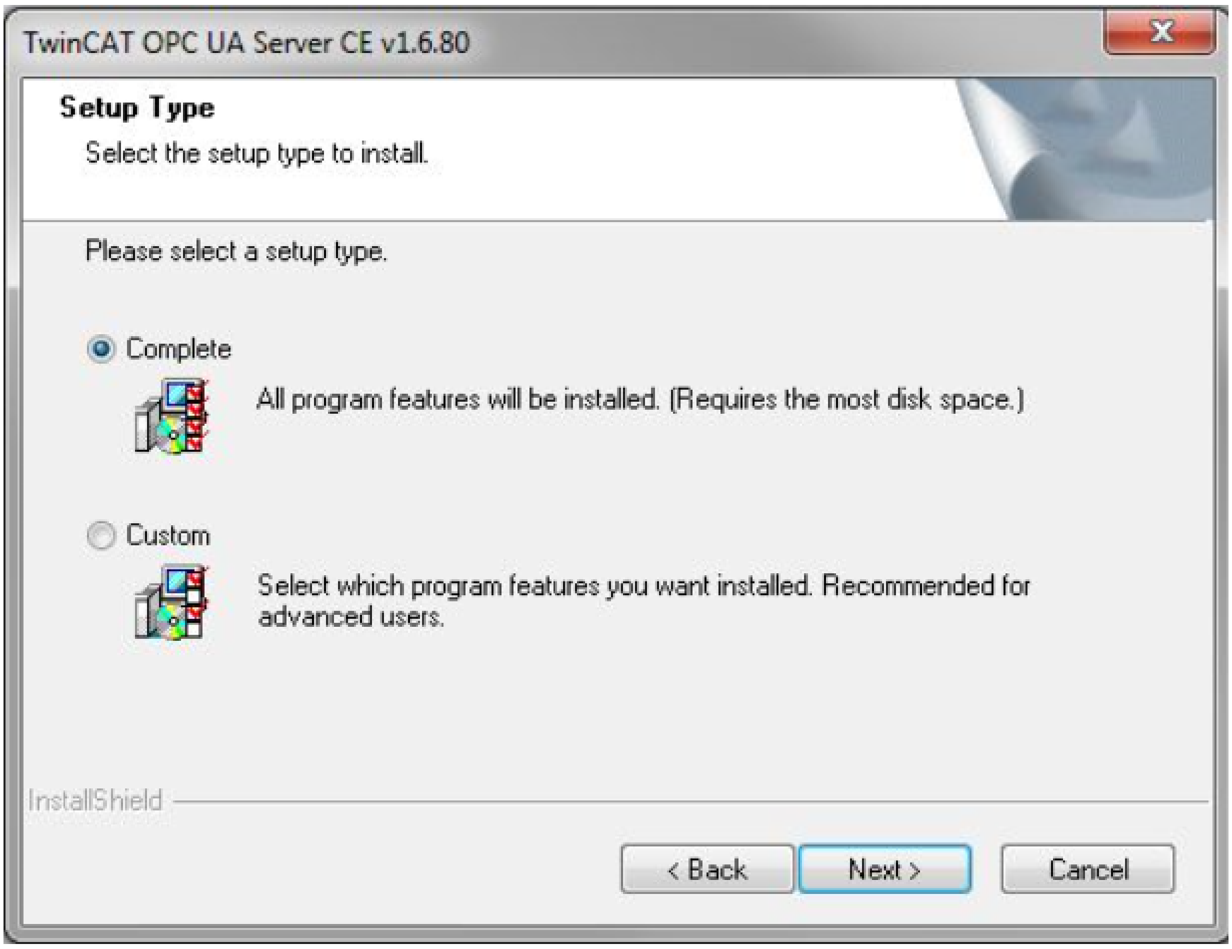
Insert the Soft-Key. (No trial version available for now)

User Name: Max Mustermann

Company Name: Mustermann Inc.

Serial Number: xxx-xxx-xxx-xxx

5. Select **Full** as the installation type and click on **Continue**



TwinCAT OPC UA Server CE v1.6.80

Setup Type

Select the setup type to install.

Please select a setup type.

[✓ Complete]

All program features will be installed. (Requires the most disk space.)

Custom

Select which program features you want installed. Recommended for advanced users.

6. Click **Install** to start the installation.

⇨ After installation you will find the setup files for Windows CE in the directory ...\TwinCAT\CE. This directory contains setup files for the following CE platforms:

- TwinCAT OPC UA Client CE\I586: OPC UA Client (PLC library) for x86-based CPUs (such as CX10xx, CP62xx, C69xx,...)
- TwinCAT OPC UA Client CE\ARMV4I: OPC UA Client (PLC library) for ARM-based CPUs (such as CX9001, CX9010, CP6608, ...)
- TwinCAT OPC UA Server CE\I586: OPC UA Server for x86-based CPUs (such as CX10xx, CP62xx, C69xx, ...)
- TwinCAT OPC UA Server CE\ARMV4I: OPC UA Server for ARM-based CPUs (such as CX9001, CX9010, CP6608, ...)

Transfer the setup file to the Windows CE device

Transfer the corresponding executable file to the Windows CE device. There are various options for transferring the setup file:

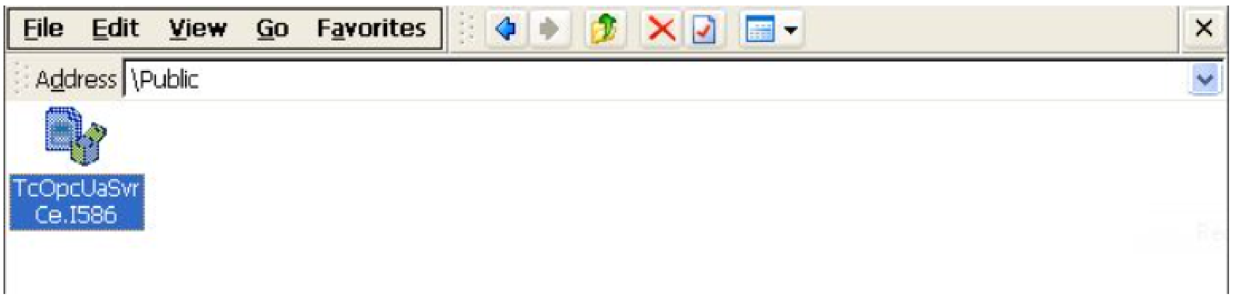
- from a shared folder
- via the integrated FTP server
- via ActiveSync
- via a CF card

Further information can be found in the Beckhoff Information System in the "Operating systems" documentation (Embedded PC > Operating systems > CE)

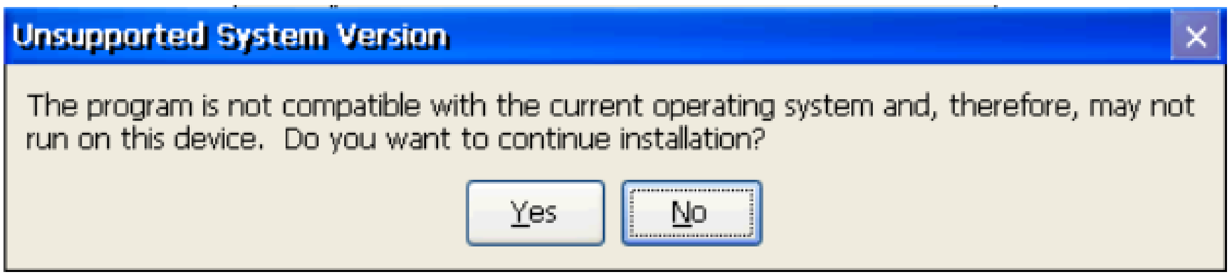
Run the setup file on the Windows CE device

Execute the transferred setup file *TcOpcUaSvrCe.xxxx.CAB* on the CE device:

1. Navigate to the directory to which you have transferred the setup file.

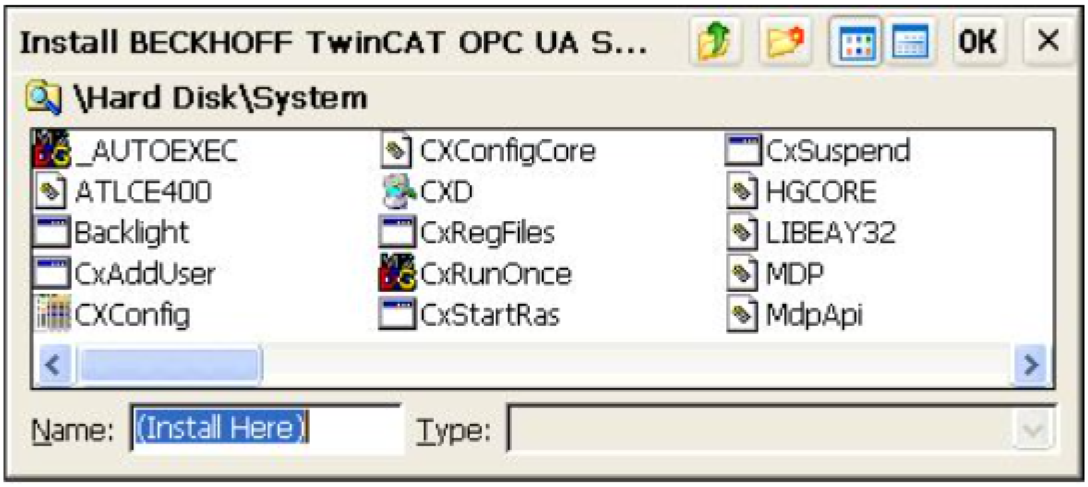


2. Double-click the CAB file. If a message dialog box appears that says this program is not compatible with the current operating system, make sure you are using the correct CAB file (ARM, I586) for your IPC/ Embedded PC.
3. If you are sure that the CAB file matches the Embedded PC/IPC, confirm this message dialog with **Yes**.


Unsupported System Version

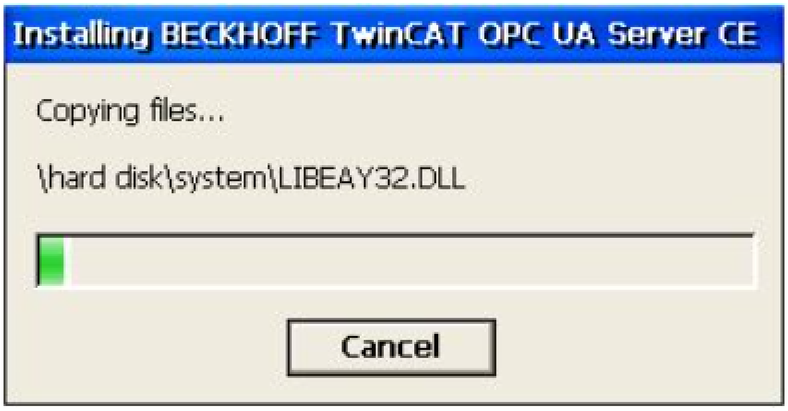
The program is not compatible with the current operating system and, therefore, may not run on this device. Do you want to continue installation?

4. Select *Hard Disk**System*\ as destination directory


Install BECKHOFF TwinCAT OPC UA S...

\Hard Disk\System

5. Click **OK** to start the installation.


Installing BECKHOFF TwinCAT OPC UA Server CE

Copying files...

\hard disk\system\LIBEAY32.DLL

⇨ After installation, the setup file is automatically deleted.

Once you have installed the OPC UA Server, this component will be available after restarting your Windows CE device.

3.6 Licensing (TC3)

The TwinCAT 3 Function can be activated as a full version or as a 7-day test version. Both license types can be activated via the TwinCAT 3 development environment (XAE).

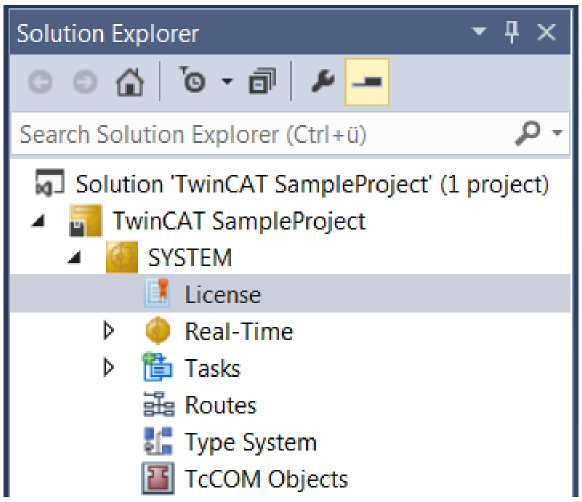
The licensing of a TwinCAT 3 Function is described below. The description is divided into the following sections:

- Licensing a 7-day test version [▶ 35]
- Licensing a full version [▶ 36]

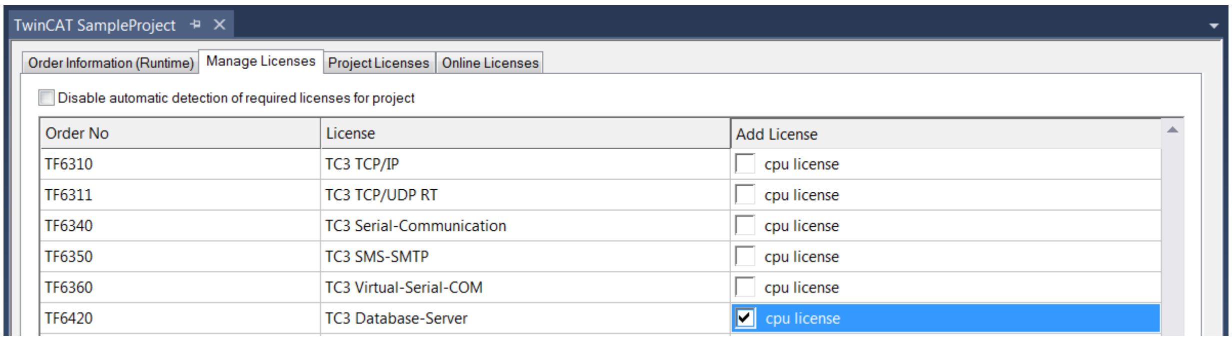
Further information on TwinCAT 3 licensing can be found in the “Licensing” documentation in the Beckhoff Information System (TwinCAT 3 > Licensing).

Licensing a 7-day test version

1. Start the TwinCAT 3 development environment (XAE).
2. Open an existing TwinCAT 3 project or create a new project.
3. If you want to activate the license for a remote device, set the desired target system. To do this, select the target system from the **Choose Target System** drop-down list in the toolbar.
   ⇨ The licensing settings always refer to the selected target system. When the project is activated on the target system, the corresponding TwinCAT 3 licenses are automatically copied to this system.
4. In the **Solution Explorer**, double-click **License** in the **SYSTEM** subtree.



   ⇨ The TwinCAT 3 license manager opens.
5. Open the **Manage Licenses** tab. In the **Add License** column, check the check box for the license you want to add to your project (e.g. “TF6420: TC3 Database Server“).

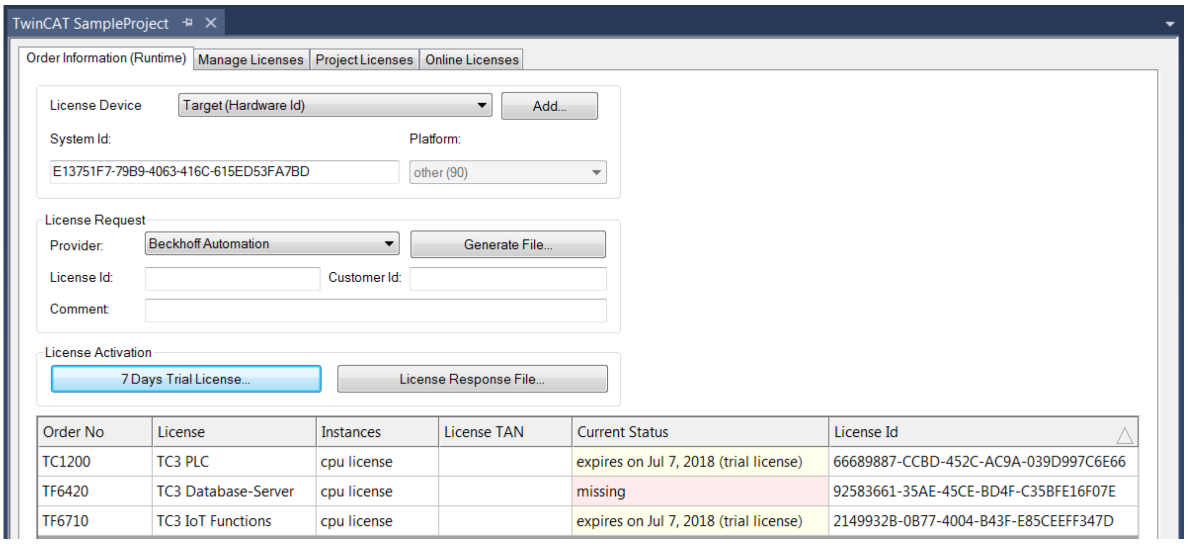


☐ Disable automatic detection of required licenses for project

Order No	License	Add License
TF6310	TC3 TCP/IP	☐ cpu license
TF6311	TC3 TCP/UDP RT	☐ cpu license
TF6340	TC3 Serial-Communication	☐ cpu license
TF6350	TC3 SMS-SMTP	☐ cpu license
TF6360	TC3 Virtual-Serial-COM	☐ cpu license
TF6420	TC3 Database-Server	☑ cpu license

6. Open the **Order Information (Runtime)** tab.

⇨ In the tabular overview of licenses, the previously selected license is displayed with the status "missing".



License Device: Target (Hardware Id)

System Id: E13751F7-79B9-4063-416C-615ED53FA7BD

Platform: other (90)

License Request

Provider: Beckhoff Automation

License Id:

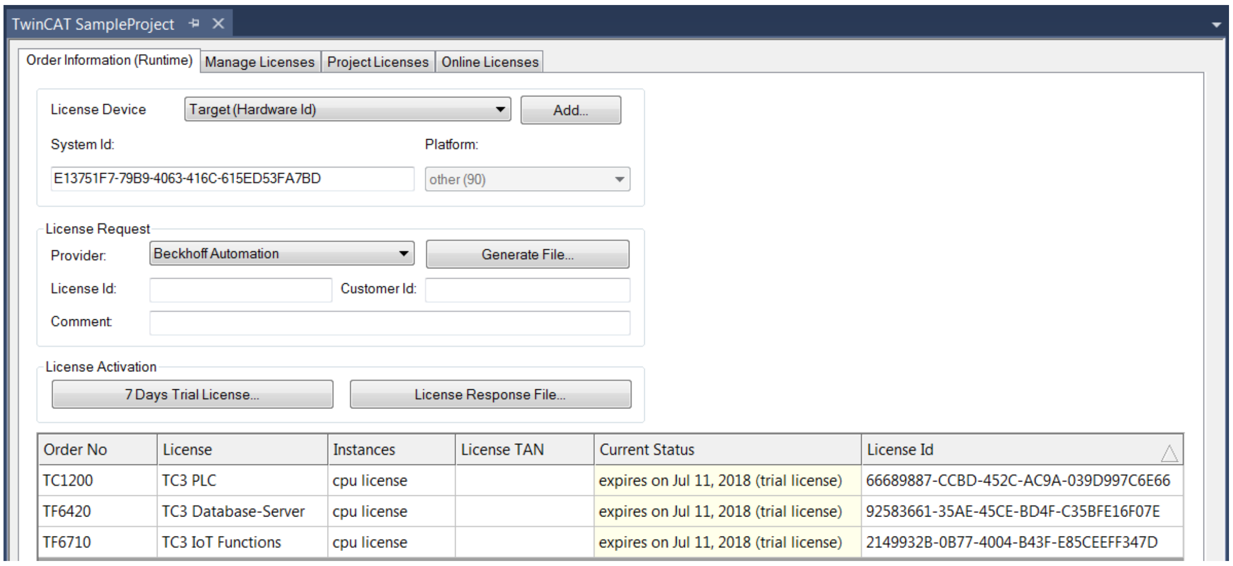
Customer Id:

Comment:

License Activation

Order No	License	Instances	License TAN	Current Status	License Id
TC1200	TC3 PLC	cpu license		expires on Jul 7, 2018 (trial license)	66689887-CCBD-452C-AC9A-039D997C6E66
TF6420	TC3 Database-Server	cpu license		missing	92583661-35AE-45CE-BD4F-C35BFE16F07E
TF6710	TC3 IoT Functions	cpu license		expires on Jul 7, 2018 (trial license)	2149932B-0B77-4004-B43F-E85CEEFF347D

7. Click **7-Day Trial License...** to activate the 7-day trial license.



License Device: Target (Hardware Id)

System Id: E13751F7-79B9-4063-416C-615ED53FA7BD

Platform: other (90)

License Request

Provider: Beckhoff Automation

License Id:

Customer Id:

Comment:

License Activation

Order No	License	Instances	License TAN	Current Status	License Id
TC1200	TC3 PLC	cpu license		expires on Jul 11, 2018 (trial license)	66689887-CCBD-452C-AC9A-039D997C6E66
TF6420	TC3 Database-Server	cpu license		expires on Jul 11, 2018 (trial license)	92583661-35AE-45CE-BD4F-C35BFE16F07E
TF6710	TC3 IoT Functions	cpu license		expires on Jul 11, 2018 (trial license)	2149932B-0B77-4004-B43F-E85CEEFF347D

⇨ A dialog box opens, prompting you to enter the security code displayed in the dialog.

8. Enter the code exactly as it appears, confirm it and acknowledge the subsequent dialog indicating successful activation.

   ⇨ In the tabular overview of licenses, the license status now indicates the expiration date of the license.

9. Restart the TwinCAT system.

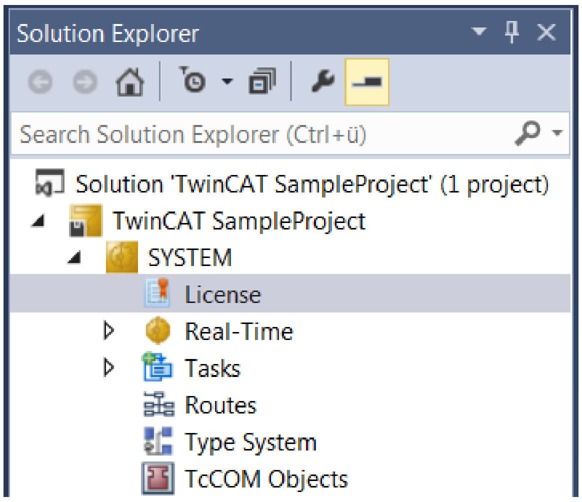
⇨ The 7-day trial version is enabled.

Licensing a full version

1. Start the TwinCAT 3 development environment (XAE).
2. Open an existing TwinCAT 3 project or create a new project.
3. If you want to activate the license for a remote device, set the desired target system. To do this, select the target system from the **Choose Target System** drop-down list in the toolbar.

   ⇨ The licensing settings always refer to the selected target system. When the project is activated on the target system, the corresponding TwinCAT 3 licenses are automatically copied to this system.

4. In the **Solution Explorer**, double-click **License** in the **SYSTEM** subtree.



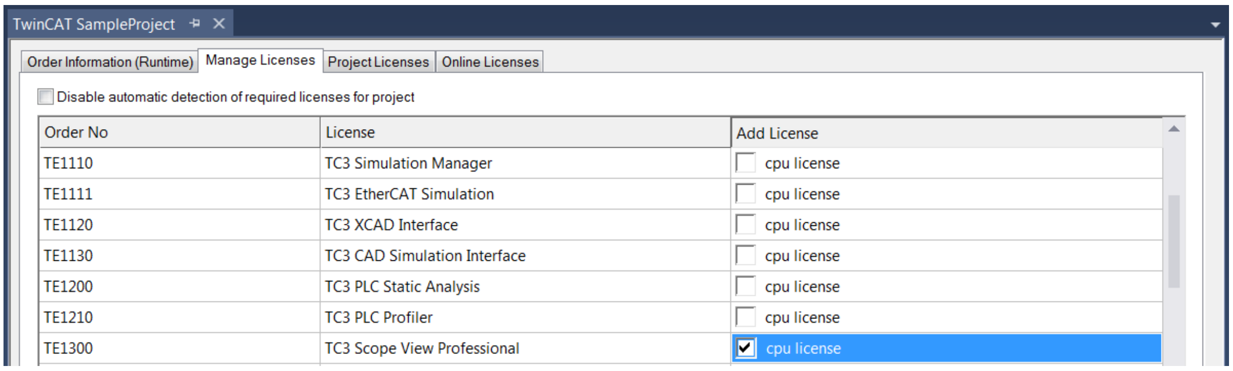
Solution Explorer

Search Solution Explorer (Ctrl+ü)

- Solution 'TwinCAT SampleProject' (1 project)
  - TwinCAT SampleProject
    - SYSTEM
      - License
      - Real-Time
      - Tasks
      - Routes
      - Type System
      - TcCOM Objects

⇨ The TwinCAT 3 license manager opens.

5. Open the **Manage Licenses** tab. In the **Add License** column, check the check box for the license you want to add to your project (e.g. “TE1300: TC3 Scope View Professional”).



TwinCAT SampleProject

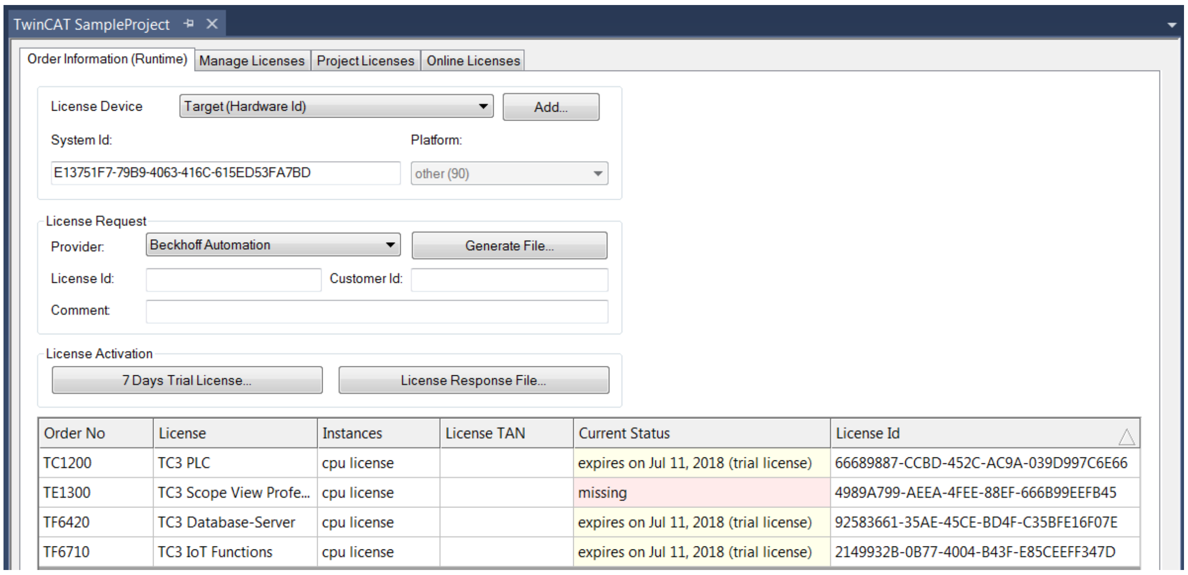
Order Information (Runtime) | Manage Licenses | Project Licenses | Online Licenses

☐ Disable automatic detection of required licenses for project

Order No	License	Add License
TE1110	TC3 Simulation Manager	☐ cpu license
TE1111	TC3 EtherCAT Simulation	☐ cpu license
TE1120	TC3 XCAD Interface	☐ cpu license
TE1130	TC3 CAD Simulation Interface	☐ cpu license
TE1200	TC3 PLC Static Analysis	☐ cpu license
TE1210	TC3 PLC Profiler	☐ cpu license
TE1300	TC3 Scope View Professional	☑ cpu license

6. Open the **Order Information** tab.

⇨ In the tabular overview of licenses, the previously selected license is displayed with the status “missing”**.**



TwinCAT SampleProject

Order Information (Runtime) | Manage Licenses | Project Licenses | Online Licenses

License Device: Target (Hardware Id) [Add...]

System Id: E13751F7-79B9-4063-416C-615ED53FA7BD

Platform: other (90)

License Request

Provider: Beckhoff Automation [Generate File...]

License Id:

Customer Id:

Comment:

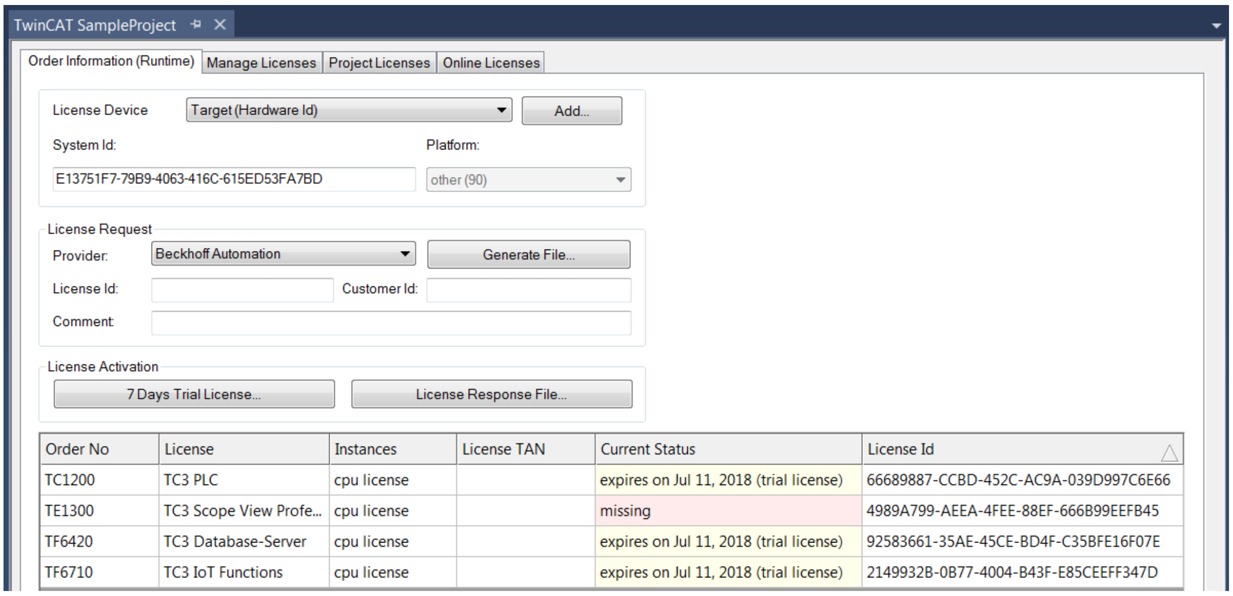
License Activation

[7 Days Trial License...] [License Response File...]

Order No	License	Instances	License TAN	Current Status	License Id
TC1200	TC3 PLC	cpu license		expires on Jul 11, 2018 (trial license)	66689887-CCBD-452C-AC9A-039D997C6E66
TE1300	TC3 Scope View Profe...	cpu license		missing	4989A799-AEEA-4FEE-88EF-666B99EEFB45
TF6420	TC3 Database-Server	cpu license		expires on Jul 11, 2018 (trial license)	92583661-35AE-45CE-BD4F-C35BFE16F07E
TF6710	TC3 IoT Functions	cpu license		expires on Jul 11, 2018 (trial license)	2149932B-0B77-4004-B43F-E85CEEFF347D

A TwinCAT 3 license is generally linked to two indices describing the platform to be licensed:
System ID: Uniquely identifies the device
Platform level: Defines the performance of the device
The corresponding **System Id** and **Platform** fields cannot be changed.

7. Enter the order number (**License Id**) for the license to be activated and optionally a separate order number (**Customer Id**), plus an optional comment for your own purposes (**Comment**). If you do not know your Beckhoff order number, please contact your Beckhoff sales contact.



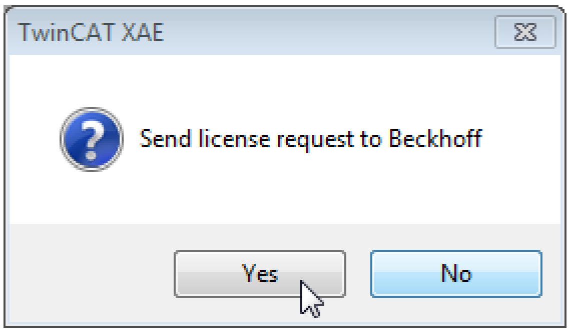
License Device: Target (Hardware Id)
System Id: E13751F7-79B9-4063-416C-615ED53FA7BD
Platform: other (90)

License Request
Provider: Beckhoff Automation
License Id:
Customer Id:
Comment:

License Activation

Order No	License	Instances	License TAN	Current Status	License Id
TC1200	TC3 PLC	cpu license		expires on Jul 11, 2018 (trial license)	66689887-CCBD-452C-AC9A-039D997C6E66
TE1300	TC3 Scope View Profe...	cpu license		missing	4989A799-AEEA-4FEE-88EF-666B99EEFB45
TF6420	TC3 Database-Server	cpu license		expires on Jul 11, 2018 (trial license)	92583661-35AE-45CE-BD4F-C35BFE16F07E
TF6710	TC3 IoT Functions	cpu license		expires on Jul 11, 2018 (trial license)	2149932B-0B77-4004-B43F-E85CEEFF347D

8. Click the **Generate File**... button to create a License Request File for the listed missing license.
   ⇨ A window opens, in which you can specify where the License Request File is to be stored. (We recommend accepting the default settings.)
9. Select a location and click **Save**.
   ⇨ A prompt appears asking whether you want to send the License Request File to the Beckhoff license server for verification:

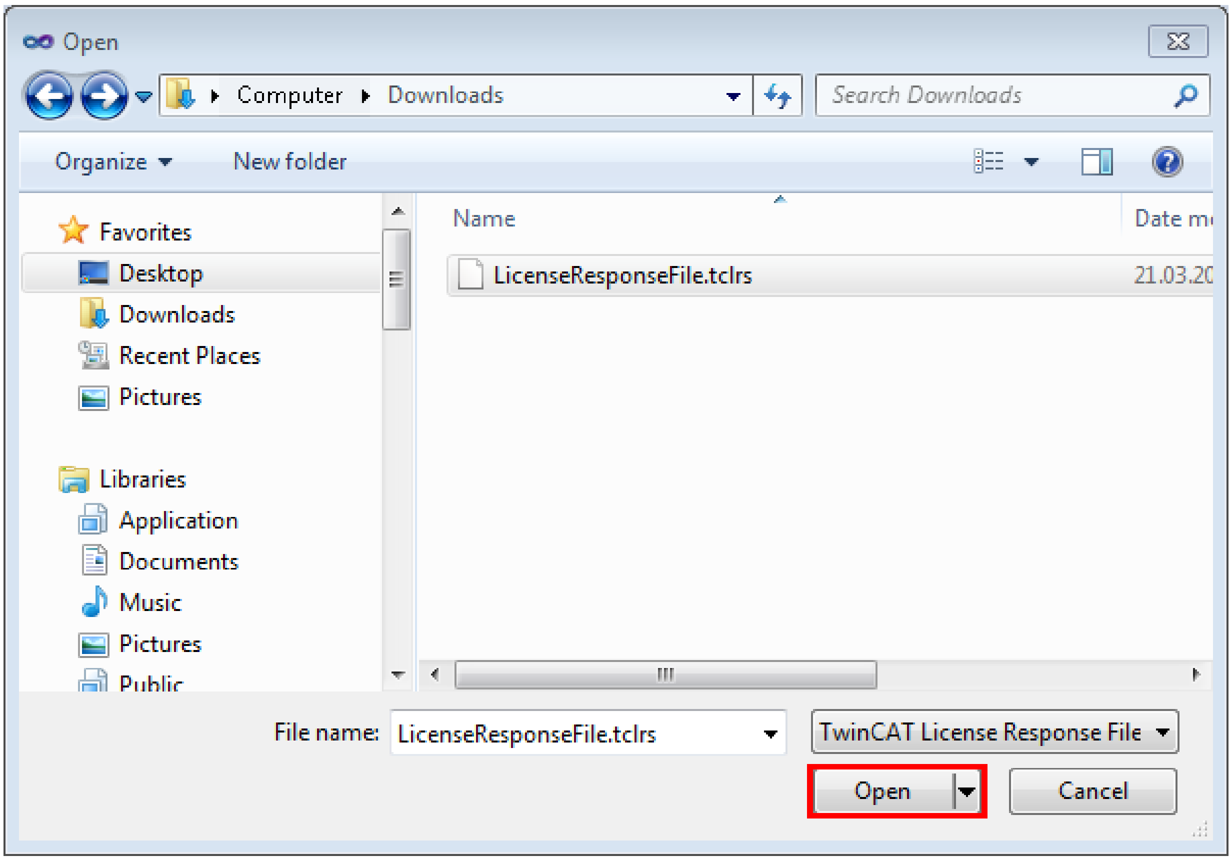


Send license request to Beckhoff

- Click **Yes** to send the License Request File. A prerequisite is that an email program is installed on your computer and that your computer is connected to the internet. When you click **Yes**, the system automatically generates a draft email containing the License Request File with all the necessary information.
- Click **No** if your computer does not have an email program installed on it or is not connected to the internet. Copy the License Request File onto a data storage device (e.g. a USB stick) and send the file from a computer with internet access and an email program to the Beckhoff license server (tclicense@beckhoff.com) by email.

10. Send the License Request File.
    ⇨ The License Request File is sent to the Beckhoff license server. After receiving the email, the server compares your license request with the specified order number and returns a License Response File by email. The Beckhoff license server returns the License Response File to the same email address from which the License Request File was sent. The License Response File differs from the License Request File only by a signature that documents the validity of the license file content. You can view the contents of the License Response File with an editor suitable for XML files (e.g. “XML Notepad”). The contents of the License Response File must not be changed, otherwise the license file becomes invalid.
11. Save the License Response File.

12. To import the license file and activate the license, click **License Response File...** in the **Order Information** tab.
13. Select the License Response File in your file directory and confirm the dialog.



⇨ The License Response File is imported and the license it contains is activated.
Existing demo licenses will be removed.

14. Restart the TwinCAT system.

⇨ The license becomes active when TwinCAT is restarted. The product can be used as a full version. During the TwinCAT restart the license file is automatically copied to the directory ...*\TwinCAT\3.1\Target \License* on the respective target system.

3.7 Licensing (TC2)

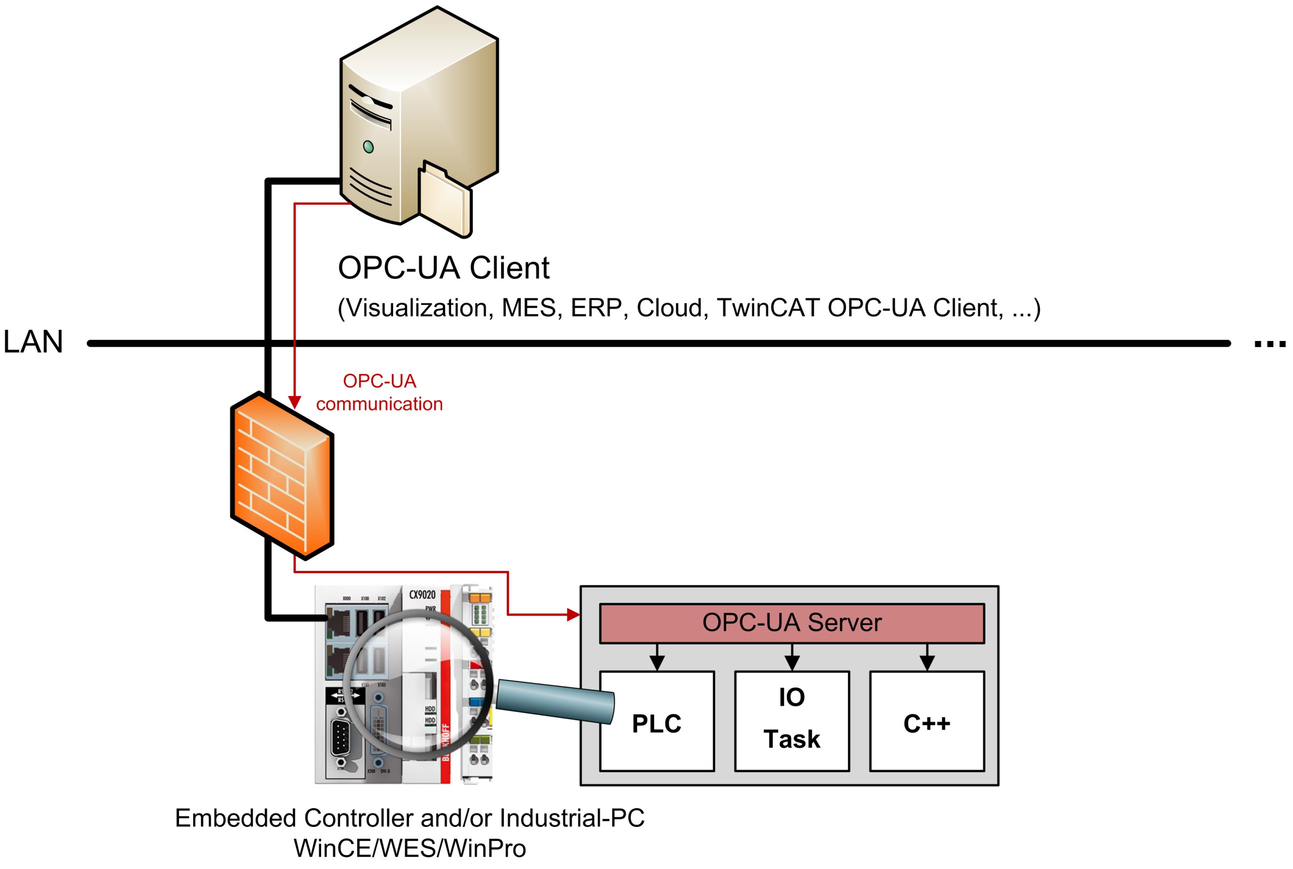
The licensing of the TwinCAT OPC UA for TwinCAT 2 takes place during the installation [▸ 25] by entering a license key.

4 Technical introduction

4.1 Server

4.1.1 Overview

The TwinCAT OPC UA Server offers a standardized communication interface for accessing symbol values from the TwinCAT PLC and C++ runtime or I/O task.


OPC-UA Client
(Visualization, MES, ERP, Cloud, TwinCAT OPC-UA Client, ...)
LAN
...
OPC-UA communication
OPC-UA Server
PLC
IO Task
C++
Embedded Controller and/or Industrial-PC
WinCE/WES/WinPro

This part of the documentation describes the basic configuration of the TwinCAT OPC UA Server.

Topical areas:

- OPC UA Configurator: Graphical user interface for configuring the OPC UA Server
- OPC UA DA (Data Access): Access to TwinCAT PLC and IO task values via OPC UA
- OPC UA HA (Historical Access): Storage of historical data, e.g. in local RAM, in a file or in a database
- OPC UA AC (Alarms and Conditions): Condition Monitoring functions for PLC variables
- OPC UA Security: Security mechanisms that are integrated in OPC UA

See also: Quick start [▸ 41]

Advanced configuration

Additional configuration parameters may be necessary depending on the type of operating environment.

Extended topical areas:

- Configuration file [▸ 126]: Describes the configuration file *ServerConfig.xml* of the OPC UA Server. This file should usually be edited via the OPC UA Configurator (see topics above).

- Configuration of PLC variables [▶ 71]: Provides an overview of all configurable options (via PLC comments) on a PLC variable.
- Other ADS devices [▶ 127]: Describes how further TwinCAT devices can be integrated in the namespace of the OPC UA Server.
- Several UA Servers [▶ 129]: Describes how to start several OPC UA Server instances on one device.
- UA over NAT [▶ 131]: Describes how OPC UA communication can be established via a NAT device, e.g. a firewall or a router.
- Changing the handling of arrays in the OPC UA namespace [▶ 71]: Describes two presentation options for PLC arrays in the OPC UA namespace.

4.1.2 Quick start

This quick start assumes that the OPC UA Server and the PLC runtime are installed on the same system.

Following successful installation and licensing, perform the following steps in order to establish an initial connection to a PLC runtime via the OPC UA Server:

- Step 1: Configure PLC variables for the OPC UA access [▶ 41]
- Step 2: Configure the OPC UA namespace [▶ 41]
- Step 3: Establish a connection to the OPC UA Server [▶ 42]

Step 1: Configure PLC variables for the OPC UA access

Open an existing PLC project and insert the following comment before the selected variables.

TwinCAT 3 (with TMC import):

```
{attribute 'OPC.UA.DA' := '1'}
bVariable : BOOL;
```

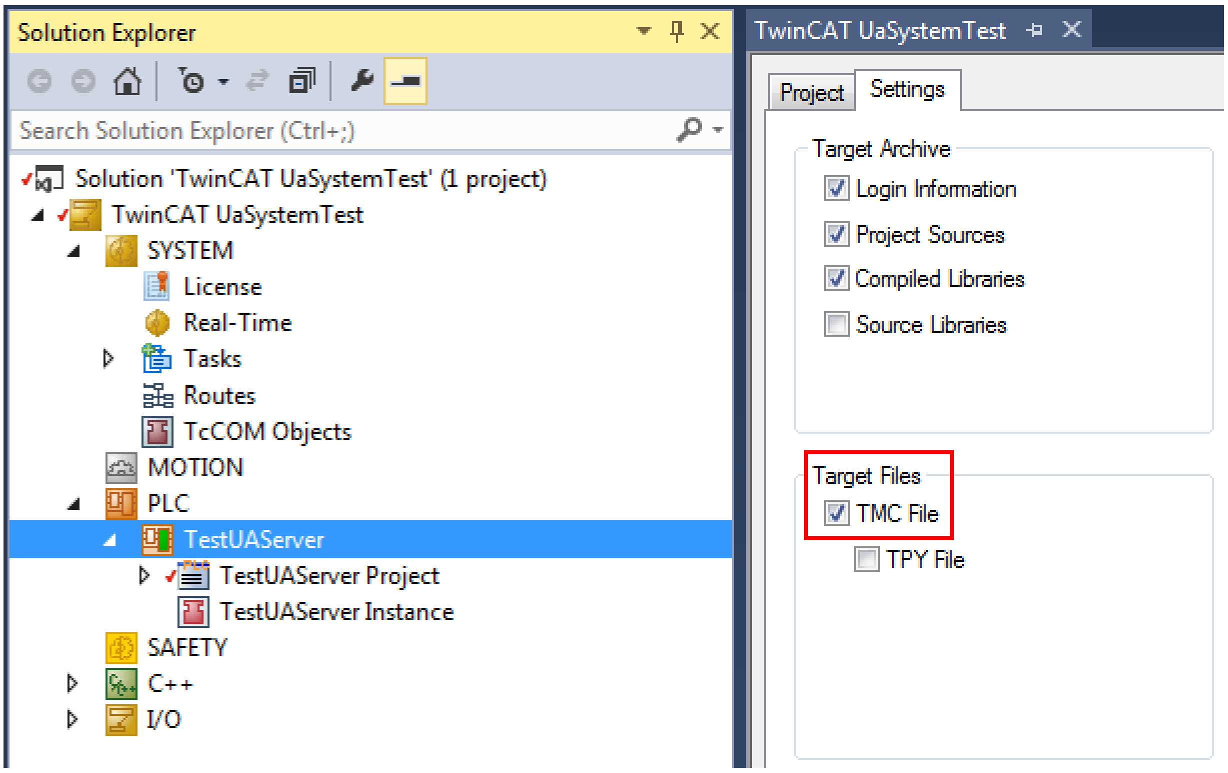
TwinCAT 2 (with TPY import):

```
bVariable : BOOL; (*~ (OPC:1:some description) *)
```

Step 2: Configure the OPC UA namespace

By default, the OPC UA Server connects to the first PLC runtime on the local system (local = the same system on which the OPC UA Server is installed) and uses its corresponding symbol file (TMC file) for configuring the OPC UA namespace.

To ensure that the symbol file is automatically transferred to the PLC runtime, activate the download of the symbol file in the settings of the PLC project.



Solution Explorer

Search Solution Explorer (Ctrl+;)

- Solution 'TwinCAT UaSystemTest' (1 project)
  - TwinCAT UaSystemTest
    - SYSTEM
      - License
      - Real-Time
      - Tasks
      - Routes
      - TcCOM Objects
    - MOTION
    - PLC
      - TestUAServer
        - TestUAServer Project
        - TestUAServer Instance
    - SAFETY
    - C++
    - I/O

TwinCAT UaSystemTest

Project | Settings

Target Archive
- [✓] Login Information
- [✓] Project Sources
- [✓] Compiled Libraries
- [ ] Source Libraries

Target Files
- [✓] TMC File
- [ ] TPY File

Each time the PLC project is activated, the symbol file is automatically copied to the *C:\TwinCAT\3.x\Boot \Plc* directory, where it is named after the corresponding ADS port of the PLC project, such as "Port_851.tmc" for the first PLC runtime. The standard configuration of the OPC UA Server automatically reads this symbol file at startup.

License

Check whether a license exists. In TwinCAT 3 you can enter a TF6100 license in the TwinCAT license management (see Licensing (TC3) [▶ 35]).

Activate the configuration and restart TwinCAT. Then log into the PLC runtime and start the PLC program.

Step 3: Establish a connection to the OPC UA Server

To connect from an OPC UA Client, the client must connect to the URL of the OPC UA Server. This URL is described as follows:

opc.xxx://<ip or name>:port

- opc = predefined
- xxx = transport to OPC UA Server, e.g."tcp"
- port = port number of the OPC UA Server, e.g. 4840 (default)

Standard port 4840

Note that the standard port 4840 may be used by other OPC UA Servers, such as the Local Discovery Server (LDS) from the OPC Foundation, which is used by some vendors with OPC UA software packages.

Samples:

- opc.tcp://CX_0215AF:4840
- opc.tcp://172.16.2.80:4840

You will find the UA Sample Client provided in the Windows Start menu or in the product installation directory, e.g. *C:\TwinCAT\Functions\TF6100-OPC-UA\Win32\SampleClient\*.

Start the UA Sample Client (with the command **Run As Administrator** on Windows systems with activated UAC, e.g. Windows 7) and establish a connection with the URL of the OPC UA Server, e.g. opc.tcp://localhost:4840, by clicking on **Receive endpoints** and then on **Connect**.

After successfully establishing the connection, you will find the PLC runtime "PLC1" with the published variables (see step 1) in the UA namespace below the node "Objects".

4.1.3 PLC

4.1.3.1 Access to PLC runtime

This section describes how to configure the namespace of the OPC UA Server. The UA namespace describes the structure into which the PLC variables are mapped. In order for a PLC variable to be reachable via the UA namespace, it must be released in the PLC program.

Note that the OPC UA Server always connects by default to the first local PLC runtime system, therefore a configuration is not necessary in most cases (see also Quick start [▸ 41]). Only in exceptional cases is a separate configuration required, for example if further runtimes are to be displayed in the UA namespace or if the PLC of a BC device needs to be accessed.

This section contains the following topics:

- General Information [▸ 43]
- Step 1: Selecting PLC variables to be publicly accessible via OPC UA [▸ 44]
- Step 2: Downloading the symbol description and activating it in the PLC project [▸ 46]
- [Optional] Step 3: Configuring the data access device in the OPC UA Server [▸ 48]
- Explicit hiding of variables [▸ 48]

General Information

Several parameters are available for configuring the OPC UA namespace and accessing PLC variables, which have different effects depending on how the PLC runtime is displayed and how the communication takes place.

You can define these parameters with the help of the OPC UA Configurator or directly in the configuration file *ServerConfig.xml*.

The following table provides an overview of all parameters that are important when accessing a PLC runtime:

Parameter	Description	Possible values
ADS Port	Defines the ADS port under which the PLC runtime is accessible. The ADS port can be read in the properties of the PLC project.	800 (BC Controller) 801 (TwinCAT 2) 811 (TwinCAT 2) … 851 (TwinCAT 3 - standard) 852 (TwinCAT 3) …

Parameter	Description	Possible values
AutoCfg	Initially defines the type of target runtime used for the communication, e.g. PLC, C++, I/O. A finer distinction can subsequently be made within these categories. Each AutoCfg option is available as an unfiltered or filtered option. Filtered means the user can determine which PLC symbols should be published via OPC UA through PLC comments (see below). When using an unfiltered option, each PLC symbol is published to OPC UA.	7 TwinCAT 2 (TPY) 8 TwinCAT 2 (TPY) filtered 4040 TwinCAT 3 (TMC) 4041 TwinCAT 3 (TMC) filtered
AutoCfgSymFile	Symbol file of the respective PLC runtime. By default, the automatically generated symbol file of the first PLC runtime of the local system is imported.	Path to the symbol file. Points by default to the symbol file (TMC) of the first local PLC runtime.
IoMode	Defines the method for accessing symbols. This is particularly important for accessing BC devices.	1 (access via handle - default) 3 (Access to BC Controller)
ArraySubItemLegacy Support	By default, subelements of an array are not mapped as separate nodes in the UA namespace. Instead, only the array is mapped as a single element. Nonetheless, UA Clients can access subelements via their "IndexRange". (Some older OPC UA Clients do not yet support this option). The flag was introduced so that access is nevertheless possible for these clients. It ensures that every array position is displayed as a separate node in the UA namespace. This leads to higher memory requirement of the OPC UA Server.	0 (disabled - default) 1 (enabled)
Disabled	Disables the PLC runtime in the UA namespace, so that the corresponding node is not displayed. It is advisable to enable this parameter if certain PLC runtimes are not yet available at the time of project planning, for example because the corresponding devices are not yet connected to the network.	0 (disabled - default) 1 (enabled)

The following section uses a sample to illustrate how the variables can be imported from a PLC program into the UA namespace. It is assumed that the OPC UA Server in its standard configuration (delivery state after installation) and the PLC runtime are on the same computer.

AutoCfg options

The information contained in this article, including the following PLC comments/pragmas, is based on TwinCAT 3 and must therefore be used with the AutoCfg options "4041 TwinCAT 3 (TMC) filtered" (standard configuration).

Note that the AutoCfg options for TwinCAT 2 use a different PLC comment format (*~ (OPC:1:description) *), which is no longer used in TwinCAT 3, but is still available for compatibility by using AutoCfg option "8".

Step 1: Selecting PLC variables to be publicly accessible via OPC UA

The OPC UA Server automatically establishes a connection to the first PLC runtime on the local system. The PLC symbols marked in the PLC program (TwinCAT PLC control) for OPC UA are provided. They identify the PLC symbols via a comment at the corresponding point (instance, structure, variable) in the PLC program code (see following samples).

Sample 1

In this sample, the PLC variables bMemFlag1, bMemFlag2, bMemAlarm2 and iReadOnly are enabled via OPC UA. The PLC variable bMemAlarm1 should not be accessible via the OPC UA Server. The PLC program in PLC control then looks like the following, depending on the TwinCAT version you are using:

TwinCAT 3 (with TMC import):

```
{attribute 'OPC.UA.DA' := '1'}
bMemFlag1 : BOOL;

{attribute 'OPC.UA.DA' := '1'}
bMemFlag2 : BOOL;

bMemAlarm1 : BOOL;

{attribute 'OPC.UA.DA' := '1'}
bMemAlarm2 : BOOL;

{attribute 'OPC.UA.DA' := '1'}
{attribute 'OPC.UA.DA.Access' := '1'}
iReadOnly : INT;
```

TwinCAT 2 (with TPY import):

```
bMemFlag1 : BOOL; (*~ (OPC:1:some description) *)

bMemFlag2 : BOOL; (*~ (OPC:1:some description) *)

bMemAlarm1 : BOOL;

bMemAlarm2 : BOOL; (*~ (OPC:1:some description) *)

iReadOnly    : INT; (*~ (OPC:1:some description)
 (OPC_PROP[0005]:1:read-only flag) *)
```

TPY formatting can also be used in TwinCAT 3 PLC projects for migration purposes, e.g. when a TwinCAT 2 project is to be conveniently converted into a TwinCAT 3 project.

Due of the additional comment `OPC.UA.DA.Access` the access level for the variable iReadOnly is set to "ReadOnly". The various options can be found in the complete list of PLC comments (see List of attributes and comments [▶ 71]).

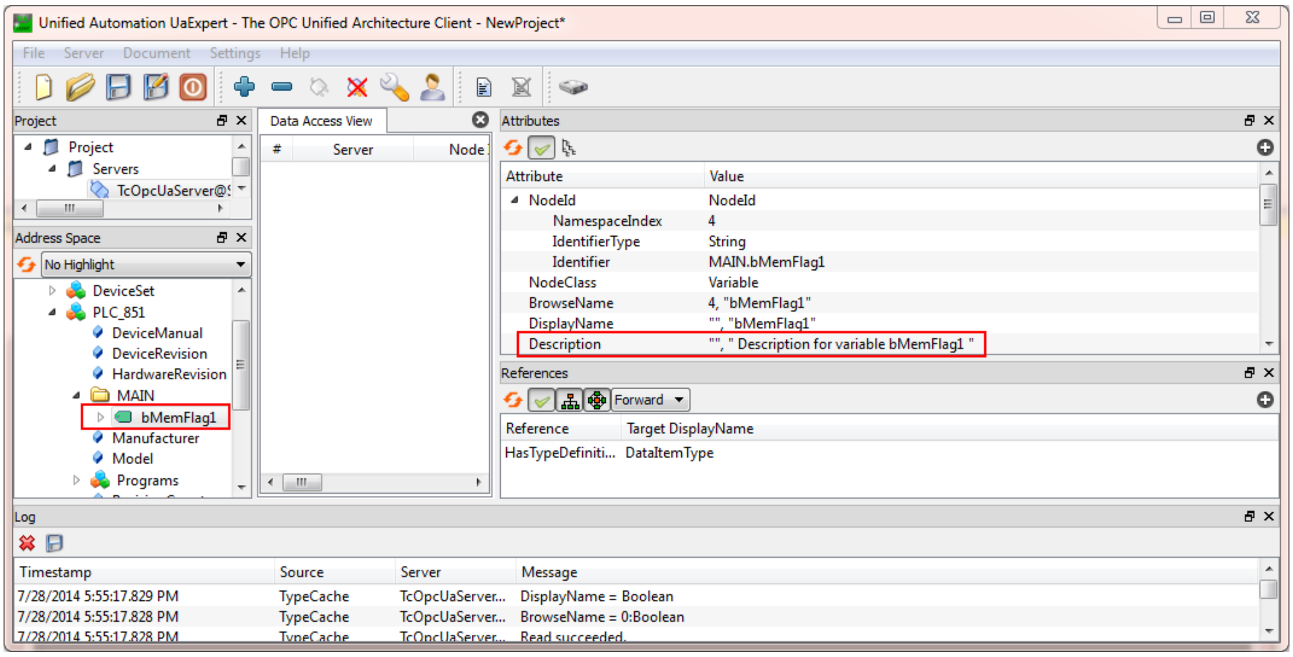
To describe a variable in the UA namespace, insert a simple PLC comment after the corresponding variable, e.g.

TwinCAT 3 (with TMC import):

```
{attribute 'OPC.UA.DA' := '1'}
bMemFlag1 : BOOL; (* Description for variable bMemFlag1 *)
```

TwinCAT 2 (with TPY import): Function not available

The comment automatically becomes the description attribute of the node in the UA namespace.



Attribute	Value
NodeId	NodeId
NamespaceIndex	4
IdentifierType	String
Identifier	MAIN.bMemFlag1
NodeClass	Variable
BrowseName	4, "bMemFlag1"
DisplayName	"", "bMemFlag1"
Description	"", " Description for variable bMemFlag1 "

Reference	Target DisplayName
HasTypeDefiniti...	DataItemType

Timestamp	Source	Server	Message
7/28/2014 5:55:17.829 PM	TypeCache	TcOpcUaServer...	DisplayName = Boolean
7/28/2014 5:55:17.828 PM	TypeCache	TcOpcUaServer...	BrowseName = 0:Boolean
7/28/2014 5:55:17.828 PM	TypeCache	TcOpcUaServer...	Read succeeded.

Sample 2:

In this sample, the two instances fbTest1 and fbTest2 of the function block FB_BLOCK1 should be available via OPC UA. When an entire instance is released, all its symbols are also available via OPC UA. The PLC program looks like the following:

TwinCAT 3 (with TMC import):

```
PROGRAM MAIN
VAR
    {attribute 'OPC.UA.DA' := '1'}
    fbTest1 : FB_BLOCK1;
    fbTest2 : FB_BLOCK1;
END_VAR
```

```
FUNCTION_BLOCK FB_BLOCK1
VAR_INPUT
    {attribute 'OPC.UA.DA' := '1'}
    ni1 : INT;
    ni2 : INT;
END_VAR
VAR_OUTPUT
    {attribute 'OPC.UA.DA' := '1'}
    no1 : INT;
    no2 : INT;
END_VAR
VAR
    {attribute 'OPC.UA.DA' := '1'}
    nx1 : INT;
    nx2 : INT;
END_VAR
```

TwinCAT 2 (with TPY import):

```
PROGRAM MAIN
VAR
    fbTest1 : FB_BLOCK1; (*~ (OPC:1:some description) *)
    fbTest2 : FB_BLOCK1;
END_VAR
```

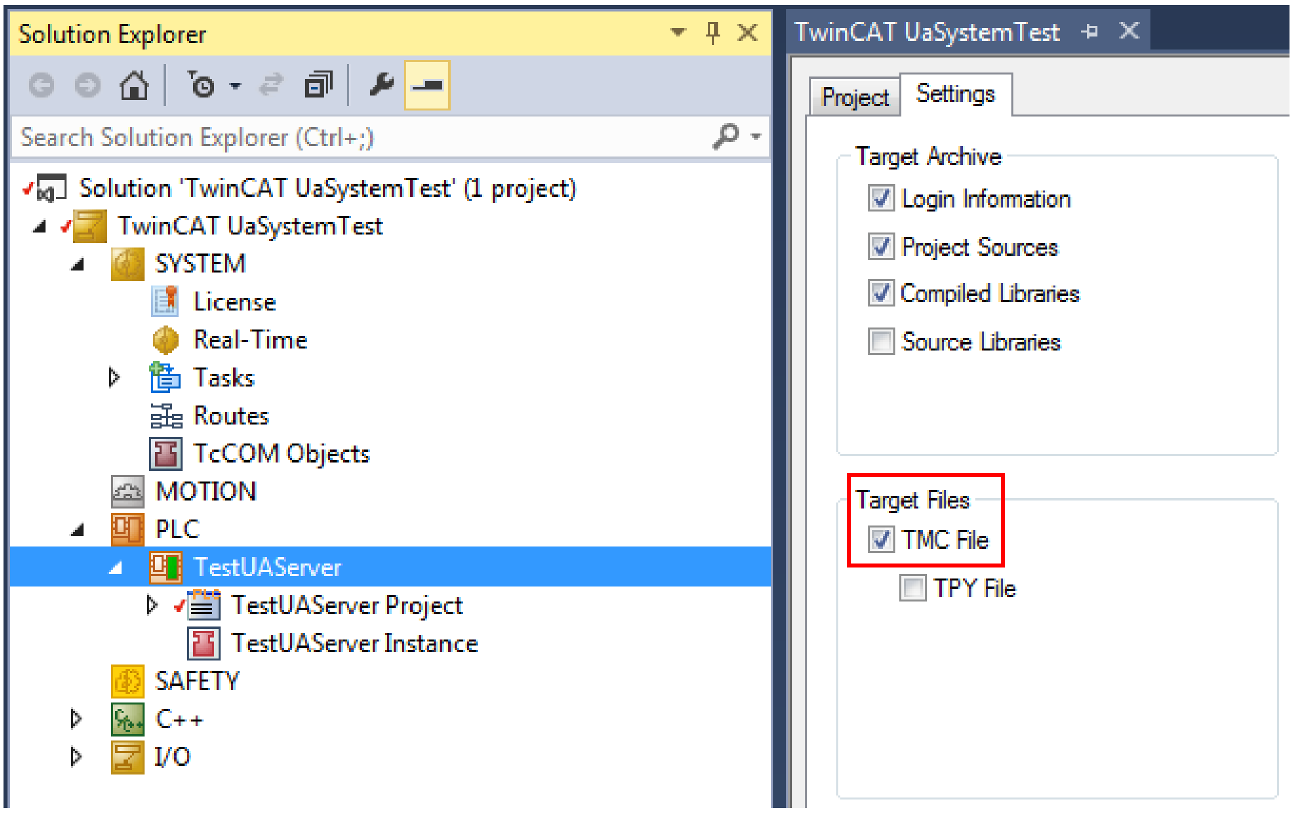
```
FUNCTION_BLOCK FB_BLOCK1
VAR_INPUT
    ni1 : INT; (*~ (OPC:1:some description) *)
    ni2 : INT;
END_VAR
VAR_OUTPUT
    no1 : INT; (*~ (OPC:1:some description) *)
    no2 : INT;
END_VAR
VAR
    nx1 : INT; (*~ (OPC:1:some description) *)
    nx2 : INT;
END_VAR
```

The instance fbTest1 is enabled for OPC UA, whereby all contained symbols are automatically enabled for OPC UA, i.e. fbTest.ni1, fbTest.ni2, etc. The instance fbTest2 is not marked for OPC UA, but the three variables contained in it - ni1, no1 and nx1 - were marked in the function block. They are therefore available in all instances via OPC UA.

Step 2: Downloading the symbol description and activating it in the PLC project

The symbol file contains information about all PLC variables available in a PLC project. The OPC UA Server needs this information to configure its namespace. By default, the current symbol information is automatically downloaded to a symbol file. This is located in the project folder of the corresponding TwinCAT project. To ensure that the symbol file is transferred to the target runtime, enable the download of the symbol file in the settings of the PLC project.

TwinCAT 3:



Solution Explorer

Search Solution Explorer (Ctrl+;)

- Solution 'TwinCAT UaSystemTest' (1 project)
  - TwinCAT UaSystemTest
    - SYSTEM
      - License
      - Real-Time
      - Tasks
      - Routes
      - TcCOM Objects
    - MOTION
    - PLC
      - TestUAServer
        - TestUAServer Project
        - TestUAServer Instance
    - SAFETY
    - C++
    - I/O

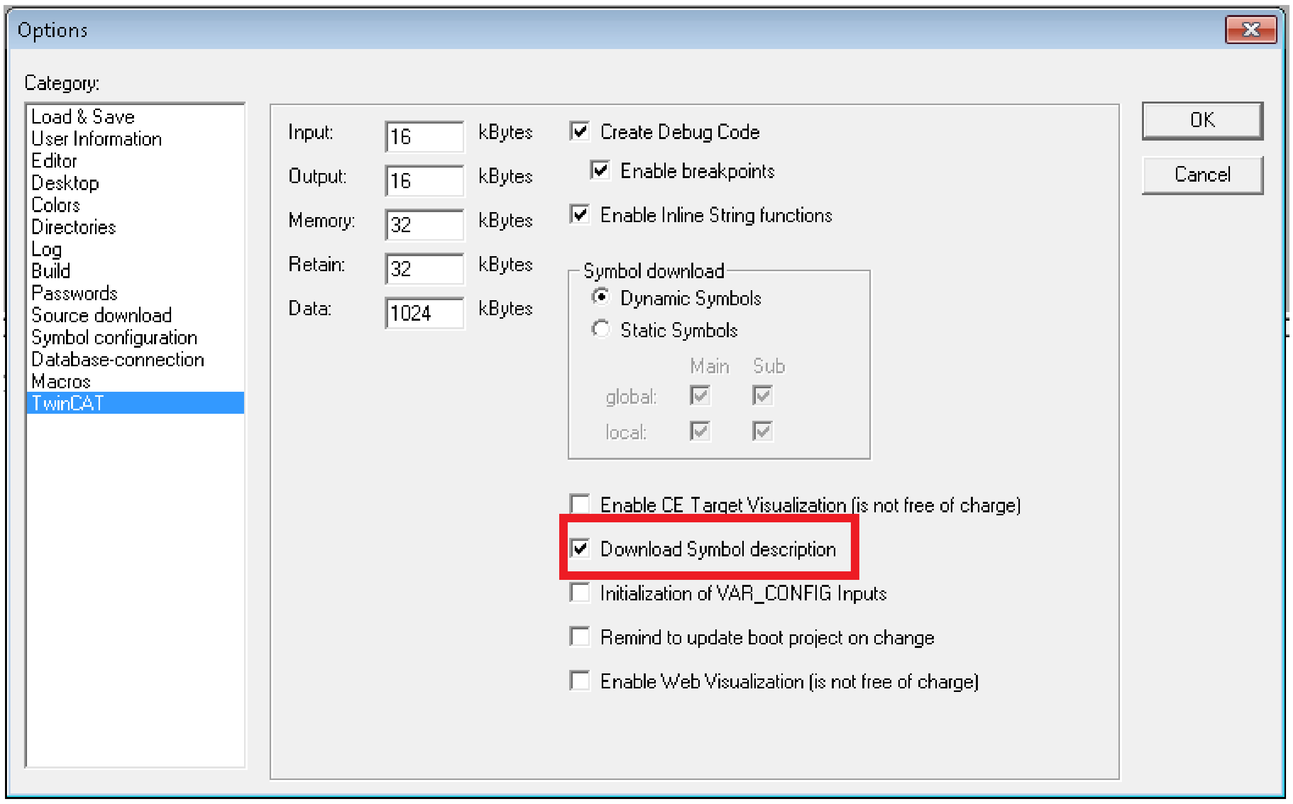
TwinCAT UaSystemTest

Project | Settings

Target Archive
- [✓] Login Information
- [✓] Project Sources
- [✓] Compiled Libraries
- [ ] Source Libraries

Target Files
- [✓] TMC File
- [ ] TPY File

TwinCAT 2:



Options

Category: Load & Save, User Information, Editor, Desktop, Colors, Directories, Log, Build, Passwords, Source download, Symbol configuration, Database-connection, Macros, TwinCAT

Input:	16	kBytes
Output:	16	kBytes
Memory:	32	kBytes
Retain:	32	kBytes
Data:	1024	kBytes

- [✓] Create Debug Code
  - [✓] Enable breakpoints
- [✓] Enable Inline String functions

Symbol download
- (●) Dynamic Symbols
- ( ) Static Symbols

	Main	Sub
global:	✓	✓
local:	✓	✓

- [ ] Enable CE Target Visualization (is not free of charge)
- [✓] Download Symbol description
- [ ] Initialization of VAR_CONFIG Inputs
- [ ] Remind to update boot project on change
- [ ] Enable Web Visualization (is not free of charge)

OK | Cancel

When creating a boot project, the symbol file is automatically copied to the boot directory of the PLC runtime. This file name provides information about the corresponding ADS port of the PLC runtime.

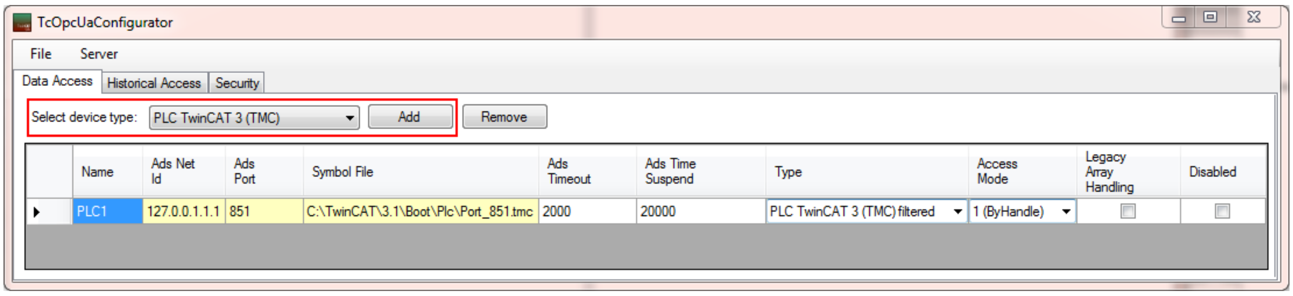
If the OPC UA Server and the PLC runtime are on the same system, all you have to do is restart the OPC UA Server. The OPC UA Server is automatically configured to load the symbol file from the TwinCAT boot directory and generate its namespace based on the symbol information contained in this file. If another symbol file is to be used, you can configure it either in the OPC UA Configurator or directly in the file *ServerConfig.xml*.

[Optional] Step 3: Configuring the data access device in the OPC UA Server

By default, the OPC UA Server connects to the first PLC runtime on the local system. If you want to publish more than one PLC runtime in the UA namespace, you must configure additional data access devices in the OPC UA Configurator. This may be necessary for the following scenarios:

- You have another (second or third) PLC runtime from which you want to publish symbol information to OPC UA.
- You have another Industrial PC or Embedded PC whose PLC runtime and corresponding symbol information you want to make publicly available to OPC UA.

To configure additional devices for data access, open the OPC UA Configurator and add new data access devices.



Select device type: PLC TwinCAT 3 (TMC)

Name	Ads Net Id	Ads Port	Symbol File	Ads Timeout	Ads Time Suspend	Type	Access Mode	Legacy Array Handling	Disabled
PLC1	127.0.0.1.1.1	851	C:\TwinCAT\3.1\Boot\Plc\Port_851.tmc	2000	20000	PLC TwinCAT 3 (TMC) filtered	1 (ByHandle)	☐	☐

Configure all further necessary parameters as described in the section General information [▶ 43]. If you are configuring a remote Industrial PC or Embedded PC, you must enter the correct parameters for AdsNetId and AdsPort and provide the corresponding symbol file of the PLC program on this PC.

Explicit hiding of symbols

In addition to the regular PLC attributes for activating a PLC symbol, such as variables, you can explicitly exclude PLC symbols from publication in the OPC UA namespace. There may be several reasons for this:

- You want to publish a data structure, but not all its subelements (Sample 1).
- You have enabled a data structure in the definition and want to hide an individual instance of this data structure (Sample 2).
- You want to speed up the startup of the OPC UA Server (Sample 3).

Sample 1:

```
{attribute 'OPC.UA.DA' := '1'}
TYPE ST_TEST :
STRUCT
    a : INT;
    {attribute 'OPC.UA.DA' := '0'}
    b : DINT;
END_STRUCT
END_TYPE

PROGRAM MAIN
VAR
    instance1 : ST_TEST;
    instance2 : ST_TEST;
END_VAR
```

In this case, the PLC attribute has been added to the structure definition, so that each instance of this structure should be available on the OPC UA Server. Subelement b (and all its subelements, if b is another data structure) will be excluded from publication on the OPC UA Server.

Sample 2:

```
{attribute 'OPC.UA.DA' := '1'}
TYPE ST_TEST :
STRUCT
  a : INT;
  b : DINT;
END_STRUCT
END_TYPE

PROGRAM MAIN
```

```
VAR
  instance1 : ST_TEST;
  {attribute 'OPC.UA.DA' := '0'}
  instance2 : ST_TEST;
END_VAR
```

Although the PLC attribute is added to the structure definition, which makes each instance of this structure available on the OPC UA Server, instance2 receives the PLC attribute to disable this inheritance. This means that only instance1 is available in the UA namespace, not instance2.

Sample 3:

During the start procedure the OPC UA Server reads the PLC symbol files to construct its namespace. To this end, all the data structures found are completely read out in order to find and integrate all type definitions. If an `OPC.UA.DA:=0` PLC attribute is detected, the analysis of all subsequent subelements stops, so that the startup can be considerably faster, depending on the size of the data structures.

4.1.3.2 Historical Access

This section describes the steps necessary to configure the variables in the namespace of the OPC UA Server for Historical Access (HA).

Historical Access is an OPC UA function, in which the variable values are either stored permanently on a data storage device (file or database) or in the device RAM, so that they can be retrieved later by the client. The way in which the OPC UA Server reads and stores the variable values can be configured.

4.1.3.2.1 Setup Version 3.x.x

General

Perform the following steps once in order to activate a value curve of PLC variables via OPC UA Historical Access.

- Activate Historical Access on the UA Server [▸ 49]
- Activate Historical Access in the TwinCAT PLC program [▸ 51]
- Activate Historical Access in the TwinCAT C++ module [▸ 52]

The section Display historical data [▸ 55] describes how you can call saved variable values via Historical Access and display them in an OPC UA Client with the help of the UA Expert software.

Requirements

The following prerequisites apply to the use of Historical Access:

- The runtime system whose symbols are to be saved for Historical Access must be configured as Data Access.
- In order to use an SQL Compact database as a storage medium, SQL Compact Runtime 3.5 SP2 must be installed on the computer on which the OPC UA Server is running.
- OPC UA version 2.0 or higher must be installed to use an SQL Server database.
- SQL Compact databases are also supported under Windows CE.
- SQL Server databases are not supported under Windows CE.

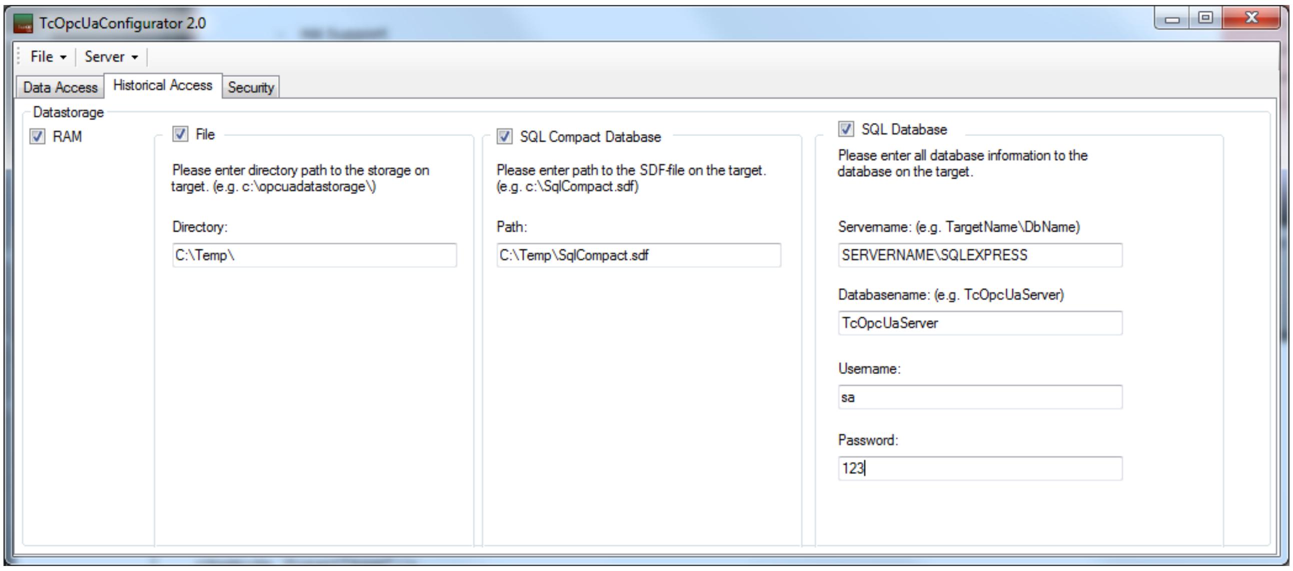
Activate Historical Access on the UA Server

Various storage media can be used for storing historic values. These must be activated once in the OPC UA Server. You can make the settings necessary for this with the help of the OPC UA Configurator or in the *ServerConfig.xml* file. Once you have activated one or more storage media on the OPC UA Server, the PLC program selects the storage medium to be used for each symbol.

The following four storage media are currently supported:

- Main memory: Stores the symbol values in the RAM of the device on which the OPC UA Server is running.
- File: Saves the values in a file with the specified path.
- SQL Compact database: Stores the values in an SQL Compact database with specified path. In this case, the SQL Compact Runtime 3.5 SP2 must be installed as a corresponding database version on the OPC UA Server.
- SQL Server database: Stores the values in an SQL Server database specified with the parameters server name, database name, user name and password.

Configuration with the help of the OPC UA Configurator:



☑ RAM

☑ File

Please enter directory path to the storage on target. (e.g. c:\opcuadatastorage\)

Directory: C:\Temp\

☑ SQL Compact Database

Please enter path to the SDF-file on the target. (e.g. c:\SqlCompact.sdf)

Path: C:\Temp\SqlCompact.sdf

☑ SQL Database

Please enter all database information to the database on the target.

Servername: (e.g. TargetName\DbName) SERVERNAME\SQLEXPRESS

Databasename: (e.g. TcOpcUaServer) TcOpcUaServer

Username: sa

Password: 123

Configuration directly via the *ServerConfig.xml* file:

Insert the XML tag shown below into the *ServerConfig.xml* file between <OpcServerConfig><UaServerConfig> and </UaServerConfig></OpcServerConfig>.

```
<HaModes>
<HaMode Type="RAM"/>
<HaMode Type="File" DataStore="C:\Temp\"/>
<HaMode Type="SQLServerCompact" DbName="C:\Temp\TcOpcUaServer.sdf"/>
<HaMode Type=”SQLServer” Servername=”SERVERNAME\SQLEXPRESS” DbName=”TcOpcUaServer” User=”sa”
Pass=”123”/>
</HaModes>
```

The parameters required for the respective mode are:

Mode	Required parameters
RAM	Stores the values in the RAM of the device on which the OPC UA Server is running. No further parameters are required. After restarting the device, the stored values are no longer available. The storage medium is therefore not a permanent medium.
File	Stores the values in multiple files whose location is determined by the DataStore parameter. Each symbol configured for this storage medium (see step 2) is assigned its own file in this directory. In addition, there is a backup copy of the file for each symbol, which is created when the buffer length (see step 2) is reached. The contents of the data file are then saved as a backup copy, and a new file is created.
SQL Server Compact	Stores the values in an SQL Compact database whose location is defined with the parameter DbName. The above requirements apply.
SQL Server	Stores the values in an SQL Server database that is specified with the parameters server name, DBName, user and pass.

Activate Historical Access in the TwinCAT PLC program

If a PLC runtime is configured as a data access device, the PLC variables that have been enabled for this purpose in the PLC program by means of comments are automatically made available by the OPC UA Server via OPC UA.

You can enable or disable individual variables for Historical Access in the same way as you can enable variables for access via OPC UA. To do this, supplement an appropriate comment in the PLC program code on the right of a symbol. The following parameters can be specified:

TwinCAT 3 (with TMC import):

Parameter	Comment	Description
Activate	{attribute 'OPC.UA.DA' := '1'}	Activates the data access symbol. This is necessary to make the symbol available for Historical Access.
Activating for HA	{attribute 'OPC.UA.HA' := '1'}	Activates the symbol for Historical Access.
Storage medium	{attribute 'OPC.UA.HA.Storage' := '1'}	Specifies the storage medium to be used for storing the symbol values (1 = RAM, 2 = File, 3 = SQL Compact Database, 4 = SQL Database).
Sampling rate	{attribute 'OPC.UA.HA.Sampling' := '50'}	Defines the rate at which the variable values are to be read from the controller and stored on the corresponding storage medium. The OPC UA Server uses predefined fixed sampling rates that can be used here. You can expand the sampling rates if necessary in the ServerConfig file. However, this is not normally necessary, and it is not advisable to change the settings, unless there is a good reason.
Buffer length	{attribute 'OPC.UA.HA.Buffer' := '10000'}	Specifies the number of elements to include in the buffer.

TwinCAT 2 (with TPY import):

Parameter	Comment	Description
Activate	(*~ (OPC:1:some description) *)	Activates the data access symbol. This is necessary to make the symbol available for Historical Access.
Storage medium	(*~ (OPC_UA_PROP[5000]:1:RAM) *)	Specifies the storage medium to be used for storing the symbol values (1 = RAM, 2 = File, 3 = SQL Compact Database, 4 = SQL Database).
Sampling rate	(*~ (OPC_UA_PROP[5000][1]:50:SamplingRate) *)	Defines the rate at which the variable values are to be read from the controller and stored on the corresponding storage medium. The OPC UA Server uses predefined fixed sampling rates that can be used here. You can expand the sampling rates if necessary in the ServerConfig file. However, this is not normally necessary, and it is not advisable to change the settings, unless there is a good reason.
Buffer length	(*~ (OPC_UA_PROP[5000][2]:5000:Buffer) *)	Specifies the number of elements to include in the buffer.

The following excerpt from a PLC program describes the use of the parameters listed above using four PLC variables:

TwinCAT 3 (with TMC import):

```
{attribute 'OPC.UA.DA' := '1'}
{attribute 'OPC.UA.HA' := '1'}
{attribute 'OPC.UA.HA.Storage' := '1'}
{attribute 'OPC.UA.HA.Sampling' := '5'}
{attribute 'OPC.UA.HA.Buffer' := '1000'}
_HistoryMem : UINT;

{attribute 'OPC.UA.DA' := '1'}
{attribute 'OPC.UA.HA' := '1'}
{attribute 'OPC.UA.HA.Storage' := '2'}
{attribute 'OPC.UA.HA.Sampling' := '20'}
{attribute 'OPC.UA.HA.Buffer' := '200'}
_HistoryFile : UINT;

{attribute 'OPC.UA.DA' := '1'}
{attribute 'OPC.UA.HA' := '1'}
{attribute 'OPC.UA.HA.Storage' := '3'}
{attribute 'OPC.UA.HA.Sampling' := '50'}
{attribute 'OPC.UA.HA.Buffer' := '10000'}
_HistoryDBcompact : UINT;

{attribute 'OPC.UA.DA' := '1'}
{attribute 'OPC.UA.HA' := '1'}
{attribute 'OPC.UA.HA.Storage' := '4'}
{attribute 'OPC.UA.HA.Sampling' := '5'}
{attribute 'OPC.UA.HA.Buffer' := '1000000'}
_HistoryDB : UINT;
```

TwinCAT 2 (with TPY import):

```
_HistoryMem : UINT; (*~ (OPC:1:available for OPC UA Clients)
 (OPC_UA_PROP[5000]:1:Memory)
 (OPC_UA_PROP[5000][1]:5:SamplingRate)
 (OPC_UA_PROP[5000][2]:1000:Buffer) *)

_HistoryFile : UINT; (*~ (OPC:1:available for OPC UA Clients)
 (OPC_UA_PROP[5000]:2:File)
 (OPC_UA_PROP[5000][1]:20:SamplingRate)
 (OPC_UA_PROP[5000][2]:200:Buffer) *)

_HistoryDBcompact : UINT; (*~ (OPC:1:available for OPC UA Clients)
 (OPC_UA_PROP[5000]:3:SQLCompact)
 (OPC_UA_PROP[5000][1]:50:SamplingRate)
 (OPC_UA_PROP[5000][2]:10000:Buffer) *)

_HistoryDB : UINT; (*~ (OPC:1:available for OPC UA Clients)
 (OPC_UA_PROP[5000]:4:SQL)
 (OPC_UA_PROP[5000][1]:5:SamplingRate)
 (OPC_UA_PROP[5000][2]:1000000:Buffer) *)
```

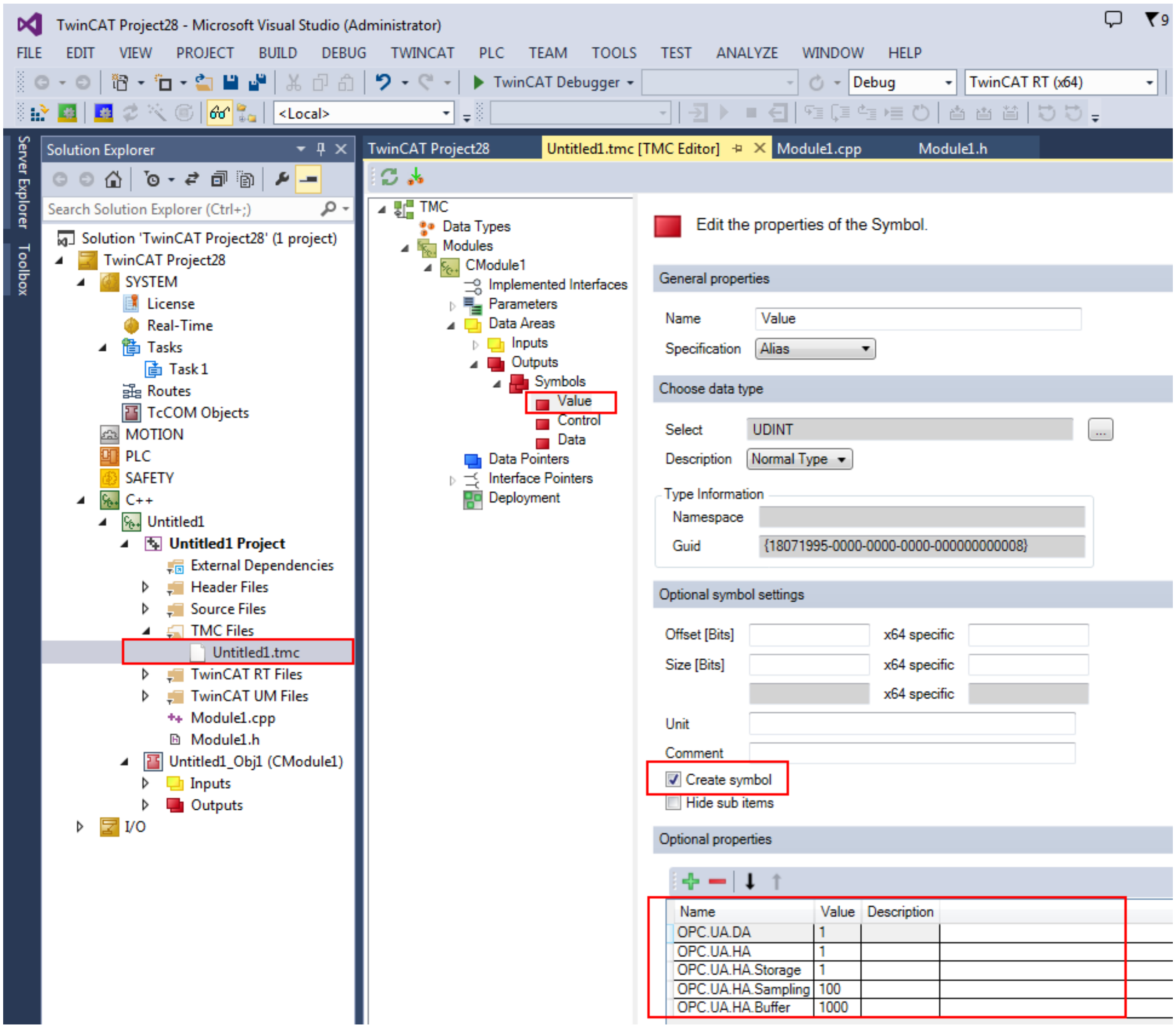
Activate Historical Access in the TwinCAT C++ module

Each TwinCAT 3 C++ symbol whose history is to be recorded must be configured accordingly. Comparable with the configuration of PLC symbols for Historical Access, the parameters for a C++ symbol must be configured:

Parameter	Comment	Description
Activate	OPC.UA.DA := 1	Activates the data access symbol. This is necessary to make the symbol available for Historical Access.
Activate HA	OPC.UA.HA := 1	Activates the symbol for Historical Access.
Storage medium	OPC.UA.HA.Storage := 1	Specifies the storage medium to be used for storing the symbol values (1 = RAM, 2 = File, 3 = SQL Compact Database, 4 = SQL Database).

Parameter	Comment	Description
Sampling rate	OPC.UA.HA.Sampling := 50	Defines the rate at which the variable values are to be read from the controller and stored on the corresponding storage medium. The OPC UA Server uses predefined fixed sampling rates that can be used here. You can expand the sampling rates if necessary in the ServerConfig file. However, this is not normally necessary, and it is not advisable to change the settings, unless there is a good reason.
Buffer length	OPC.UA.HA.Buffer := 10000	Specifies the number of elements to include in the buffer.

In TwinCAT 3 C++ these parameters are configured in the TMC code editor of the corresponding C++ module. Using the TMC code editor, navigate to the symbol to be enabled for Historical Access and add these parameters to the **Optional Properties** of Symbol Properties. Make sure that you have selected the **Create symbol** check box.



Edit the properties of the Symbol.

General properties

Name: Value
Specification: Alias

Choose data type

Select: UDINT
Description: Normal Type
Type Information
Namespace:
Guid: {18071995-0000-0000-0000-000000000008}

Optional symbol settings

Create symbol [✓]
Hide sub items [ ]

Optional properties

Name	Value	Description
OPC.UA.DA	1	
OPC.UA.HA	1	
OPC.UA.HA.Storage	1	
OPC.UA.HA.Sampling	100	
OPC.UA.HA.Buffer	1000	

Requirements

Products	Setup versions	Target platform
TS6100, TF6100	3.x.x	IPC or CX (x86, x64, ARM)

4.1.3.2.2 Setup Version 4.x.x

From setup version 4.x.x historical access can no longer be configured via attributes or comments. The advantage of this is that the Historical Access functionality is now also available for runtimes that previously did not support this, such as variables of a BC9191 Controller. Furthermore, a Historical Access configuration can now also be carried out retrospectively, i.e. without having to adapt the PLC project.

The historical access configuration is now XML-based (TcUaHaConfig.xml) and can be carried out using the OPC UA Server Configurator. The XML file is located in the same directory as *TcOpcUaServer.exe*.

See also:

- Configuration > Setup version 4.x.x > Overview [▶ 192] (OPC UA Server Configurator) and Configuring historical access [▶ 200])

XML file structure (TCUaHaConfig.xml)

```
<HistoryConfig>
  <HistoryController>
    <HistoryAdapter Type="Volatile" Id="0" File="" Server="" Database="" User="" Password=""/>
    <HistoryAdapter Type="File" Id="1" File="historydb.dat" Server="" Database="" User=""
Password=""/>
    <HistoryAdapter Type="SQL" Id="2" File="" Server="" Database="database123" User="user"
Password="pwd"/>
    <HistoryAdapter Type="SQLCompact" Id="3" File="historydb.xyz" Server="" Database="" User=""
Password=""/>
    <HistoryAdapter Type="SQLCompact" Id="4" File="historydb2.xyz" Server="" Database="" User=""
Password=""/>
  </HistoryController>
  <HistoryNodes>
    <HistoryNode HistoryWriteable="true" SamplingRate="0" MaxSamples="100000" AdapterId="0"
NS="urn:BeckhoffAutomation:Ua:PLC1" NodeId="s=PRG_HA_SingleNode.nMyValue"/>
    <HistoryNode HistoryWriteable="true" SamplingRate="0" MaxSamples="100000" AdapterId="0"
NS="urn:BeckhoffAutomation:Ua:PLC1" NodeId="s=PRG_HA_MultipleNodes.nMyValue1"/>
    <HistoryNode HistoryWriteable="true" SamplingRate="0" MaxSamples="100000" AdapterId="0"
NS="urn:BeckhoffAutomation:Ua:PLC1" NodeId="s=PRG_HA_MultipleNodes.nMyValue2"/>
    <HistoryNode HistoryWriteable="true" SamplingRate="0" MaxSamples="100000" AdapterId="0"
NS="urn:BeckhoffAutomation:Ua:PLC1" NodeId="s=PRG_HA_MultipleNodes.fMyValue3"/>
  </HistoryNodes>
</HistoryConfig>
```

You can configure all supported data memory in the HistoryController area. The following locations are currently supported:

- RAM ("Volatile")
- File
- SQL Compact
- SQL Server (not supported under Windows CE)

Each HistoryAdapter (except for "Volatile") can be used multiple times, e.g. if the historical values for individual variables are to be stored in different data memories.

In the HistoryNodes area, the individual nodes are configured, assigned to a data memory and assigned a SamplingRate and a maximum size for the ring buffer to be used in the data memory.

The attribute `HistoryWriteable="true"` ensures that the data memory for this node can be filled with values via a HistoryUpdate() call. Such a call is provided, for example, by the TwinCAT OPC UA Client via the function block UA_HistoryUpdate. If you configure a node with this attribute, then the server doesn't normally "sample" the values itself, but receives them from other clients. In such a configuration the SamplingRate is therefore usually 0.

Requirements

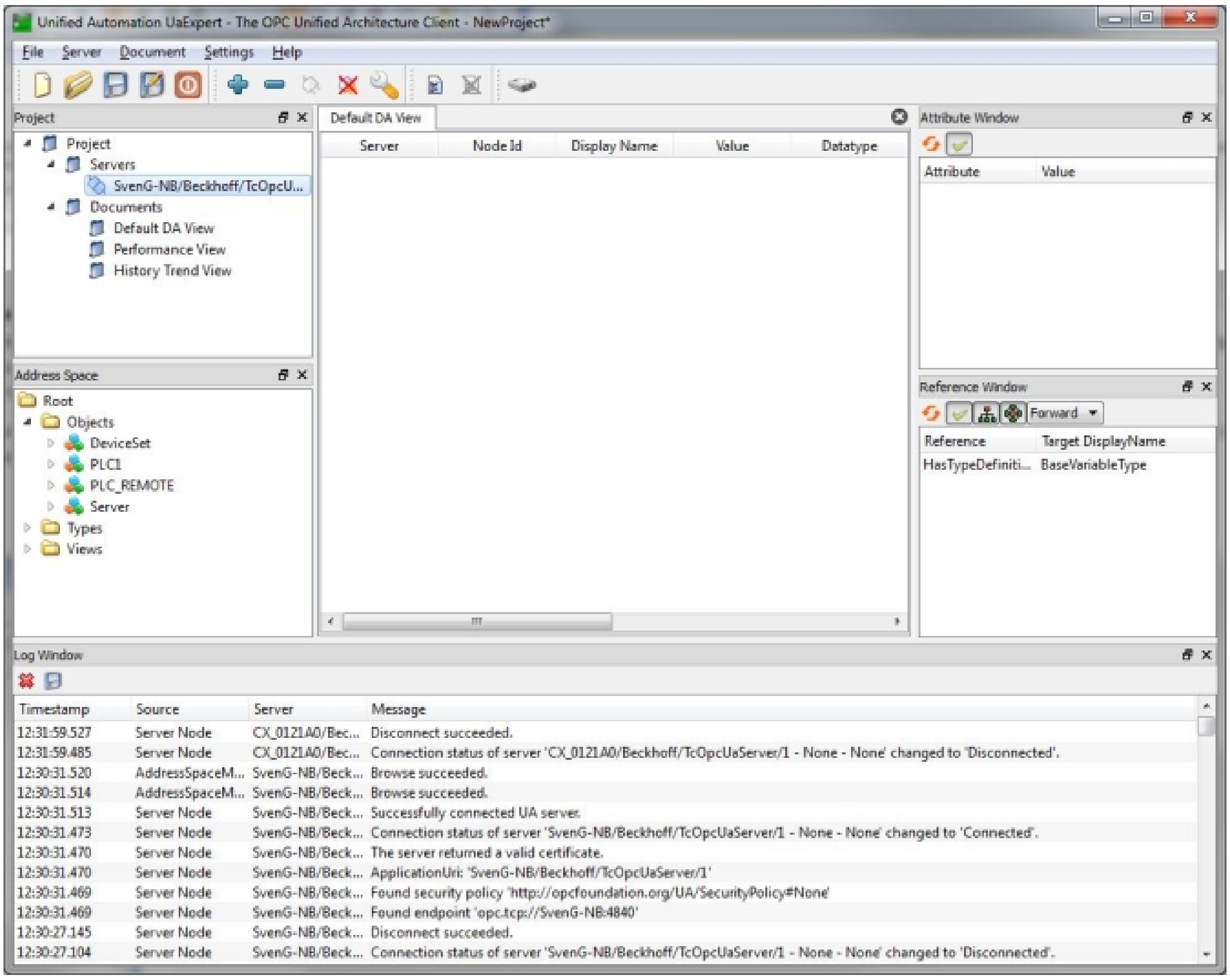
Products	Setup versions	Target platform
TF6100	4.x.x	IPC or CX (x86, x64, ARM)

4.1.3.2.3 Display historical data

Displaying Historical Access values in an OPC UA Client

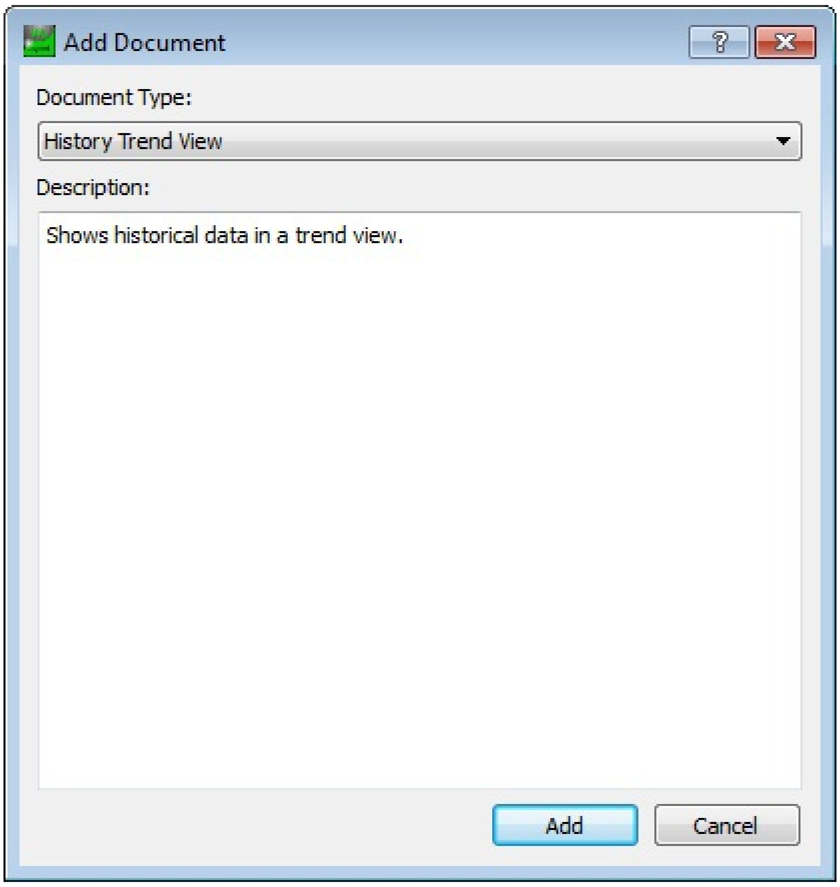
The following step-by-step instructions describe how to configure the UA Expert software in order to access historical data.

1. Start the UA Expert software and connect to the OPC UA Server.



```
Timestamp     Source             Server               Message
12:31:59.527  Server Node        CX_0121A0/Bec...     Disconnect succeeded.
12:31:59.485  Server Node        CX_0121A0/Bec...     Connection status of server 'CX_0121A0/Beckhoff/TcOpcUaServer/1 - None - None' changed to 'Disconnected'.
12:30:31.520  AddressSpaceM...   SvenG-NB/Beck...     Browse succeeded.
12:30:31.514  AddressSpaceM...   SvenG-NB/Beck...     Browse succeeded.
12:30:31.513  Server Node        SvenG-NB/Beck...     Successfully connected UA server.
12:30:31.473  Server Node        SvenG-NB/Beck...     Connection status of server 'SvenG-NB/Beckhoff/TcOpcUaServer/1 - None - None' changed to 'Connected'.
12:30:31.470  Server Node        SvenG-NB/Beck...     The server returned a valid certificate.
12:30:31.470  Server Node        SvenG-NB/Beck...     ApplicationUri: 'SvenG-NB/Beckhoff/TcOpcUaServer/1'
12:30:31.469  Server Node        SvenG-NB/Beck...     Found security policy 'http://opcfoundation.org/UA/SecurityPolicy#None'
12:30:31.469  Server Node        SvenG-NB/Beck...     Found endpoint 'opc.tcp://SvenG-NB:4840'
12:30:27.145  Server Node        SvenG-NB/Beck...     Disconnect succeeded.
12:30:27.104  Server Node        SvenG-NB/Beck...     Connection status of server 'SvenG-NB/Beckhoff/TcOpcUaServer/1 - None - None' changed to 'Disconnected'.
```

2. Add a new History Trend View.

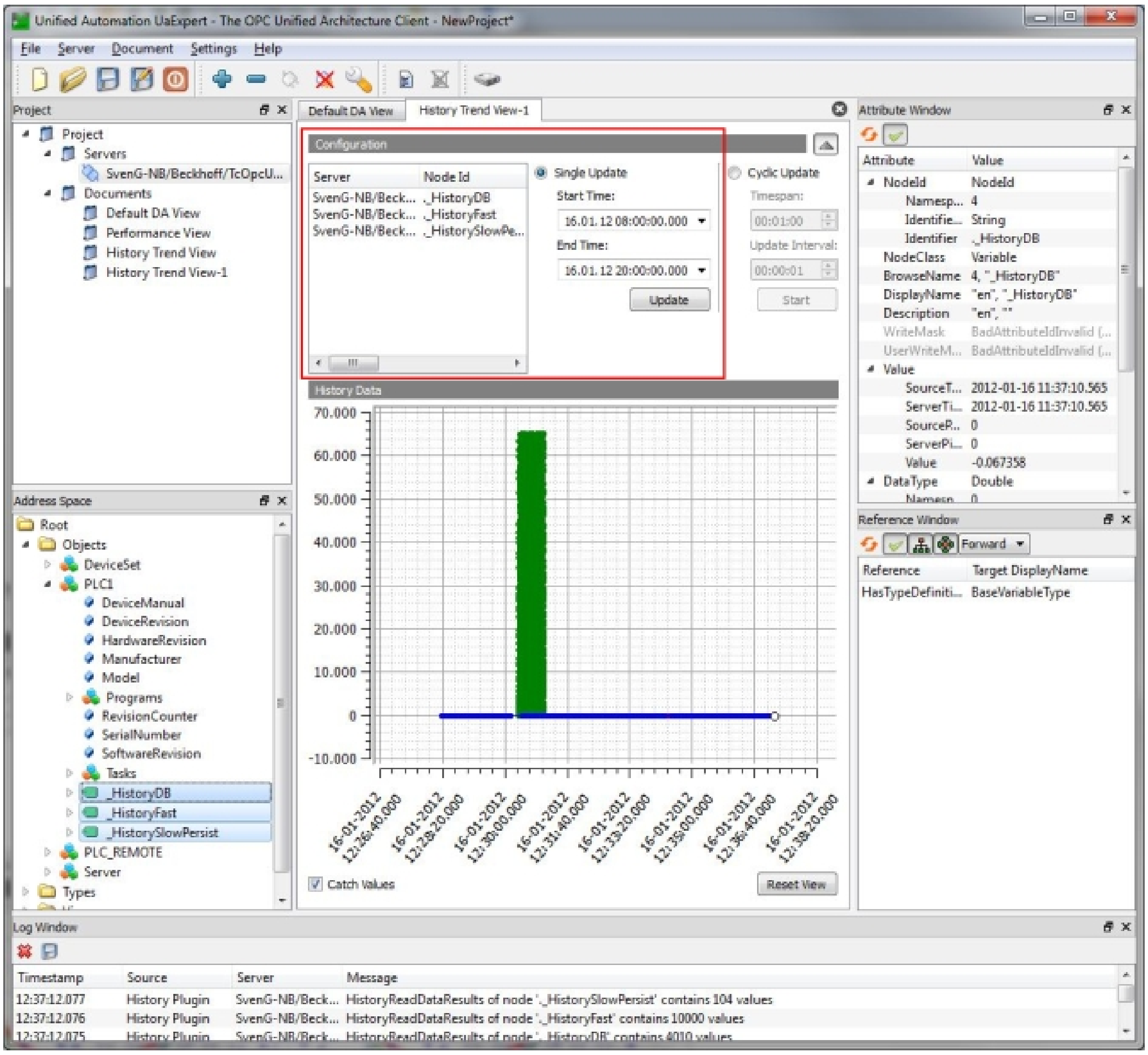


Document Type: History Trend View

Description: Shows historical data in a trend view.

3. Browse the PLC1 namespace and use drag & drop to add the PLC variables _HistoryDB, _HistoryDBcompact, _HistoryFast and _HistorySlowPersist.

⇨ You can now use the **Start Time** and **End Time** controls to specify the desired time period for which the symbol values are to be displayed, or you can start a **Cyclic Update** for these variables if necessary.



Server	Node Id
SvenG-NB/Beck...	._HistoryDB
SvenG-NB/Beck...	._HistoryFast
SvenG-NB/Beck...	._HistorySlowPe...

Start Time: 16.01.12 08:00:00.000
End Time: 16.01.12 20:00:00.000
Timespan: 00:01:00
Update Interval: 00:00:01

Attribute	Value
NodeId	NodeId
Namesp...	4
Identifie...	String
Identifier	._HistoryDB
NodeClass	Variable
BrowseName	4, "_HistoryDB"
DisplayName	"en", "_HistoryDB"
Description	"en", ""
WriteMask	BadAttributeIdInvalid (...
UserWriteM...	BadAttributeIdInvalid (...
Value	
SourceT...	2012-01-16 11:37:10.565
ServerTi...	2012-01-16 11:37:10.565
SourceP...	0
ServerPi...	0
Value	-0.067358
DataType	Double
Namesp...	0

Reference	Target DisplayName
HasTypeDefiniti...	BaseVariableType

Timestamp	Source	Server	Message
12:37:12.077	History Plugin	SvenG-NB/Beck...	HistoryReadDataResults of node '._HistorySlowPersist' contains 104 values
12:37:12.076	History Plugin	SvenG-NB/Beck...	HistoryReadDataResults of node '._HistoryFast' contains 10000 values
12:37:12.075	History Plugin	SvenG-NB/Beck...	HistoryReadDataResults of node '._HistoryDB' contains 4010 values

4.1.3.3 Alarms and Conditions

The steps required to configure OPC UA Alarms and Conditions (A&C) on the OPC UA Server are described in this section. The basic concept is independent of TwinCAT runtime, which means the configuration steps for PLC, C++, TcCOM runtime or only I/O task are the same.

OPC UA Alarms and Conditions (Part 9 of the OPC UA specification) describes a model for monitoring process values and outputting alarms and events when a runtime symbol changes its state.

Requirements

The following requirements apply to the use of OPC UA Alarms and Conditions:

- The runtime symbol to be monitored must be available in the namespace.
- The OPC UA Client must support Alarms and Conditions. In this section UA Expert (from Unified Automation) is used as the reference UA Client.

General Information

Execute the following steps once to release a symbol for Alarms and Conditions:

- Step 1: Activating runtime symbol for data access [▶ 58] (so that the symbol is generally accessible via OPC UA.)
- Step 2: Activating A&C for a symbol [▶ 58]
- Step 3: Transferring your own user data with an event [▶ 59]
- Step 4: Triggering an event using the FireEvent method [▶ 60]
- Step 5: Configuring multilingual alarm texts [▶ 62]
- Step 6: Registering A&C with a reference OPC UA Client [▶ 62]

These steps are explained in more detail below. At the end of this section you will find information about receiving configured alarms via A&C with the UA Expert Reference Client.

Supported alarm types

The implementation of OPC UA Alarms and Conditions currently supports the following alarm types:

- LimitAlarmType: Define different limits for a symbol. If a limit is reached, the UA Server issues an alarm.
- OffNormalAlarmType: Define a value that is "normal". If the current value deviates from the "normal" value, the UA Server issues an alarm.

Step 1: Activating runtime symbol for data access

A variable must be available in the OPC UA Server in order for it to be configured for A&C. To do this in the case of a PLC variable, provide it with an attribute (see Access to PLC runtime [▶ 43]).

Step 2: Activating A&C for a symbol

You can configure a runtime symbol for A&C with the OPC UA Server Configurator. The Configurator has a simple graphical user interface for editing the XML file on which it is based. Depending on the setup version the configurator is available in two variants: standalone [▶ 190] and integrated in Visual Studio [▶ 202].

The following program extract shows an example of this XML file to better understand the general behavior and structure of the A&C implementation.

```
<TcUaAcConfig>
  <ConditionController Name="ConditionController1" >
    <Condition Name="Counter" Severity="200">
    <LimitAlarmType LowLowLimit="-10" LowLimit="0" HighLimit="10" HighHighLimit="20"
MessageNormal="100" MessageLowLow="10" MessageLow="11" MessageHigh="12" MessageHighHigh="13"/>
      <ItemToMonitor SamplingRate="100" NS="urn:[NodeName]:BeckhoffAutomation:Ua:PLC1"
NodeId="s=MAIN.nCounter1" />
    </Condition>
    <Condition Name="Switch" Severity="500">
      <OffNormalAlarmType Normal="0" MessageNormal="100" MessageOffNormal="20" />
      <ItemToMonitor SamplingRate="100" NS="urn:[NodeName]:BeckhoffAutomation:Ua:PLC1"
NodeId="s=MAIN.bSwitch" />
    </Condition>
    <Condition Name="Struct" Severity="300">
      <LimitAlarmType LowLowLimit="-10" LowLimit="0" HighLimit="10" HighHighLimit="20"
MessageNormal="100" MessageLowLow="10" MessageLow="11" MessageHigh="12" MessageHighHigh="13"/>
      <ItemToMonitor SamplingRate="100" NS="urn:[NodeName]:BeckhoffAutomation:Ua:PLC1"
NodeId="s=MAIN.stStruct" />
    </Condition>
  </ConditionController>
  <ConditionController Name="ConditionController2" >
    <Condition Name="Counter2" Severity="200">
      <LimitAlarmType LowLowLimit="-10" LowLimit="0" HighLimit="10" HighHighLimit="20"
MessageNormal="100" MessageLowLow="10" MessageLow="11" MessageHigh="12" MessageHighHigh="13"/>
      <ItemToMonitor SamplingRate="100" NS="urn:[NodeName]:BeckhoffAutomation:Ua:PLC1"
NodeId="s=MAIN.nCounter2" />
    </Condition>
  </ConditionController>
</TcUaAcConfig>
```

A "condition" defines the runtime symbol to be monitored and the alarm limits and texts. Each condition is organized in a so-called "ConditionController", the object that the OPC UA Clients subsequently subscribe to.

When creating a condition, you must specify the NamespaceName (NS) and NodeID for referring to the UA node to be monitored. The Configurator provides a simple browsing mechanism for selecting a node. In XML, the placeholder [NodeName] in NamespaceName can be used to switch the XML file between different hardware systems. NamespaceName always contains the host name of the IPC or Embedded PC on which the OPC UA Server is running. If [NodeName] is selected, this tag will be replaced by the host name of the current IPC or Embedded PC on which the UA Server is running.

SamplingRate determines how often the UA Server should request a value from the node to determine whether one of the alarm limits has been reached.

The alarm texts are identified by an ID. The ID uniquely identifies an alarm text from the resource file (see Step 5: Configuring multilingual alarm texts [▸ 62]).

Step 3: Transferring own user data with an alarm

Alarms can contain fields with their own user data, which complement the data output with the alarm. These user data fields can be created and filled out in the runtime application. To do this, create a STRUCT and name its first element "value". In case of an alarm, all the following elements are then sent in an additional user data field.

Sample PLC application:

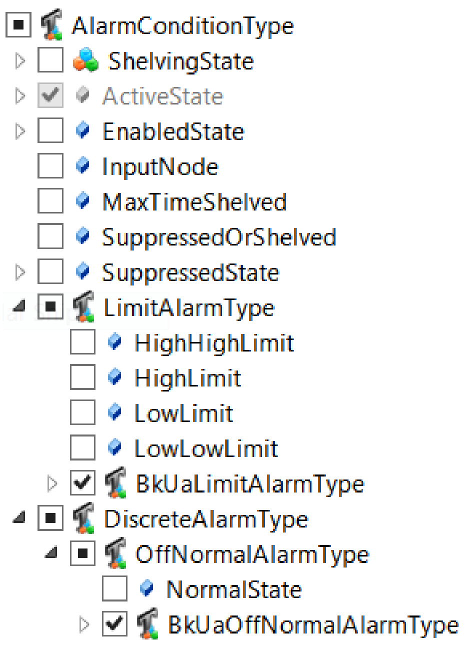
```
TYPE ST_CustomStruct :
STRUCT
  value : INT;
  data  : ST_SomeStruct;
END_STRUCT
END_TYPE

TYPE ST_SomeStruct :
STRUCT
  Data1 : INT;
  Data2 : REAL;
  Data3 : LREAL;
END_STRUCT
END_TYPE
```

The instance of ST_CustomStruct is then enabled for data access via the regular mechanism, e.g. in TwinCAT 3: To do this the STRUCT must be activated as a StructuredType [▸ 68].

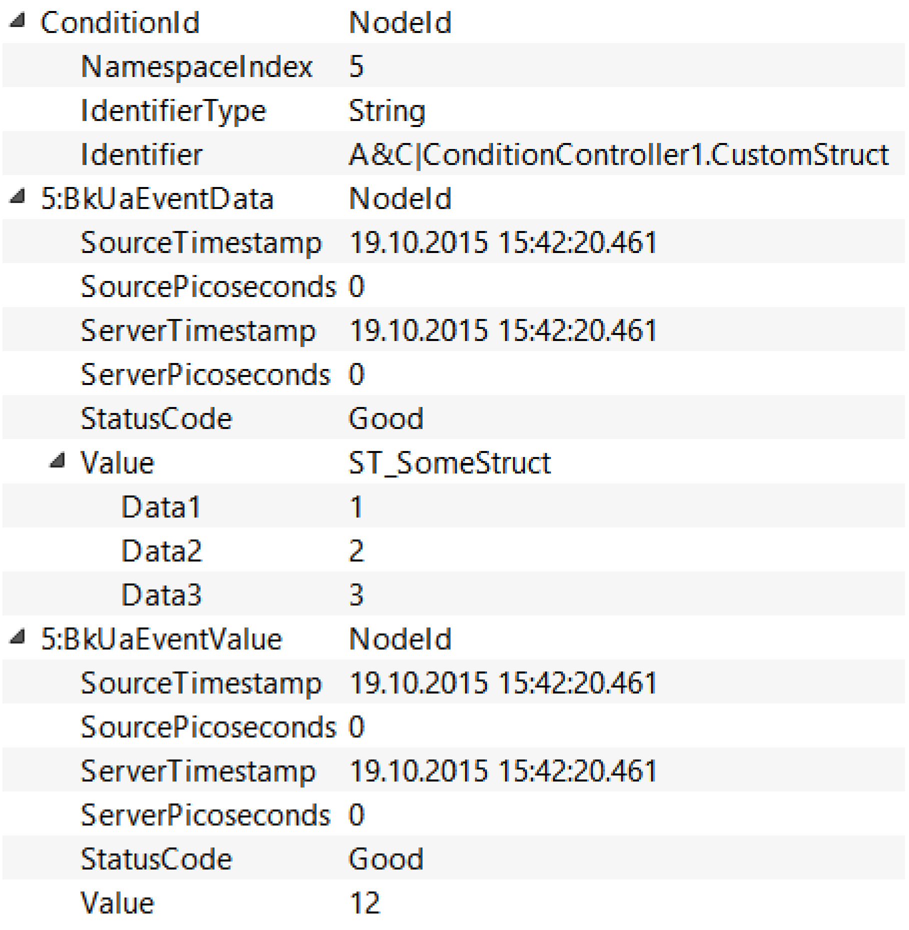
```
PROGRAM MAIN
VAR
  {attribute 'OPC.UA.DA' := '1'}
  {attribute 'OPC.UA.DA.StructuredType' := '1'}
  stCustomStruct : ST_CustomStruct;
END_VAR
```

When logging on to a ConditionController, the OPC UA Clients must subscribe to special AlarmConditionTypes, i.e. "BkUaLimitAlarmType" and "BkUaOffNormalAlarmType", so that they can receive the special user data fields when an alarm is received.



```
■ AlarmConditionType
▷ ☐ ShelvingState
▷ ☑ ActiveState
▷ ☐ EnabledState
  ☐ InputNode
  ☐ MaxTimeShelved
  ☐ SuppressedOrShelved
▷ ☐ SuppressedState
◢ ■ LimitAlarmType
    ☐ HighHighLimit
    ☐ HighLimit
    ☐ LowLimit
    ☐ LowLowLimit
  ▷ ☑ BkUaLimitAlarmType
◢ ■ DiscreteAlarmType
  ◢ ■ OffNormalAlarmType
      ☐ NormalState
    ▷ ☑ BkUaOffNormalAlarmType
```

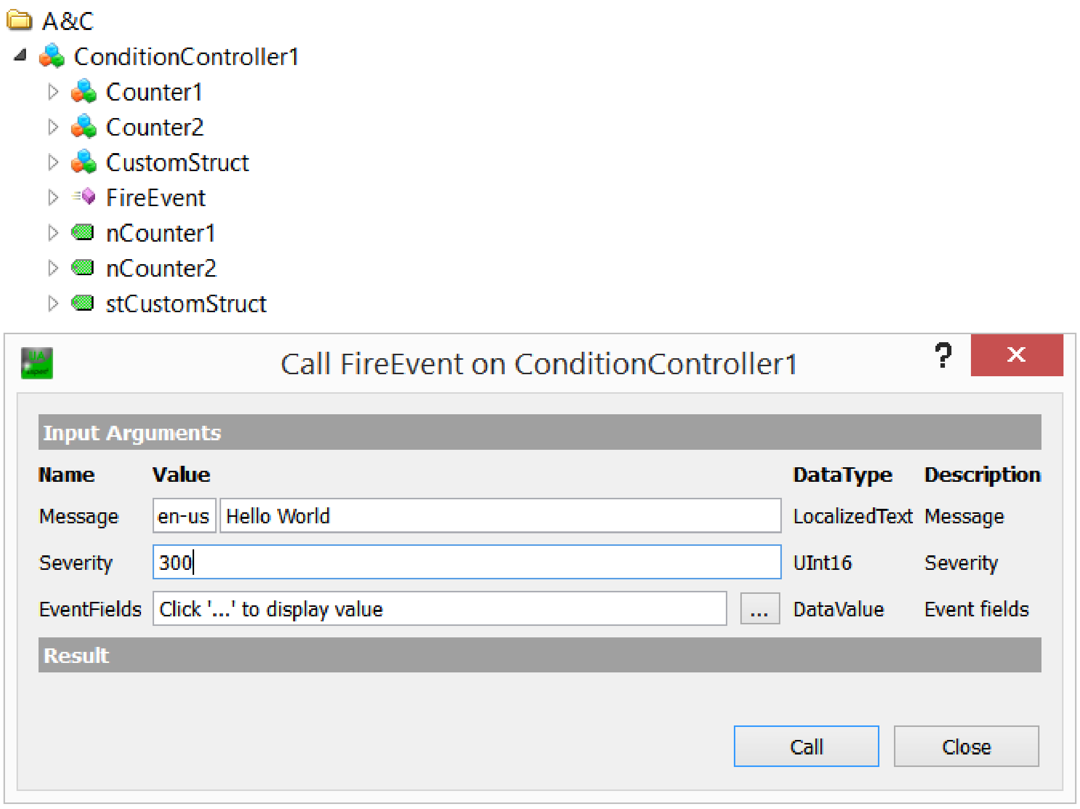
The OPC UA Client then receives the user data in the fields BkUaEventData and BkUaEventValue of the incoming alarm. In the above sample, these are the value of the PLC variables and the user data represented by the PLC structure ST_SomeStruct.



◢ ConditionId	NodeId
NamespaceIndex	5
IdentifierType	String
Identifier	A&C|ConditionController1.CustomStruct
◢ 5:BkUaEventData	NodeId
SourceTimestamp	19.10.2015 15:42:20.461
SourcePicoseconds	0
ServerTimestamp	19.10.2015 15:42:20.461
ServerPicoseconds	0
StatusCode	Good
◢ Value	ST_SomeStruct
Data1	1
Data2	2
Data3	3
◢ 5:BkUaEventValue	NodeId
SourceTimestamp	19.10.2015 15:42:20.461
SourcePicoseconds	0
ServerTimestamp	19.10.2015 15:42:20.461
ServerPicoseconds	0
StatusCode	Good
Value	12

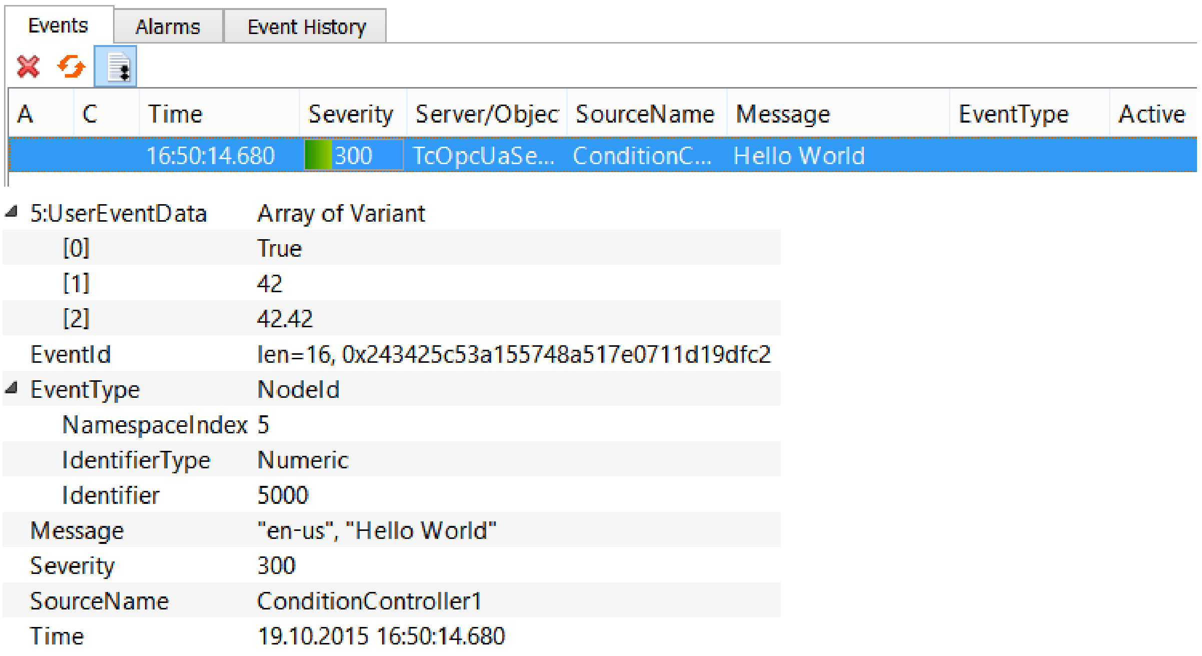
Step 4: Triggering an event using the FireEvent method

Each ConditionController includes a FireEvent method with which the OPC UA Clients can trigger a general event with user-defined EventFields.



Name	Value	DataType	Description
Message	en-us Hello World	LocalizedText	Message
Severity	300	UInt16	Severity
EventFields	Click '...' to display value	DataValue	Event fields

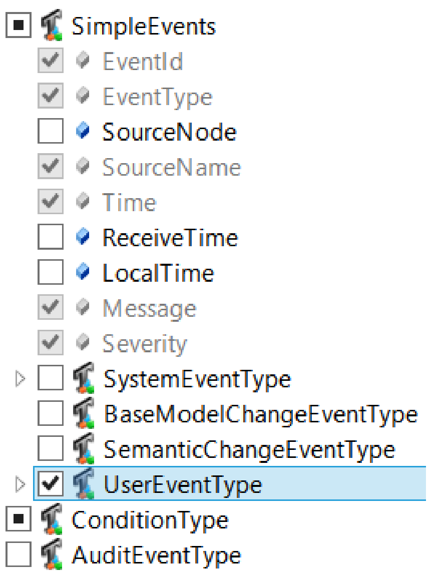
If the method is executed, an event is output on the OPC UA Server. Other OPC UA Clients can receive these events when they subscribe to the corresponding ConditionController.



A	C	Time	Severity	Server/Objec	SourceName	Message	EventType	Active
		16:50:14.680	300	TcOpcUaSe...	ConditionC...	Hello World		

5:UserEventData	Array of Variant
[0]	True
[1]	42
[2]	42.42
EventId	len=16, 0x243425c53a155748a517e0711d19dfc2
EventType	NodeId
NamespaceIndex	5
IdentifierType	Numeric
Identifier	5000
Message	"en-us", "Hello World"
Severity	300
SourceName	ConditionController1
Time	19.10.2015 16:50:14.680

The user-defined EventFields are appended to the event as "UserEventData". This data can be received by OPC UA Clients that are logged on to the SimpleEventType "UserEventType".



Step 5: Configuring multilingual alarm texts

The A&C implementation in the OPC UA Server supports the use of multilingual alarm texts. The alarm text that is used depends on the language with which the UA Client connects to the Server.

Alarm texts are configured in XML files. There is a separate file for each language. These files are located in the res (Resource) folder on the OPC UA Server. With the configurator you can simply add or remove alarm texts without having to directly edit the XML files. Each alarm text has its own ID, which occurs only once in the file.

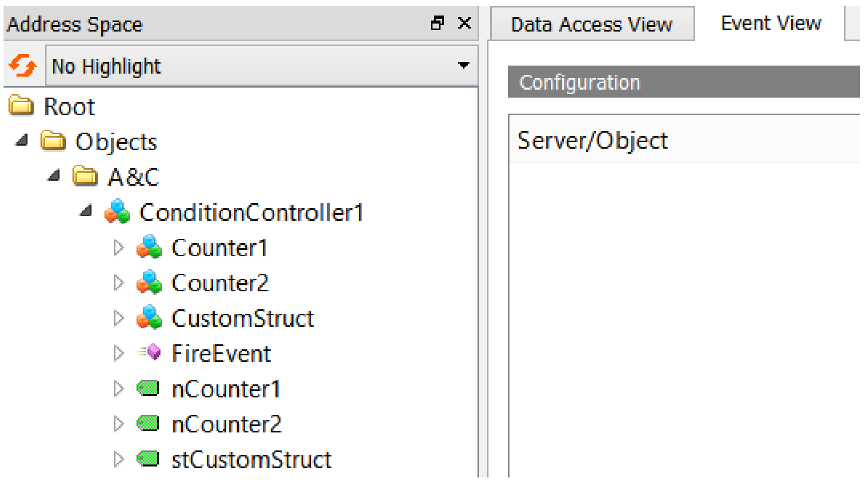
Sample:

```
<TcOpcUaSvrRes Lang="en">
  <Text ID="0">Text not available</Text>
  <Text ID="1">Some alarm text</Text>
  <Text ID="2">Value is High range</Text>
  <Text ID="3">Value is HighHigh range</Text>
  <Text ID="4">Value is OffNormal</Text>
  ...
</TcOpcUaSvrRes>
```

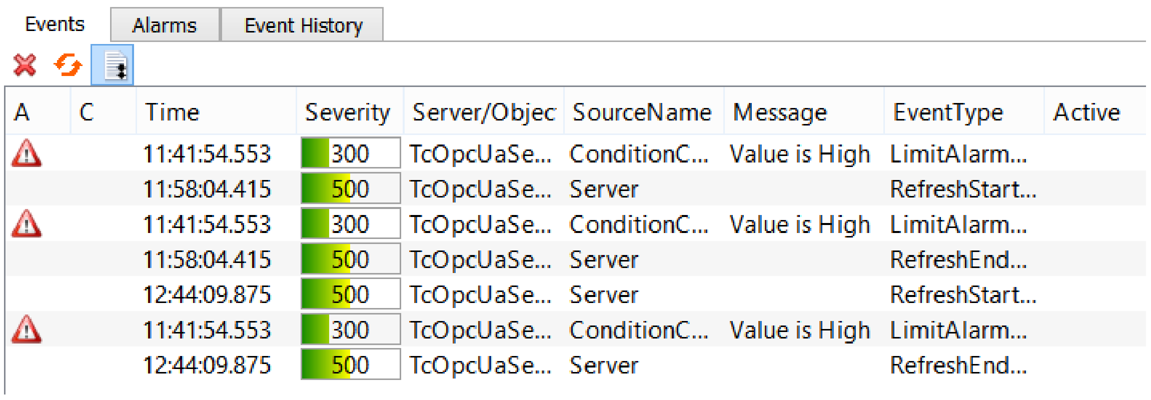
Step 6: Registering A&C with a reference OPC UA Client

An OPC UA Client must log on to a ConditionController so that it can receive events for conditions that are configured for this particular ConditionController. The UA Expert provides functions for subscribing to and receiving UA events.

After you have started the UA Expert and established a connection with the OPC UA Server, add a new document view, Event View, to your working area.



You can then subscribe to a ConditionController by dragging the corresponding object in Event View. The events for this ConditionController are displayed in the Event Window.



A	C	Time	Severity	Server/Objec	SourceName	Message	EventType	Active
⚠		11:41:54.553	300	TcOpcUaSe...	ConditionC...	Value is High	LimitAlarm...	
		11:58:04.415	500	TcOpcUaSe...	Server		RefreshStart...	
⚠		11:41:54.553	300	TcOpcUaSe...	ConditionC...	Value is High	LimitAlarm...	
		11:58:04.415	500	TcOpcUaSe...	Server		RefreshEnd...	
		12:44:09.875	500	TcOpcUaSe...	Server		RefreshStart...	
⚠		11:41:54.553	300	TcOpcUaSe...	ConditionC...	Value is High	LimitAlarm...	
		12:44:09.875	500	TcOpcUaSe...	Server		RefreshEnd...	

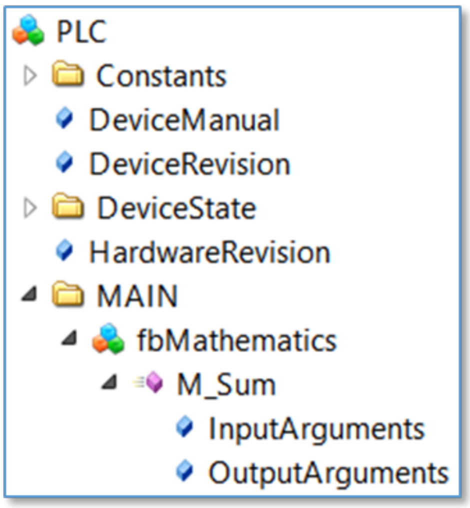
It may be necessary to subscribe to special alarm and/or event types in order to receive all fields of an incoming event or alarm.

4.1.3.4 Method Call

Method calls are a fundamental part of the OPC UA specification. With the introduction of these functionalities into the PLC world, TwinCAT 3 offers the possibility of efficiently executing RPC calls in the IEC61131 world and thus reduces the classic handshake patterns for communication between devices. The OPC UA Server imports PLC methods such as OPC UA methods via its TMC import.

IEC61131 method

Although the PLC method appears to be a normal method in the UA namespace, it is still an IEC61131 method that runs within the real-time context and therefore falls under the context of a real-time task. The PLC developer must therefore take precautions so that the execution time of the method matches the task cycle time.



Methods in IEC61131-3

Methods in the IEC61131 world are always configured below a function block. At high-level language level, the function block can be regarded as the surrounding class of the method. You have to declare the method itself with a special PLC attribute so that the TwinCAT system knows that the method is to be activated for a remote method call.

```
{attribute 'TcRpcEnable':='1'}
METHOD M_Sum : INT
VAR_INPUT
  a : INT;
  b : INT;
END_VAR
```

Filtered or unfiltered mode

Depending on the import mode used, you also have to activate the surrounding function block for the OPC UA access. This can be done by using the normal PLC attributes for OPC UA access.

Note that you only need to use the PLC attributes if the filtered mode is used to enable symbols from the PLC to be accessed via OPC UA. This is the default setting of the OPC UA Server.

Sample:

The method M_Sum is located in the function block FB_Mathematics. The declaration of the function block instance uses the PLC attribute that has enabled the function block and thus the method for OPC UA access.

```
PROGRAM MAIN
VAR
  {attribute 'OPC.UA.DA':='1'}
  fbMathematics : FB_Mathematics;
END_VAR
```

4.1.3.5 AnalogItemTypes

AnalogItemTypes are part of the OPC UA specification and allow meta information such as units to be attached to a variable. You can define these items of meta information in the form of PLC attributes in the TwinCAT 3 PLC.

The following parameters can be set:

- EngineeringUnits: Units defined by the OPC UA specification
- EURange: Maximum value range of the variables
- InstrumentRange: Normal value range of the variables
- WriteBehavior: Behavior if the value range is exceeded during a write operation.

The following sample shows how the fillLevel variable is configured as an AnalogItemType. The following parameters are hereby set:

- Unit: 20529 ("Percent", defined in the OPC UA specification)
- Max. value range: 0 to 100
- Normal value range: 10 to 90
- Write behavior: 1 (Clamping)

```
{attribute 'OPC.UA.DA' := '1'}
{attribute 'OPC.UA.DA.AnalogItemType' := '1'}
{attribute 'OPC.UA.DA.AnalogItemType.EngineeringUnits' := '20529'}
{attribute 'OPC.UA.DA.AnalogItemType.EURange' := '0:100'}
{attribute 'OPC.UA.DA.AnalogItemType.InstrumentRange' := '10:90'}
{attribute 'OPC.UA.DA.AnalogItemType.WriteBehavior' := '1'}
fillLevel : UINT;
```

EngineeringUnits can be configured using the IDs specified in OPC UA (Part 8 of the OPC UA specification). The IDs are based on the widely used and accepted "Codes for Units of Measurement (Recommendation N.20)" published by the "United Nations Center for Trade Facilitation and Electronic Business". CommonCode, which specifies the three-digit alphanumeric ID, is converted by OPC UA according to specification into an Int32 value and referenced (extract from OPC UA specification v1.02, pseudo-code):

```
Int32 unitId = 0;
Int32 c;
for (i=0; i<=3;i++)
{
  c = CommonCode[i];
  if (c == 0)
    break; // end of Common Code
  unitId = unitId << 8; // shift left
  unitId = unitId | c; // OR operation
}
```

Write behavior

When writing an AnalogItemType variable, you can define how the OPC UA Server should handle the new value in relation to the value range. The following options are available:

- 0: All values are allowed and are accepted during a write operation.
- 1: The value to be written is truncated according to the value range.
- 2: The value to be written is rejected if it exceeds the value range.

4.1.3.6 Type system

One of the biggest advantages of OPC UA is the meta-model, which can be used to provide base types as well as to extend the type system with custom models. The same mechanism is used to represent real objects (nodes) so that OPC UA Clients can determine an object type.

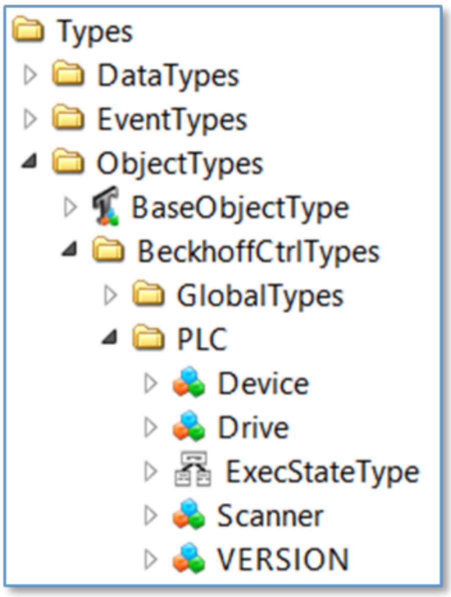
The OPC UA Server publishes type information from the IEC61131 world in its namespace. This includes not only base types such as BOOL, INT, DINT, or REAL, but also extended type information such as the current class (function block) or structure that represents an object.

Type information

Type information is part of the UA namespace. The OPC UA Server extends the basic type information as follows:

- Local type information that is only valid for one runtime is stored in the same namespace as the runtime symbols.
- Global type information that can be valid for different runtimes is stored in a separate global namespace.

The type system is also virtually available and can be viewed in the Types area of the OPC UA Server:

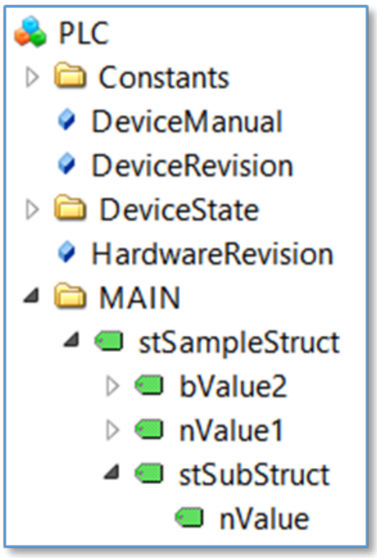


Every non-standard data type is entered in the BeckhoffCtrlTypes area.

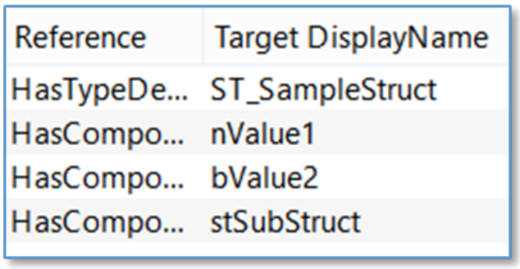
Basic principles

Assuming the TwinCAT 3 PLC consists of a PLC program with different STRUCTs. Each STRUCT is represented as a node in a UA namespace, with each element of the structure as a subordinate node.

In this sample the STRUCT stSampleStruct consists of three subordinate elements: one variable nValue1 of the type INT, one variable bValue2 of the type BOOL and a further STRUCT stSubStruct, which contains only one subordinate element (variable nValue of the type INT).

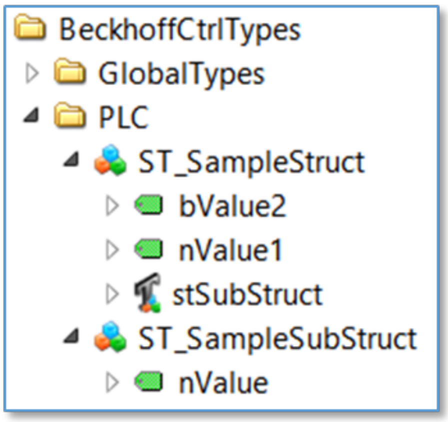


The structure itself is a regular variable in the UA namespace and has the data type ByteString. The clients can therefore simply be connected to the root element (the structure itself), and its values can be read/ written by interpreting the ByteString. To simplify the interpretation of each subordinate element, the type system contains more information about the structure itself, primarily in the instance reference:

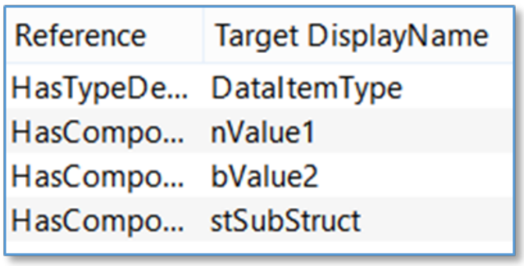


Reference	Target DisplayName
HasTypeDe...	ST_SampleStruct
HasCompo...	nValue1
HasCompo...	bValue2
HasCompo...	stSubStruct

In addition, the type system contains more information about ST_SampleStruct:



And in the references of ST_SampleStruct:

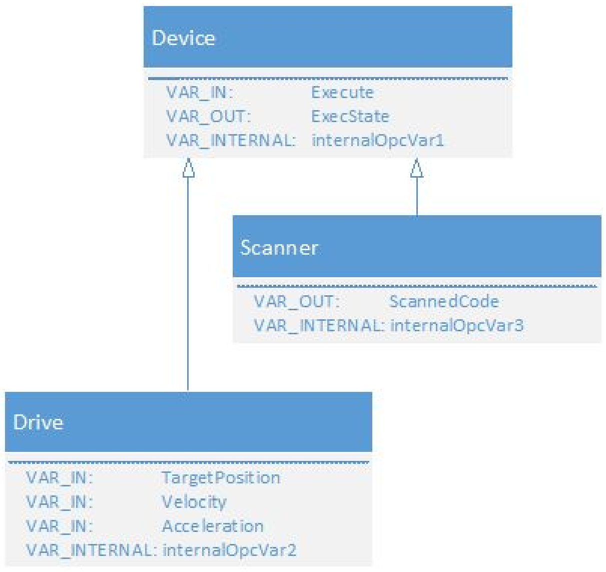


Reference	Target DisplayName
HasTypeDe...	DataItemType
HasCompo...	nValue1
HasCompo...	bValue2
HasCompo...	stSubStruct

Overall, the type system can offer very useful information if a Client wants to interpret the structure further.

Object-orientated extensions

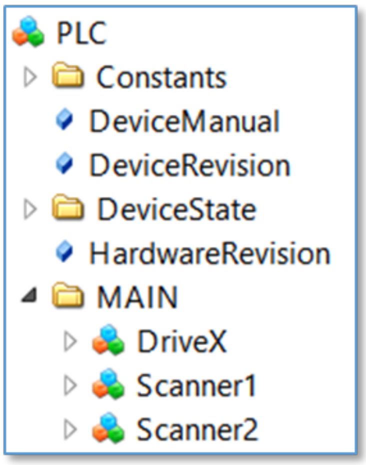
Assuming the TwinCAT 3 PLC runtime contains a PLC program whose structure can be visualized as follows:



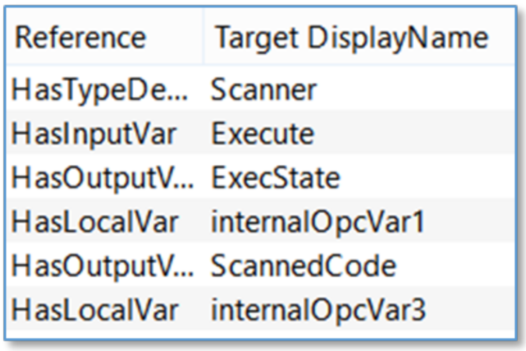
The two function blocks Scanner and Drive are derived from the base class Device by using object-oriented extensions of IEC61131-3. The MAIN program now contains the following declarations:

```
PROGRAM MAIN
VAR
  {attribute 'OPC.UA.DA':='1'}
  Scanner1 : Scanner;
  {attribute 'OPC.UA.DA':='1'}
  Scanner2 : Scanner;
  {attribute 'OPC.UA.DA':='1'}
  DriveX : Drive;
END_VAR
```

All three objects are of type Device, but also of their special data type. The OPC UA Server imports the objects as follows:



The basic data type can now be determined in the reference of each object, e.g. object Scanner1:



Reference	Target DisplayName
HasTypeDe...	Scanner
HasInputVar	Execute
HasOutputV...	ExecState
HasLocalVar	internalOpcVar1
HasOutputV...	ScannedCode
HasLocalVar	internalOpcVar3

According to the basic IEC61131 program, the object Scanner1 is of the data type Scanner and also shows which variables are contained and what type the variable is: input, output or internal (local) variable. The diagram above shows that not only the variables of the actual function block are displayed here, but also the derived variables of the base class. The entire IEC61131-3 inheritance chain is represented in the UA namespace.

4.1.3.7 StructuredTypes

StructuredTypes allow you to read or write structures without interpreting each byte, because the UA Server returns the information type of each element of the structure. Based on complex functions in modern OPC UA SDKs, OPC UA Clients can search and interpret this structural information.

From version 2.2.x of the OPC UA Server, structures of the TwinCAT 3 runtime (TMC and TMI import only) are generated as a StructuredType in the UA namespace.

Sample:

STRUCT ST_Communication:

```
TYPE ST_Communication :
STRUCT
  a : INT;
  b : INT;
  c : INT;
END_STRUCT
END_TYPE
```

Program MAIN:

```
PROGRAM MAIN
VAR
  {attribute 'OPC.UA.DA' := '1'}
  {attribute 'OPC.UA.DA.StructuredType' := '1'}
  stCommunication : ST_Communication;
END_VAR
```

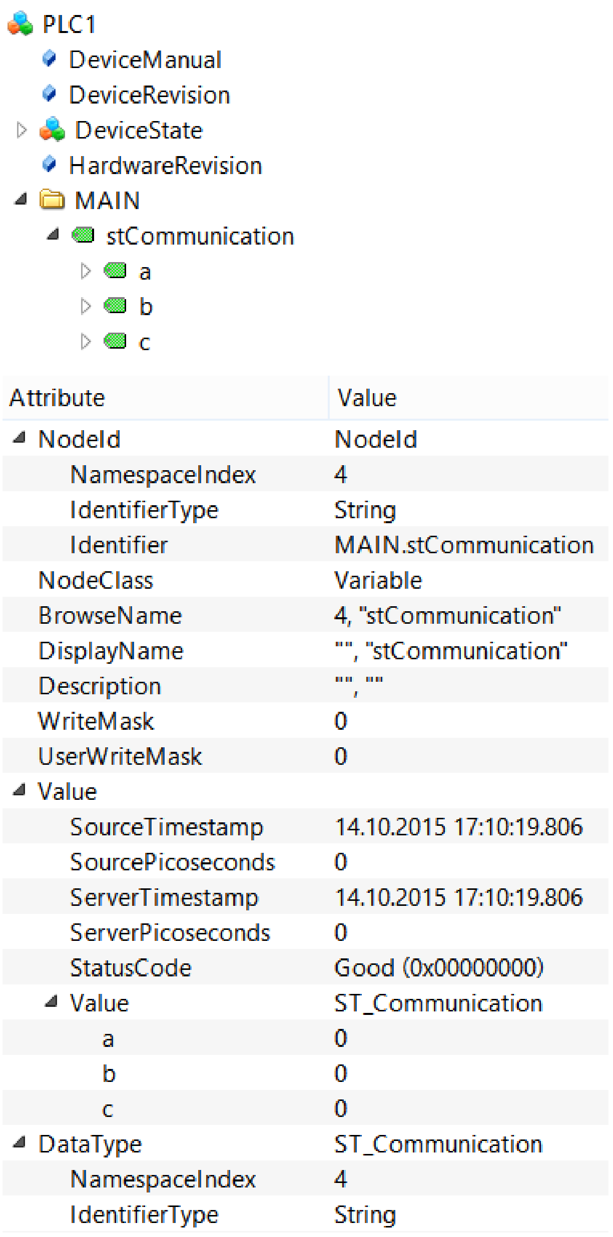
Filtered mode

If filters are used to make symbols available via OPC UA, a STRUCT or function block must be fully available in the UA namespace in order to be displayed as a StructuredType.

Pointers and references

If pointers and references are used in the structure, then they cannot be converted into a StructuredType. The OPC UA Server then illustrates these structures as regular FolderTypes with the corresponding member variables.

The instance stCommunication is then displayed in the UA namespace as a StructuredType:



- PLC1
  - DeviceManual
  - DeviceRevision
  - DeviceState
  - HardwareRevision
  - MAIN
    - stCommunication
      - a
      - b
      - c

Attribute	Value
NodeId	NodeId
NamespaceIndex	4
IdentifierType	String
Identifier	MAIN.stCommunication
NodeClass	Variable
BrowseName	4, "stCommunication"
DisplayName	"", "stCommunication"
Description	"", ""
WriteMask	0
UserWriteMask	0
Value	
SourceTimestamp	14.10.2015 17:10:19.806
SourcePicoseconds	0
ServerTimestamp	14.10.2015 17:10:19.806
ServerPicoseconds	0
StatusCode	Good (0x00000000)
Value	ST_Communication
a	0
b	0
c	0
DataType	ST_Communication
NamespaceIndex	4
IdentifierType	String

Alternatively, the STRUCT definition can also be assigned the PLC attribute to make all instances of STRUCT available as StructuredType.

```
{attribute 'OPC.UA.DA.StructuredType' := '1'}
TYPE ST_Communication :
STRUCT
  a : INT;
  b : INT;
  c : INT;
END_STRUCT
END_TYPE
```

In order to deactivate StructuredType of a certain instance, use the following attribute:

```
PROGRAM MAIN
VAR
  {attribute 'OPC.UA.DA' := '1'}
  {attribute 'OPC.UA.DA.StructuredType' := '0'}
  stCommunication : ST_Communication;
END_VAR
```

Function block StructuredType

In addition, each function block of the TwinCAT 3 PLC also contains a child node, FunctionBlock, which contains the entire function block as a StructuredType.

Sample:

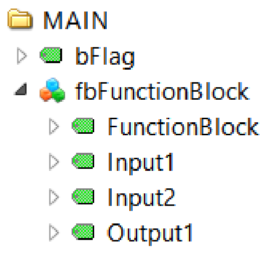
Function block:

```
FUNCTION_BLOCK FB_FunctionBlock
VAR_INPUT
  Input1 : INT;
  Input2 : LREAL;
END_VAR
VAR_OUTPUT
  Output1 : LREAL;
END_VAR
```

Instance of the function block:

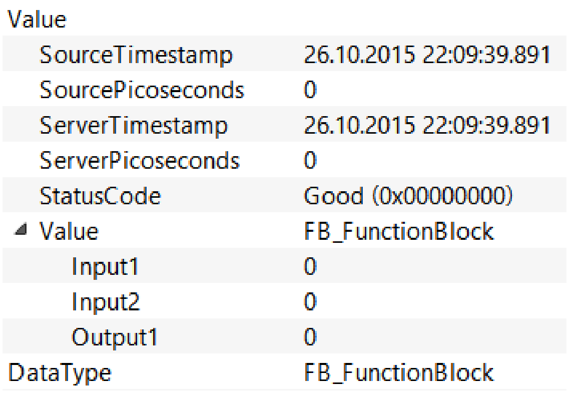
```
PROGRAM MAIN
VAR
  {attribute 'OPC.UA.DA' := '1'}
  {attribute 'OPC.UA.DA.StructuredType' := '1'}
  fbFunctionBlock : FB_FunctionBlock;
END_VAR
```

Instance of the function block in the OPC UA namespace:



```
MAIN
  ▷ bFlag
  ◢ fbFunctionBlock
      ▷ FunctionBlock
      ▷ Input1
      ▷ Input2
      ▷ Output1
```

FunctionBlock node with StructuredType:



Value	
SourceTimestamp	26.10.2015 22:09:39.891
SourcePicoseconds	0
ServerTimestamp	26.10.2015 22:09:39.891
ServerPicoseconds	0
StatusCode	Good (0x00000000)
◢ Value	FB_FunctionBlock
Input1	0
Input2	0
Output1	0
DataType	FB_FunctionBlock

Alternatively, the function block can also receive a PLC attribute to make all instances of the function block available as StructuredType.

```
{attribute 'OPC.UA.DA.StructuredType' := '1'}
FUNCTION_BLOCK FB_FunctionBlock
VAR_INPUT
  Input1 : INT;
  Input2 : LREAL;
END_VAR
VAR_OUTPUT
  Output1 : LREAL;
END_VAR
```

In order to deactivate StructuredType of a certain instance, use the following attribute:

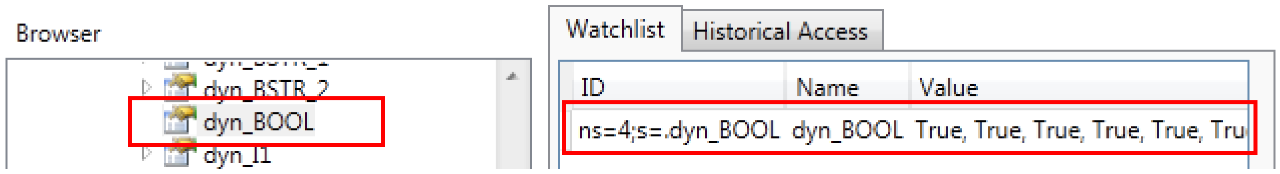
```
PROGRAM MAIN
VAR
  {attribute 'OPC.UA.DA' := '1'}
  {attribute 'OPC.UA.DA.StructuredType' := '0'}
  fbFunctionBlock : FB_FunctionBlock;
END_VAR
```

Maximum size of the structure

The maximum size of a structure is 16 kByte by default. Each STRUCT constantly exchanges data with the basic ADS device, i.e. a large ADS message is sent with each read/write command of a StructuredType. To prevent the ADS router from being flooded with large messages, the maximum size is limited. You can change this specification in the file *ServerConfig.xml*. To do this, the key <MaxStructureSize> must be added to OpcServerConfig\UaServerConfig\, and a new value for the maximum size of a structure must be set in bytes. If a structure exceeds <MaxStructureSize>, it is imported as FolderType, where each structure item is available as a single node.

4.1.3.8 Use of arrays

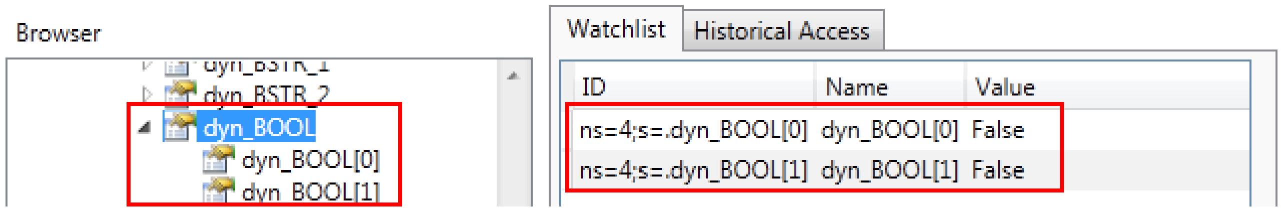
By default arrays are regarded as individual nodes in the UA namespace. This means that if you define, for example, an array dyn_BOOL[10] in the PLC (and have also enabled it for OPC UA), it will subsequently appear in the UA namespace as follows:



ID	Name	Value
ns=4;s=.dyn_BOOL	dyn_BOOL	True, True, True, True, True, Tru

The advantage of this approach is a considerable reduction in the complexity of the UA namespace and in memory consumption, since not every position of an array needs to be made available as an individual node in the namespace. However, modern UA Clients can continue to access the individual array positions via the so-called "RangeOffset".

In order to support older UA Clients that don’t offer this feature, however, you can also make the positions of an array available as individual nodes in the UA namespace. It is illustrated as follows:



ID	Name	Value
ns=4;s=.dyn_BOOL[0]	dyn_BOOL[0]	False
ns=4;s=.dyn_BOOL[1]	dyn_BOOL[1]	False

This setting is available by activating the **Legacy Array Handling** option in the UA Configurator within the respective namespace configuration.

Depending on the scope of the PLC project, the UA namespace can become significantly more complex, which in turn is reflected in an increased memory utilization of the UA Server.

Changes to the settings listed above only become active after restarting the UA Server.

4.1.3.9 List of attributes and comments

The runtime variables for various OPC UA functions (e.g. data access or historical access) are configured directly in the PLC program or in the TMC code editor (if using TwinCAT 3 C++). The advantage is that PLC developers can decide directly in the program with which they are familiar whether and how a variable is to be enabled for OPC UA. You activate a variable by inserting a comment and the corresponding OPC UA tag in front of the variable, e.g.:

TwinCAT 3 PLC (TMC):

```
{attribute 'OPC.UA.DA' := '1'}
bVariable : BOOL;
```

TwinCAT 2 PLC (TPY):

```
bVariable : BOOL; (*~ (OPC:1:available)*)
```

You can find a detailed description of the use of attributes and comments in the sections concerning the corresponding runtime components:

- PLC [▸ 43]
- C++ [▸ 75]
- I/O [▸ 94]
- Matlab/Simulink [▸ 106]

The following table shows an overview of all definable tags and their meaning. The subsections of the corresponding function contain a detailed description of the functional principle.

TwinCAT 3 (TMC):

OPC UA function	**PLC tag**	**C++ TMC code editor (Optional features)**	**Meaning**
Data Access (DA)	{attribute 'OPC.UA.DA' := '0'}	Name: OPC.UA.DA Value: 0	Locks a variable for OPC UA, whereupon it is no longer visible in the UA namespace.
Data Access (DA)	{attribute 'OPC.UA.DA' := '1'}	Name: OPC.UA.DA Value: 1	Enables a variable for OPC UA, whereupon it becomes visible in the UA namespace. This tag must always be set if you want to use a variable for UA.
Data Access (DA)	{attribute 'OPC.UA.DA.Access' := 'x'}	Name: OPC.UA.DA.Access Value: see right column	Sets read/write access for a variable, depending on parameter "x". 0 = none 1 = read-only 2 = write access only 3 = read and write access (default if no tag is used)
Data Access (DA)	{attribute 'OPC.UA.DA.Description' := 'x'}	Name: OPC.UA.DA.Description Value: see right column	Sets a text for the OPC UA attribute "Description".
Data Access (DA)	{attribute 'OPC.UA.DA.StructuredType' := '0'}	Name: OPC.UA.DA.StructuredType Value: 0	Disables StructuredType for a STRUCT or a function block
Data Access (DA)	{attribute 'OPC.UA.DA.StructuredType' := '1'}	Name: OPC.UA.DA.StructuredType Value: 1	Enables StructuredType for a STRUCT or a function block
Data Access	{attribute 'OPC.UA.DA.Status' := 'quality'}	Name: OPC.UA.DA.Status Value: quality	Manually define the StatusCode of a symbol in the UA namespace. Only for data structures. The value "quality" specifies which DINT subelement of the data structure determines the StatusCode . See corresponding article in the documentation for more information.
Historical Access (HA) [▸ 49]	{attribute 'OPC.UA.HA' := '1'}	Name: OPC.UA.HA Value: 1	Enables a variable for Historical Access. Must be used with {attribute 'OPC.UA.DA' := '1'}.

OPC UA function	PLC tag	C++ TMC code editor (Optional features)	Meaning
Historical Access (HA) [▶ 49]	{attribute 'OPC.UA.HA.Storage' := 'x'}	Name: OPC.UA.HA.Storage Value: see right column	Defines the storage location for Historical Access, depending on parameter "x" 1 = RAM 2 = File 3 = SQL Compact Database 4 = SQL Server Database
Historical Access (HA) [▶ 49]	{attribute 'OPC.UA.HA.Sampling' := 'x'}	Name: OPC.UA.HA.Sampling Value: see right column	Determines the sampling rate at which the variable values are to be stored, in [ms] depending on the parameter "x"
Historical Access (HA) [▶ 49]	{attribute 'OPC.UA.HA.Buffer' := 'x'}	Name: OPC.UA.HA.Buffer Value: see right column	Defines the maximum number of values that remain in the data memory, depending on the parameter "x".

TwinCAT 2 (TPY):

OPC UA function	PLC tag	Meaning
Data Access (DA)	(*~ (OPC:0:not available) *)	Locks a variable for OPC UA, whereupon it is no longer visible in the UA namespace.
Data Access (DA)	(*~ (OPC:1:available) *)	Enables a variable for OPC UA, whereupon it becomes visible in the UA namespace. This tag must always be set if you want to use a variable for UA.
Data Access (DA)	(*~ (OPC_PROP[0005]:1: read-only) *)	Sets the write protection for a variable. Must be used together with (*~ (OPC: 1: available) *).
Data Access (DA)	(*~ (OPC_UA_PROP[5100] : x: Alias name) *)	Specifies x as node name in the UA namespace, so-called alias mapping.
Historical Access (HA) [▶ 49]	(*~ (OPC_UA_PROP[5000]:x:Storage media) *)	Enables a variable for "Historical Access". Must be used together with (*~ (OPC: 1: available) *). x defines the storage medium for saving the data values: 1 = RAM 2 = File 3 = SQL Compact Database 4 = SQL Server Database
Historical Access (HA) [▶ 49]	(*~ (OPC_UA_PROP[5000][1]:x:SamplingRate) *)	Determines the sampling rate at which the variable values are to be stored, in [ms] depending on the parameter "x"
Historical Access (HA) [▶ 49]	(*~ (OPC_UA_PROP[5000][2]:x:Buffer) *)	Defines the maximum number of values that remain in the data memory, depending on the parameter "x".

4.1.3.10 StatusCode

The OPC UA Server enables a PLC application to change the OPC UA StatusCode.

Perform the following steps to configure and affect the StatusCode of a variable:

- Creating a PLC structure [▶ 74]
- Overwriting StatusCode [▶ 74]

Creating a PLC structure

So that the data consistency is guaranteed, the concept of changing the OPC UA StatusCode is based on StructuredTypes (see StructuredTypes [▶ 68]). Each variable whose UA StatusCode is to be affected must be included in a STRUCT.

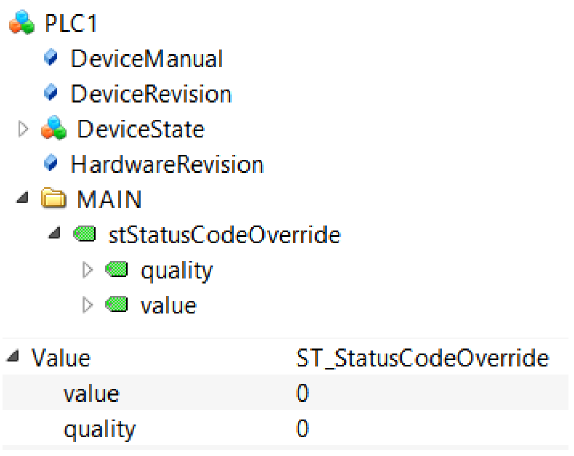
Create a new STRUCT and add a PLC attribute before the STRUCT definition. The STRUCT contains the variable itself and a variable of the data type DINT, which represents the StatusCode and to which reference is made in the attribute in front of the STRUCT definition.

```
{attribute 'OPC.UA.DA.STATUS' := 'quality'}
TYPE ST_StatusCodeOverride :
STRUCT
  value   : REAL;
  quality: DINT;
END_STRUCT
END_TYPE
```

Now create an instance of this STRUCT, for example in the MAIN program, and add the regular PLC attribute so that this instance becomes available via OPC UA.

```
PROGRAM MAIN
VAR
  {attribute 'OPC.UA.DA' := '1'}
  stStatusCodeOverride : ST_StatusCodeOverride;
END_VAR
```

This instance is now available inside the OPC UA namespace as a StructuredType.

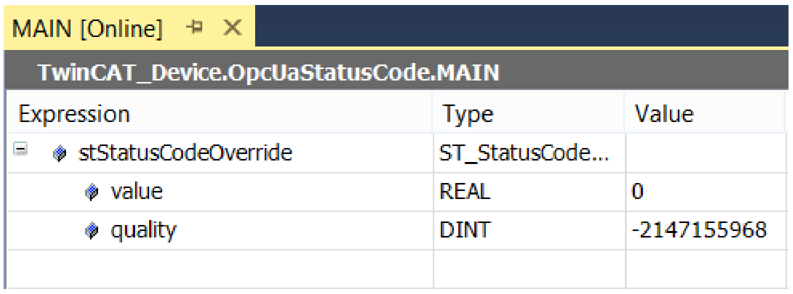


```
PLC1
  DeviceManual
  DeviceRevision
  DeviceState
  HardwareRevision
  MAIN
    stStatusCodeOverride
      quality
      value
```

Value	ST_StatusCodeOverride
value	0
quality	0

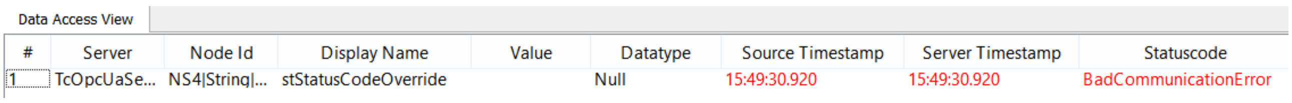
Overwriting StatusCode

To overwrite the UA StatusCode for the STRUCT, simply edit the value of the variable "quality". If you set this, for example, to "-2147155968", the StatusCode of the STRUCT changes to "BadCommunicationError".



TwinCAT_Device.OpcUaStatusCode.MAIN

Expression	Type	Value
stStatusCodeOverride	ST_StatusCode...	
value	REAL	0
quality	DINT	-2147155968



Data Access View

#	Server	Node Id	Display Name	Value	Datatype	Source Timestamp	Server Timestamp	Statuscode
1	TcOpcUaSe...	NS4|String|...	stStatusCodeOverride		Null	15:49:30.920	15:49:30.920	BadCommunicationError

The value must be determined according to the definition in the OPC UA specification.

The following table lists only some of the available StatusCodes and their corresponding decimal representation. Consult the official OPC UA specification for a more comprehensive list of the StatusCodes.

StatusCode	Hex	Decimal
BadUnexpectedError	0x80010000	-2147418112
BadInternalError	0x80020000	-2147352576
BadCommunicationError	0x80050000	-2147155968
BadTimeout	0x800A0000	-2146828288
BadServiceNotSupported	0x800B0000	-2146762752

UA StatusCodes

When calculating the decimal representation of other UA StatusCodes on the basis of their hexadecimal representation, make sure that your computer is set to DWORD, e.g. the Windows computer ("Programmer" view).

4.1.4 C++

4.1.4.1 Access to C++ runtime

This section describes how to configure the namespace of the TwinCAT OPC UA Server in order to obtain access to the symbols of a TwinCAT 3 C++ module. For this you can either use the OPC UA Configurator or carry out the configuration directly in the file *ServerConfig.xml* of the OPC UA Server.

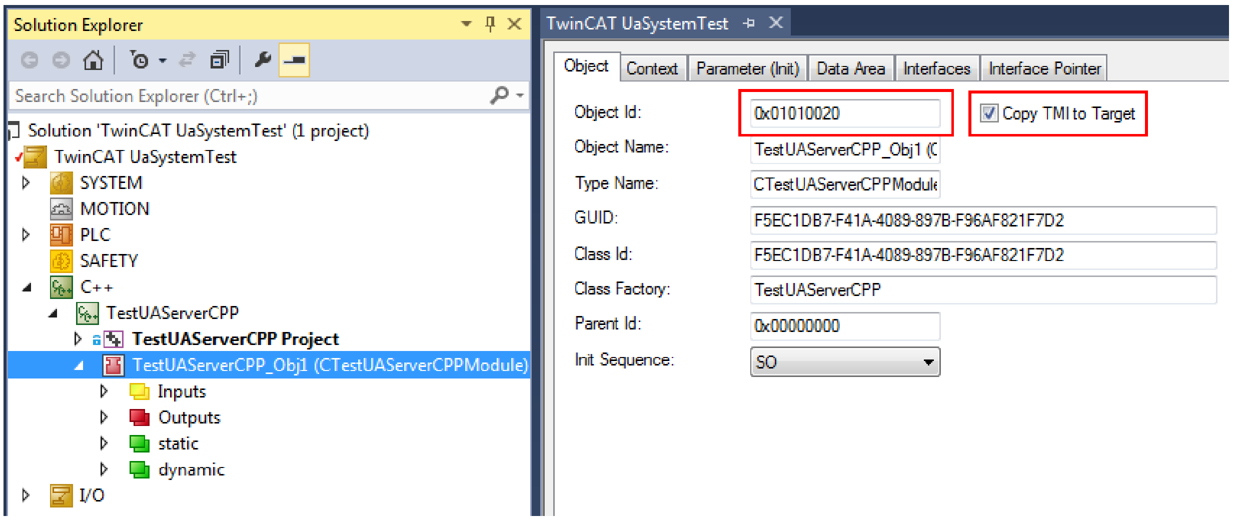
This section contains the following topics:

- Step 1: Project-related settings [▸ 75]
- Step 2: Configuring the UA Server [▸ 76]

Step 1: Project-related settings

To configure certain symbols contained in an instance of a C++ module in such a way that they are accessible via OPC UA, the following settings are necessary in the corresponding C++ module instance in TwinCAT XAE.

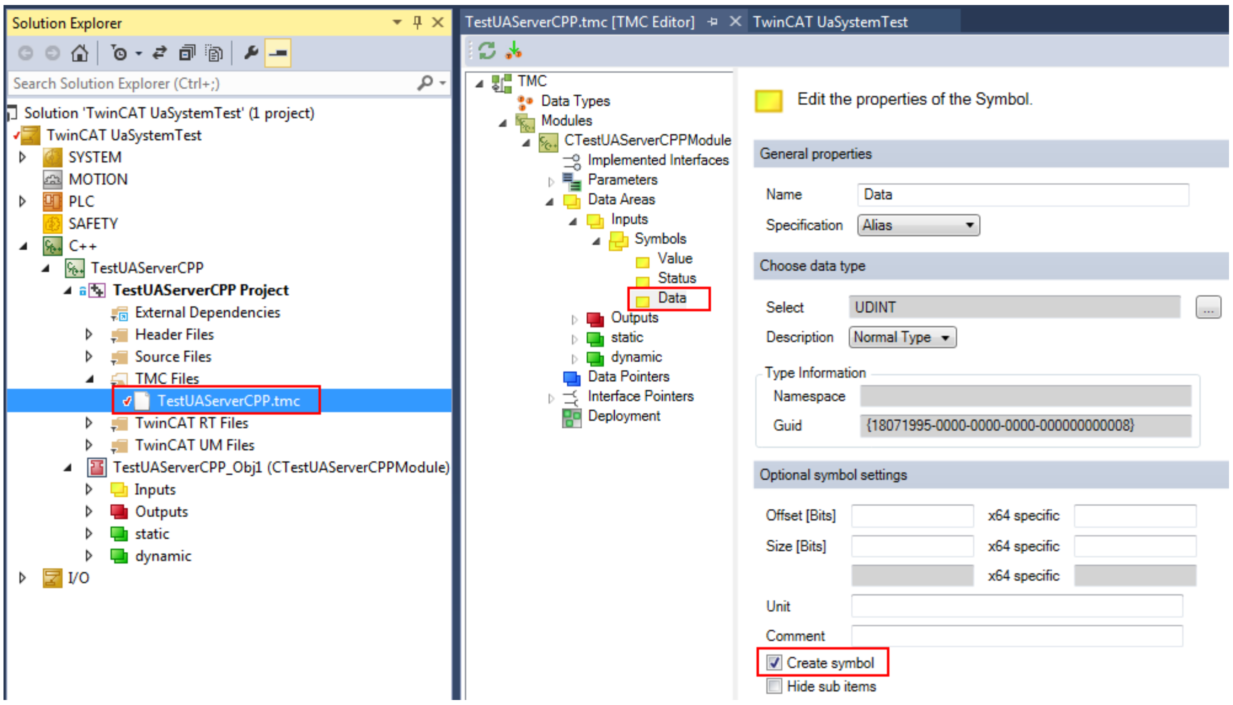
1. The OPC UA Server requires the TMI symbol file, which is not passed to the target runtime by default. Enable the transfer by setting the following options:



Object Id: 0x01010020
☑ Copy TMI to Target
Object Name: TestUAServerCPP_Obj1 (C
Type Name: CTestUAServerCPPModule
GUID: F5EC1DB7-F41A-4089-897B-F96AF821F7D2
Class Id: F5EC1DB7-F41A-4089-897B-F96AF821F7D2
Class Factory: TestUAServerCPP
Parent Id: 0x00000000
Init Sequence: SO

⇨ The generated TMI file is named after the Object ID, e.g. "Obj_01010020.tmi" and stored in the TwinCAT boot directory, e.g. *C:\TwinCAT\Boot\Tmi\*.

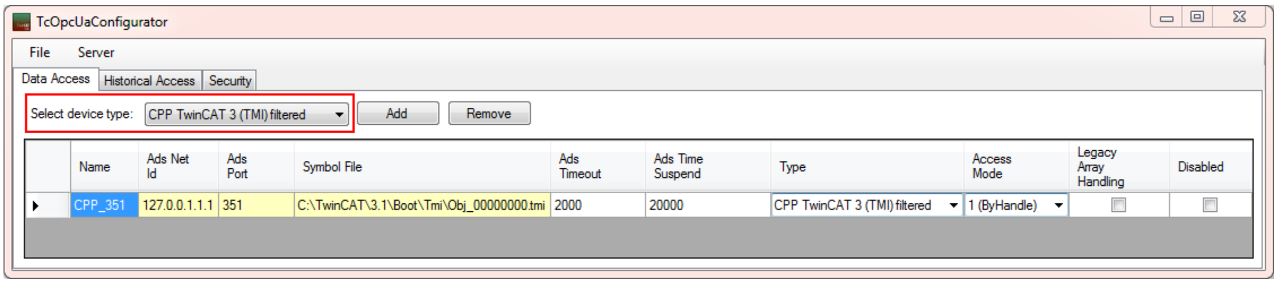
2. Select which symbols are to be made accessible to the corresponding variables by checking the **Create symbol** checkbox in the TMC code generator. Then execute the TMC CodeGenerator.



Name: Data
Specification: Alias
Select: UDINT
Description: Normal Type
Namespace:
Guid: {18071995-0000-0000-0000-000000000008}
Offset [Bits]: x64 specific:
Size [Bits]: x64 specific:
x64 specific:
Unit:
Comment:
☑ Create symbol
☐ Hide sub items

Step 2: Configuring the UA Server

You can configure and parameterize the access to TwinCAT 3 C++ modules simply with the help of the OPC UA Configurator. To do this, add a new device of the type "CPP TwinCAT 3 (TMI) filtered" in the **Data Access** area.



Select device type: CPP TwinCAT 3 (TMI) filtered

Name	Ads Net Id	Ads Port	Symbol File	Ads Timeout	Ads Time Suspend	Type	Access Mode	Legacy Array Handling	Disabled
CPP_351	127.0.0.1.1.1	351	C:\TwinCAT\3.1\Boot\Tmi\Obj_00000000.tmi	2000	20000	CPP TwinCAT 3 (TMI) filtered	1 (ByHandle)	☐	☐

The **Symbol File** field must point to the TMI file created for the C++ module instance. This TMI file is located in the TwinCAT boot directory, e.g. *C:\TwinCAT\Boot\Tmi\*, and is named after the ObjectID of the TwinCAT 3 C++ module instance.

The configurable parameters describe the following functions:

Parameter	Description	Possible values
ADS Port	Defines the ADS port under which the C++ module instance is accessible. The ADS port can be read in the properties of the C++ task.	351 352 …

Parameter	Description	Possible values
AutoCfg	First defines the type of the target runtime used for communication, e.g. PLC, C++, I/O. A further distinction can then be made within these categories. Each AutoCfg option is available as an unfiltered or filtered option. Filtered means the user can determine which C++ symbols are made publicly accessible via OPC UA. When using an unfiltered option, each C++ symbol is published to OPC UA.	4020 CPP TwinCAT 3 (TMI) 4021 CPP TwinCAT 3 (TMI) filtered
AutoCfgSymFile	Symbol file (TMI) of the corresponding C++ module instance.	Path to the symbol file. (TMI)
IoMode	Defines the method for accessing symbols.	1 (access via handle - default)
ArraySubItemLegacy Support	By default, subelements of an array are not mapped as separate nodes in the UA namespace. Instead, only the array is mapped as a single element. Nonetheless, UA Clients can access subelements via their "IndexRange". (Some older OPC UA Clients do not yet support this option). The flag was introduced so that access is nevertheless possible for these clients. It ensures that every array position is displayed as a separate node in the UA namespace. This leads to higher memory requirement of the OPC UA Server.	0 (disabled - default) 1 (enabled)
Disabled	Disables the C++ module instance in the UA namespace, so that the corresponding node is not displayed. It is advisable to enable this parameter if certain C++ runtimes are not yet available at the time of project planning, for example because the corresponding devices are not yet connected to the network.	0 (disabled - default) 1 (enabled)

4.1.4.2 Historical Access

4.1.4.2.1 Setup Version 4.x.x

From setup version 4.x.x historical access can no longer be configured via attributes or comments. The advantage of this is that the Historical Access functionality is now also available for runtimes that previously did not support this, such as variables of a BC9191 Controller. Furthermore, a Historical Access configuration can now also be carried out retrospectively, i.e. without having to adapt the PLC project.

The historical access configuration is now XML-based (TcUaHaConfig.xml) and can be carried out using the OPC UA Server Configurator. The XML file is located in the same directory as *TcOpcUaServer.exe*.

See also:

- Configuration > Setup version 4.x.x > Overview [▶ 192] (OPC UA Server Configurator) and Configuring historical access [▶ 200])

XML file structure (TCUaHaConfig.xml)

```
<HistoryConfig>
  <HistoryController>
    <HistoryAdapter Type="Volatile" Id="0" File="" Server="" Database="" User="" Password=""/>
    <HistoryAdapter Type="File" Id="1" File="historydb.dat" Server="" Database="" User=""
Password=""/>
    <HistoryAdapter Type="SQL" Id="2" File="" Server="" Database="database123" User="user"
Password="pwd"/>
    <HistoryAdapter Type="SQLCompact" Id="3" File="historydb.xyz" Server="" Database="" User=""
Password=""/>
    <HistoryAdapter Type="SQLCompact" Id="4" File="historydb2.xyz" Server="" Database="" User=""
Password=""/>
  </HistoryController>
  <HistoryNodes>
    <HistoryNode HistoryWriteable="true" SamplingRate="0" MaxSamples="100000" AdapterId="0"
NS="urn:BeckhoffAutomation:Ua:PLC1" NodeId="s=PRG_HA_SingleNode.nMyValue"/>
    <HistoryNode HistoryWriteable="true" SamplingRate="0" MaxSamples="100000" AdapterId="0"
NS="urn:BeckhoffAutomation:Ua:PLC1" NodeId="s=PRG_HA_MultipleNodes.nMyValue1"/>
    <HistoryNode HistoryWriteable="true" SamplingRate="0" MaxSamples="100000" AdapterId="0"
NS="urn:BeckhoffAutomation:Ua:PLC1" NodeId="s=PRG_HA_MultipleNodes.nMyValue2"/>
    <HistoryNode HistoryWriteable="true" SamplingRate="0" MaxSamples="100000" AdapterId="0"
NS="urn:BeckhoffAutomation:Ua:PLC1" NodeId="s=PRG_HA_MultipleNodes.fMyValue3"/>
  </HistoryNodes>
</HistoryConfig>
```

You can configure all supported data memory in the HistoryController area. The following locations are currently supported:

- RAM ("Volatile")
- File
- SQL Compact
- SQL Server (not supported under Windows CE)

Each HistoryAdapter (except for "Volatile") can be used multiple times, e.g. if the historical values for individual variables are to be stored in different data memories.

In the HistoryNodes area, the individual nodes are configured, assigned to a data memory and assigned a SamplingRate and a maximum size for the ring buffer to be used in the data memory.

The attribute `HistoryWriteable="true"` ensures that the data memory for this node can be filled with values via a HistoryUpdate() call. Such a call is provided, for example, by the TwinCAT OPC UA Client via the function block UA_HistoryUpdate. If you configure a node with this attribute, then the server doesn't normally "sample" the values itself, but receives them from other clients. In such a configuration the SamplingRate is therefore usually 0.

Requirements

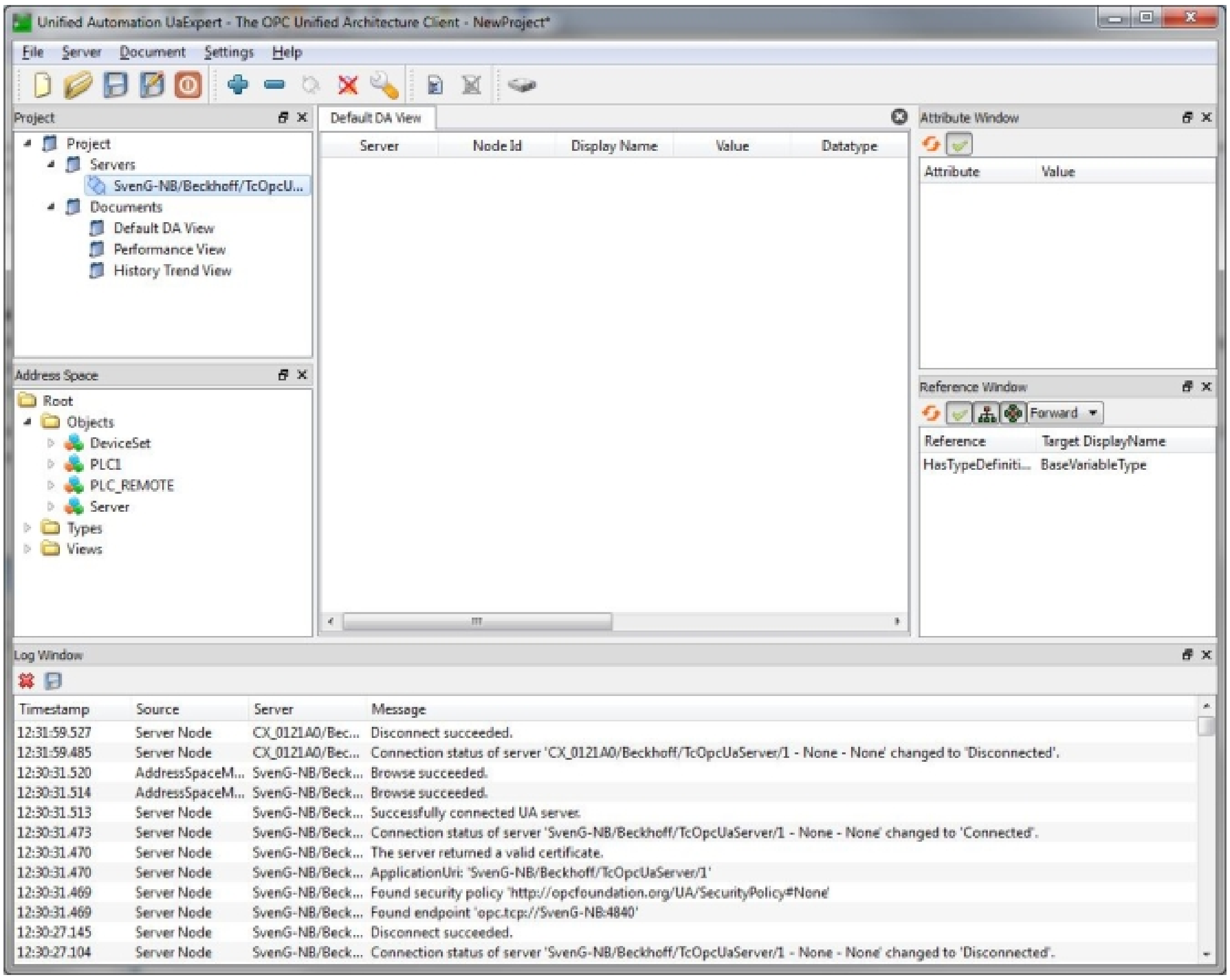
Products	Setup versions	Target platform
TF6100	4.x.x	IPC or CX (x86, x64, ARM)

4.1.4.2.2 Display historical data

Displaying Historical Access values in an OPC UA Client

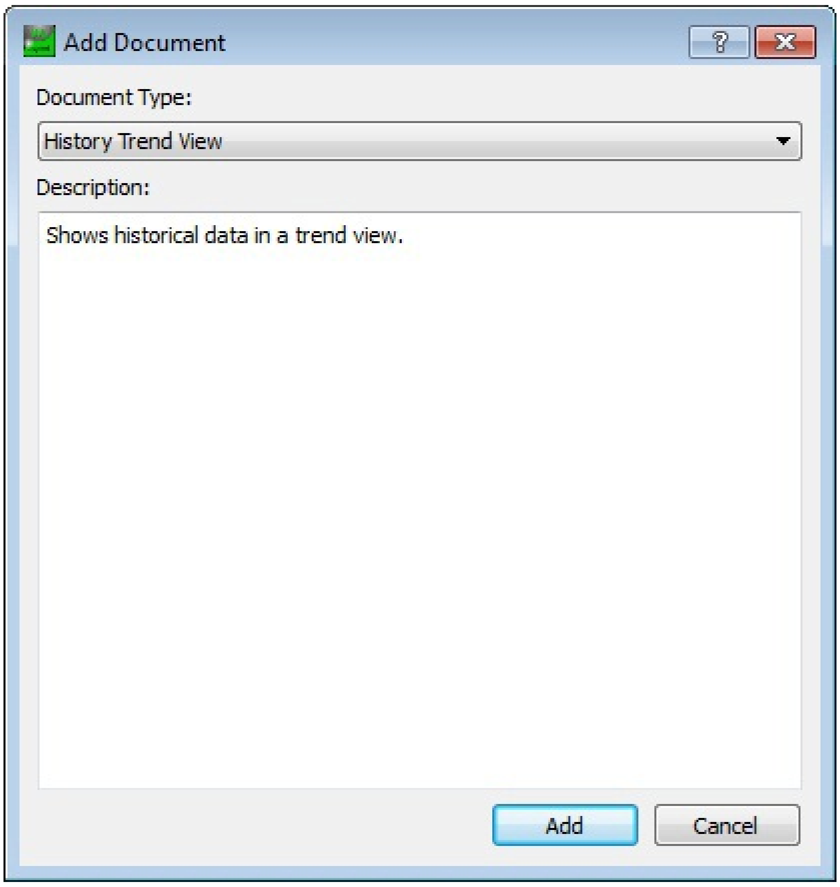
The following step-by-step instructions describe how to configure the UA Expert software in order to access historical data.

1. Start the UA Expert software and connect to the OPC UA Server.



```
Timestamp     Source            Server            Message
12:31:59.527  Server Node       CX_0121A0/Bec...  Disconnect succeeded.
12:31:59.485  Server Node       CX_0121A0/Bec...  Connection status of server 'CX_0121A0/Beckhoff/TcOpcUaServer/1 - None - None' changed to 'Disconnected'.
12:30:31.520  AddressSpaceM...  SvenG-NB/Beck...  Browse succeeded.
12:30:31.514  AddressSpaceM...  SvenG-NB/Beck...  Browse succeeded.
12:30:31.513  Server Node       SvenG-NB/Beck...  Successfully connected UA server.
12:30:31.473  Server Node       SvenG-NB/Beck...  Connection status of server 'SvenG-NB/Beckhoff/TcOpcUaServer/1 - None - None' changed to 'Connected'.
12:30:31.470  Server Node       SvenG-NB/Beck...  The server returned a valid certificate.
12:30:31.470  Server Node       SvenG-NB/Beck...  ApplicationUri: 'SvenG-NB/Beckhoff/TcOpcUaServer/1'
12:30:31.469  Server Node       SvenG-NB/Beck...  Found security policy 'http://opcfoundation.org/UA/SecurityPolicy#None'
12:30:31.469  Server Node       SvenG-NB/Beck...  Found endpoint 'opc.tcp://SvenG-NB:4840'
12:30:27.145  Server Node       SvenG-NB/Beck...  Disconnect succeeded.
12:30:27.104  Server Node       SvenG-NB/Beck...  Connection status of server 'SvenG-NB/Beckhoff/TcOpcUaServer/1 - None - None' changed to 'Disconnected'.
```

2. Add a new History Trend View.

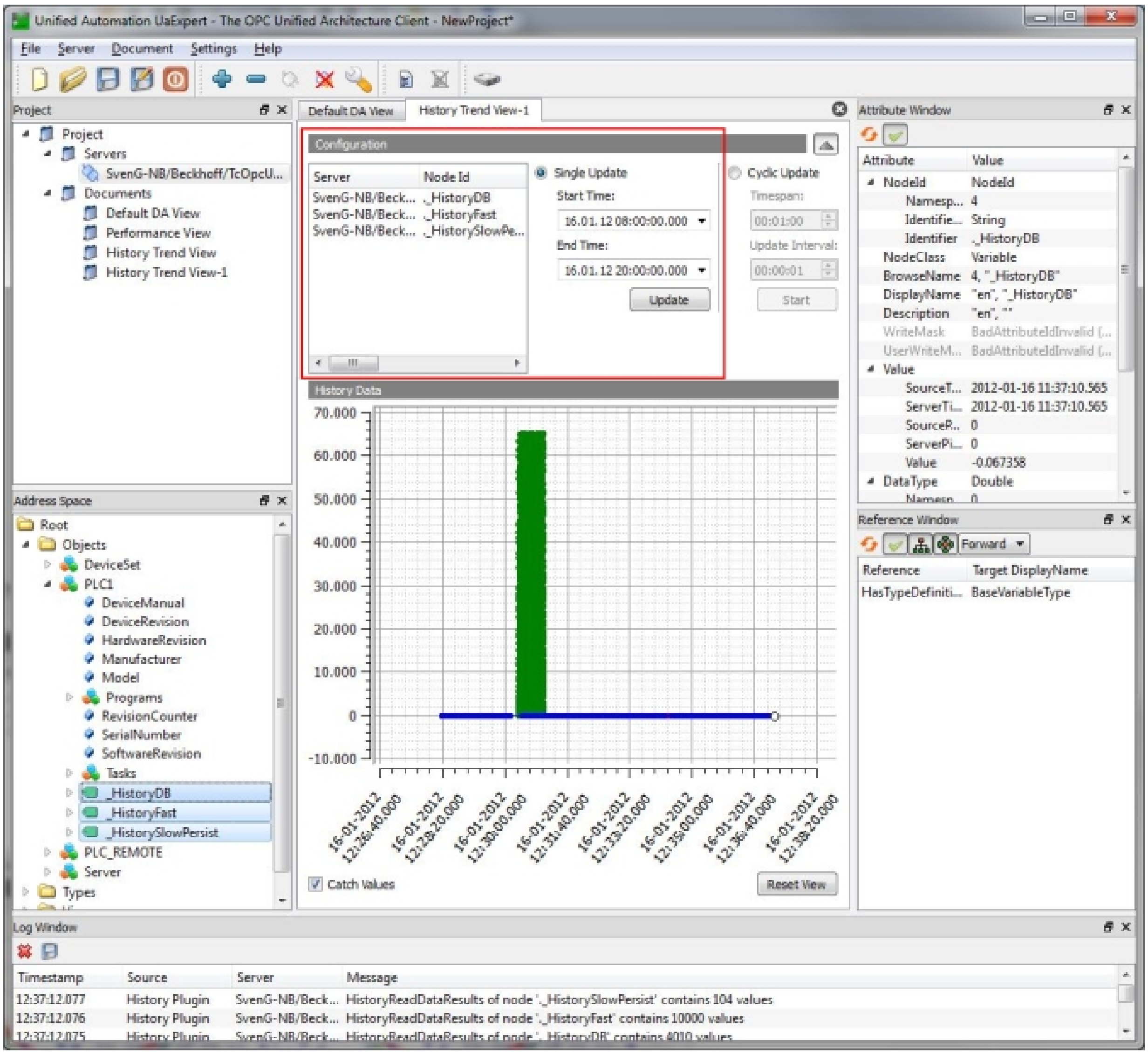


Document Type: History Trend View

Description: Shows historical data in a trend view.

3. Browse the PLC1 namespace and use drag & drop to add the PLC variables _HistoryDB, _HistoryDBcompact, _HistoryFast and _HistorySlowPersist.

⇨ You can now use the **Start Time** and **End Time** controls to specify the desired time period for which the symbol values are to be displayed, or you can start a **Cyclic Update** for these variables if necessary.



Server	Node Id
SvenG-NB/Beck...	._HistoryDB
SvenG-NB/Beck...	._HistoryFast
SvenG-NB/Beck...	._HistorySlowPe...

[✓ Single Update] Cyclic Update

Start Time: 16.01.12 08:00:00.000

End Time: 16.01.12 20:00:00.000

Timespan: 00:01:00

Update Interval: 00:00:01

Attribute	Value
NodeId	NodeId
Namesp...	4
Identifie...	String
Identifier	._HistoryDB
NodeClass	Variable
BrowseName	4, "._HistoryDB"
DisplayName	"en", "._HistoryDB"
Description	"en", ""
WriteMask	BadAttributeIdInvalid (...
UserWriteM...	BadAttributeIdInvalid (...
Value	
SourceT...	2012-01-16 11:37:10.565
ServerTi...	2012-01-16 11:37:10.565
SourceP...	0
ServerPi...	0
Value	-0.067358
DataType	Double
Namesp...	0

Reference	Target DisplayName
HasTypeDefiniti...	BaseVariableType

```
Timestamp      Source          Server             Message
12:37:12.077   History Plugin  SvenG-NB/Beck...   HistoryReadDataResults of node '._HistorySlowPersist' contains 104 values
12:37:12.076   History Plugin  SvenG-NB/Beck...   HistoryReadDataResults of node '._HistoryFast' contains 10000 values
12:37:12.075   History Plugin  SvenG-NB/Beck...   HistoryReadDataResults of node '._HistoryDB' contains 4010 values
```

4.1.4.3 Alarms and Conditions

The steps required to configure OPC UA Alarms and Conditions (A&C) on the OPC UA Server are described in this section. The basic concept is independent of TwinCAT runtime, which means the configuration steps for PLC, C++, TcCOM runtime or only I/O task are the same.

OPC UA Alarms and Conditions (Part 9 of the OPC UA specification) describes a model for monitoring process values and outputting alarms and events when a runtime symbol changes its state.

Requirements

The following requirements apply to the use of OPC UA Alarms and Conditions:

- The runtime symbol to be monitored must be available in the namespace.
- The OPC UA Client must support Alarms and Conditions. In this section UA Expert (from Unified Automation) is used as the reference UA Client.

General Information

Execute the following steps once to release a symbol for Alarms and Conditions:

- Step 1: Activating runtime symbol for data access [▶ 82] (so that the symbol is generally accessible via OPC UA.)
- Step 2: Activating A&C for a symbol [▶ 82]
- Step 3: Transferring your own user data with an event [▶ 83]
- Step 4: Triggering an event using the FireEvent method [▶ 84]
- Step 5: Configuring multilingual alarm texts [▶ 86]
- Step 6: Registering A&C with a reference OPC UA Client [▶ 86]

These steps are explained in more detail below. At the end of this section you will find information about receiving configured alarms via A&C with the UA Expert Reference Client.

Supported alarm types

The implementation of OPC UA Alarms and Conditions currently supports the following alarm types:

- LimitAlarmType: Define different limits for a symbol. If a limit is reached, the UA Server issues an alarm.
- OffNormalAlarmType: Define a value that is "normal". If the current value deviates from the "normal" value, the UA Server issues an alarm.

Step 1: Activating runtime symbol for data access

A variable must be available in the OPC UA Server in order for it to be configured for A&C. To do this in the case of a PLC variable, provide it with an attribute (see Access to PLC runtime [▶ 43]).

Step 2: Activating A&C for a symbol

You can configure a runtime symbol for A&C with the OPC UA Server Configurator. The Configurator has a simple graphical user interface for editing the XML file on which it is based. Depending on the setup version the configurator is available in two variants: standalone [▶ 190] and integrated in Visual Studio [▶ 202].

The following program extract shows an example of this XML file to better understand the general behavior and structure of the A&C implementation.

```
<TcUaAcConfig>
  <ConditionController Name="ConditionController1" >
    <Condition Name="Counter" Severity="200">
    <LimitAlarmType LowLowLimit="-10" LowLimit="0" HighLimit="10" HighHighLimit="20"
MessageNormal="100" MessageLowLow="10" MessageLow="11" MessageHigh="12" MessageHighHigh="13"/>
      <ItemToMonitor SamplingRate="100" NS="urn:[NodeName]:BeckhoffAutomation:Ua:PLC1"
NodeId="s=MAIN.nCounter1" />
    </Condition>
    <Condition Name="Switch" Severity="500">
      <OffNormalAlarmType Normal="0" MessageNormal="100" MessageOffNormal="20" />
      <ItemToMonitor SamplingRate="100" NS="urn:[NodeName]:BeckhoffAutomation:Ua:PLC1"
NodeId="s=MAIN.bSwitch" />
    </Condition>
    <Condition Name="Struct" Severity="300">
      <LimitAlarmType LowLowLimit="-10" LowLimit="0" HighLimit="10" HighHighLimit="20"
MessageNormal="100" MessageLowLow="10" MessageLow="11" MessageHigh="12" MessageHighHigh="13"/>
      <ItemToMonitor SamplingRate="100" NS="urn:[NodeName]:BeckhoffAutomation:Ua:PLC1"
NodeId="s=MAIN.stStruct" />
    </Condition>
  </ConditionController>
  <ConditionController Name="ConditionController2" >
    <Condition Name="Counter2" Severity="200">
      <LimitAlarmType LowLowLimit="-10" LowLimit="0" HighLimit="10" HighHighLimit="20"
MessageNormal="100" MessageLowLow="10" MessageLow="11" MessageHigh="12" MessageHighHigh="13"/>
      <ItemToMonitor SamplingRate="100" NS="urn:[NodeName]:BeckhoffAutomation:Ua:PLC1"
NodeId="s=MAIN.nCounter2" />
    </Condition>
  </ConditionController>
</TcUaAcConfig>
```

A "condition" defines the runtime symbol to be monitored and the alarm limits and texts. Each condition is organized in a so-called "ConditionController", the object that the OPC UA Clients subsequently subscribe to.

When creating a condition, you must specify the NamespaceName (NS) and NodeID for referring to the UA node to be monitored. The Configurator provides a simple browsing mechanism for selecting a node. In XML, the placeholder [NodeName] in NamespaceName can be used to switch the XML file between different hardware systems. NamespaceName always contains the host name of the IPC or Embedded PC on which the OPC UA Server is running. If [NodeName] is selected, this tag will be replaced by the host name of the current IPC or Embedded PC on which the UA Server is running.

SamplingRate determines how often the UA Server should request a value from the node to determine whether one of the alarm limits has been reached.

The alarm texts are identified by an ID. The ID uniquely identifies an alarm text from the resource file (see Step 5: Configuring multilingual alarm texts [▶ 86]).

Step 3: Transferring own user data with an alarm

Alarms can contain fields with their own user data, which complement the data output with the alarm. These user data fields can be created and filled out in the runtime application. To do this, create a STRUCT and name its first element "value". In case of an alarm, all the following elements are then sent in an additional user data field.

Sample PLC application:

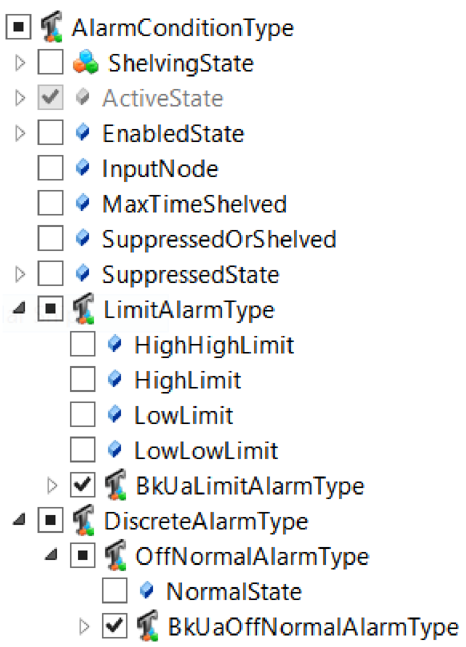
```
TYPE ST_CustomStruct :
STRUCT
  value : INT;
  data  : ST_SomeStruct;
END_STRUCT
END_TYPE

TYPE ST_SomeStruct :
STRUCT
  Data1 : INT;
  Data2 : REAL;
  Data3 : LREAL;
END_STRUCT
END_TYPE
```

The instance of ST_CustomStruct is then enabled for data access via the regular mechanism, e.g. in TwinCAT 3: To do this the STRUCT must be activated as a StructuredType [▶ 88].

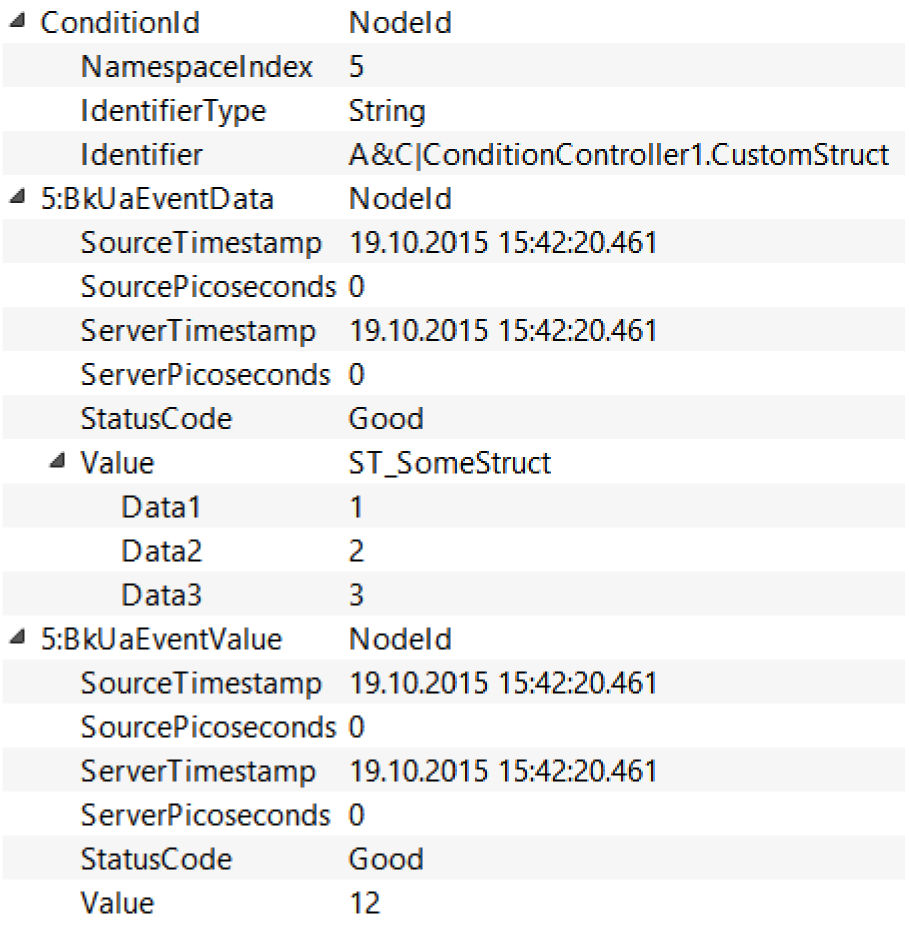
```
PROGRAM MAIN
VAR
  {attribute 'OPC.UA.DA' := '1'}
  {attribute 'OPC.UA.DA.StructuredType' := '1'}
  stCustomStruct : ST_CustomStruct;
END_VAR
```

When logging on to a ConditionController, the OPC UA Clients must subscribe to special AlarmConditionTypes, i.e. "BkUaLimitAlarmType" and "BkUaOffNormalAlarmType", so that they can receive the special user data fields when an alarm is received.



```
■ AlarmConditionType
  ▷ ☐ ShelvingState
  ▷ ✓ ActiveState
  ▷ ☐ EnabledState
    ☐ InputNode
    ☐ MaxTimeShelved
    ☐ SuppressedOrShelved
  ▷ ☐ SuppressedState
  ◢ ■ LimitAlarmType
      ☐ HighHighLimit
      ☐ HighLimit
      ☐ LowLimit
      ☐ LowLowLimit
    ▷ ✓ BkUaLimitAlarmType
  ◢ ■ DiscreteAlarmType
    ◢ ■ OffNormalAlarmType
        ☐ NormalState
      ▷ ✓ BkUaOffNormalAlarmType
```

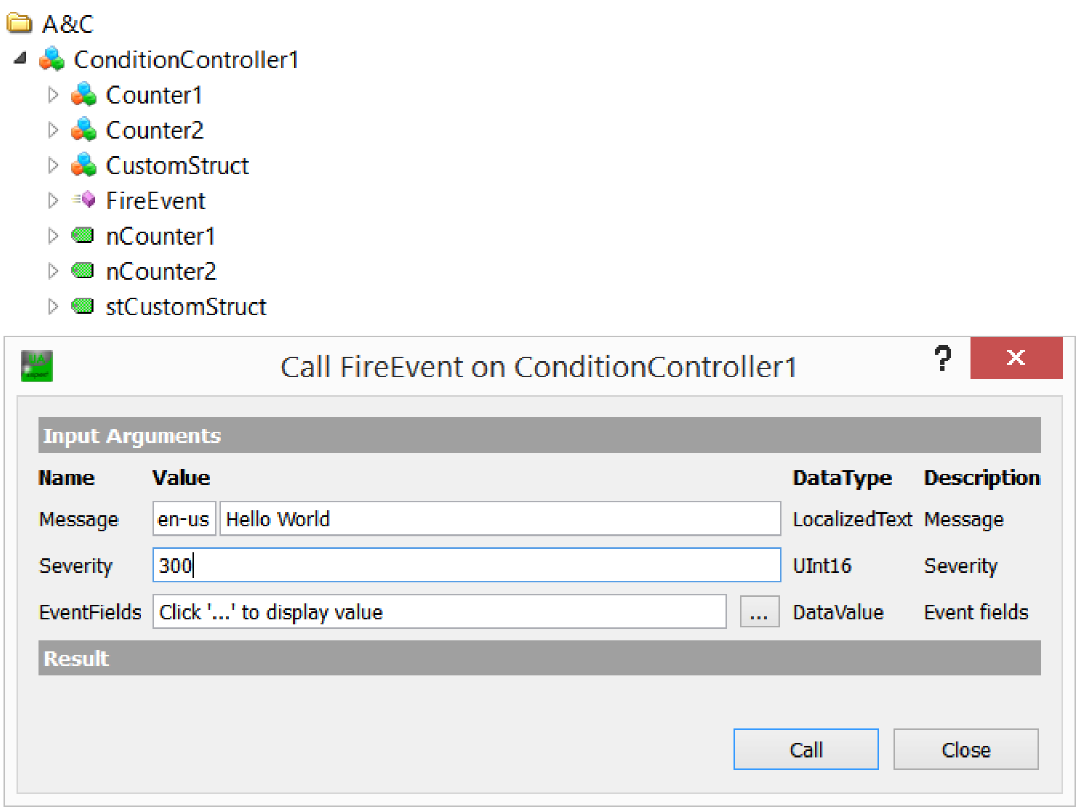
The OPC UA Client then receives the user data in the fields BkUaEventData and BkUaEventValue of the incoming alarm. In the above sample, these are the value of the PLC variables and the user data represented by the PLC structure ST_SomeStruct.



◢ ConditionId	NodeId
NamespaceIndex	5
IdentifierType	String
Identifier	A&C|ConditionController1.CustomStruct
◢ 5:BkUaEventData	NodeId
SourceTimestamp	19.10.2015 15:42:20.461
SourcePicoseconds	0
ServerTimestamp	19.10.2015 15:42:20.461
ServerPicoseconds	0
StatusCode	Good
◢ Value	ST_SomeStruct
Data1	1
Data2	2
Data3	3
◢ 5:BkUaEventValue	NodeId
SourceTimestamp	19.10.2015 15:42:20.461
SourcePicoseconds	0
ServerTimestamp	19.10.2015 15:42:20.461
ServerPicoseconds	0
StatusCode	Good
Value	12

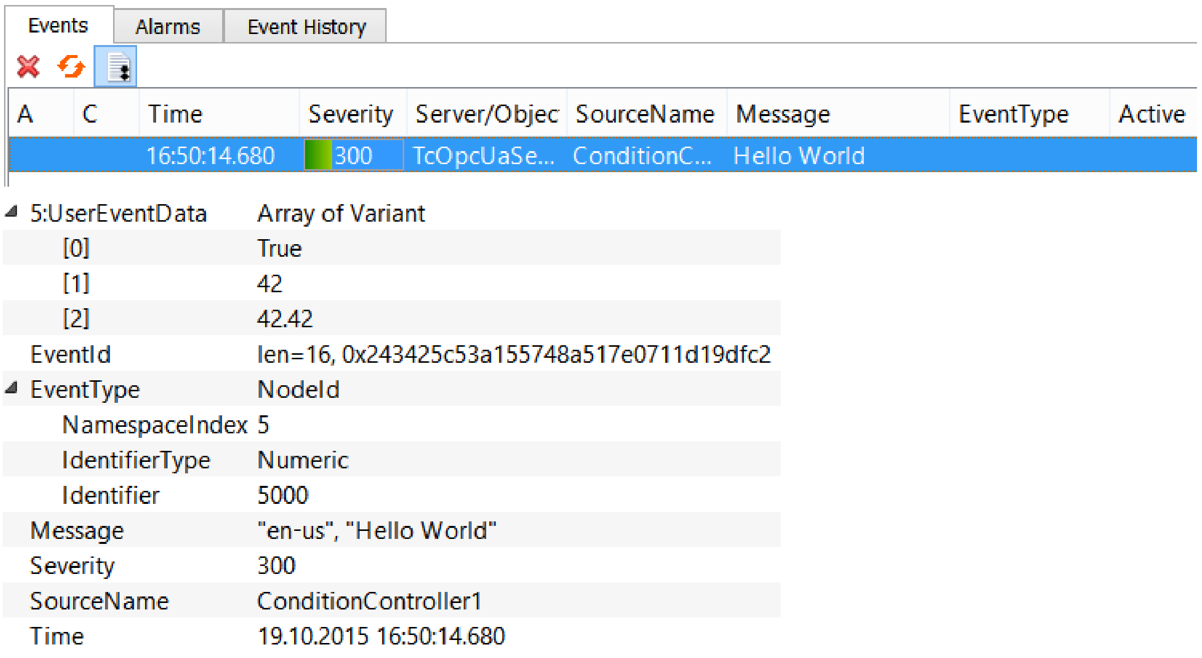
Step 4: Triggering an event using the FireEvent method

Each ConditionController includes a FireEvent method with which the OPC UA Clients can trigger a general event with user-defined EventFields.



Name	Value	DataType	Description
Message	en-us Hello World	LocalizedText	Message
Severity	300	UInt16	Severity
EventFields	Click '...' to display value	DataValue	Event fields

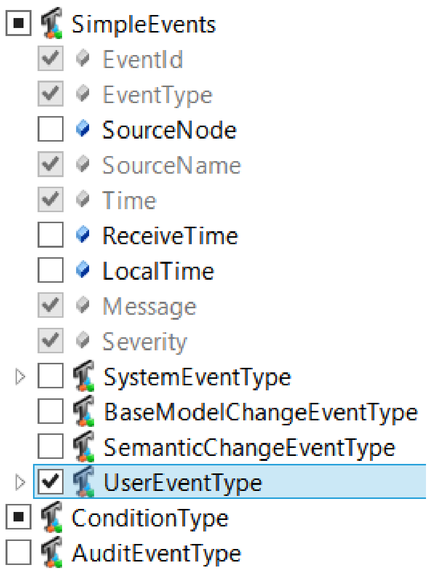
If the method is executed, an event is output on the OPC UA Server. Other OPC UA Clients can receive these events when they subscribe to the corresponding ConditionController.



A	C	Time	Severity	Server/Objec	SourceName	Message	EventType	Active
		16:50:14.680	300	TcOpcUaSe...	ConditionC...	Hello World		

5:UserEventData	Array of Variant
[0]	True
[1]	42
[2]	42.42
EventId	len=16, 0x243425c53a155748a517e0711d19dfc2
EventType	NodeId
NamespaceIndex	5
IdentifierType	Numeric
Identifier	5000
Message	"en-us", "Hello World"
Severity	300
SourceName	ConditionController1
Time	19.10.2015 16:50:14.680

The user-defined EventFields are appended to the event as "UserEventData". This data can be received by OPC UA Clients that are logged on to the SimpleEventType "UserEventType".



Step 5: Configuring multilingual alarm texts

The A&C implementation in the OPC UA Server supports the use of multilingual alarm texts. The alarm text that is used depends on the language with which the UA Client connects to the Server.

Alarm texts are configured in XML files. There is a separate file for each language. These files are located in the res (Resource) folder on the OPC UA Server. With the configurator you can simply add or remove alarm texts without having to directly edit the XML files. Each alarm text has its own ID, which occurs only once in the file.

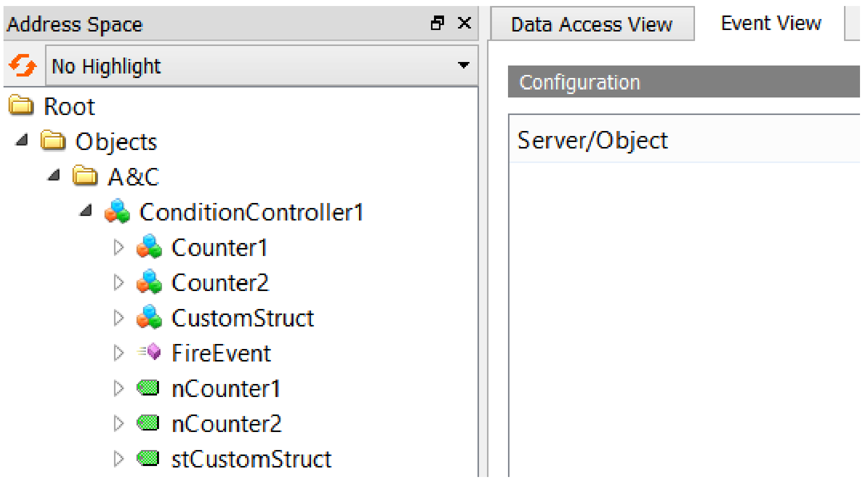
Sample:

```
<TcOpcUaSvrRes Lang="en">
  <Text ID="0">Text not available</Text>
  <Text ID="1">Some alarm text</Text>
  <Text ID="2">Value is High range</Text>
  <Text ID="3">Value is HighHigh range</Text>
  <Text ID="4">Value is OffNormal</Text>
  ...
</TcOpcUaSvrRes>
```

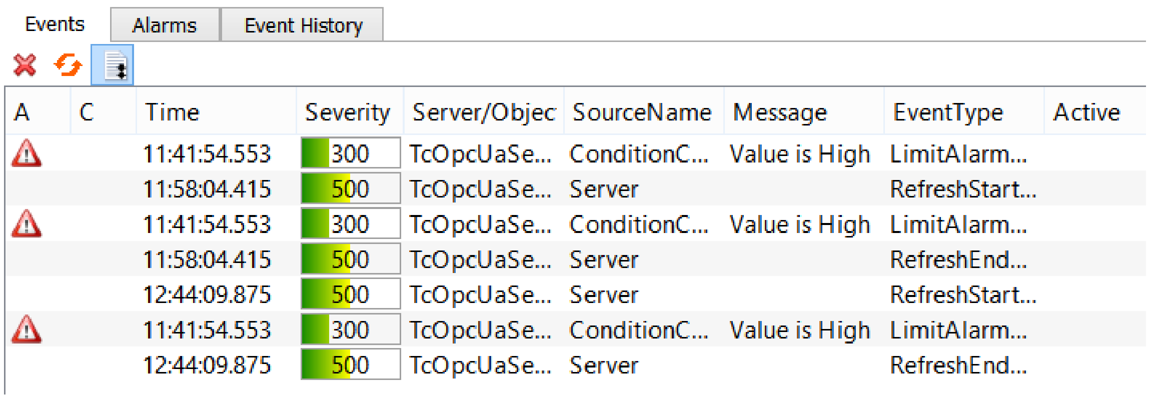
Step 6: Registering A&C with a reference OPC UA Client

An OPC UA Client must log on to a ConditionController so that it can receive events for conditions that are configured for this particular ConditionController. The UA Expert provides functions for subscribing to and receiving UA events.

After you have started the UA Expert and established a connection with the OPC UA Server, add a new document view, Event View, to your working area.



You can then subscribe to a ConditionController by dragging the corresponding object in Event View. The events for this ConditionController are displayed in the Event Window.



A	C	Time	Severity	Server/Objec	SourceName	Message	EventType	Active
⚠		11:41:54.553	300	TcOpcUaSe...	ConditionC...	Value is High	LimitAlarm...	
		11:58:04.415	500	TcOpcUaSe...	Server		RefreshStart...	
⚠		11:41:54.553	300	TcOpcUaSe...	ConditionC...	Value is High	LimitAlarm...	
		11:58:04.415	500	TcOpcUaSe...	Server		RefreshEnd...	
		12:44:09.875	500	TcOpcUaSe...	Server		RefreshStart...	
⚠		11:41:54.553	300	TcOpcUaSe...	ConditionC...	Value is High	LimitAlarm...	
		12:44:09.875	500	TcOpcUaSe...	Server		RefreshEnd...	

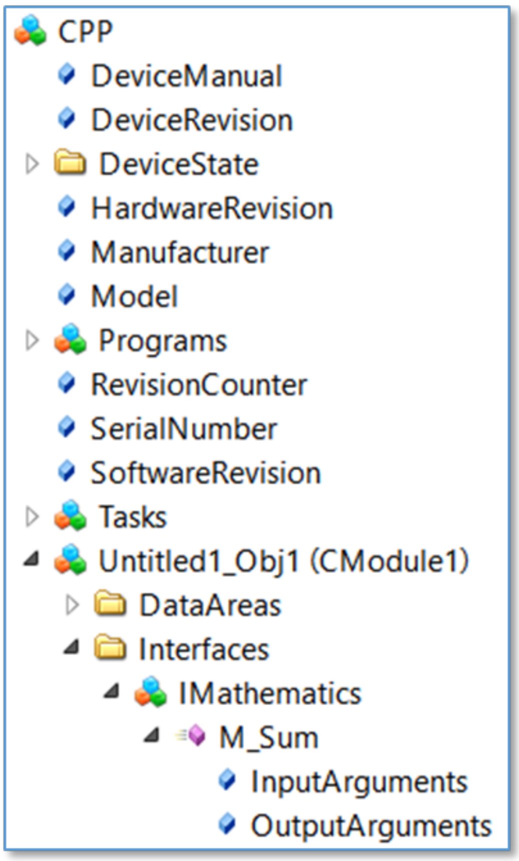
It may be necessary to subscribe to special alarm and/or event types in order to receive all fields of an incoming event or alarm.

4.1.4.4 Method Call

Method calls are a fundamental part of the OPC UA specification. With the introduction of these functionalities into the PLC world, TwinCAT 3 offers the possibility of efficient execution of RPC calls in the C ++ real-time context and thus reduces the classic handshake patterns for communication between devices. The OPC UA Server imports C++ methods such as OPC UA methods via its TMI import.

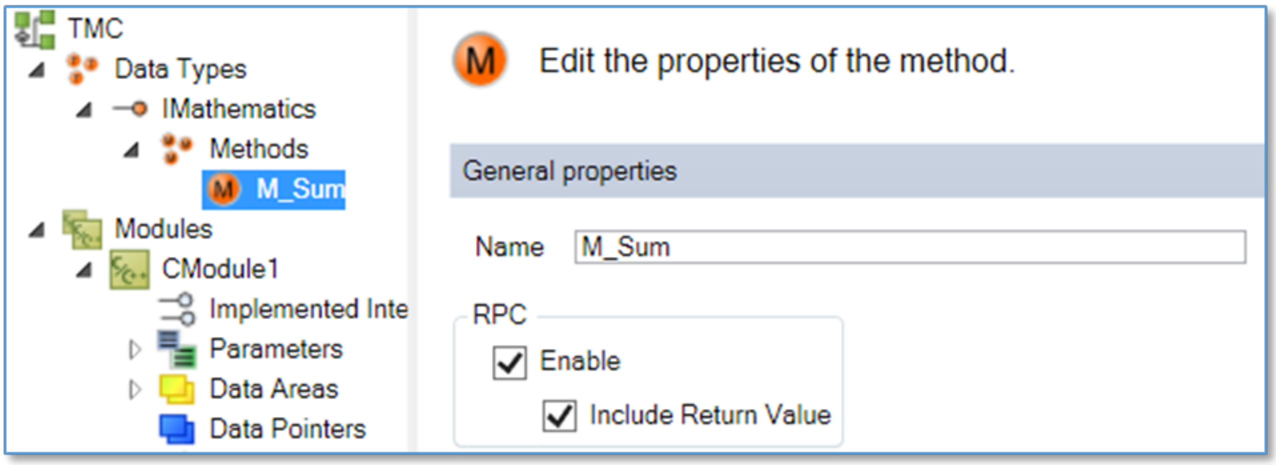
Real-time method

Although the C++ method appears to be a normal method in the UA namespace, it is still a real-time method that runs within the real-time context and therefore falls under the context of a real-time task. The TwinCAT C++ developer must therefore take precautions so that the execution time of the method matches the task cycle time.



Methods in TwinCAT 3 C++

TwinCAT modules could implement interfaces with predefined methods (see TcCOM modules). The method itself must be enabled for RPC calls during its definition (see TwinCAT Module Class Wizard documentation) so that the OPC UA Server knows that it is ready for execution.



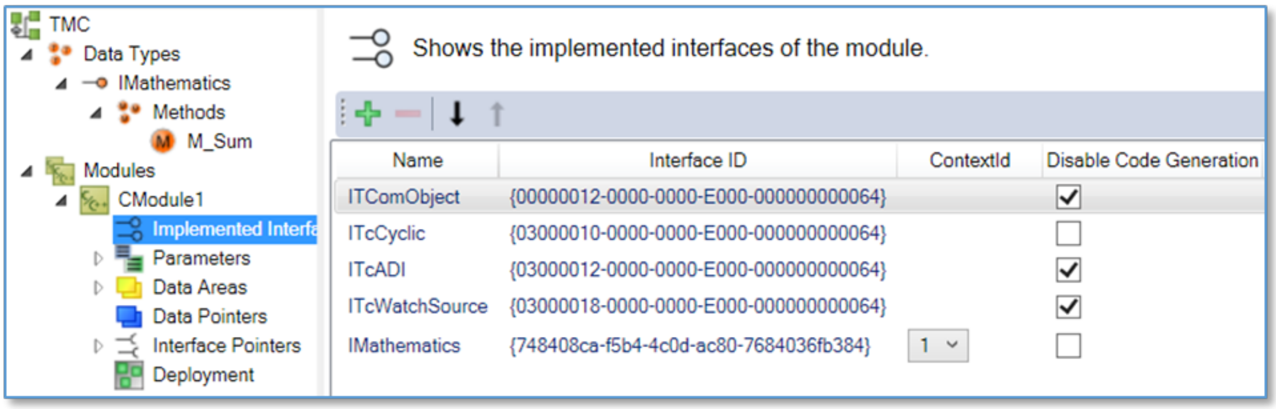
Name: M_Sum

RPC

☑ Enable

☑ Include Return Value

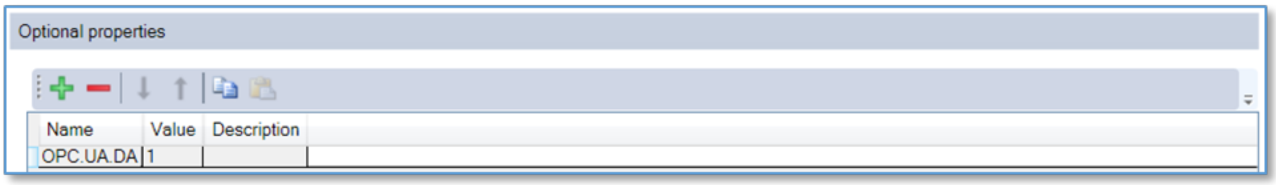
So that the return value of the method is available, the corresponding option **Include Return Value** must be activated. Note that the interface under which the method was created must be implemented.



Name	Interface ID	ContextId	Disable Code Generation
ITComObject	{00000012-0000-0000-E000-000000000064}		☑
ITcCyclic	{03000010-0000-0000-E000-000000000064}		☐
ITcADI	{03000012-0000-0000-E000-000000000064}		☑
ITcWatchSource	{03000018-0000-0000-E000-000000000064}		☑
IMathematics	{748408ca-f5b4-4c0d-ac80-7684036fb384}	1	☐

Filtered or unfiltered mode

Depending on the import mode used, you have to declare the method for the access via OPC UA. This can be done by using the TMC Code editor and the usual OPC UA attributes as optional properties.



Name	Value	Description
OPC.UA.DA	1	

4.1.4.5 StructuredTypes

StructuredTypes allow you to read or write structures without interpreting each byte, because the UA Server returns the information type of each element of the structure. Based on complex functions in modern OPC UA SDKs, OPC UA Clients can search and interpret this structural information.

From version 2.2.x of the OPC UA Server, structures of the TwinCAT 3 runtime (TMC and TMI import only) are generated as a StructuredType in the UA namespace.

Sample:

STRUCT ST_Communication:

```
TYPE ST_Communication :
STRUCT
  a : INT;
  b : INT;
  c : INT;
END_STRUCT
END_TYPE
```

Program MAIN:

```
PROGRAM MAIN
VAR
  {attribute 'OPC.UA.DA' := '1'}
  {attribute 'OPC.UA.DA.StructuredType' := '1'}
  stCommunication : ST_Communication;
END_VAR
```

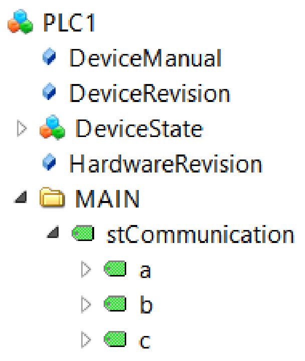
Filtered mode

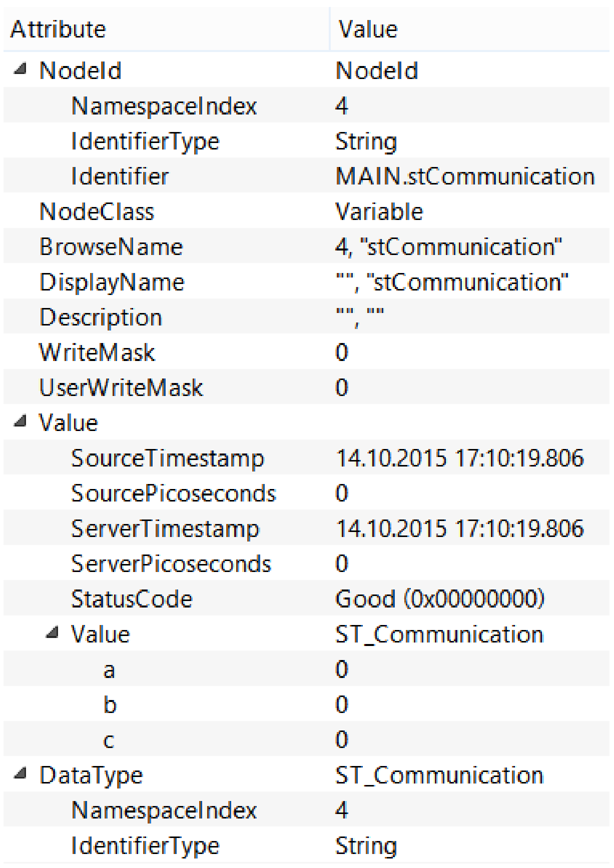
If filters are used to make symbols available via OPC UA, a STRUCT or function block must be fully available in the UA namespace in order to be displayed as a StructuredType.

Pointers and references

If pointers and references are used in the structure, then they cannot be converted into a StructuredType. The OPC UA Server then illustrates these structures as regular FolderTypes with the corresponding member variables.

The instance stCommunication is then displayed in the UA namespace as a StructuredType:





Attribute	Value
◢ NodeId	NodeId
NamespaceIndex	4
IdentifierType	String
Identifier	MAIN.stCommunication
NodeClass	Variable
BrowseName	4, "stCommunication"
DisplayName	"", "stCommunication"
Description	"", ""
WriteMask	0
UserWriteMask	0
◢ Value	
SourceTimestamp	14.10.2015 17:10:19.806
SourcePicoseconds	0
ServerTimestamp	14.10.2015 17:10:19.806
ServerPicoseconds	0
StatusCode	Good (0x00000000)
◢ Value	ST_Communication
a	0
b	0
c	0
◢ DataType	ST_Communication
NamespaceIndex	4
IdentifierType	String

Alternatively, the STRUCT definition can also be assigned the PLC attribute to make all instances of STRUCT available as StructuredType.

```
{attribute 'OPC.UA.DA.StructuredType' := '1'}
TYPE ST_Communication :
STRUCT
  a : INT;
  b : INT;
  c : INT;
END_STRUCT
END_TYPE
```

In order to deactivate StructuredType of a certain instance, use the following attribute:

```
PROGRAM MAIN
VAR
  {attribute 'OPC.UA.DA' := '1'}
  {attribute 'OPC.UA.DA.StructuredType' := '0'}
  stCommunication : ST_Communication;
END_VAR
```

Function block StructuredType

In addition, each function block of the TwinCAT 3 PLC also contains a child node, FunctionBlock, which contains the entire function block as a StructuredType.

Sample:

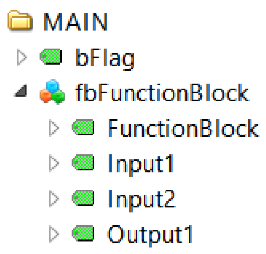
Function block:

```
FUNCTION_BLOCK FB_FunctionBlock
VAR_INPUT
  Input1 : INT;
  Input2 : LREAL;
END_VAR
VAR_OUTPUT
  Output1 : LREAL;
END_VAR
```

Instance of the function block:

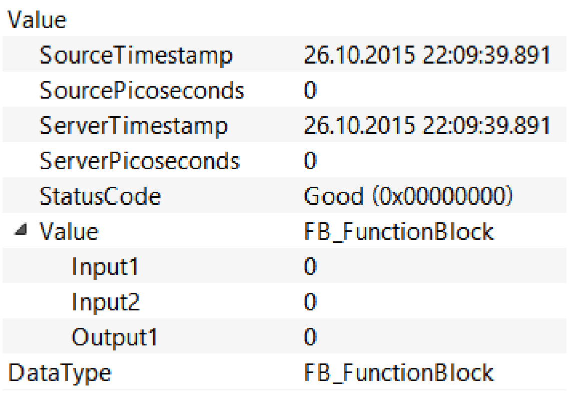
```
PROGRAM MAIN
VAR
  {attribute 'OPC.UA.DA' := '1'}
  {attribute 'OPC.UA.DA.StructuredType' := '1'}
  fbFunctionBlock : FB_FunctionBlock;
END_VAR
```

Instance of the function block in the OPC UA namespace:



```
MAIN
  bFlag
  fbFunctionBlock
    FunctionBlock
    Input1
    Input2
    Output1
```

FunctionBlock node with StructuredType:



Value	
SourceTimestamp	26.10.2015 22:09:39.891
SourcePicoseconds	0
ServerTimestamp	26.10.2015 22:09:39.891
ServerPicoseconds	0
StatusCode	Good (0x00000000)
Value	FB_FunctionBlock
Input1	0
Input2	0
Output1	0
DataType	FB_FunctionBlock

Alternatively, the function block can also receive a PLC attribute to make all instances of the function block available as StructuredType.

```
{attribute 'OPC.UA.DA.StructuredType' := '1'}
FUNCTION_BLOCK FB_FunctionBlock
VAR_INPUT
  Input1 : INT;
  Input2 : LREAL;
END_VAR
VAR_OUTPUT
  Output1 : LREAL;
END_VAR
```

In order to deactivate StructuredType of a certain instance, use the following attribute:

```
PROGRAM MAIN
VAR
  {attribute 'OPC.UA.DA' := '1'}
  {attribute 'OPC.UA.DA.StructuredType' := '0'}
  fbFunctionBlock : FB_FunctionBlock;
END_VAR
```

Maximum size of the structure

The maximum size of a structure is 16 kByte by default. Each STRUCT constantly exchanges data with the basic ADS device, i.e. a large ADS message is sent with each read/write command of a StructuredType. To prevent the ADS router from being flooded with large messages, the maximum size is limited. You can change this specification in the file *ServerConfig.xml*. To do this, the key <MaxStructureSize> must be added to OpcServerConfig\UaServerConfig\, and a new value for the maximum size of a structure must be set in bytes. If a structure exceeds <MaxStructureSize>, it is imported as FolderType, where each structure item is available as a single node.

4.1.4.6 List of attributes and comments

The runtime variables for various OPC UA functions (e.g. data access or historical access) are configured directly in the PLC program or in the TMC code editor (if using TwinCAT 3 C++). The advantage is that PLC developers can decide directly in the program with which they are familiar whether and how a variable is to be enabled for OPC UA. You activate a variable by inserting a comment and the corresponding OPC UA tag in front of the variable, e.g.:

TwinCAT 3 PLC (TMC):

```
{attribute 'OPC.UA.DA' := '1'}
bVariable : BOOL;
```

TwinCAT 2 PLC (TPY):

```
bVariable : BOOL; (*~ (OPC:1:available)*)
```

You can find a detailed description of the use of attributes and comments in the sections concerning the corresponding runtime components:

- PLC [▸ 43]
- C++ [▸ 75]
- I/O [▸ 94]
- Matlab/Simulink [▸ 106]

The following table shows an overview of all definable tags and their meaning. The subsections of the corresponding function contain a detailed description of the functional principle.

TwinCAT 3 (TMC):

OPC UA function	PLC tag	C++ TMC code editor (Optional features)	Meaning
Data Access (DA)	{attribute 'OPC.UA.DA' := '0'}	Name: OPC.UA.DA Value: 0	Locks a variable for OPC UA, whereupon it is no longer visible in the UA namespace.
Data Access (DA)	{attribute 'OPC.UA.DA' := '1'}	Name: OPC.UA.DA Value: 1	Enables a variable for OPC UA, whereupon it becomes visible in the UA namespace. This tag must always be set if you want to use a variable for UA.
Data Access (DA)	{attribute 'OPC.UA.DA.Access' := 'x'}	Name: OPC.UA.DA.Access Value: see right column	Sets read/write access for a variable, depending on parameter "x". 0 = none 1 = read-only 2 = write access only 3 = read and write access (default if no tag is used)
Data Access (DA)	{attribute 'OPC.UA.DA.Description' := 'x'}	Name: OPC.UA.DA.Description Value: see right column	Sets a text for the OPC UA attribute "Description".
Data Access (DA)	{attribute 'OPC.UA.DA.StructuredType' := '0'}	Name: OPC.UA.DA.StructuredType Value: 0	Disables StructuredType for a STRUCT or a function block

OPC UA function	PLC tag	C++ TMC code editor (Optional features)	Meaning
Data Access (DA)	{attribute 'OPC.UA.DA.StructuredType' := '1'}	Name: OPC.UA.DA.StructuredType Value: 1	Enables StructuredType for a STRUCT or a function block
Data Access	{attribute 'OPC.UA.DA.Status' := 'quality'}	Name: OPC.UA.DA.Status Value: quality	Manually define the StatusCode of a symbol in the UA namespace. Only for data structures. The value "quality" specifies which DINT subelement of the data structure determines the StatusCode . See corresponding article in the documentation for more information.
Historical Access (HA) [▶ 49]	{attribute 'OPC.UA.HA' := '1'}	Name: OPC.UA.HA Value: 1	Enables a variable for Historical Access. Must be used with {attribute 'OPC.UA.DA' := '1'}.
Historical Access (HA) [▶ 49]	{attribute 'OPC.UA.HA.Storage' := 'x'}	Name: OPC.UA.HA.Storage Value: see right column	Defines the storage location for Historical Access, depending on parameter "x" 1 = RAM 2 = File 3 = SQL Compact Database 4 = SQL Server Database
Historical Access (HA) [▶ 49]	{attribute 'OPC.UA.HA.Sampling' := 'x'}	Name: OPC.UA.HA.Sampling Value: see right column	Determines the sampling rate at which the variable values are to be stored, in [ms] depending on the parameter "x"
Historical Access (HA) [▶ 49]	{attribute 'OPC.UA.HA.Buffer' := 'x'}	Name: OPC.UA.HA.Buffer Value: see right column	Defines the maximum number of values that remain in the data memory, depending on the parameter "x".

TwinCAT 2 (TPY):

OPC UA function	PLC tag	Meaning
Data Access (DA)	(*~ (OPC:0:not available) *)	Locks a variable for OPC UA, whereupon it is no longer visible in the UA namespace.
Data Access (DA)	(*~ (OPC:1:available) *)	Enables a variable for OPC UA, whereupon it becomes visible in the UA namespace. This tag must always be set if you want to use a variable for UA.
Data Access (DA)	(*~ (OPC_PROP[0005]:1: read-only) *)	Sets the write protection for a variable. Must be used together with (*~ (OPC: 1: available) *).
Data Access (DA)	(*~ (OPC_UA_PROP[5100] : x: Alias name) *)	Specifies x as node name in the UA namespace, so-called alias mapping.

OPC UA function	PLC tag	Meaning
Historical Access (HA) [▶ 49]	(*~ (OPC_UA_PROP[5000]:x:Storage media) *)	Enables a variable for "Historical Access". Must be used together with (*~ (OPC: 1: available) *). x defines the storage medium for saving the data values: 1 = RAM 2 = File 3 = SQL Compact Database 4 = SQL Server Database
Historical Access (HA) [▶ 49]	(*~ (OPC_UA_PROP[5000][1]:x:SamplingRate) *)	Determines the sampling rate at which the variable values are to be stored, in [ms] depending on the parameter "x"
Historical Access (HA) [▶ 49]	(*~ (OPC_UA_PROP[5000][2]:x:Buffer) *)	Defines the maximum number of values that remain in the data memory, depending on the parameter "x".

4.1.5 I/O

4.1.5.1 Access to IO task

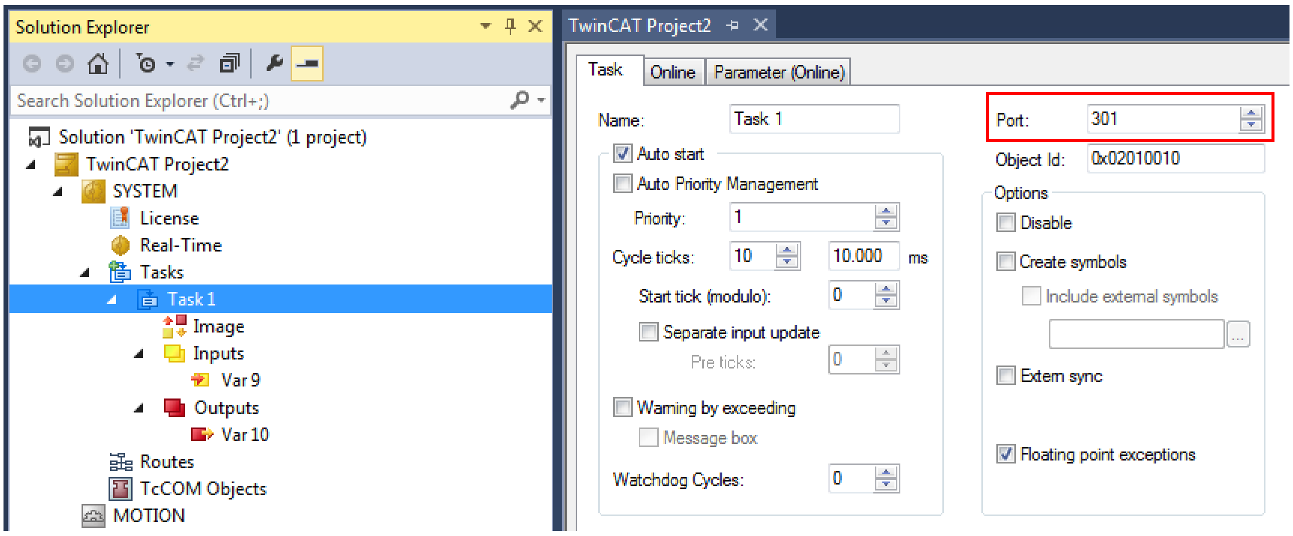
This section describes how you can make IO task variables accessible via the data access functions of the TwinCAT OPC UA Server.

This section contains the following topics:

- General Information [▶ 94]
- Step 1: Configure a TwinCAT I/O task to enable symbol handling. [▶ 95]
- Step 2: Add the I/O task to the OPC UA namespace [▶ 96]

General Information

You can publish the variables of a task with process image via OPC UA.



Name: Task 1
Port: 301
Object Id: 0x02010010
[✓] Auto start
[ ] Auto Priority Management
Priority: 1
Cycle ticks: 10 10.000 ms
Start tick (modulo): 0
[ ] Separate input update
Pre ticks: 0
[ ] Warning by exceeding
[ ] Message box
Watchdog Cycles: 0
Options
[ ] Disable
[ ] Create symbols
[ ] Include external symbols
[ ] Extern sync
[✓] Floating point exceptions

The way in which the variables are displayed and how they are communicated with is determined by various parameters. You can define these parameters with the help of the OPC UA Configurator or directly in the configuration file *ServerConfig.xml*.

The following table provides an overview of all parameters that are important when accessing the I/O task.

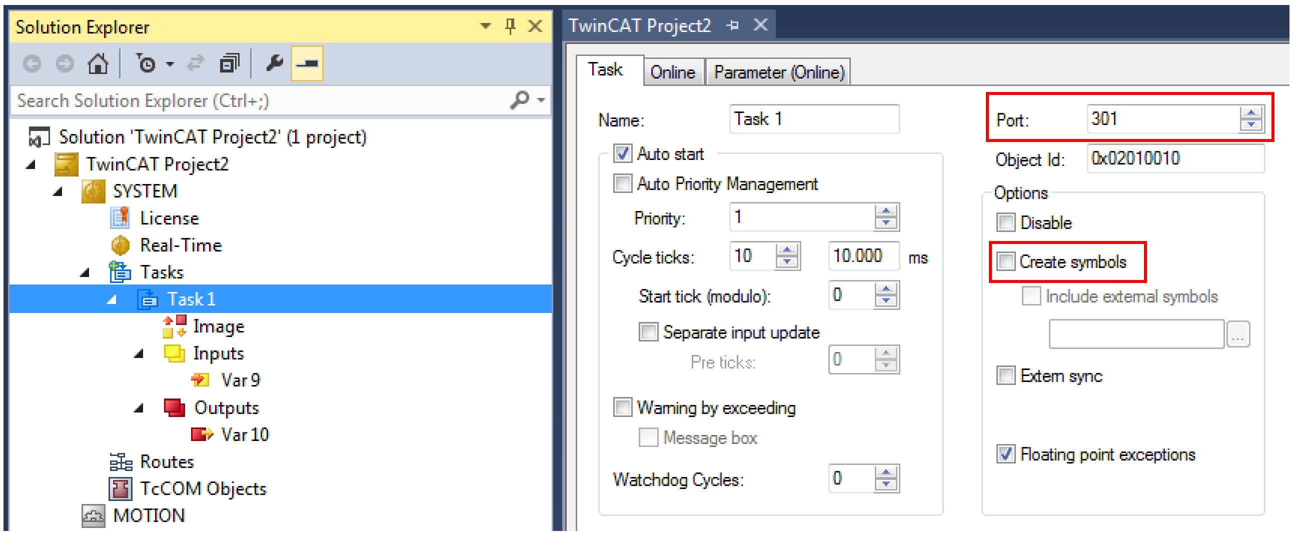
Parameter	Description	Possible values
ADS Port	Defines the ADS port under which the I/O task is accessible. The ADS port can be read out in the properties of the I/O task.	301 302 …
AutoCfg	Initially defines the type of target runtime used for the communication, e.g. PLC, C++, I/O. A finer distinction can subsequently be made within these categories. Each AutoCfg option is available as an unfiltered or filtered option. Filtered means that users can determine which I/O variables are made publicly available to the OPC UA via comments. When using an unfiltered option, each task variable is published to OPC UA.	107 I/O TwinCAT 2/3 108 I/O TwinCAT 2/3 filtered
AutoCfgSymFile	Specifies the path of the CurrentConfig.xml file, which is normally located in the folder *C:\TwinCAT\3.1\Boot\*.	Path of CurrentConfig.xml
Disabled	Disables the I/O task in the UA namespace, so that the corresponding node is not displayed. It is advisable to enable this parameter if certain runtimes are not yet available at the time of project planning, for example, because the corresponding devices are not yet connected to the network.	0 (disabled - default) 1 (enabled)

The following steps describe how to import variables from an I/O task into the UA namespace. This requires the OPC UA Server and the runtime to be on the same computer.

Step 1: Configure a TwinCAT I/O task to enable symbol handling.

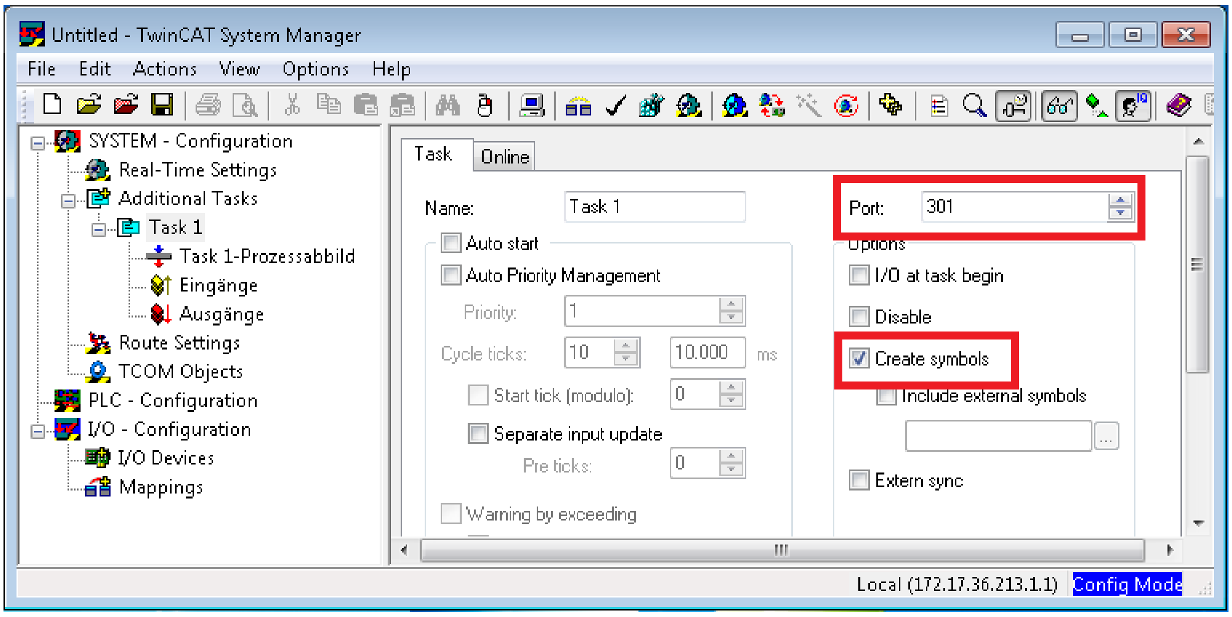
Open the I/O task settings and enable the **Create symbols** option. At this point, note the ADS port of the I/O task (in the example, this is 301).

TwinCAT 3:



Name: Task 1
Auto start [✓]
Auto Priority Management [ ]
Priority: 1
Cycle ticks: 10 10.000 ms
Start tick (modulo): 0
Separate input update [ ]
Pre ticks: 0
Warning by exceeding [ ]
Message box [ ]
Watchdog Cycles: 0
Port: 301
Object Id: 0x02010010
Options
Disable [ ]
Create symbols [ ]
Include external symbols [ ]
Extern sync [ ]
Floating point exceptions [✓]

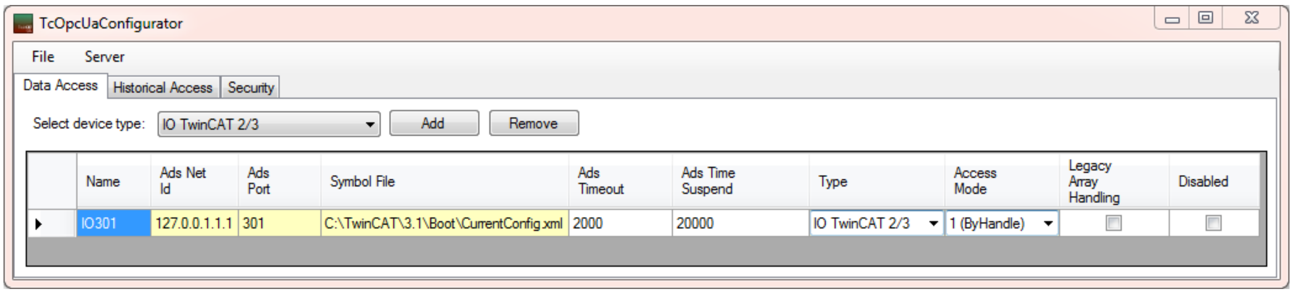
TwinCAT 2:



Name: Task 1
Port: 301
☐ Auto start
☐ Auto Priority Management
Priority: 1
Cycle ticks: 10 10.000 ms
☐ Start tick (modulo): 0
☐ Separate input update
Pre ticks: 0
☐ Warning by exceeding
☐ I/O at task begin
☐ Disable
☑ Create symbols
☐ Include external symbols
☐ Extern sync
Local (172.17.36.213.1.1) Config Mode

Step 2: Add the I/O task to the OPC UA namespace

Configure the OPC UA Server so that it offers access to the I/O task. Using the OPC UA Configurator, add a new "IO TwinCAT 2/3" device and configure its settings for AdsNetId, AdsPort and SymbolFile (path to the *CurrentConfig.xml* file).



Select device type: IO TwinCAT 2/3

Name	Ads Net Id	Ads Port	Symbol File	Ads Timeout	Ads Time Suspend	Type	Access Mode	Legacy Array Handling	Disabled
IO301	127.0.0.1.1.1	301	C:\TwinCAT\3.1\Boot\CurrentConfig.xml	2000	20000	IO TwinCAT 2/3	1 (ByHandle)	☐	☐

After setting the corresponding properties, restart the OPC UA Server to enable these settings.

4.1.5.2 Historical Access

4.1.5.2.1 Setup Version 4.x.x

From setup version 4.x.x historical access can no longer be configured via attributes or comments. The advantage of this is that the Historical Access functionality is now also available for runtimes that previously did not support this, such as variables of a BC9191 Controller. Furthermore, a Historical Access configuration can now also be carried out retrospectively, i.e. without having to adapt the PLC project.

The historical access configuration is now XML-based (TcUaHaConfig.xml) and can be carried out using the OPC UA Server Configurator. The XML file is located in the same directory as *TcOpcUaServer.exe*.

See also:

- Configuration > Setup version 4.x.x > Overview [▸ 192] (OPC UA Server Configurator) and Configuring historical access [▸ 200])

XML file structure (TCUaHaConfig.xml)

```
<HistoryConfig>
  <HistoryController>
    <HistoryAdapter Type="Volatile" Id="0" File="" Server="" Database="" User="" Password=""/>
    <HistoryAdapter Type="File" Id="1" File="historydb.dat" Server="" Database="" User=""
Password=""/>
```

```
    <HistoryAdapter Type="SQL" Id="2" File="" Server="" Database="database123" User="user"
Password="pwd"/>
    <HistoryAdapter Type="SQLCompact" Id="3" File="historydb.xyz" Server="" Database="" User=""
Password=""/>
    <HistoryAdapter Type="SQLCompact" Id="4" File="historydb2.xyz" Server="" Database="" User=""
Password=""/>
  </HistoryController>
  <HistoryNodes>
    <HistoryNode HistoryWriteable="true" SamplingRate="0" MaxSamples="100000" AdapterId="0"
NS="urn:BeckhoffAutomation:Ua:PLC1" NodeId="s=PRG_HA_SingleNode.nMyValue"/>
    <HistoryNode HistoryWriteable="true" SamplingRate="0" MaxSamples="100000" AdapterId="0"
NS="urn:BeckhoffAutomation:Ua:PLC1" NodeId="s=PRG_HA_MultipleNodes.nMyValue1"/>
    <HistoryNode HistoryWriteable="true" SamplingRate="0" MaxSamples="100000" AdapterId="0"
NS="urn:BeckhoffAutomation:Ua:PLC1" NodeId="s=PRG_HA_MultipleNodes.nMyValue2"/>
    <HistoryNode HistoryWriteable="true" SamplingRate="0" MaxSamples="100000" AdapterId="0"
NS="urn:BeckhoffAutomation:Ua:PLC1" NodeId="s=PRG_HA_MultipleNodes.fMyValue3"/>
  </HistoryNodes>
</HistoryConfig>
```

You can configure all supported data memory in the HistoryController area. The following locations are currently supported:

- RAM ("Volatile")
- File
- SQL Compact
- SQL Server (not supported under Windows CE)

Each HistoryAdapter (except for "Volatile") can be used multiple times, e.g. if the historical values for individual variables are to be stored in different data memories.

In the HistoryNodes area, the individual nodes are configured, assigned to a data memory and assigned a SamplingRate and a maximum size for the ring buffer to be used in the data memory.

The attribute `HistoryWriteable="true"` ensures that the data memory for this node can be filled with values via a HistoryUpdate() call. Such a call is provided, for example, by the TwinCAT OPC UA Client via the function block UA_HistoryUpdate. If you configure a node with this attribute, then the server doesn't normally "sample" the values itself, but receives them from other clients. In such a configuration the SamplingRate is therefore usually 0.

Requirements

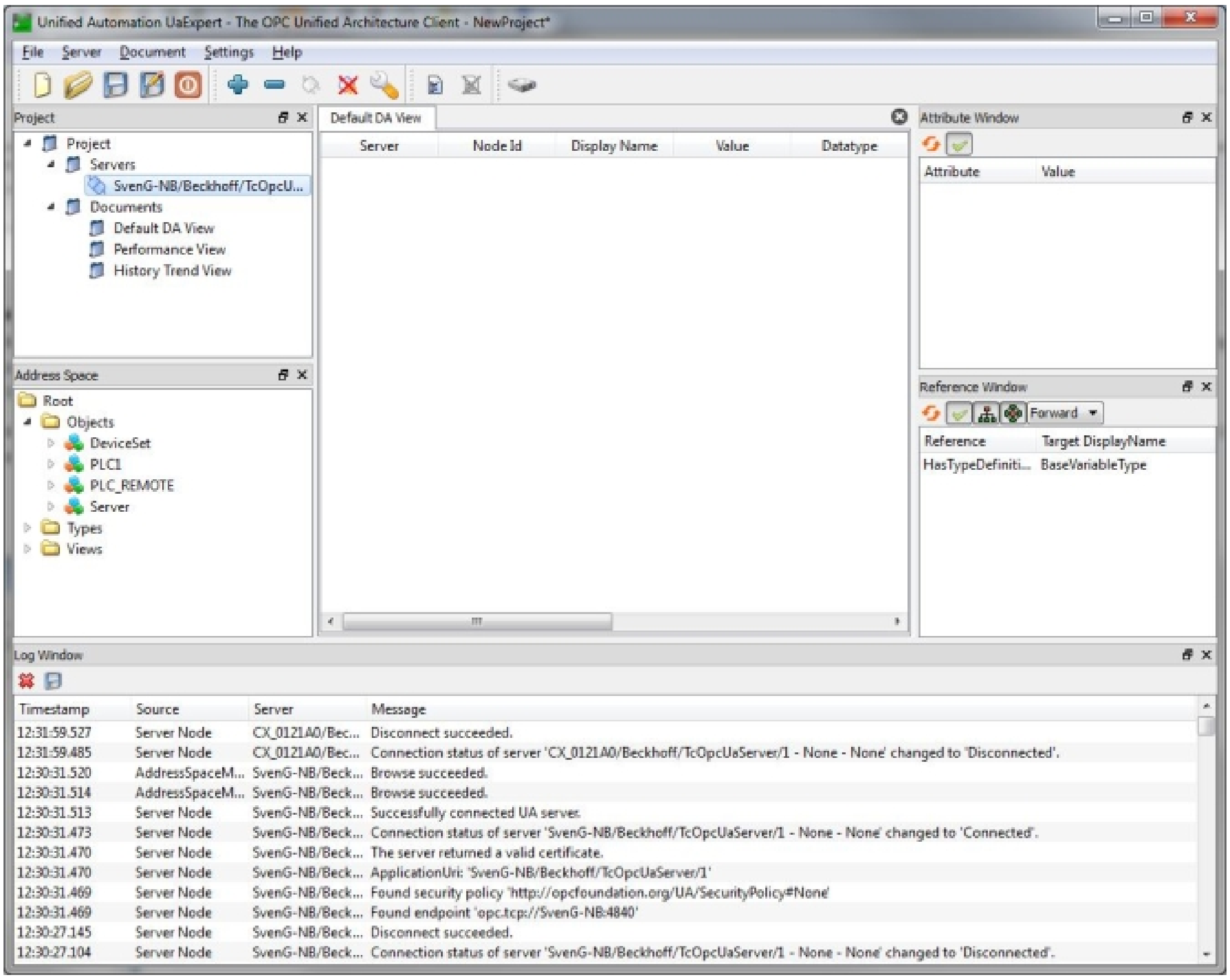
Products	Setup versions	Target platform
TF6100	4.x.x	IPC or CX (x86, x64, ARM)

4.1.5.2.2 Display historical data

Displaying Historical Access values in an OPC UA Client

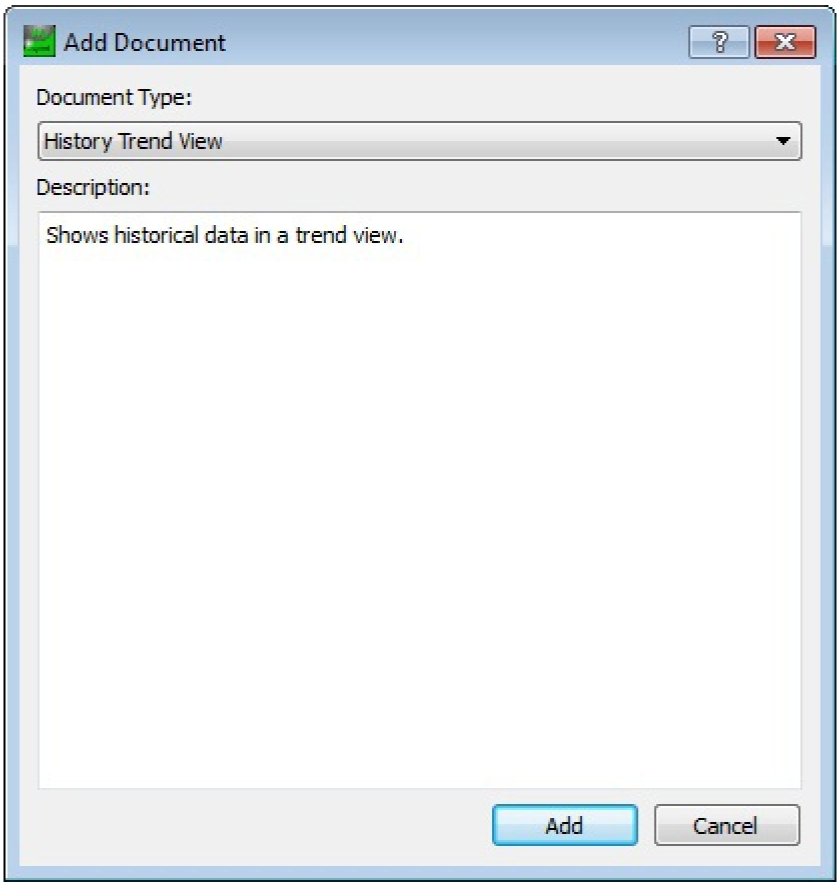
The following step-by-step instructions describe how to configure the UA Expert software in order to access historical data.

1. Start the UA Expert software and connect to the OPC UA Server.



```
Timestamp     Source           Server             Message
12:31:59.527  Server Node      CX_0121A0/Bec...   Disconnect succeeded.
12:31:59.485  Server Node      CX_0121A0/Bec...   Connection status of server 'CX_0121A0/Beckhoff/TcOpcUaServer/1 - None - None' changed to 'Disconnected'.
12:30:31.520  AddressSpaceM... SvenG-NB/Beck...   Browse succeeded.
12:30:31.514  AddressSpaceM... SvenG-NB/Beck...   Browse succeeded.
12:30:31.513  Server Node      SvenG-NB/Beck...   Successfully connected UA server.
12:30:31.473  Server Node      SvenG-NB/Beck...   Connection status of server 'SvenG-NB/Beckhoff/TcOpcUaServer/1 - None - None' changed to 'Connected'.
12:30:31.470  Server Node      SvenG-NB/Beck...   The server returned a valid certificate.
12:30:31.470  Server Node      SvenG-NB/Beck...   ApplicationUri: 'SvenG-NB/Beckhoff/TcOpcUaServer/1'
12:30:31.469  Server Node      SvenG-NB/Beck...   Found security policy 'http://opcfoundation.org/UA/SecurityPolicy#None'
12:30:31.469  Server Node      SvenG-NB/Beck...   Found endpoint 'opc.tcp://SvenG-NB:4840'
12:30:27.145  Server Node      SvenG-NB/Beck...   Disconnect succeeded.
12:30:27.104  Server Node      SvenG-NB/Beck...   Connection status of server 'SvenG-NB/Beckhoff/TcOpcUaServer/1 - None - None' changed to 'Disconnected'.
```

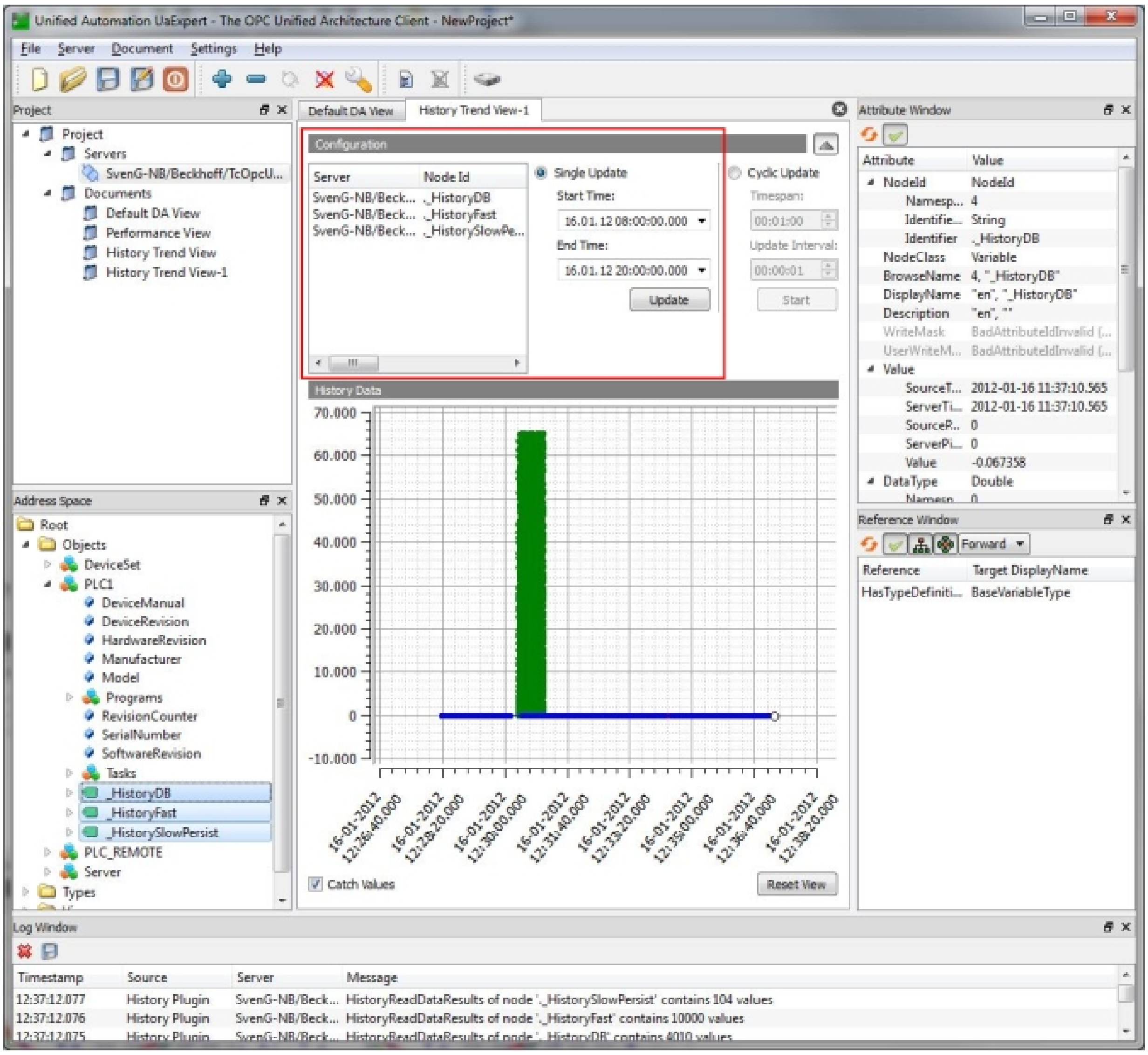
2. Add a new History Trend View.



Shows historical data in a trend view.

3. Browse the PLC1 namespace and use drag & drop to add the PLC variables _HistoryDB, _HistoryDBcompact, _HistoryFast and _HistorySlowPersist.

⇨ You can now use the **Start Time** and **End Time** controls to specify the desired time period for which the symbol values are to be displayed, or you can start a **Cyclic Update** for these variables if necessary.



Server	Node Id
SvenG-NB/Beck...	._HistoryDB
SvenG-NB/Beck...	._HistoryFast
SvenG-NB/Beck...	._HistorySlowPe...

Start Time: 16.01.12 08:00:00.000
End Time: 16.01.12 20:00:00.000
Timespan: 00:01:00
Update Interval: 00:00:01

Attribute	Value
NodeId	NodeId
Namesp...	4
Identifie...	String
Identifier	._HistoryDB
NodeClass	Variable
BrowseName	4, "._HistoryDB"
DisplayName	"en", "._HistoryDB"
Description	"en", ""
WriteMask	BadAttributeIdInvalid (...
UserWriteM...	BadAttributeIdInvalid (...
Value	
SourceT...	2012-01-16 11:37:10.565
ServerTi...	2012-01-16 11:37:10.565
SourceP...	0
ServerPi...	0
Value	-0.067358
DataType	Double

Reference	Target DisplayName
HasTypeDefiniti...	BaseVariableType

```
Timestamp     Source          Server             Message
12:37:12.077  History Plugin  SvenG-NB/Beck...   HistoryReadDataResults of node '._HistorySlowPersist' contains 104 values
12:37:12.076  History Plugin  SvenG-NB/Beck...   HistoryReadDataResults of node '._HistoryFast' contains 10000 values
12:37:12.075  History Plugin  SvenG-NB/Beck...   HistoryReadDataResults of node '._HistoryDB' contains 4010 values
```

4.1.5.3 Alarms and Conditions

The steps required to configure OPC UA Alarms and Conditions (A&C) on the OPC UA Server are described in this section. The basic concept is independent of TwinCAT runtime, which means the configuration steps for PLC, C++, TcCOM runtime or only I/O task are the same.

OPC UA Alarms and Conditions (Part 9 of the OPC UA specification) describes a model for monitoring process values and outputting alarms and events when a runtime symbol changes its state.

Requirements

The following requirements apply to the use of OPC UA Alarms and Conditions:

- The runtime symbol to be monitored must be available in the namespace.
- The OPC UA Client must support Alarms and Conditions. In this section UA Expert (from Unified Automation) is used as the reference UA Client.

General Information

Execute the following steps once to release a symbol for Alarms and Conditions:

- Step 1: Activating runtime symbol for data access [▸ 101] (so that the symbol is generally accessible via OPC UA.)
- Step 2: Activating A&C for a symbol [▸ 101]
- Step 3: Transferring your own user data with an event [▸ 102]
- Step 4: Triggering an event using the FireEvent method [▸ 103]
- Step 5: Configuring multilingual alarm texts [▸ 105]
- Step 6: Registering A&C with a reference OPC UA Client [▸ 105]

These steps are explained in more detail below. At the end of this section you will find information about receiving configured alarms via A&C with the UA Expert Reference Client.

Supported alarm types

The implementation of OPC UA Alarms and Conditions currently supports the following alarm types:

- LimitAlarmType: Define different limits for a symbol. If a limit is reached, the UA Server issues an alarm.
- OffNormalAlarmType: Define a value that is "normal". If the current value deviates from the "normal" value, the UA Server issues an alarm.

Step 1: Activating runtime symbol for data access

A variable must be available in the OPC UA Server in order for it to be configured for A&C. To do this in the case of a PLC variable, provide it with an attribute (see Access to PLC runtime [▸ 43]).

Step 2: Activating A&C for a symbol

You can configure a runtime symbol for A&C with the OPC UA Server Configurator. The Configurator has a simple graphical user interface for editing the XML file on which it is based. Depending on the setup version the configurator is available in two variants: standalone [▸ 190] and integrated in Visual Studio [▸ 202].

The following program extract shows an example of this XML file to better understand the general behavior and structure of the A&C implementation.

```
<TcUaAcConfig>
  <ConditionController Name="ConditionController1" >
    <Condition Name="Counter" Severity="200">
    <LimitAlarmType LowLowLimit="-10" LowLimit="0" HighLimit="10" HighHighLimit="20"
MessageNormal="100" MessageLowLow="10" MessageLow="11" MessageHigh="12" MessageHighHigh="13"/>
      <ItemToMonitor SamplingRate="100" NS="urn:[NodeName]:BeckhoffAutomation:Ua:PLC1"
NodeId="s=MAIN.nCounter1" />
    </Condition>
    <Condition Name="Switch" Severity="500">
      <OffNormalAlarmType Normal="0" MessageNormal="100" MessageOffNormal="20" />
      <ItemToMonitor SamplingRate="100" NS="urn:[NodeName]:BeckhoffAutomation:Ua:PLC1"
NodeId="s=MAIN.bSwitch" />
    </Condition>
    <Condition Name="Struct" Severity="300">
      <LimitAlarmType LowLowLimit="-10" LowLimit="0" HighLimit="10" HighHighLimit="20"
MessageNormal="100" MessageLowLow="10" MessageLow="11" MessageHigh="12" MessageHighHigh="13"/>
      <ItemToMonitor SamplingRate="100" NS="urn:[NodeName]:BeckhoffAutomation:Ua:PLC1"
NodeId="s=MAIN.stStruct" />
    </Condition>
  </ConditionController>
  <ConditionController Name="ConditionController2" >
    <Condition Name="Counter2" Severity="200">
      <LimitAlarmType LowLowLimit="-10" LowLimit="0" HighLimit="10" HighHighLimit="20"
MessageNormal="100" MessageLowLow="10" MessageLow="11" MessageHigh="12" MessageHighHigh="13"/>
      <ItemToMonitor SamplingRate="100" NS="urn:[NodeName]:BeckhoffAutomation:Ua:PLC1"
NodeId="s=MAIN.nCounter2" />
    </Condition>
  </ConditionController>
</TcUaAcConfig>
```

A "condition" defines the runtime symbol to be monitored and the alarm limits and texts. Each condition is organized in a so-called "ConditionController", the object that the OPC UA Clients subsequently subscribe to.

When creating a condition, you must specify the NamespaceName (NS) and NodeID for referring to the UA node to be monitored. The Configurator provides a simple browsing mechanism for selecting a node. In XML, the placeholder [NodeName] in NamespaceName can be used to switch the XML file between different hardware systems. NamespaceName always contains the host name of the IPC or Embedded PC on which the OPC UA Server is running. If [NodeName] is selected, this tag will be replaced by the host name of the current IPC or Embedded PC on which the UA Server is running.

SamplingRate determines how often the UA Server should request a value from the node to determine whether one of the alarm limits has been reached.

The alarm texts are identified by an ID. The ID uniquely identifies an alarm text from the resource file (see Step 5: Configuring multilingual alarm texts [▶ 105]).

Step 3: Transferring own user data with an alarm

Alarms can contain fields with their own user data, which complement the data output with the alarm. These user data fields can be created and filled out in the runtime application. To do this, create a STRUCT and name its first element "value". In case of an alarm, all the following elements are then sent in an additional user data field.

Sample PLC application:

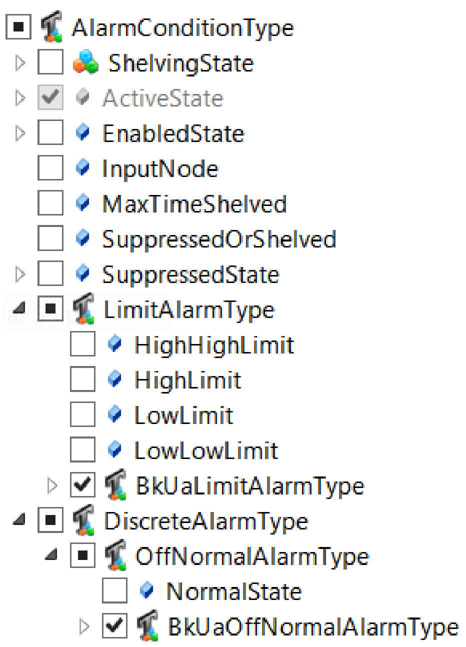
```
TYPE ST_CustomStruct :
STRUCT
  value : INT;
  data  : ST_SomeStruct;
END_STRUCT
END_TYPE

TYPE ST_SomeStruct :
STRUCT
  Data1 : INT;
  Data2 : REAL;
  Data3 : LREAL;
END_STRUCT
END_TYPE
```

The instance of ST_CustomStruct is then enabled for data access via the regular mechanism, e.g. in TwinCAT 3: To do this the STRUCT must be activated as a StructuredType [▶ 88].

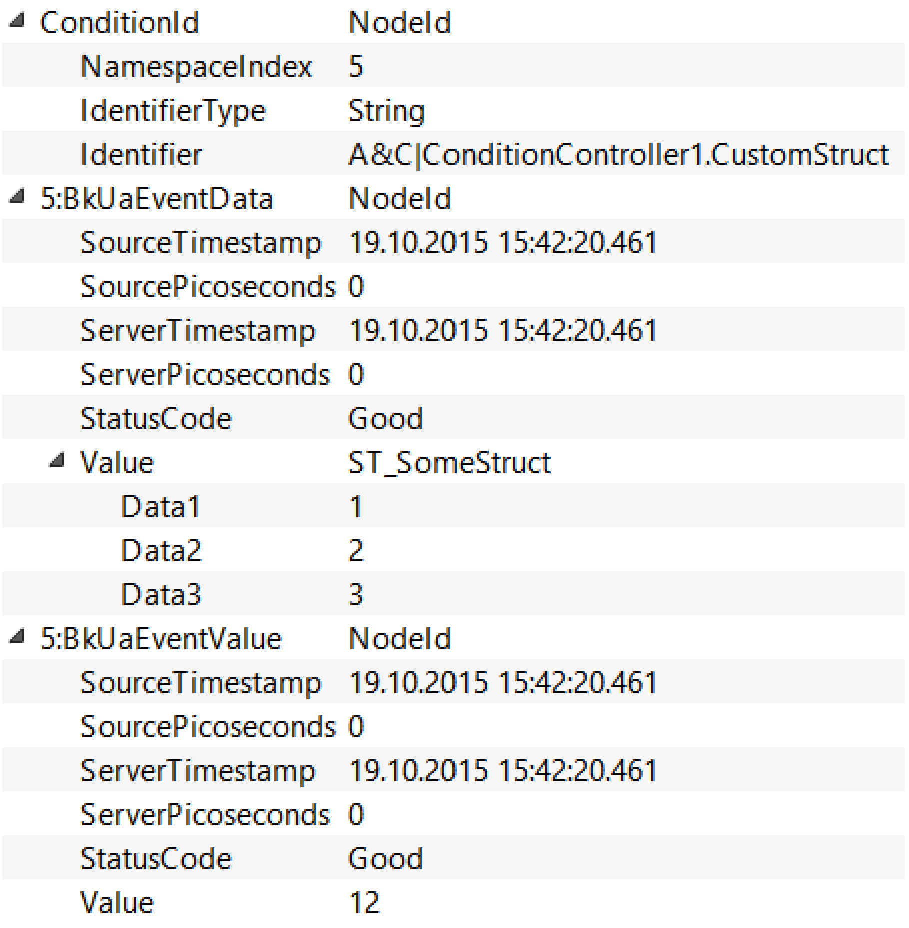
```
PROGRAM MAIN
VAR
  {attribute 'OPC.UA.DA' := '1'}
  {attribute 'OPC.UA.DA.StructuredType' := '1'}
  stCustomStruct : ST_CustomStruct;
END_VAR
```

When logging on to a ConditionController, the OPC UA Clients must subscribe to special AlarmConditionTypes, i.e. "BkUaLimitAlarmType" and "BkUaOffNormalAlarmType", so that they can receive the special user data fields when an alarm is received.



- ■ AlarmConditionType
  - ☐ ShelvingState
  - ☑ ActiveState
  - ☐ EnabledState
  - ☐ InputNode
  - ☐ MaxTimeShelved
  - ☐ SuppressedOrShelved
  - ☐ SuppressedState
  - ■ LimitAlarmType
    - ☐ HighHighLimit
    - ☐ HighLimit
    - ☐ LowLimit
    - ☐ LowLowLimit
    - ☑ BkUaLimitAlarmType
  - ■ DiscreteAlarmType
    - ■ OffNormalAlarmType
      - ☐ NormalState
      - ☑ BkUaOffNormalAlarmType

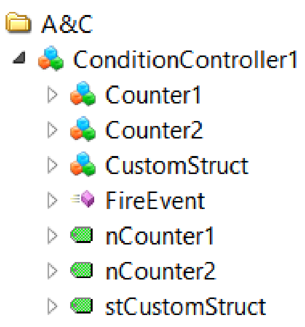
The OPC UA Client then receives the user data in the fields BkUaEventData and BkUaEventValue of the incoming alarm. In the above sample, these are the value of the PLC variables and the user data represented by the PLC structure ST_SomeStruct.

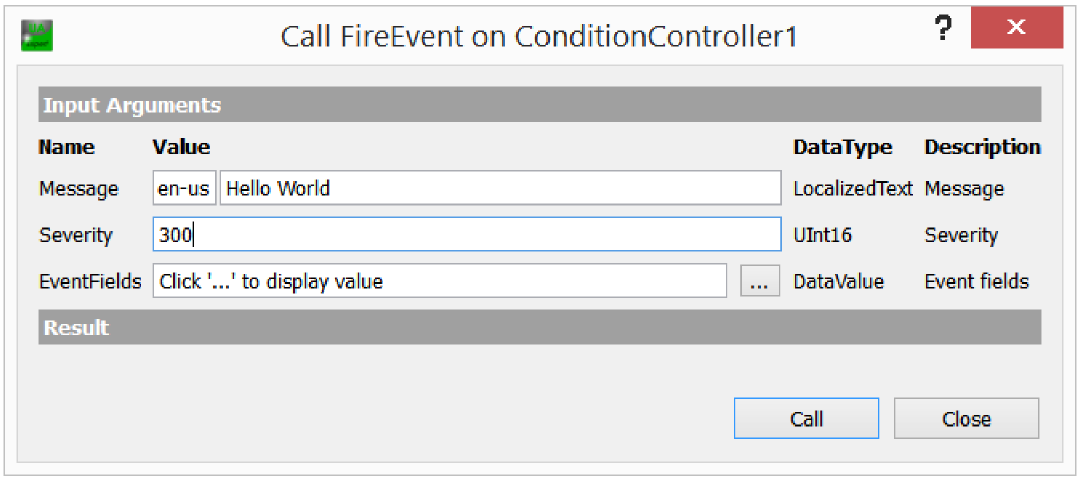


ConditionId	NodeId
NamespaceIndex	5
IdentifierType	String
Identifier	A&C|ConditionController1.CustomStruct
5:BkUaEventData	NodeId
SourceTimestamp	19.10.2015 15:42:20.461
SourcePicoseconds	0
ServerTimestamp	19.10.2015 15:42:20.461
ServerPicoseconds	0
StatusCode	Good
Value	ST_SomeStruct
Data1	1
Data2	2
Data3	3
5:BkUaEventValue	NodeId
SourceTimestamp	19.10.2015 15:42:20.461
SourcePicoseconds	0
ServerTimestamp	19.10.2015 15:42:20.461
ServerPicoseconds	0
StatusCode	Good
Value	12

Step 4: Triggering an event using the FireEvent method

Each ConditionController includes a FireEvent method with which the OPC UA Clients can trigger a general event with user-defined EventFields.



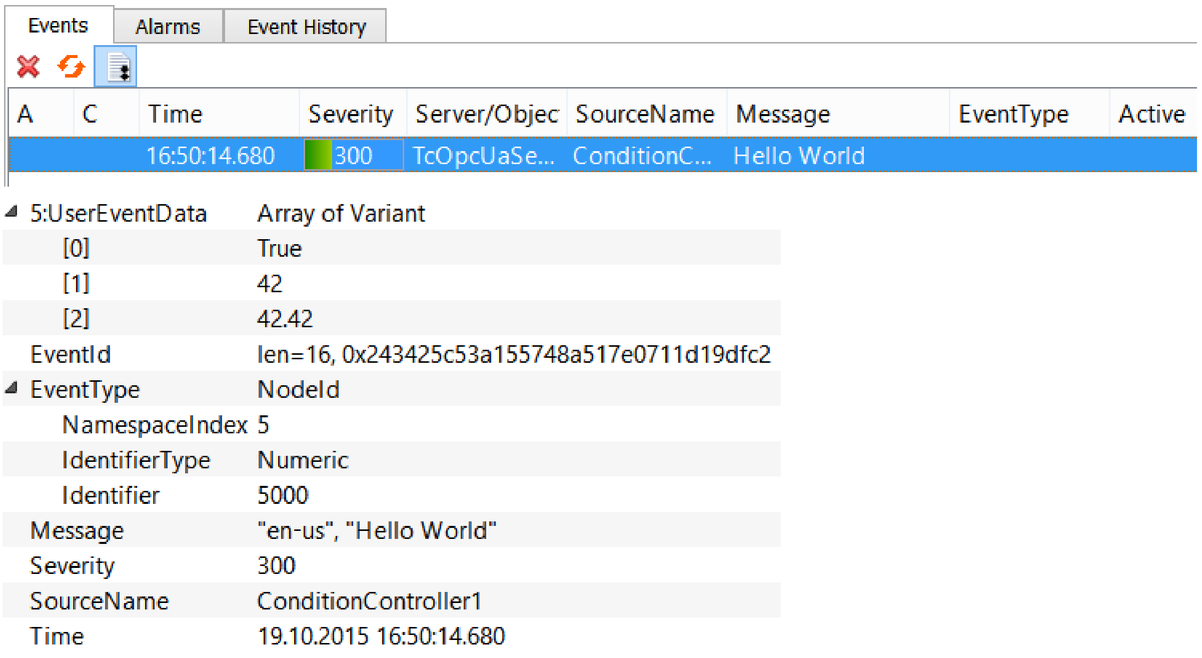

Call FireEvent on ConditionController1

Input Arguments

Name	Value	DataType	Description
Message	en-us Hello World	LocalizedText	Message
Severity	300	UInt16	Severity
EventFields	Click '...' to display value	DataValue	Event fields

Result

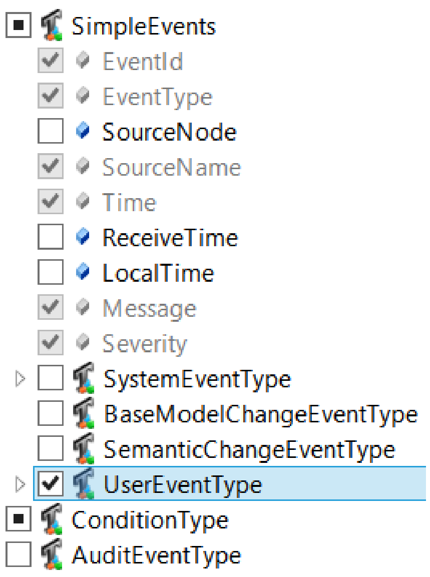
If the method is executed, an event is output on the OPC UA Server. Other OPC UA Clients can receive these events when they subscribe to the corresponding ConditionController.



A	C	Time	Severity	Server/Objec	SourceName	Message	EventType	Active
		16:50:14.680	300	TcOpcUaSe...	ConditionC...	Hello World		

5:UserEventData	Array of Variant
[0]	True
[1]	42
[2]	42.42
EventId	len=16, 0x243425c53a155748a517e0711d19dfc2
EventType	NodeId
NamespaceIndex	5
IdentifierType	Numeric
Identifier	5000
Message	"en-us", "Hello World"
Severity	300
SourceName	ConditionController1
Time	19.10.2015 16:50:14.680

The user-defined EventFields are appended to the event as "UserEventData". This data can be received by OPC UA Clients that are logged on to the SimpleEventType "UserEventType".



Step 5: Configuring multilingual alarm texts

The A&C implementation in the OPC UA Server supports the use of multilingual alarm texts. The alarm text that is used depends on the language with which the UA Client connects to the Server.

Alarm texts are configured in XML files. There is a separate file for each language. These files are located in the res (Resource) folder on the OPC UA Server. With the configurator you can simply add or remove alarm texts without having to directly edit the XML files. Each alarm text has its own ID, which occurs only once in the file.

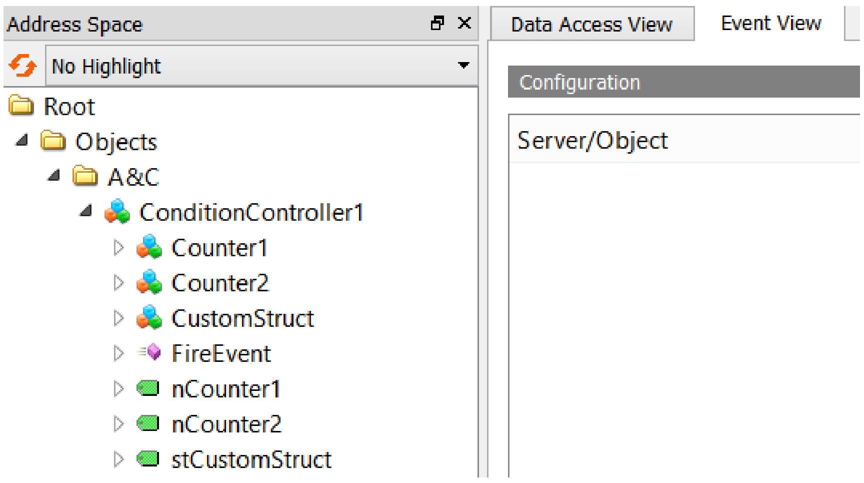
Sample:

```
<TcOpcUaSvrRes Lang="en">
  <Text ID="0">Text not available</Text>
  <Text ID="1">Some alarm text</Text>
  <Text ID="2">Value is High range</Text>
  <Text ID="3">Value is HighHigh range</Text>
  <Text ID="4">Value is OffNormal</Text>
  ...
</TcOpcUaSvrRes>
```

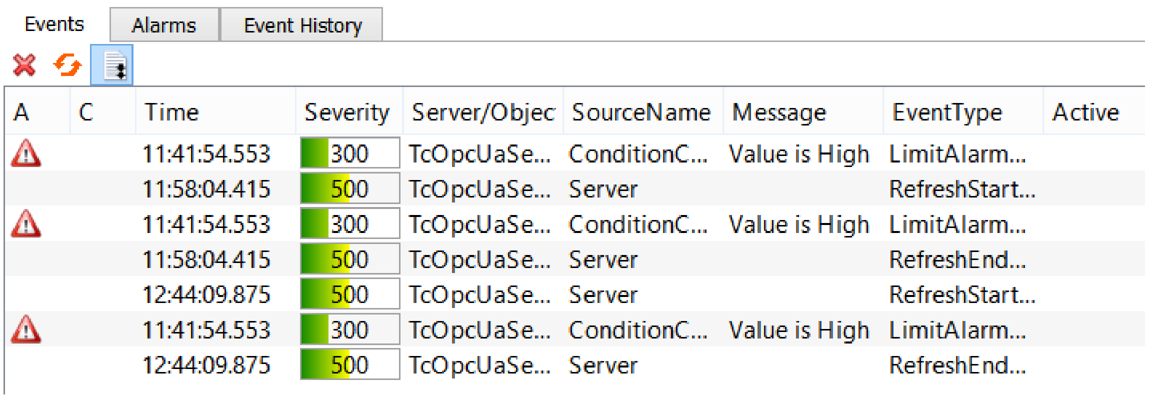
Step 6: Registering A&C with a reference OPC UA Client

An OPC UA Client must log on to a ConditionController so that it can receive events for conditions that are configured for this particular ConditionController. The UA Expert provides functions for subscribing to and receiving UA events.

After you have started the UA Expert and established a connection with the OPC UA Server, add a new document view, Event View, to your working area.



You can then subscribe to a ConditionController by dragging the corresponding object in Event View. The events for this ConditionController are displayed in the Event Window.



A	C	Time	Severity	Server/Objec	SourceName	Message	EventType	Active
⚠		11:41:54.553	300	TcOpcUaSe...	ConditionC...	Value is High	LimitAlarm...	
		11:58:04.415	500	TcOpcUaSe...	Server		RefreshStart...	
⚠		11:41:54.553	300	TcOpcUaSe...	ConditionC...	Value is High	LimitAlarm...	
		11:58:04.415	500	TcOpcUaSe...	Server		RefreshEnd...	
		12:44:09.875	500	TcOpcUaSe...	Server		RefreshStart...	
⚠		11:41:54.553	300	TcOpcUaSe...	ConditionC...	Value is High	LimitAlarm...	
		12:44:09.875	500	TcOpcUaSe...	Server		RefreshEnd...	

It may be necessary to subscribe to special alarm and/or event types in order to receive all fields of an incoming event or alarm.

4.1.6 Matlab/Simulink

4.1.6.1 Access to Matlab/Simulink modules

This section describes how to use TwinCAT 3 TMI files for TcCOM modules to construct the namespace of the OPC UA Server.

This section contains the following topics:

- General Information [▶ 106]
- Generating and importing a TMI file for TcCom modules [▶ 106]
- Filtered or unfiltered mode [▶ 109]

General Information

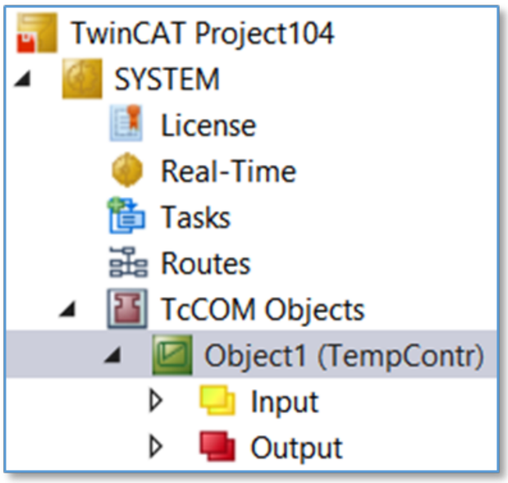
If you use TwinCAT 3 C++ or the Matlab/Simulink integration, you can generate a symbol file (TMI file) for the arising TcCom modules from the TwinCAT 3 XAE.

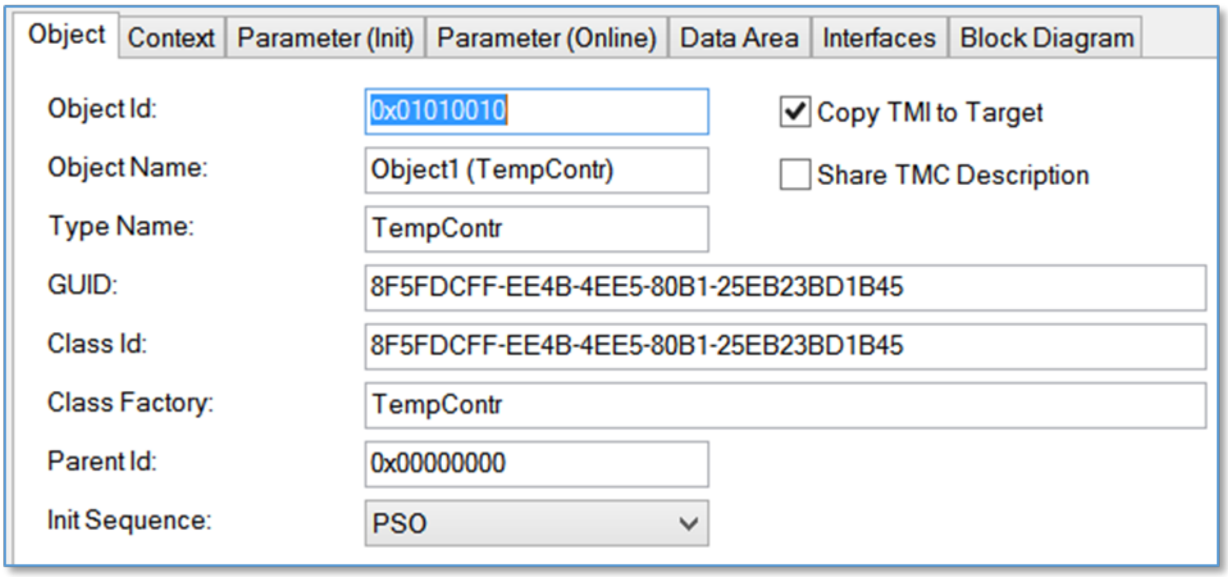
The OPC UA Server imports this TMI file and generates its namespace using the symbol information contained in the file. The namespace can then consist of variables (= symbols) that are contained in the respective TcCom module.

The import of TMI files is configured in the OPC UA Configurator.

Generating and importing a TMI file for TcCom modules

In order for a TMI file to be generated and for TwinCAT to automatically copy the TMI file to the target system, enable the option **Copy TMI to Target** in the settings of the TcCOM module:



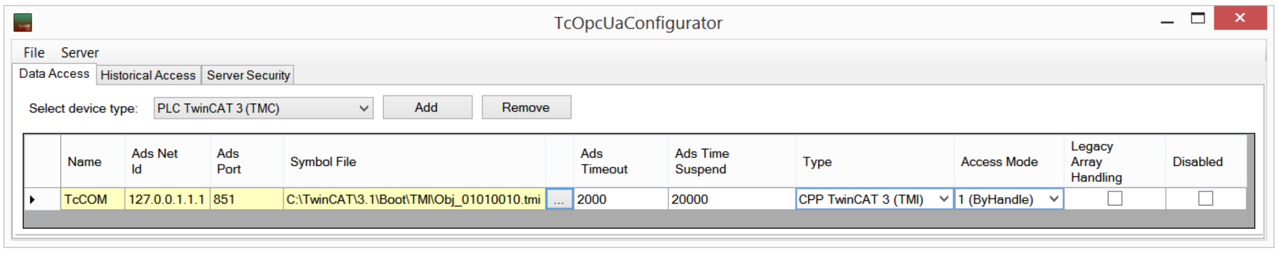


Object Id:	0x01010010
Object Name:	Object1 (TempContr)
Type Name:	TempContr
GUID:	8F5FDCFF-EE4B-4EE5-80B1-25EB23BD1B45
Class Id:	8F5FDCFF-EE4B-4EE5-80B1-25EB23BD1B45
Class Factory:	TempContr
Parent Id:	0x00000000
Init Sequence:	PSO

✓ Copy TMI to Target

☐ Share TMC Description

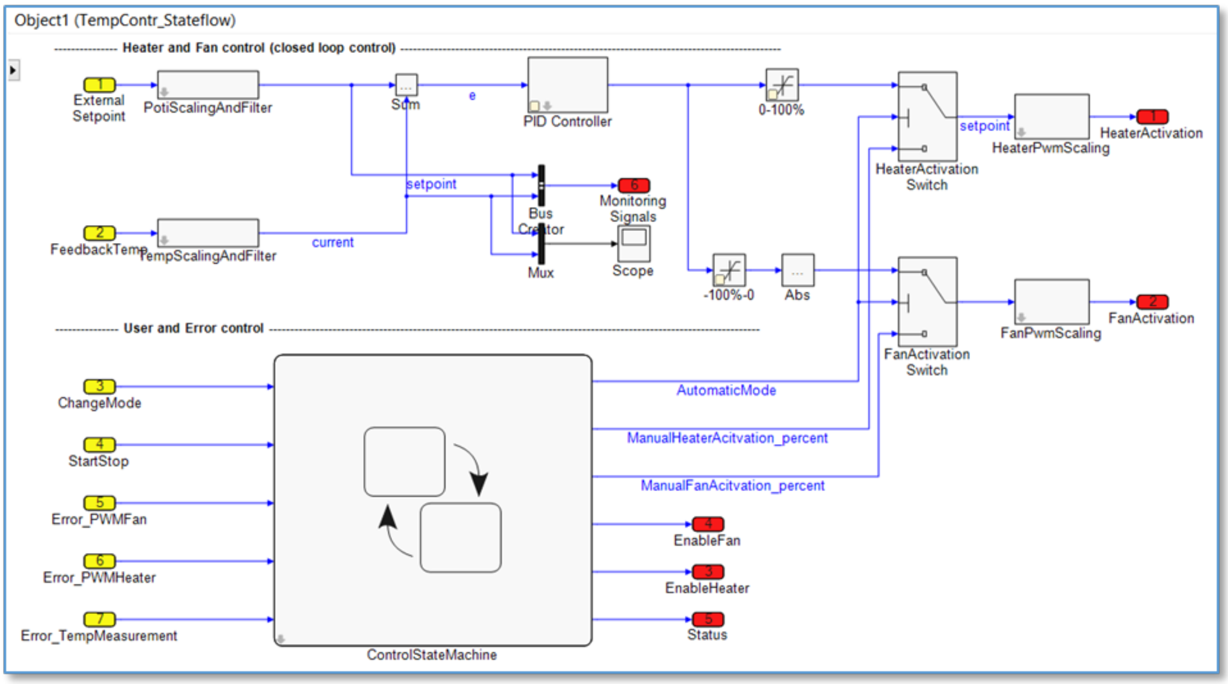
After activating the project, the boot directory of the target system contains the TMI file for the TcCOM module. This file can then be imported into the OPC UA namespace with the help of the OPC UA Configurator.

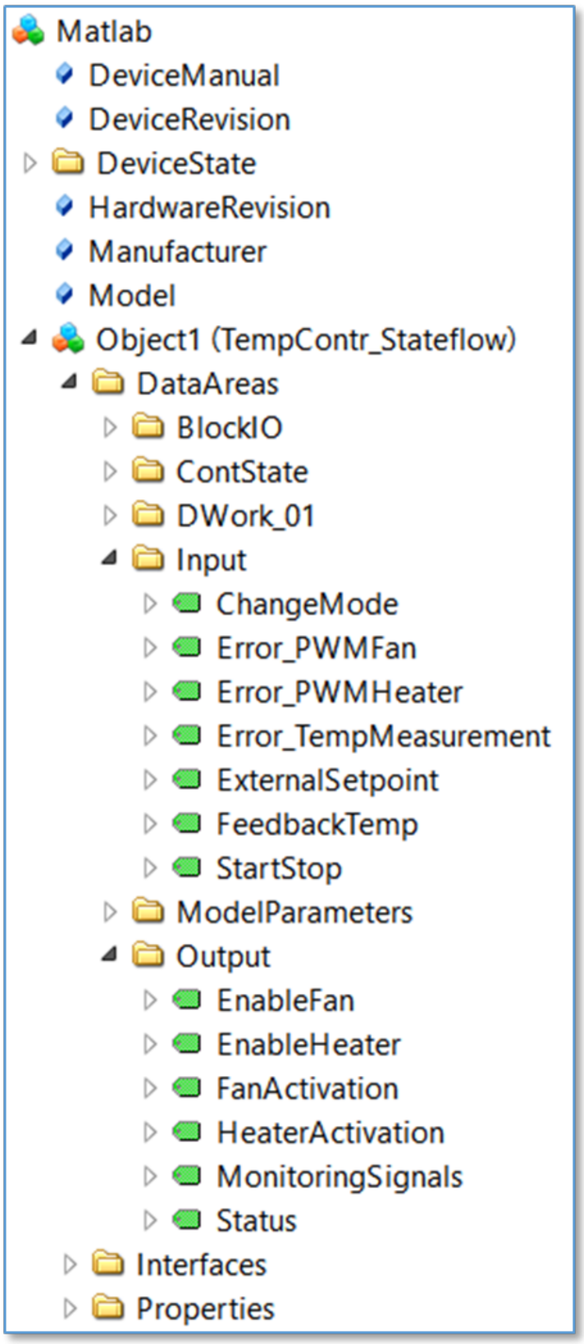


Select device type: PLC TwinCAT 3 (TMC)

Name	Ads Net Id	Ads Port	Symbol File		Ads Timeout	Ads Time Suspend	Type	Access Mode	Legacy Array Handling	Disabled
TcCOM	127.0.0.1.1.1	851	C:\TwinCAT\3.1\Boot\TMI\Obj_01010010.tmi	...	2000	20000	CPP TwinCAT 3 (TMI)	1 (ByHandle)	☐	☐

The configured OPC UA namespace then contains a representation of the Matlab/Simulink block diagram:


Object1 (TempContr_Stateflow)
Heater and Fan control (closed loop control)
1
External Setpoint
PotiScalingAndFilter
Sum
e
PID Controller
0-100%
HeaterActivation Switch
setpoint
HeaterPwmScaling
1
HeaterActivation
setpoint
Bus Creator
6
Monitoring Signals
2
FeedbackTemp
TempScalingAndFilter
current
Mux
Scope
-100%-0
Abs
FanActivation Switch
FanPwmScaling
2
FanActivation
User and Error control
3
ChangeMode
4
StartStop
5
Error_PWMFan
6
Error_PWMHeater
7
Error_TempMeasurement
ControlStateMachine
AutomaticMode
ManualHeaterAcitvation_percent
ManualFanAcitvation_percent
4
EnableFan
3
EnableHeater
5
Status



```
Matlab
    DeviceManual
    DeviceRevision
    DeviceState
    HardwareRevision
    Manufacturer
    Model
    Object1 (TempContr_Stateflow)
        DataAreas
            BlockIO
            ContState
            DWork_01
            Input
                ChangeMode
                Error_PWMFan
                Error_PWMHeater
                Error_TempMeasurement
                ExternalSetpoint
                FeedbackTemp
                StartStop
            ModelParameters
            Output
                EnableFan
                EnableHeater
                FanActivation
                HeaterActivation
                MonitoringSignals
                Status
        Interfaces
        Properties
```

Filtered or unfiltered mode

Note that there is currently no filtered mode for generic TcCOM module instances. Filtered mode is only possible for TwinCAT 3 C++ module instances if the TMC Code Editor is used. TwinCAT Matlab/Simulink integration offers several filter options. These are described in the documentation TE1400 in section Data exchange.

If a generic TcCOM module instance (e.g. a Matlab/Simulink module) is to be imported, the unfiltered option must be used and therefore all symbols of the modules are published in the OPC UA namespace.

4.1.6.2 Historical Access

4.1.6.2.1 Setup Version 4.x.x

From setup version 4.x.x historical access can no longer be configured via attributes or comments. The advantage of this is that the Historical Access functionality is now also available for runtimes that previously did not support this, such as variables of a BC9191 Controller. Furthermore, a Historical Access configuration can now also be carried out retrospectively, i.e. without having to adapt the PLC project.

The historical access configuration is now XML-based (TcUaHaConfig.xml) and can be carried out using the OPC UA Server Configurator. The XML file is located in the same directory as *TcOpcUaServer.exe*.

See also:

- Configuration > Setup version 4.x.x > Overview [▸ 192] (OPC UA Server Configurator) and Configuring historical access [▸ 200])

XML file structure (TCUaHaConfig.xml)

```
<HistoryConfig>
  <HistoryController>
    <HistoryAdapter Type="Volatile" Id="0" File="" Server="" Database="" User="" Password=""/>
    <HistoryAdapter Type="File" Id="1" File="historydb.dat" Server="" Database="" User=""
Password=""/>
    <HistoryAdapter Type="SQL" Id="2" File="" Server="" Database="database123" User="user"
Password="pwd"/>
    <HistoryAdapter Type="SQLCompact" Id="3" File="historydb.xyz" Server="" Database="" User=""
Password=""/>
    <HistoryAdapter Type="SQLCompact" Id="4" File="historydb2.xyz" Server="" Database="" User=""
Password=""/>
  </HistoryController>
  <HistoryNodes>
    <HistoryNode HistoryWriteable="true" SamplingRate="0" MaxSamples="100000" AdapterId="0"
NS="urn:BeckhoffAutomation:Ua:PLC1" NodeId="s=PRG_HA_SingleNode.nMyValue"/>
    <HistoryNode HistoryWriteable="true" SamplingRate="0" MaxSamples="100000" AdapterId="0"
NS="urn:BeckhoffAutomation:Ua:PLC1" NodeId="s=PRG_HA_MultipleNodes.nMyValue1"/>
    <HistoryNode HistoryWriteable="true" SamplingRate="0" MaxSamples="100000" AdapterId="0"
NS="urn:BeckhoffAutomation:Ua:PLC1" NodeId="s=PRG_HA_MultipleNodes.nMyValue2"/>
    <HistoryNode HistoryWriteable="true" SamplingRate="0" MaxSamples="100000" AdapterId="0"
NS="urn:BeckhoffAutomation:Ua:PLC1" NodeId="s=PRG_HA_MultipleNodes.fMyValue3"/>
  </HistoryNodes>
</HistoryConfig>
```

You can configure all supported data memory in the HistoryController area. The following locations are currently supported:

- RAM ("Volatile")
- File
- SQL Compact
- SQL Server (not supported under Windows CE)

Each HistoryAdapter (except for "Volatile") can be used multiple times, e.g. if the historical values for individual variables are to be stored in different data memories.

In the HistoryNodes area, the individual nodes are configured, assigned to a data memory and assigned a SamplingRate and a maximum size for the ring buffer to be used in the data memory.

The attribute `HistoryWriteable="true"` ensures that the data memory for this node can be filled with values via a HistoryUpdate() call. Such a call is provided, for example, by the TwinCAT OPC UA Client via the function block UA_HistoryUpdate. If you configure a node with this attribute, then the server doesn't normally "sample" the values itself, but receives them from other clients. In such a configuration the SamplingRate is therefore usually 0.

Requirements

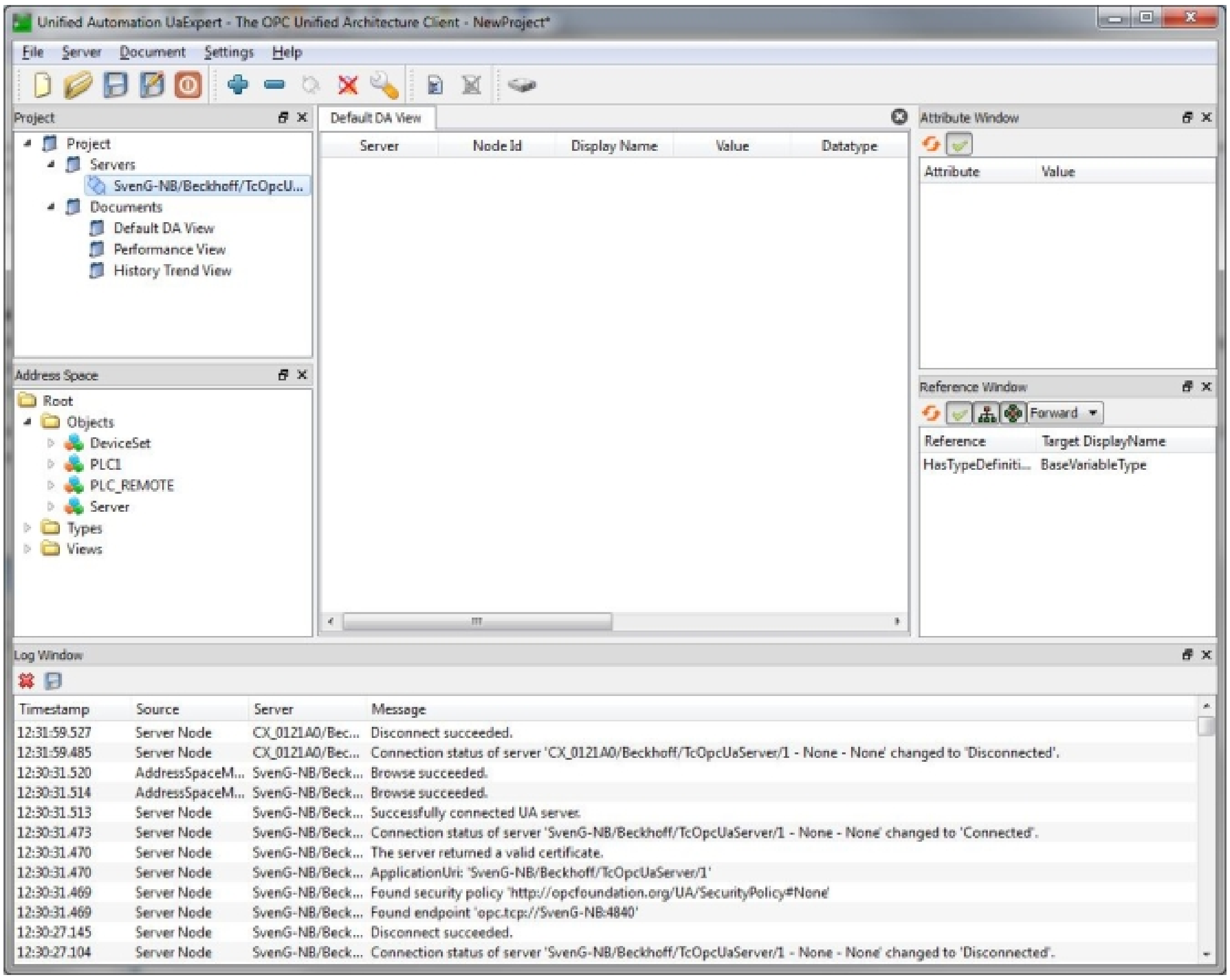
Products	Setup versions	Target platform
TF6100	4.x.x	IPC or CX (x86, x64, ARM)

4.1.6.2.2 Display historical data

Displaying Historical Access values in an OPC UA Client

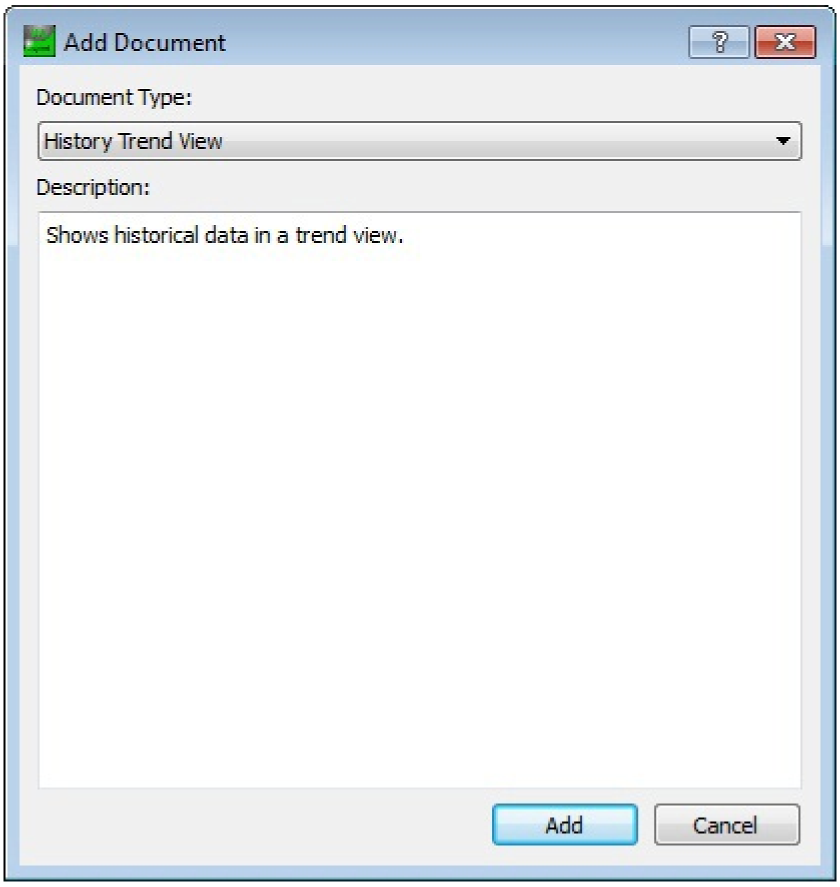
The following step-by-step instructions describe how to configure the UA Expert software in order to access historical data.

1. Start the UA Expert software and connect to the OPC UA Server.



```
Timestamp     Source           Server              Message
12:31:59.527  Server Node      CX_0121A0/Bec...    Disconnect succeeded.
12:31:59.485  Server Node      CX_0121A0/Bec...    Connection status of server 'CX_0121A0/Beckhoff/TcOpcUaServer/1 - None - None' changed to 'Disconnected'.
12:30:31.520  AddressSpaceM... SvenG-NB/Beck...    Browse succeeded.
12:30:31.514  AddressSpaceM... SvenG-NB/Beck...    Browse succeeded.
12:30:31.513  Server Node      SvenG-NB/Beck...    Successfully connected UA server.
12:30:31.473  Server Node      SvenG-NB/Beck...    Connection status of server 'SvenG-NB/Beckhoff/TcOpcUaServer/1 - None - None' changed to 'Connected'.
12:30:31.470  Server Node      SvenG-NB/Beck...    The server returned a valid certificate.
12:30:31.470  Server Node      SvenG-NB/Beck...    ApplicationUri: 'SvenG-NB/Beckhoff/TcOpcUaServer/1'
12:30:31.469  Server Node      SvenG-NB/Beck...    Found security policy 'http://opcfoundation.org/UA/SecurityPolicy#None'
12:30:31.469  Server Node      SvenG-NB/Beck...    Found endpoint 'opc.tcp://SvenG-NB:4840'
12:30:27.145  Server Node      SvenG-NB/Beck...    Disconnect succeeded.
12:30:27.104  Server Node      SvenG-NB/Beck...    Connection status of server 'SvenG-NB/Beckhoff/TcOpcUaServer/1 - None - None' changed to 'Disconnected'.
```

2. Add a new History Trend View.



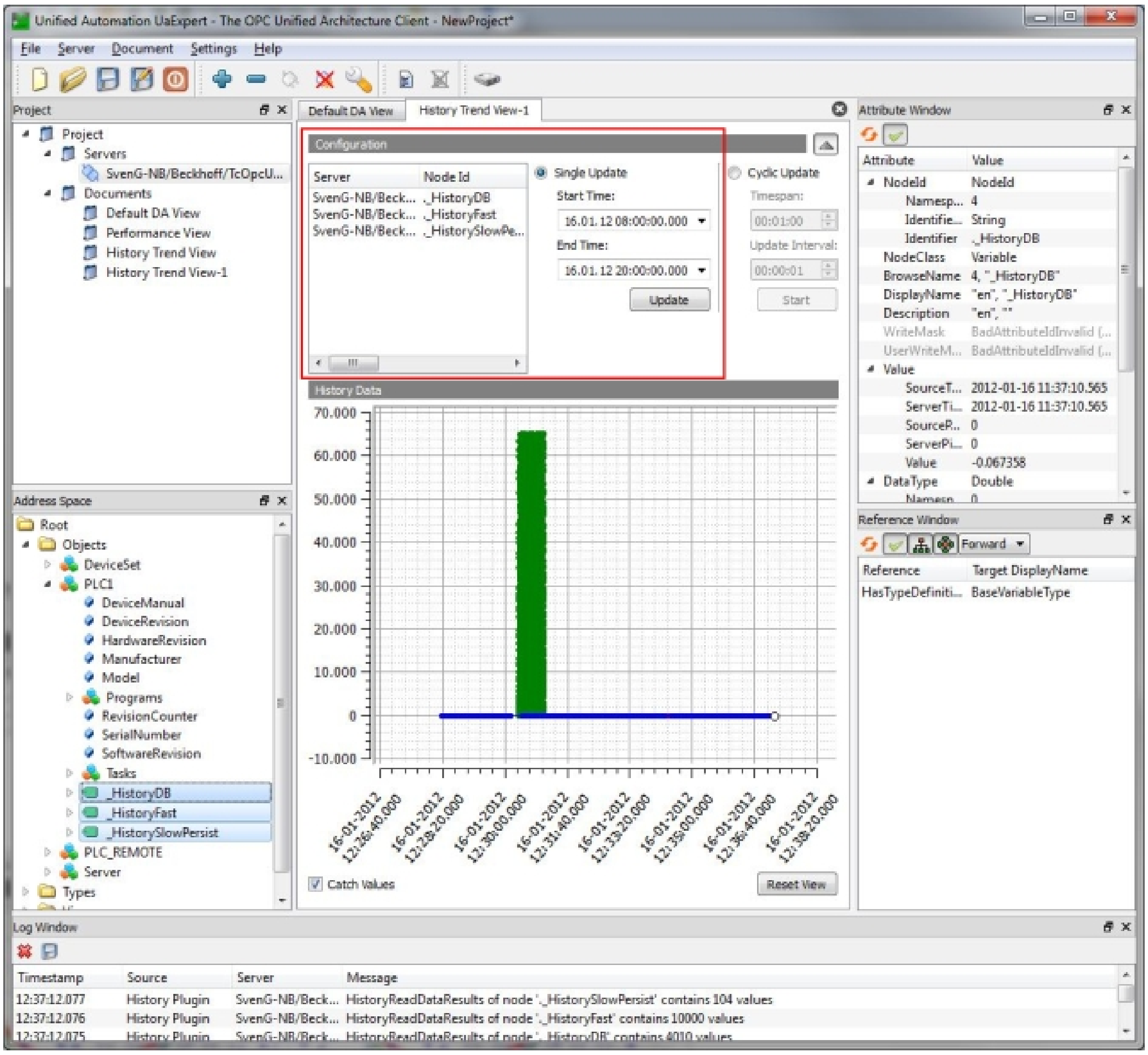
Document Type: History Trend View

Description:

Shows historical data in a trend view.

3. Browse the PLC1 namespace and use drag & drop to add the PLC variables _HistoryDB, _HistoryDBcompact, _HistoryFast and _HistorySlowPersist.

⇨ You can now use the **Start Time** and **End Time** controls to specify the desired time period for which the symbol values are to be displayed, or you can start a **Cyclic Update** for these variables if necessary.



Server	Node Id
SvenG-NB/Beck...	._HistoryDB
SvenG-NB/Beck...	._HistoryFast
SvenG-NB/Beck...	._HistorySlowPe...

Attribute	Value
NodeId	NodeId
Namesp...	4
Identifie...	String
Identifier	._HistoryDB
NodeClass	Variable
BrowseName	4, "_HistoryDB"
DisplayName	"en", "_HistoryDB"
Description	"en", ""
WriteMask	BadAttributeIdInvalid (...
UserWriteM...	BadAttributeIdInvalid (...
Value	
SourceT...	2012-01-16 11:37:10.565
ServerTi...	2012-01-16 11:37:10.565
SourceP...	0
ServerPi...	0
Value	-0.067358
DataType	Double
Namesp...	0

Reference	Target DisplayName
HasTypeDefiniti...	BaseVariableType

```
Timestamp     Source          Server            Message
12:37:12.077  History Plugin  SvenG-NB/Beck...  HistoryReadDataResults of node '._HistorySlowPersist' contains 104 values
12:37:12.076  History Plugin  SvenG-NB/Beck...  HistoryReadDataResults of node '._HistoryFast' contains 10000 values
12:37:12.075  History Plugin  SvenG-NB/Beck...  HistoryReadDataResults of node '._HistoryDB' contains 4010 values
```

4.1.6.3 Alarms and Conditions

The steps required to configure OPC UA Alarms and Conditions (A&C) on the OPC UA Server are described in this section. The basic concept is independent of TwinCAT runtime, which means the configuration steps for PLC, C++, TcCOM runtime or only I/O task are the same.

OPC UA Alarms and Conditions (Part 9 of the OPC UA specification) describes a model for monitoring process values and outputting alarms and events when a runtime symbol changes its state.

Requirements

The following requirements apply to the use of OPC UA Alarms and Conditions:

- The runtime symbol to be monitored must be available in the namespace.
- The OPC UA Client must support Alarms and Conditions. In this section UA Expert (from Unified Automation) is used as the reference UA Client.

General Information

Execute the following steps once to release a symbol for Alarms and Conditions:

- Step 1: Activating runtime symbol for data access [▶ 114] (so that the symbol is generally accessible via OPC UA.)
- Step 2: Activating A&C for a symbol [▶ 114]
- Step 3: Transferring your own user data with an event [▶ 115]
- Step 4: Triggering an event using the FireEvent method [▶ 116]
- Step 5: Configuring multilingual alarm texts [▶ 118]
- Step 6: Registering A&C with a reference OPC UA Client [▶ 118]

These steps are explained in more detail below. At the end of this section you will find information about receiving configured alarms via A&C with the UA Expert Reference Client.

Supported alarm types

The implementation of OPC UA Alarms and Conditions currently supports the following alarm types:

- LimitAlarmType: Define different limits for a symbol. If a limit is reached, the UA Server issues an alarm.
- OffNormalAlarmType: Define a value that is "normal". If the current value deviates from the "normal" value, the UA Server issues an alarm.

Step 1: Activating runtime symbol for data access

A variable must be available in the OPC UA Server in order for it to be configured for A&C. To do this in the case of a PLC variable, provide it with an attribute (see Access to PLC runtime [▶ 43]).

Step 2: Activating A&C for a symbol

You can configure a runtime symbol for A&C with the OPC UA Server Configurator. The Configurator has a simple graphical user interface for editing the XML file on which it is based. Depending on the setup version the configurator is available in two variants: standalone [▶ 190] and integrated in Visual Studio [▶ 202].

The following program extract shows an example of this XML file to better understand the general behavior and structure of the A&C implementation.

```
<TcUaAcConfig>
  <ConditionController Name="ConditionController1" >
    <Condition Name="Counter" Severity="200">
    <LimitAlarmType LowLowLimit="-10" LowLimit="0" HighLimit="10" HighHighLimit="20"
MessageNormal="100" MessageLowLow="10" MessageLow="11" MessageHigh="12" MessageHighHigh="13"/>
      <ItemToMonitor SamplingRate="100" NS="urn:[NodeName]:BeckhoffAutomation:Ua:PLC1"
NodeId="s=MAIN.nCounter1" />
    </Condition>
    <Condition Name="Switch" Severity="500">
      <OffNormalAlarmType Normal="0" MessageNormal="100" MessageOffNormal="20" />
      <ItemToMonitor SamplingRate="100" NS="urn:[NodeName]:BeckhoffAutomation:Ua:PLC1"
NodeId="s=MAIN.bSwitch" />
    </Condition>
    <Condition Name="Struct" Severity="300">
      <LimitAlarmType LowLowLimit="-10" LowLimit="0" HighLimit="10" HighHighLimit="20"
MessageNormal="100" MessageLowLow="10" MessageLow="11" MessageHigh="12" MessageHighHigh="13"/>
      <ItemToMonitor SamplingRate="100" NS="urn:[NodeName]:BeckhoffAutomation:Ua:PLC1"
NodeId="s=MAIN.stStruct" />
    </Condition>
  </ConditionController>
  <ConditionController Name="ConditionController2" >
    <Condition Name="Counter2" Severity="200">
      <LimitAlarmType LowLowLimit="-10" LowLimit="0" HighLimit="10" HighHighLimit="20"
MessageNormal="100" MessageLowLow="10" MessageLow="11" MessageHigh="12" MessageHighHigh="13"/>
      <ItemToMonitor SamplingRate="100" NS="urn:[NodeName]:BeckhoffAutomation:Ua:PLC1"
NodeId="s=MAIN.nCounter2" />
    </Condition>
  </ConditionController>
</TcUaAcConfig>
```

A "condition" defines the runtime symbol to be monitored and the alarm limits and texts. Each condition is organized in a so-called "ConditionController", the object that the OPC UA Clients subsequently subscribe to.

When creating a condition, you must specify the NamespaceName (NS) and NodeID for referring to the UA node to be monitored. The Configurator provides a simple browsing mechanism for selecting a node. In XML, the placeholder [NodeName] in NamespaceName can be used to switch the XML file between different hardware systems. NamespaceName always contains the host name of the IPC or Embedded PC on which the OPC UA Server is running. If [NodeName] is selected, this tag will be replaced by the host name of the current IPC or Embedded PC on which the UA Server is running.

SamplingRate determines how often the UA Server should request a value from the node to determine whether one of the alarm limits has been reached.

The alarm texts are identified by an ID. The ID uniquely identifies an alarm text from the resource file (see Step 5: Configuring multilingual alarm texts [▶ 118]).

Step 3: Transferring own user data with an alarm

Alarms can contain fields with their own user data, which complement the data output with the alarm. These user data fields can be created and filled out in the runtime application. To do this, create a STRUCT and name its first element "value". In case of an alarm, all the following elements are then sent in an additional user data field.

Sample PLC application:

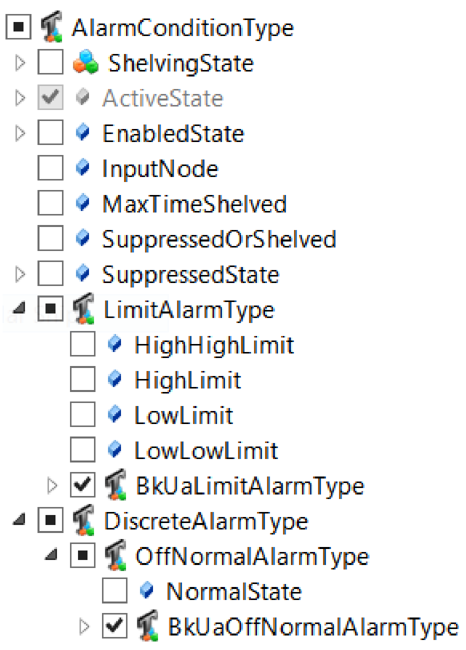
```
TYPE ST_CustomStruct :
STRUCT
  value : INT;
  data  : ST_SomeStruct;
END_STRUCT
END_TYPE

TYPE ST_SomeStruct :
STRUCT
  Data1 : INT;
  Data2 : REAL;
  Data3 : LREAL;
END_STRUCT
END_TYPE
```

The instance of ST_CustomStruct is then enabled for data access via the regular mechanism, e.g. in TwinCAT 3: To do this the STRUCT must be activated as a StructuredType [▶ 88].

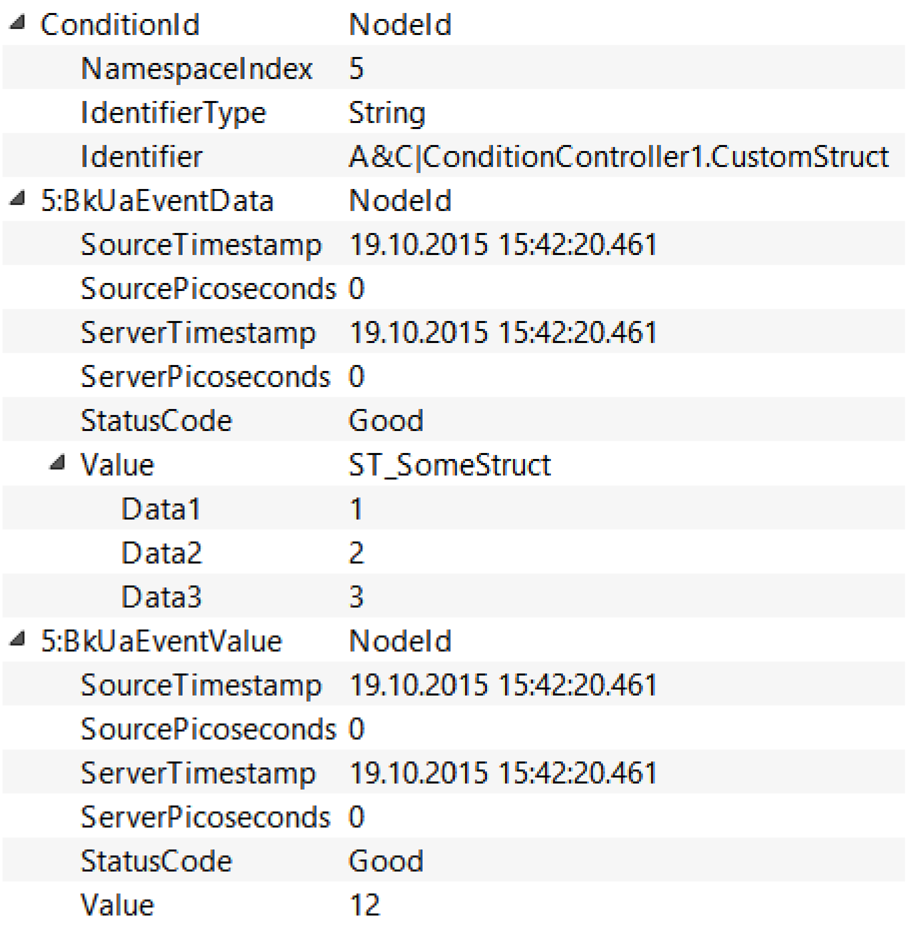
```
PROGRAM MAIN
VAR
  {attribute 'OPC.UA.DA' := '1'}
  {attribute 'OPC.UA.DA.StructuredType' := '1'}
  stCustomStruct : ST_CustomStruct;
END_VAR
```

When logging on to a ConditionController, the OPC UA Clients must subscribe to special AlarmConditionTypes, i.e. "BkUaLimitAlarmType" and "BkUaOffNormalAlarmType", so that they can receive the special user data fields when an alarm is received.



- ▣ AlarmConditionType
  - ☐ ShelvingState
  - ✓ ActiveState
  - ☐ EnabledState
  - ☐ InputNode
  - ☐ MaxTimeShelved
  - ☐ SuppressedOrShelved
  - ☐ SuppressedState
  - ▣ LimitAlarmType
    - ☐ HighHighLimit
    - ☐ HighLimit
    - ☐ LowLimit
    - ☐ LowLowLimit
    - ✓ BkUaLimitAlarmType
  - ▣ DiscreteAlarmType
    - ▣ OffNormalAlarmType
      - ☐ NormalState
      - ✓ BkUaOffNormalAlarmType

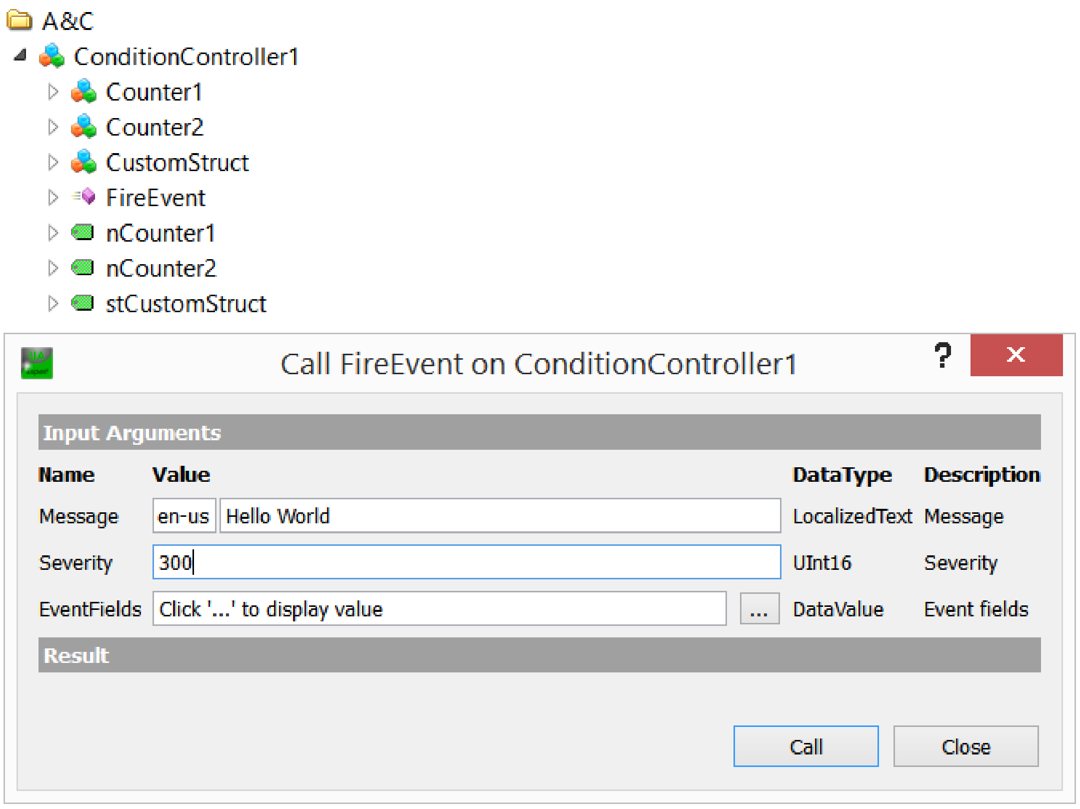
The OPC UA Client then receives the user data in the fields BkUaEventData and BkUaEventValue of the incoming alarm. In the above sample, these are the value of the PLC variables and the user data represented by the PLC structure ST_SomeStruct.



ConditionId	NodeId
NamespaceIndex	5
IdentifierType	String
Identifier	A&C|ConditionController1.CustomStruct
5:BkUaEventData	NodeId
SourceTimestamp	19.10.2015 15:42:20.461
SourcePicoseconds	0
ServerTimestamp	19.10.2015 15:42:20.461
ServerPicoseconds	0
StatusCode	Good
Value	ST_SomeStruct
Data1	1
Data2	2
Data3	3
5:BkUaEventValue	NodeId
SourceTimestamp	19.10.2015 15:42:20.461
SourcePicoseconds	0
ServerTimestamp	19.10.2015 15:42:20.461
ServerPicoseconds	0
StatusCode	Good
Value	12

Step 4: Triggering an event using the FireEvent method

Each ConditionController includes a FireEvent method with which the OPC UA Clients can trigger a general event with user-defined EventFields.



A&C
- ConditionController1
  - Counter1
  - Counter2
  - CustomStruct
  - FireEvent
  - nCounter1
  - nCounter2
  - stCustomStruct

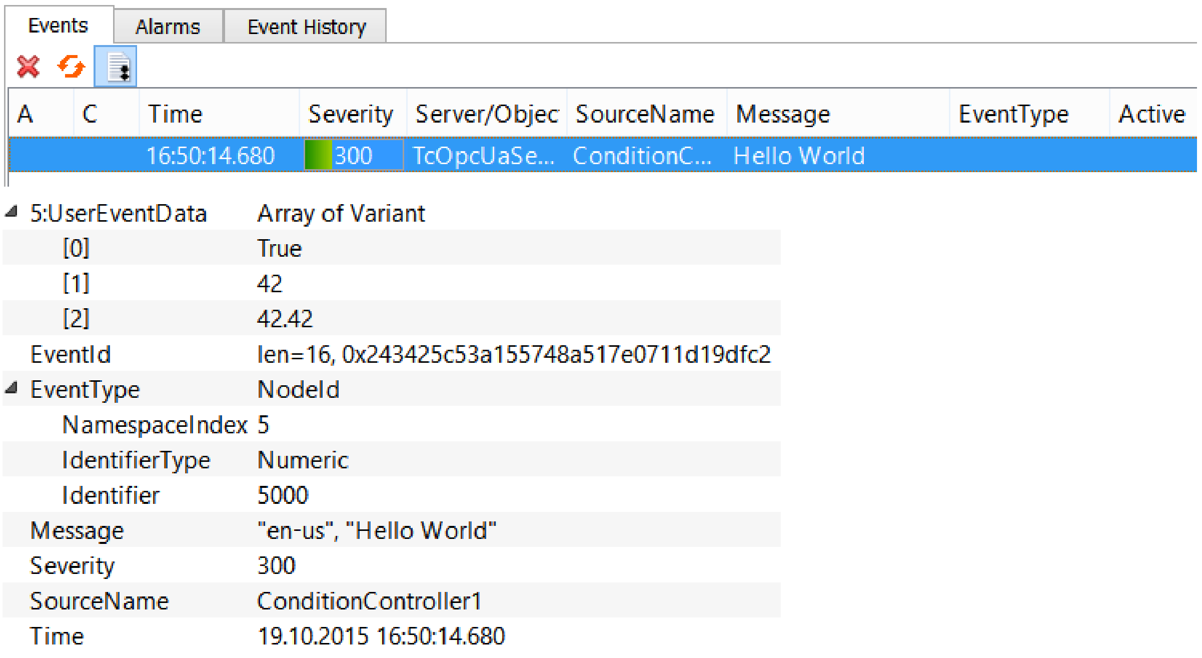
Call FireEvent on ConditionController1

Input Arguments

Name	Value	DataType	Description
Message	en-us Hello World	LocalizedText	Message
Severity	300	UInt16	Severity
EventFields	Click '...' to display value	DataValue	Event fields

Result

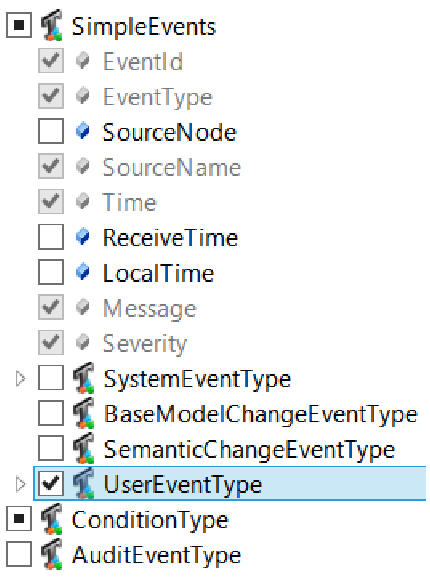
If the method is executed, an event is output on the OPC UA Server. Other OPC UA Clients can receive these events when they subscribe to the corresponding ConditionController.



A	C	Time	Severity	Server/Objec	SourceName	Message	EventType	Active
		16:50:14.680	300	TcOpcUaSe...	ConditionC...	Hello World		

5:UserEventData	Array of Variant
[0]	True
[1]	42
[2]	42.42
EventId	len=16, 0x243425c53a155748a517e0711d19dfc2
EventType	NodeId
NamespaceIndex	5
IdentifierType	Numeric
Identifier	5000
Message	"en-us", "Hello World"
Severity	300
SourceName	ConditionController1
Time	19.10.2015 16:50:14.680

The user-defined EventFields are appended to the event as "UserEventData". This data can be received by OPC UA Clients that are logged on to the SimpleEventType "UserEventType".



Step 5: Configuring multilingual alarm texts

The A&C implementation in the OPC UA Server supports the use of multilingual alarm texts. The alarm text that is used depends on the language with which the UA Client connects to the Server.

Alarm texts are configured in XML files. There is a separate file for each language. These files are located in the res (Resource) folder on the OPC UA Server. With the configurator you can simply add or remove alarm texts without having to directly edit the XML files. Each alarm text has its own ID, which occurs only once in the file.

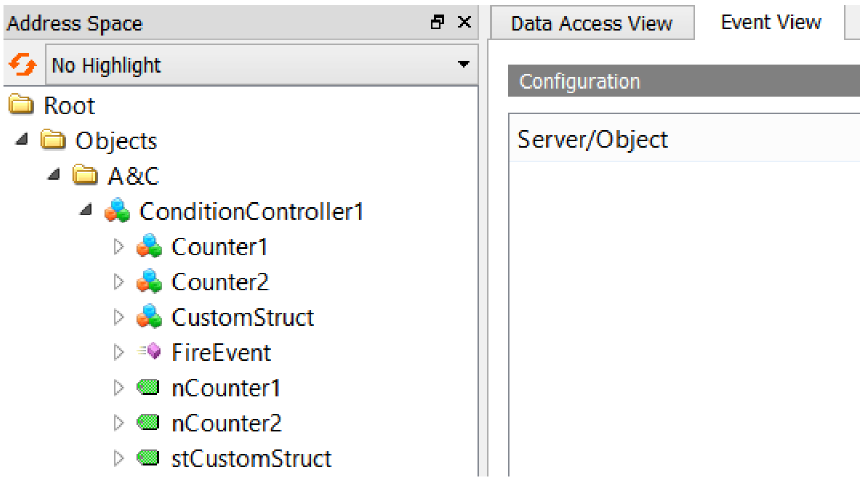
Sample:

```
<TcOpcUaSvrRes Lang="en">
  <Text ID="0">Text not available</Text>
  <Text ID="1">Some alarm text</Text>
  <Text ID="2">Value is High range</Text>
  <Text ID="3">Value is HighHigh range</Text>
  <Text ID="4">Value is OffNormal</Text>
  ...
</TcOpcUaSvrRes>
```

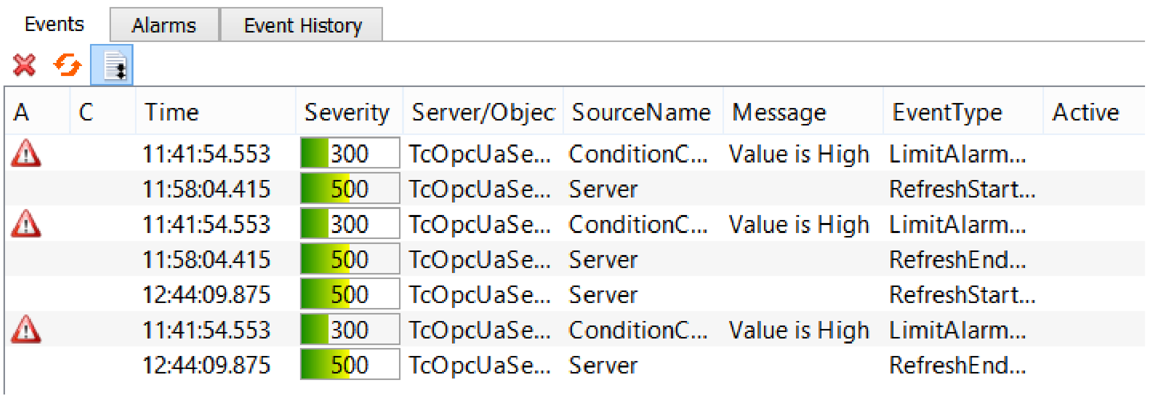
Step 6: Registering A&C with a reference OPC UA Client

An OPC UA Client must log on to a ConditionController so that it can receive events for conditions that are configured for this particular ConditionController. The UA Expert provides functions for subscribing to and receiving UA events.

After you have started the UA Expert and established a connection with the OPC UA Server, add a new document view, Event View, to your working area.



You can then subscribe to a ConditionController by dragging the corresponding object in Event View. The events for this ConditionController are displayed in the Event Window.



A	C	Time	Severity	Server/Objec	SourceName	Message	EventType	Active
⚠		11:41:54.553	300	TcOpcUaSe...	ConditionC...	Value is High	LimitAlarm...	
		11:58:04.415	500	TcOpcUaSe...	Server		RefreshStart...	
⚠		11:41:54.553	300	TcOpcUaSe...	ConditionC...	Value is High	LimitAlarm...	
		11:58:04.415	500	TcOpcUaSe...	Server		RefreshEnd...	
		12:44:09.875	500	TcOpcUaSe...	Server		RefreshStart...	
⚠		11:41:54.553	300	TcOpcUaSe...	ConditionC...	Value is High	LimitAlarm...	
		12:44:09.875	500	TcOpcUaSe...	Server		RefreshEnd...	

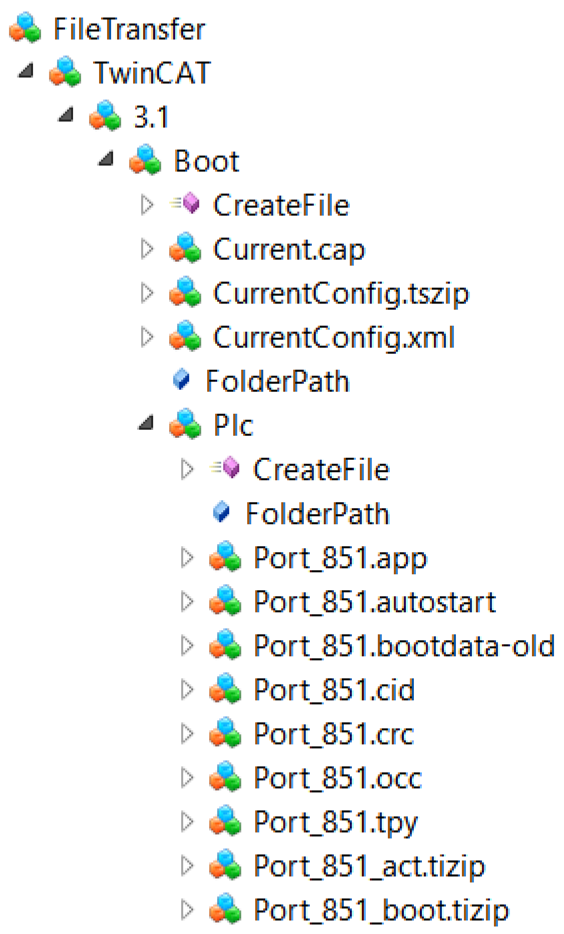
It may be necessary to subscribe to special alarm and/or event types in order to receive all fields of an incoming event or alarm.

4.1.7 File Transfer

4.1.7.1 Access to files and folders via OPC UA

From OPC UA specification version 1.02, OPC UA contains a specialized ObjectType for file transfer, which is described in Appendix C of the specification. This special ObjectType called "FileType" describes the information model for the data transfer. Files can be modeled as simple variables in OPC UA with ByteStrings. FileType is a file with methods for accessing the file. The OPC UA specification provides further information about FileType and the structure and handling of the underlying methods and properties for accessing a file in the OPC UA namespace.

Beckhoff has implemented a generic way to load files and folders from a local hard disk into the OPC UA namespace. Each file is represented by a FileType and allows read and write operations for this file. In addition, each folder contains a CreateFile() method to create new files on the hard disk and a separate FolderPath to specify the actual path to the folder on the OPC UA Server.


FileTransfer
TwinCAT
3.1
Boot
CreateFile
Current.cap
CurrentConfig.tszip
CurrentConfig.xml
FolderPath
Plc
CreateFile
FolderPath
Port_851.app
Port_851.autostart
Port_851.bootdata-old
Port_851.cid
Port_851.crc
Port_851.occ
Port_851.tpy
Port_851_act.tizip
Port_851_boot.tizip

FileTransfer in the OPC UA Server Device Manager

Only the OPC UA Server of the Beckhoff Device Manager (IPC diagnostics) has this function. The TwinCAT OPC UA Server also provides some parts of this file transfer. However, the general function that enables disclosure of all files and folders is only available in the OPC UA Server, which is part of the device manager that is automatically available on every Beckhoff Industrial PC or Embedded PC. See the Device manager documentation for more information.

Configuration

FileType objects are created in a separate namespace called "FileTransfer". An XML file (*files.xml*) is used to configure this namespace and to select the files and folders available via OPC UA. The file must be located in the same directory as the executable file of the OPC UA Server. The system must be restarted in order to activate the configuration. The XML file contains information about the folder path and a search mask that defines which files are published in the OPC UA namespace:

```
<Files>
  <FolderObject DisplayName="TwinCAT">
    <FolderObject DisplayName="3.1">
      <FolderObject DisplayName="Boot" Path="c:/TwinCAT/3.1/Boot" Search="*.*" >
        <FolderObject DisplayName="Plc" Path="c:/TwinCAT/3.1/Boot/Plc" Search="*.*" ></FolderObject>
        <FolderObject DisplayName="Tmi" Path="c:/TwinCAT/3.1/Boot/Tmi" Search="*.*" ></FolderObject>
      </FolderObject>
    </FolderObject>
  </FolderObject>
</Files>
```

Sample: Reading a file with UA Expert

General file handling is described in Appendix C of the OPC UA specification. Reading a file via UA can be divided into the following steps:

- Calling the Open method of a file. This method returns a file handle that must be saved for later access. The mode defines whether the file is read or written to (see File modes [▶ 121]).

- Determining the file size with the property "Size". In this way, the entire file can be read when the Read method is called.
- Calling the Read method. Inserting the file handle and file size as inputs. Selecting the destination folder in which the file contents are to be saved AFTER the method call.
- Calling the Close method to enable the file handle.

File modes

The following table shows all available file modes.

Field	Bit	Description
Read	1	The file is opened for reading. If this bit is not set, Read cannot be executed.
Write	4	The file is opened for writing. If this bit is not set, Write cannot be executed.
EraseExisting	6	The existing file contents are deleted, and an empty file is made available.
Append	10	The file is opened and positioned at the end, otherwise it is moved to the beginning. This position can be changed with SetPosition.

4.1.8 Security

4.1.8.1 Overview

Security was a central requirement in the development of OPC UA. It is addressed in various areas:

- Encryption
- Integrity
- Authentication

The confidentiality of the exchanged information is secured by the encryption of the exchanged messages. Modern cryptographic algorithms are used for this. In order to be able to cope with future security requirements as well, even stronger and more modern algorithms can subsequently be added to an application without changing the protocol.

Different security levels can be selected according to the requirements of the respective application. In some areas it is sufficient to sign the messages in order to prevent changes being made by third parties, while additional encryption of the messages is necessary in other cases where the data must also not be read by third parties.

TF6100 OPC UA offers the following security levels (security endpoints) that clients can connect to:

- None: No security
- Sign: signed messages
- Sign & Encrypt: signed and encrypted messages

The signing of messages prevents a third party from changing the contents of a message. This prevents, for example, a write statement to open a switch being falsified by a third party and the switch being closed instead.

OPC UA applications identify themselves via so-called software and application instance certificates. With the aid of software certificates it is possible to grant certain client applications extended access to the information on an OPC UA Server, for example for the engineering of an OPC UA Server. Application instance certificates can be used to ensure that an OPC UA Server communicates only with preconfigured clients. A client can ensure by means of the server’s application instance certificate that it is speaking to the correct server (similar to the certificates of a Web browser). The taking into account of these certificates is optional, i.e. an OPC UA server can also grant the same access to each client, depending on the user rights.

The TwinCAT OPC UA Client and TwinCAT OPC UA Server generate a self-signed certificate during the first startup process. This certificate consists of a private key and a public key. The private key is saved in the *\PKI\CA\private* directory and the corresponding public key in the *\PKI\CA\certs* directory. Any OPC UA Client wishing to establish a secure connection with the Server via one of the security endpoints (Sign, Sign&Encrypt) must know the public key of the OPC UA Server. Conversely the OPC UA Server must know the Client's public key. This so-called key exchange is described in the following sections:

- Authentication [▶ 122]
- Certificate exchange [▶ 122]
- Access rights [▶ 123]

4.1.8.2 Authentication

In addition to certificate exchange, the TwinCAT OPC UA Server offers the option of authentication via user name/password. An OPC UA Client must then use a valid user name/password combination for successful connection to the server.

The TwinCAT OPC UA Server uses mechanisms of the operating system for validating user names and passwords. If the computer on which the TwinCAT OPC UA Server is running is a member of a Windows domain (e.g. Active Directory), a request is made to the corresponding domain controller. If the computer is a single device, the local user database of the operating system is checked.

The configuration of the TwinCAT OPC UA Server allows two settings:

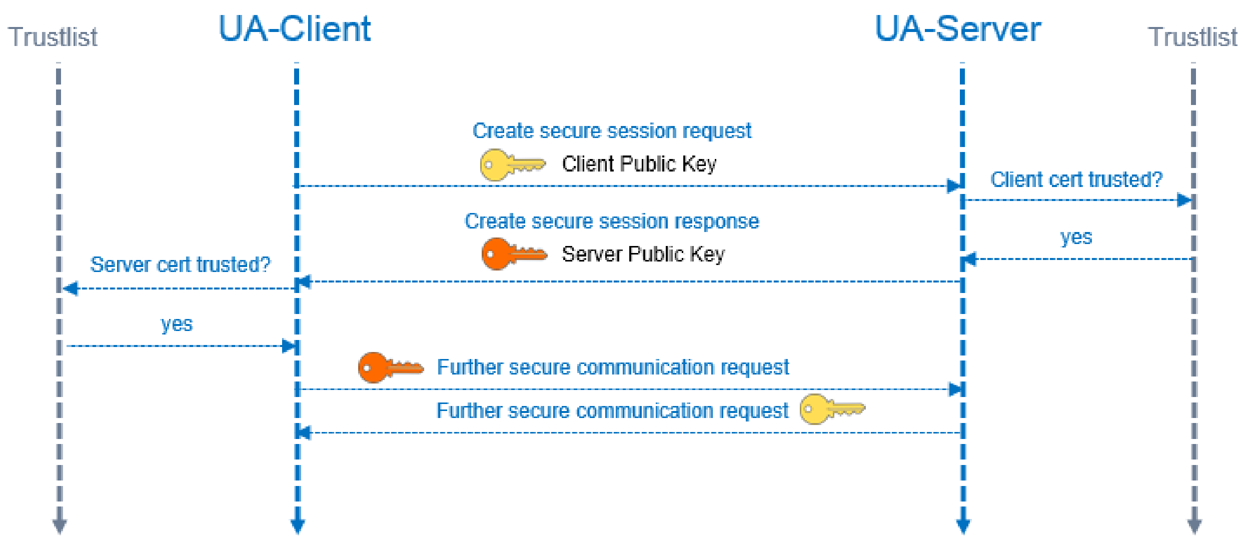
- Enable/Disable anonymous access (without user authentication)
- Enable/Disable user name/password validation

You can make both settings in the OPC UA Server Configurator.

4.1.8.3 Certificate exchange

The communication between an OPC UA Client and an OPC UA Server can optionally be secured by communication with a secure endpoint. By default, both the TwinCAT OPC UA Server and the TwinCAT OPC UA Client generate a machine-specific, self-signed certificate for authentication the first time they are started.

If you want to use such an encrypted communication connection in your environment, you need to establish a trust relationship between OPC UA Server and OPC UA Client.


Trustlist
UA-Client
UA-Server
Trustlist
Create secure session request
Client Public Key
Client cert trusted?
Create secure session response
yes
Server Public Key
Server cert trusted?
yes
Further secure communication request
Further secure communication request

Trust settings on the server

If you want to authenticate one or more OPC UA Clients to the OPC UA Server via certificates, the OPC UA Server must trust the public keys of the clients. This can be done via the file system, for example. The server manages the trust settings for certificates via the file system.

- Trusted certificates: %InstallDir%\Server\PKI\CA\trusted\certs
- Rejected certificates: %InstallDir%\Server\PKI\CA\rejected\certs

By moving client certificates between these directories, the trust settings can be adjusted accordingly.

Reading a client certificate

A simple way of accessing the client certificate is described below. To do this, you establish a connection to the UA Server using a secure endpoint (for example, Basic128Rsa15/Sign&Encrypt) without first copying the client certificate to the UA Server. This connection is naturally rejected by the UA Server, since it does not trust the UA Client at this point in time. In this case, the TwinCAT OPC UA Client would return the error 0xE4DD0102, for example. However, after rejecting the connection request, the UA Server stores a copy of the client certificate in the above-mentioned "Rejected" directory. The name of the certificate corresponds to the "Thumbprint" of the certificate and can thus be clearly assigned to the client certificate. You can now move this file directly to the above-mentioned "Trusted" directory so that the UA Server can trust the UA Client and accept the connection for all other connection requests.

Announcing OPC UA Servers to the OPC UA Client

Depending on the OPC UA Client employed, different steps may need to be taken so that the OPC UA Client trusts the OPC UA Server. Typically, for client applications with a graphical user interface, a warning message is displayed the first time you connect to the server, whereby the server certificate can then be classified as trustworthy.

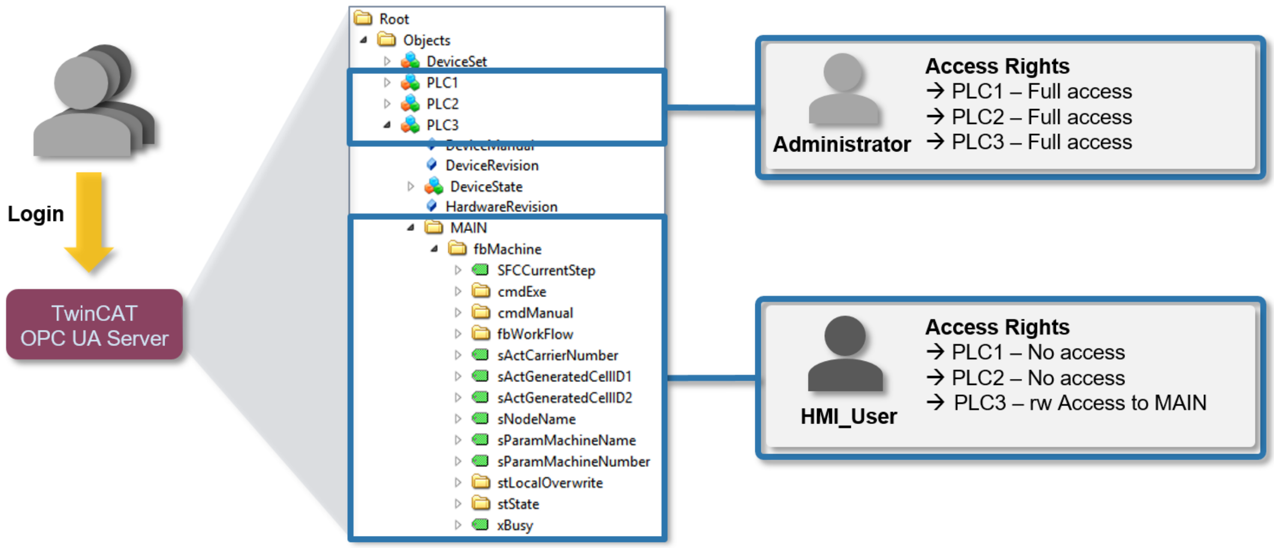
The following instruction is therefore only valid for the TwinCAT OPC UA Client.

The public key of the OPC UA Server is located in the following directory as a DER file: *%InstallDir%\Server \PKI\CA\own\certs*

In the case of the TwinCAT OPC UA Client, copy the file into the corresponding "Trusted" directory: *%InstallDir%\Client\PKI\CA\certs*

4.1.8.4 Access rights

The TwinCAT OPC UA Server enables the configuration of permissions at namespace and node level. This allows you to fine-granulate the access to ADS devices (for example, to different PLC runtimes) as well as variables. These security settings are available for all ADS devices that can be displayed in the server namespace.



See also:

Configuration > Setup version 4.x.x. > Configuration of the security settings in the configurator [▶ 208].

Requirements

Products	Setup versions	Target platform
TF6100	4.x.x	IPC or CX (x86, x64, ARM)

4.1.9 Symbol Caching

From version 3.2 (file version) the server application creates a binary cache file when a TwinCAT symbol file (TMC file) is loaded for the first time.

Depending on the size of the symbol file and the underlying hardware architecture, the parsing of a symbol file can take a very long time, leading to a lengthy start-up time of the server process. Each symbol file must be parsed at least twice in order to compile information on the type system and all instances. In order to ensure that the server will start faster next time, a cache file is created containing information about all the TwinCAT symbols that are to be exported to the namespace of the server.

The cache file is created in a compressed binary format that can be loaded and edited more efficiently than an XML-based symbol file and is saved in the /symbolfiles/ subdirectory of the server. As long as the source symbol file does not change, the cache file will be used by the server in place of the symbol file.

If the source symbol file changes (on account of the checksum inside the file), a new cache file is created.

The standard start-up process of the server consists of the following steps:

1. The server application loads the configuration file *TcUaDaConfig.xml*, which contains a list of all the namespaces (TwinCAT runtimes) that are to be created.
2. The server application loads each TMC symbol file from the configuration and checks whether a binary cache file has already been created for this symbol file.
3. If no binary cache file has been created yet, the server parses the symbol file and creates a corresponding cache file.
4. If a binary cache file has already been created, the server validates the checksum of the symbol file and compares it with the checksum of the cache file to determine whether changes have been made in comparison with the symbol file (e.g. due to a new or updated PLC program).
5. If the checksums differ, the symbol file is parsed again and a new cache file is created.
6. If the checksums are the same, the binary cache file is loaded instead of the symbol file.

This work sequence is normally completely transparent, as it is executed automatically in the background as the default behavior of the OPC UA Server. However, you can change the behavior of the Server in step no. 5 via the following configuration option in the file *TcUaDaConfig.xml*:

```
<CacheFileMode>...</CacheFileMode>
```

This configuration option can have the following values:

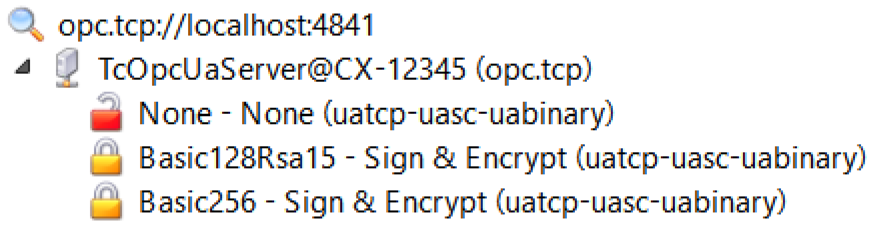
no_cache	Deactivates the cache file
dont_update	Creates a cache file, but does not update the file if the contents of the symbol file change
always_create	Creates a new cache file at each start

The "dont_update" option can be useful if, for example, all OPC UA symbols are saved in a PLC library. Even though the PLC application that uses the library can change (which leads to a changed symbol file), this does not negatively affect the OPC UA namespace. A new cache file therefore does not necessarily have to be created.

4.1.10 Miscellaneous

4.1.10.1 Endpoints and default port

By default, TwinCAT OPC UA Server provides the following endpoints and standard port for OPC UA Clients to connect to:


opc.tcp://localhost:4841
TcOpcUaServer@CX-12345 (opc.tcp)
None - None (uatcp-uasc-uabinary)
Basic128Rsa15 - Sign & Encrypt (uatcp-uasc-uabinary)
Basic256 - Sign & Encrypt (uatcp-uasc-uabinary)

Standard port

Note that the standard port 4840 may be used by other OPC UA Servers, such as the Local Discovery Server (LDS) from the OPC Foundation, which is used by some vendors with OPC UA software packages.

Endpoint	Description
None - None	No security
Basic128Rsa15 - Sign & Encrypt	Signature and encryption of messages for medium security requirements. Requires certificate exchange and trust lists management. See also: Server > Security [▶ 121]
Basic256 - Sign & Encrypt	Signature and encryption of messages for medium to high security requirements. Requires more computing time (higher CPU requirements). Requires certificate exchange and trust lists management. See also: Server > Security [▶ 121]

4.1.10.2 Generating NamespaceURI

The NamespaceURI is important for the unique identification of a node in the UA namespace. Each NodeID consists of a NamespaceIndex, an IdentifierType and a identifier. Since the NamespaceIndex can be dynamically generated by an OPC UA Server, a UA Client should always use the static NamespaceURI to resolve it into its actual NamespaceIndex. All NamespaceURIs and their corresponding NamespaceIndex items are listed in the so-called NamespaceArray of an OPC UA Server.

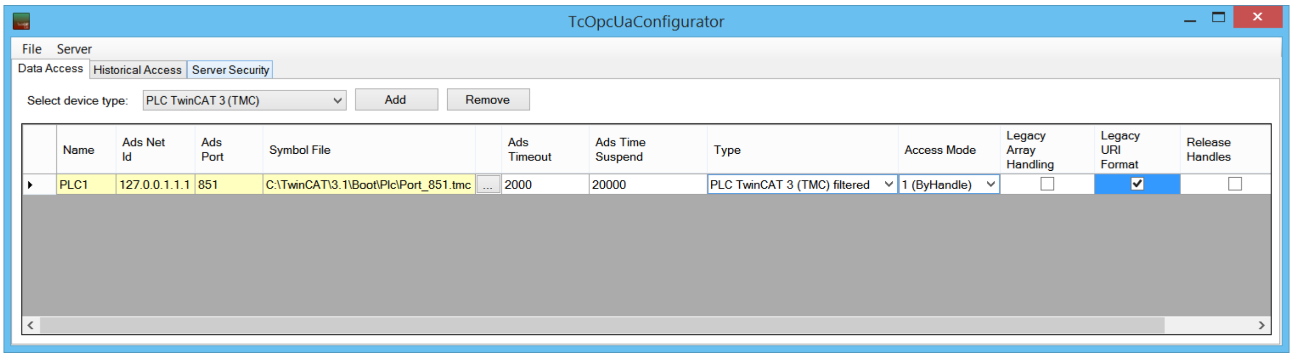
For more information about the NamespaceArray and its contents see How to determine communication parameters [▶ 150].

For each TwinCAT runtime, the NamespaceURI is generated according to the following naming scheme:

urn:[NodeName]:BeckhoffAutomation:Ua:[Name]

- [NodeName] stands for the host name of the computer on which the TwinCAT OPC UA Server is running
- [Name] is a string that can be defined via the configuration file of the server or via the TwinCAT OPC UA Configurator.

Note that older versions of the OPC UA Server do not use the URI-based format to generate the NamespaceURI, but merely the <Name> attribute, as was defined in the configuration file of the server. Because this may be an insurmountable change when upgrading from older versions, the OPC UA Server offers a so-called Legacy URI format change, which you can activate for every TwinCAT runtime in the configuration file of the Server or via the OPC UA Configurator.



Select device type: PLC TwinCAT 3 (TMC)

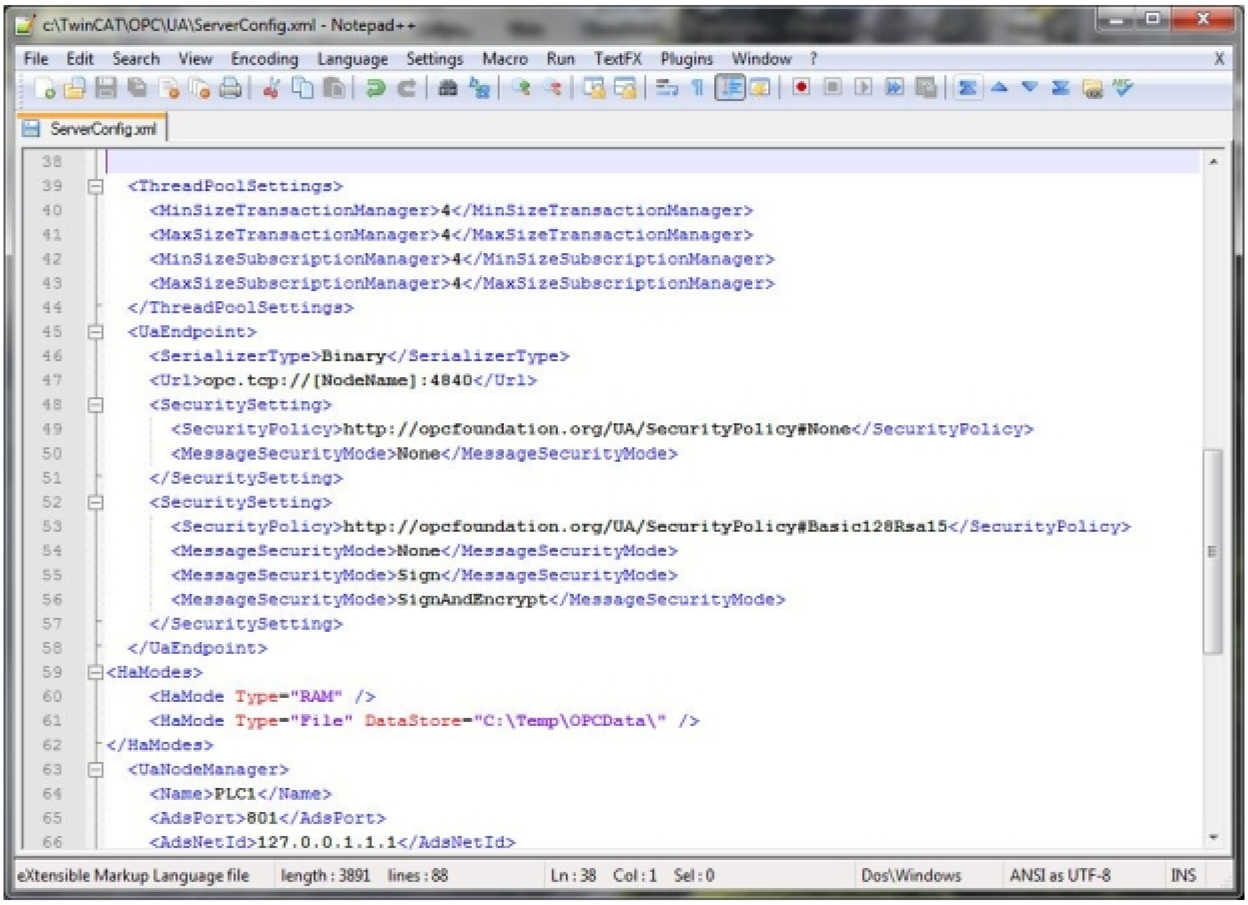
	Name	Ads Net Id	Ads Port	Symbol File		Ads Timeout	Ads Time Suspend	Type	Access Mode	Legacy Array Handling	Legacy URI Format	Release Handles
▸	PLC1	127.0.0.1.1.1	851	C:\TwinCAT\3.1\Boot\Plc\Port_851.tmc	...	2000	20000	PLC TwinCAT 3 (TMC) filtered	1 (ByHandle)	☐	✓	☐

```
<UaNodeManager>
  <Name>PLC1</Name>
  <AdsPort>851</AdsPort>
  <AdsNetId>127.0.0.1.1.1</AdsNetId>
  <AdsTimeout>2000</AdsTimeout>
  <AdsTimeSuspend>20000</AdsTimeSuspend>
  <AutoCfg>4041</AutoCfg>
  <IoMode>1</IoMode>
  <AutoCfgSymFile>C:\TwinCAT\3.1\Boot\Plc\Port_851.tmc</AutoCfgSymFile>
  <Disabled>0</Disabled>
  <ArraySubItemLegacySupport>0</ArraySubItemLegacySupport>
  <MaxGetHandle>100</MaxGetHandle>
  <ReleaseAdsVarHandles>0</ReleaseAdsVarHandles>
  <LegacyUriFormat>1</LegacyUriFormat>
</UaNodeManager>
```

If the legacy URI format change is set to "0", the Namespace Name is generated according to the scheme described above. If it is set to "1", the Namespace Name is then the same as the XML tag <Name> and is thus generated in the same way as in older versions of the OPC UA Server.

4.1.10.3 Editing the configuration file

You can make all settings for the OPC UA Server with the help of the OPC UA Configurator, which is delivered via the setup from UA Server version 1.6.80 (see Configuration > Setup version 3.x.x > Overview [▶ 187]). The Configurator facilitates the configuration of the UA Server via its configuration file *ServerConfig.xml*. The configuration file contains a hierarchical, XML-based structure of all configurable parameters of the UA Server. The following figure illustrates the structure of the file by way of example:


c:\TwinCAT\OPC\UA\ServerConfig.xml - Notepad++
File Edit Search View Encoding Language Settings Macro Run TextFX Plugins Window ?
ServerConfig.xml
eXtensible Markup Language file length : 3891 lines : 88 Ln : 38 Col : 1 Sel : 0 Dos\Windows ANSI as UTF-8 INS

```

  <ThreadPoolSettings>
    <MinSizeTransactionManager>4</MinSizeTransactionManager>
    <MaxSizeTransactionManager>4</MaxSizeTransactionManager>
    <MinSizeSubscriptionManager>4</MinSizeSubscriptionManager>
    <MaxSizeSubscriptionManager>4</MaxSizeSubscriptionManager>
  </ThreadPoolSettings>
  <UaEndpoint>
    <SerializerType>Binary</SerializerType>
    <Url>opc.tcp://[NodeName]:4840</Url>
    <SecuritySetting>
      <SecurityPolicy>http://opcfoundation.org/UA/SecurityPolicy#None</SecurityPolicy>
      <MessageSecurityMode>None</MessageSecurityMode>
    </SecuritySetting>
    <SecuritySetting>
      <SecurityPolicy>http://opcfoundation.org/UA/SecurityPolicy#Basic128Rsa15</SecurityPolicy>
      <MessageSecurityMode>None</MessageSecurityMode>
      <MessageSecurityMode>Sign</MessageSecurityMode>
      <MessageSecurityMode>SignAndEncrypt</MessageSecurityMode>
    </SecuritySetting>
  </UaEndpoint>
<HaModes>
    <HaMode Type="RAM" />
    <HaMode Type="File" DataStore="C:\Temp\OPCData\" />
</HaModes>
  <UaNodeManager>
    <Name>PLC1</Name>
    <AdsPort>801</AdsPort>
    <AdsNetId>127.0.0.1.1.1</AdsNetId>
```

In general, the use of the OPC UA Configurator is recommended, as this can help to avoid errors during configuration.

The configuration file *ServerConfig.xml* is always located in the same directory as *TcOpcUaServer.exe*, for example:

- Windows XP / 7 with TwinCAT 2: %TwinCATDIR%\OPC\UA\
- Windows XP / 7 with TwinCAT 3: %TwinCATDIR%\Functions\TF6100-OPC-UA\Win32\Server\
- Windows CE TwinCAT 2: HardDisk\System
- Windows CE TwinCAT 3: HardDisk\TwinCAT\Functions\TF6100-OPC-UA\Server\

4.1.10.4 Configuring several TwinCAT runtimes

You can optionally add further TwinCAT ADS devices to the UA namespace. Through the configuration of the namespace the OPC UA Server knows which TwinCAT ADS device (e.g. PLC, IO task, C++ runtime, etc.) is to be made available via the OPC UA Server and shows its symbol information in the UA namespace accordingly. This configuration only needs to be done once.

Adding/removing further ADS devices to/from the UA namespace

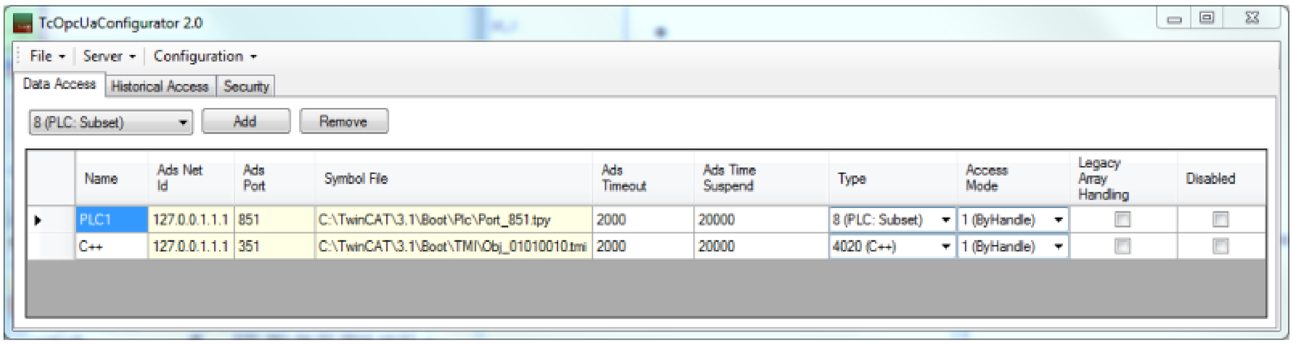
The TwinCAT OPC UA Server is configured by default such that it connects to the first PLC runtime of the local system. In order to make further ADS devices or further runtimes available via the UA namespace, carry out the following steps:

- If the device to be added is a local device:
    - Add the remote device for configuring the OPC UA Server, e.g. via the OPC UA Configurator or by directly editing the *ServerConfig.xml* file
- If the device to be added is a remote device:
    - Manufacture ADS routes between the computer on which the OPC UA Server is running and the remote system

- Add the remote device to the configuration of the OPC UA Server, e.g. via the OPC UA Configurator or by directly editing *ServerConfig.xml* file

Using the OPC UA Configurator

Use the OPC UA Configurator to add further ADS devices to the UA namespace.



Name	Ads Net Id	Ads Port	Symbol File	Ads Timeout	Ads Time Suspend	Type	Access Mode	Legacy Array Handling	Disabled
PLC1	127.0.0.1.1.1	851	C:\TwinCAT\3.1\Boot\Plc\Port_851.tpy	2000	20000	8 (PLC: Subset)	1 (ByHandle)	☐	☐
C++	127.0.0.1.1.1	351	C:\TwinCAT\3.1\Boot\TMI\Obj_01010010.tmi	2000	20000	4020 (C++)	1 (ByHandle)	☐	☐

In the **Data Access** area of the OPC UA Configurator you can manage the ADS devices integrated in the UA namespace, add new devices or remove existing devices.

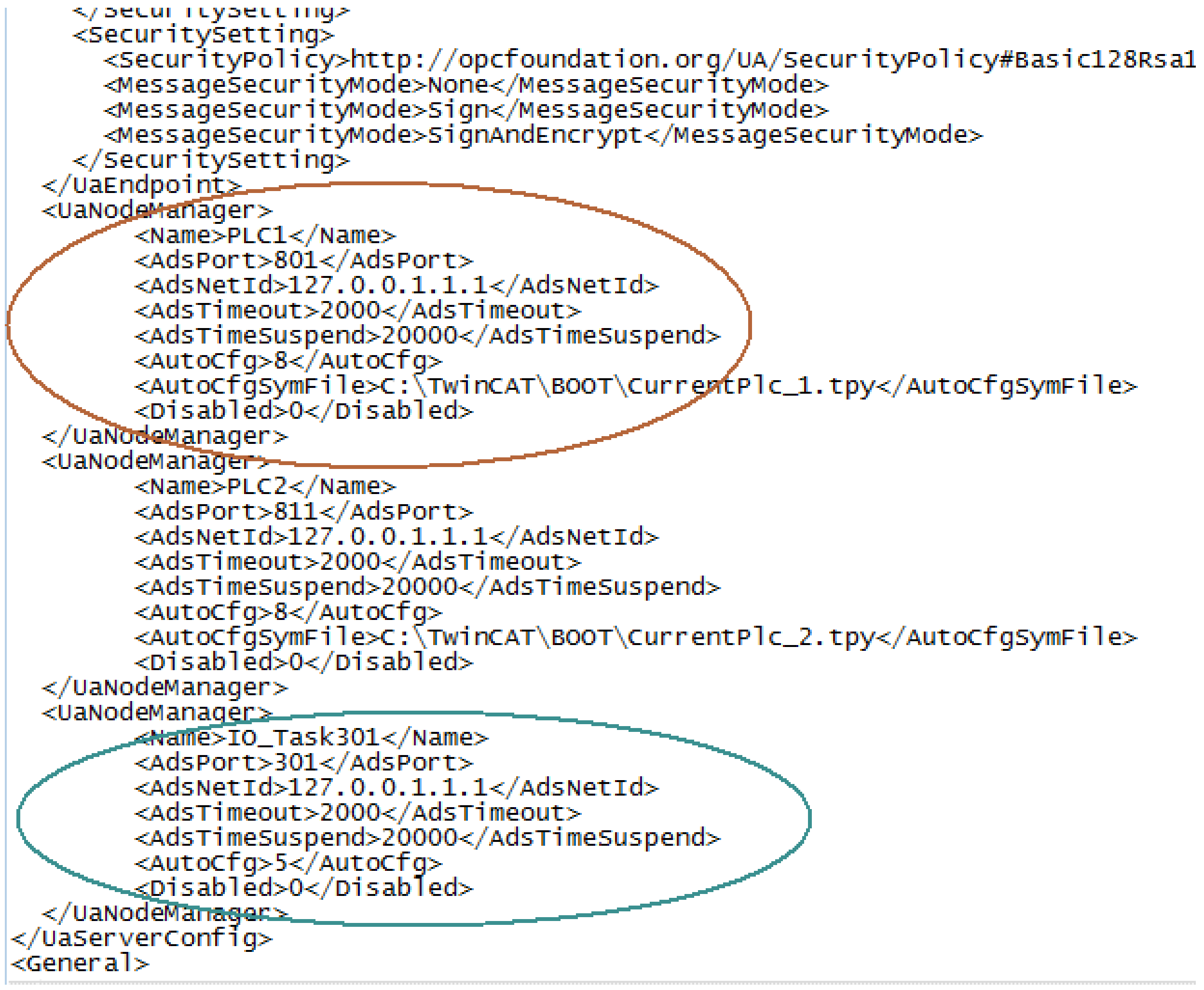
To add a device, first select the device type from the associated drop-down list (e.g."8 (PLC): Subset)"), then click **Add**. An additional entry is added to the list of the ADS devices. Make the required settings. If you want to enable the configuration immediately, click **Activate** on the **Configuration** menu.

Depending on the device to be added, you have to make different settings for the individual parameters (see Configuration > Setup version 3.x.x > Data Access [▶ 188]).

Note that you must restart the OPC UA Server for the changes to take effect. You can also do this via the Configurator by selecting **Restart** from the **Server** menu. Alternatively you can also restart TwinCAT.

[Optional] Editing ServerConfig.xml

Alternatively, you can edit the *ServerConfig.xml* file directly to add additional ADS devices to the UA namespace. In this configuration file there is a section, UaNodeManager, which defines the access to an ADS device in each case, e.g.:



```
      </SecuritySetting>
      <SecuritySetting>
        <SecurityPolicy>http://opcfoundation.org/UA/SecurityPolicy#Basic128Rsa1
        <MessageSecurityMode>None</MessageSecurityMode>
        <MessageSecurityMode>Sign</MessageSecurityMode>
        <MessageSecurityMode>SignAndEncrypt</MessageSecurityMode>
      </SecuritySetting>
    </UaEndpoint>
    <UaNodeManager>
          <Name>PLC1</Name>
          <AdsPort>801</AdsPort>
          <AdsNetId>127.0.0.1.1.1</AdsNetId>
          <AdsTimeout>2000</AdsTimeout>
          <AdsTimeSuspend>20000</AdsTimeSuspend>
          <AutoCfg>8</AutoCfg>
          <AutoCfgSymFile>C:\TwinCAT\BOOT\CurrentPlc_1.tpy</AutoCfgSymFile>
          <Disabled>0</Disabled>
    </UaNodeManager>
    <UaNodeManager>
          <Name>PLC2</Name>
          <AdsPort>811</AdsPort>
          <AdsNetId>127.0.0.1.1.1</AdsNetId>
          <AdsTimeout>2000</AdsTimeout>
          <AdsTimeSuspend>20000</AdsTimeSuspend>
          <AutoCfg>8</AutoCfg>
          <AutoCfgSymFile>C:\TwinCAT\BOOT\CurrentPlc_2.tpy</AutoCfgSymFile>
          <Disabled>0</Disabled>
    </UaNodeManager>
    <UaNodeManager>
          <Name>IO_Task301</Name>
          <AdsPort>301</AdsPort>
          <AdsNetId>127.0.0.1.1.1</AdsNetId>
          <AdsTimeout>2000</AdsTimeout>
          <AdsTimeSuspend>20000</AdsTimeSuspend>
          <AutoCfg>5</AutoCfg>
          <Disabled>0</Disabled>
    </UaNodeManager>
  </UaServerConfig>
  <General>
```

Depending on the device to be added, you have to make different settings for the individual parameters (see Configuration > Setup version 3.x.x > Data Access [▶ 188]).

Note that you must restart the OPC UA Server for the changes to take effect. You can also do this via the Configurator by selecting **Restart** from the **Server** menu. Alternatively you can also restart TwinCAT.

4.1.10.5 Configuring several Server instances

It is possible to install several TwinCAT OPC UA Servers on one system. The configuration steps necessary for this are described in this section. The configuration is identical under Windows CE and Windows XP-based operating systems and consists of the following steps:

- Step 1: Installing the TwinCAT OPC UA Server [▶ 130]
- Step 2: Copying of the installed UA Server files into a new directory [▶ 130]
- Step 3: Adaptation of the configuration file of the second UA Server [▶ 130]

Only the paths for the two operating system variants are different. Differences are pointed out in the description of the individual steps.

The TwinCAT OPC UA Server enables OPC UA to be used directly from the PLC level. An ADS server was integrated into the product for this so that PLC programmers can transmit data by function block over OPC UA. If you would like to have several TwinCAT OPC UA Servers running in parallel on one system, then please note that only the first instance of the UA Server can be addressed by function block.

Step 1: Installing the TwinCAT OPC UA Server

The initial installation of the TwinCAT OPC UA Server is carried out via the setup file available on the Beckhoff FTP server. The UA Server is installed by default in the following directory:

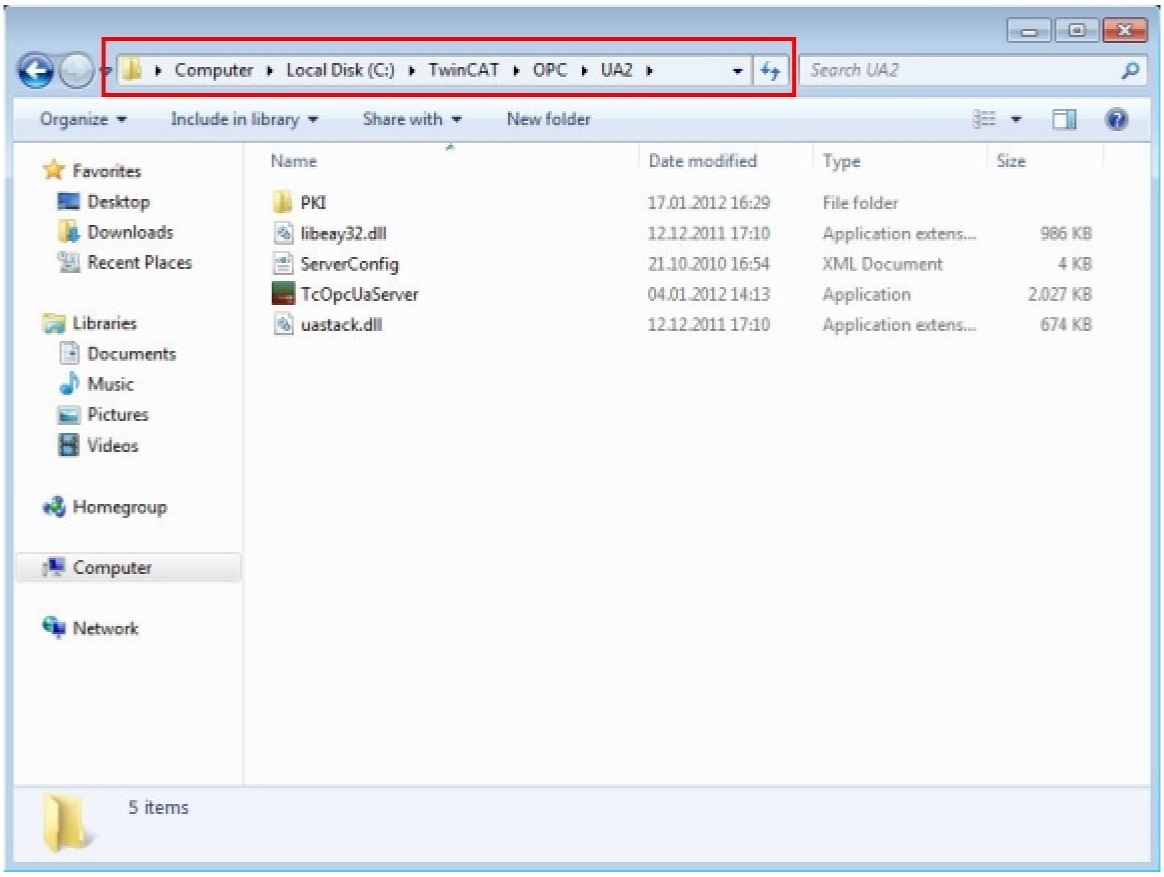
- **Windows XP:** %TwinCATDIR%\Function\TF6100-OPC-UA\Win32\Server\
- **Windows CE:** HDD\System

Step 2: Copying of the installed UA Server files into a new directory

The installed UA Server files must be copied to a new directory. Create a new directory named "Server2" in a folder of your choice, e.g.

- **Windows XP:** %TWINCATDIR%\Function\TF6100-OPC-UA\Win32\Server2\
- **Windows CE:** HDD\System\UA2

Copy the contents of the original folder into the newly created directory. The contents of this folder look like this:



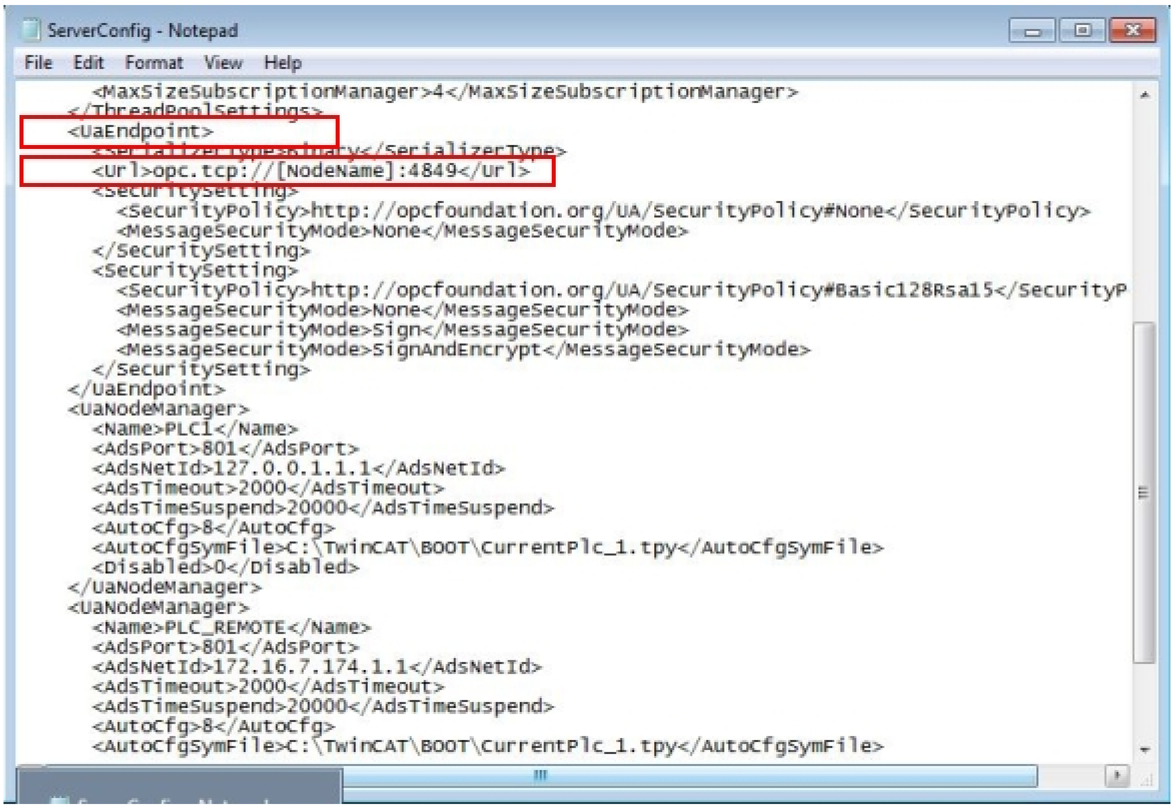
Name	Date modified	Type	Size
PKI	17.01.2012 16:29	File folder	
libeay32.dll	12.12.2011 17:10	Application extens...	986 KB
ServerConfig	21.10.2010 16:54	XML Document	4 KB
TcOpcUaServer	04.01.2012 14:13	Application	2.027 KB
uastack.dll	12.12.2011 17:10	Application extens...	674 KB

Step 3: Adaptation of the configuration file of the second UA Server

So that the second UA Server uses a different TCP port, you have to adapt its configuration file once. To do this, open the file *ServerConfig.xml* with a text editor of your choice, e.g. Notepad.

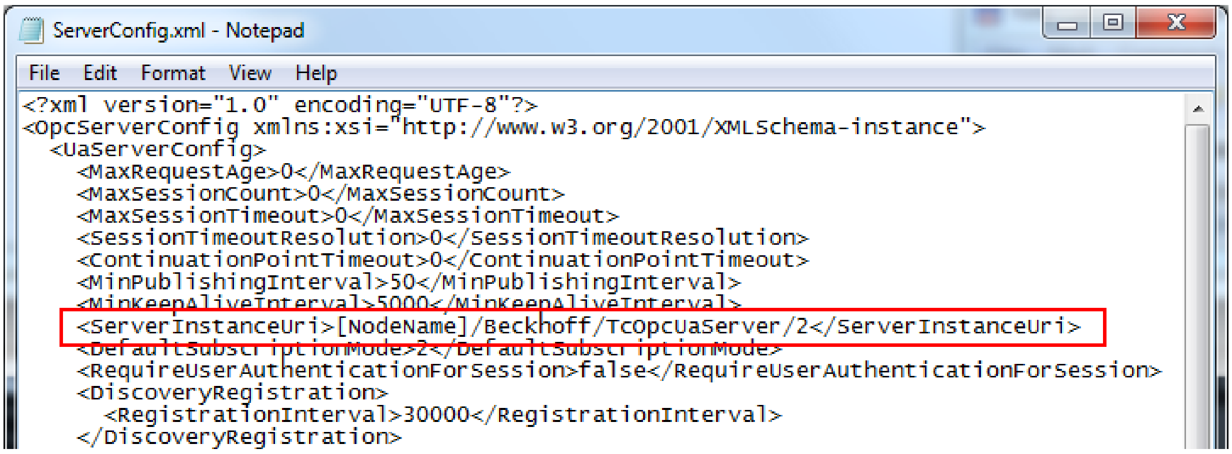
Further information on the configuration file of the UA Server can be found in the section Editing the configuration file [▸ 126].

1. In this file, search for the entry <Url>opc.tcp://[NodeName]:4840</Url> below the node <UaEndpoint> and change its port number (4840) to a TCP port of your choice, e.g. "4849":



```
    <MaxSizeSubscriptionManager>4</MaxSizeSubscriptionManager>
  </ThreadPoolSettings>
  <UaEndpoint>
    <SerializerType>Binary</SerializerType>
    <Url>opc.tcp://[NodeName]:4849</Url>
    <SecuritySetting>
      <SecurityPolicy>http://opcfoundation.org/UA/SecurityPolicy#None</SecurityPolicy>
      <MessageSecurityMode>None</MessageSecurityMode>
    </SecuritySetting>
    <SecuritySetting>
      <SecurityPolicy>http://opcfoundation.org/UA/SecurityPolicy#Basic128Rsa15</SecurityP
      <MessageSecurityMode>None</MessageSecurityMode>
      <MessageSecurityMode>Sign</MessageSecurityMode>
      <MessageSecurityMode>SignAndEncrypt</MessageSecurityMode>
    </SecuritySetting>
  </UaEndpoint>
  <UaNodeManager>
    <Name>PLC1</Name>
    <AdsPort>801</AdsPort>
    <AdsNetId>127.0.0.1.1.1</AdsNetId>
    <AdsTimeout>2000</AdsTimeout>
    <AdsTimeSuspend>20000</AdsTimeSuspend>
    <AutoCfg>8</AutoCfg>
    <AutoCfgSymFile>C:\TwinCAT\BOOT\CurrentPlc_1.tpy</AutoCfgSymFile>
    <Disabled>0</Disabled>
  </UaNodeManager>
  <UaNodeManager>
    <Name>PLC_REMOTE</Name>
    <AdsPort>801</AdsPort>
    <AdsNetId>172.16.7.174.1.1</AdsNetId>
    <AdsTimeout>2000</AdsTimeout>
    <AdsTimeSuspend>20000</AdsTimeSuspend>
    <AutoCfg>8</AutoCfg>
    <AutoCfgSymFile>C:\TwinCAT\BOOT\CurrentPlc_1.tpy</AutoCfgSymFile>
```

2. In addition, adapt the node <ServerInstanceUri>[NodeName]/Beckhoff/TcOpcUaServer/ 1 </ServerInstanceUri> and issue a new number, e.g.:



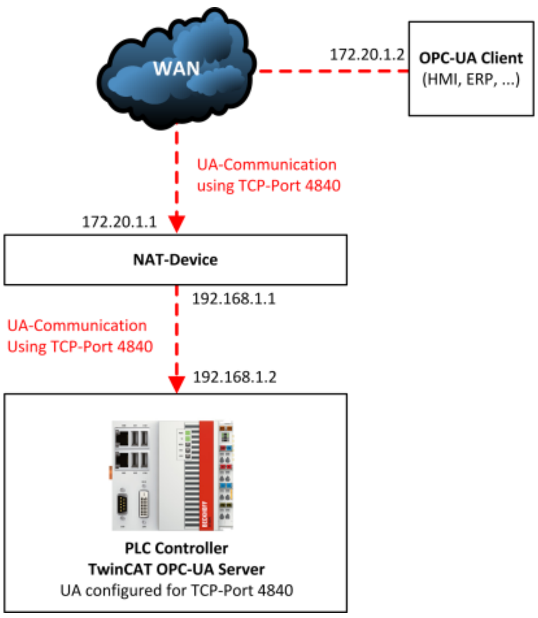
```
<?xml version="1.0" encoding="UTF-8"?>
<OpcServerConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <UaServerConfig>
    <MaxRequestAge>0</MaxRequestAge>
    <MaxSessionCount>0</MaxSessionCount>
    <MaxSessionTimeout>0</MaxSessionTimeout>
    <SessionTimeoutResolution>0</SessionTimeoutResolution>
    <ContinuationPointTimeout>0</ContinuationPointTimeout>
    <MinPublishingInterval>50</MinPublishingInterval>
    <MinKeepAliveInterval>5000</MinKeepAliveInterval>
    <ServerInstanceUri>[NodeName]/Beckhoff/TcOpcUaServer/2</ServerInstanceUri>
    <DefaultSubscriptionMode>2</DefaultSubscriptionMode>
    <RequireUserAuthenticationForSession>false</RequireUserAuthenticationForSession>
    <DiscoveryRegistration>
      <RegistrationInterval>30000</RegistrationInterval>
    </DiscoveryRegistration>
```

⇨ You can now start the second TwinCAT OPC UA Server by executing the file:

- **Windows XP**: %TWINCATDIR%\Function\TF6100-OPC-UA\Win32\Server2\TcOpcUaServer.exe -console

4.1.10.6 Configuring firewalls

To make OPC UA communication also possible via a NAT device, e.g. an Internet router, this device must be able to forward the UA port used to the TwinCAT OPC UA Server (so-called "port forwarding"). By default the TwinCAT OPC UA Server is configured for UA communication via the TCP port 4840; however, you can adapt this configuration yourself if necessary, either via the server configuration file [▸ 126] or using the OPC UA Configurator [▸ 187]. The following figure illustrates the relationship between port forwarding and the UA Server.

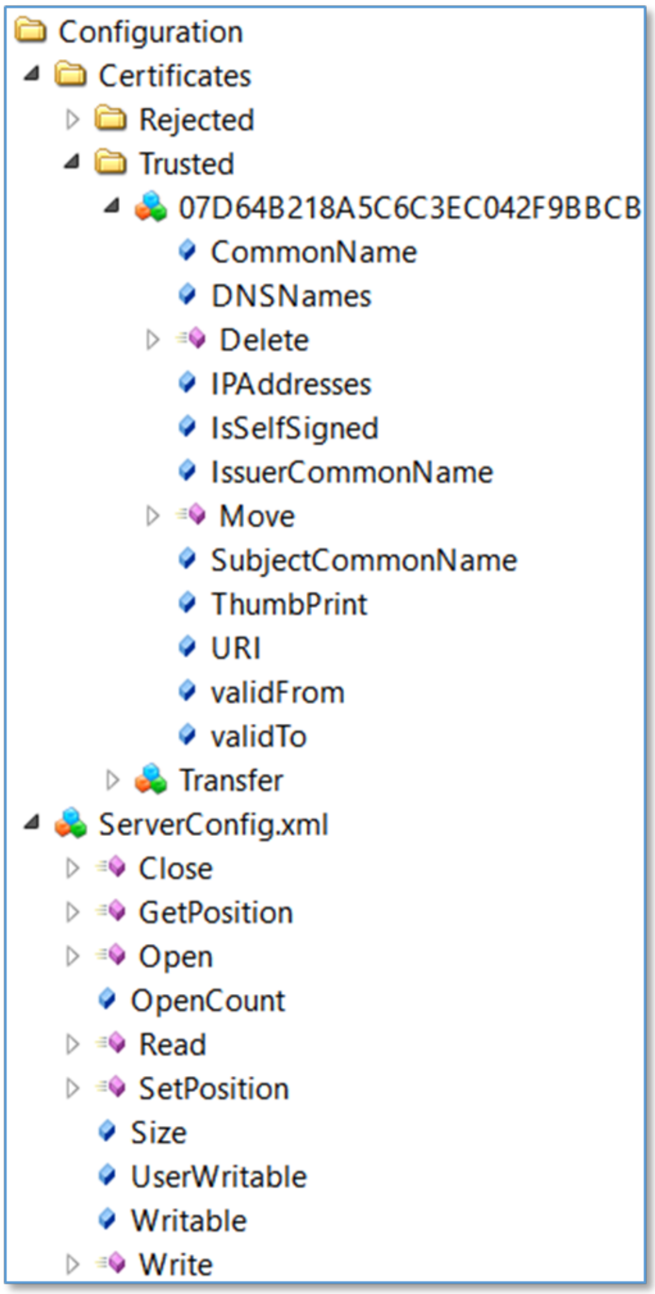


In this example, the OPC UA Client establishes a UA connection to the NAT device via TCP port 4840, which then forwards this communication connection to the TwinCAT OPC UA Server via port forwarding. The NAT device thus only needs to forward the UA port configured in the TwinCAT OPC UA Server to the target device. The port used by the UA Server can either be viewed in the server configuration file [▸ 126] or conveniently through the UA Configurator [▸ 187](in delivery state this is always TCP port 4840). For the appropriate configuration of your NAT device for port forwarding please refer to its documentation.

4.1.10.7 Configuring the namespace

As of version 2.1.x, the TwinCAT OPC UA Server offers separate configuration of the namespace containing the following functionalities:

- Management of the server configuration file *ServerConfig.xml* (read/write access) of the OPC UA Server
- Certificate management for trusted/rejected client certificates



Management of the server configuration file

The server configuration file *ServerConfig.xml* is published in the namespace as a regular OPC UA FileType and therefore offers the corresponding methods and properties for gaining access to the file.

Management of client certificates

Each client certificate known to the server is published in the namespace as an OPC UA certificate type. Certificates are divided into "rejected" and "trusted" certificates, which is represented by a separate folder in the namespace.

A certificate can be moved between the trust lists by calling the Move() method.

In addition, various properties provide additional information about the certificates themselves for easier identification.

Using the namespace configuration

The namespace configuration can only be used by authenticated users. This means that an OPC UA Client must authenticate itself with the OPC UA Server by providing a valid username/password combination. Valid user names are either local users or users from a Windows domain if the computer running the OPC UA Server is a member of a Windows domain.

In its configuration file the OPC UA Server provides the following XML tags for configuring the access to the namespace configuration:

```
<ServerConfigNamespace Exposed="True" MinUserlevel="14" />
<Users>
  <User Level="15">Administrator</User>
  <User Level="10">domain\RegularUser</User>
</Users>
```

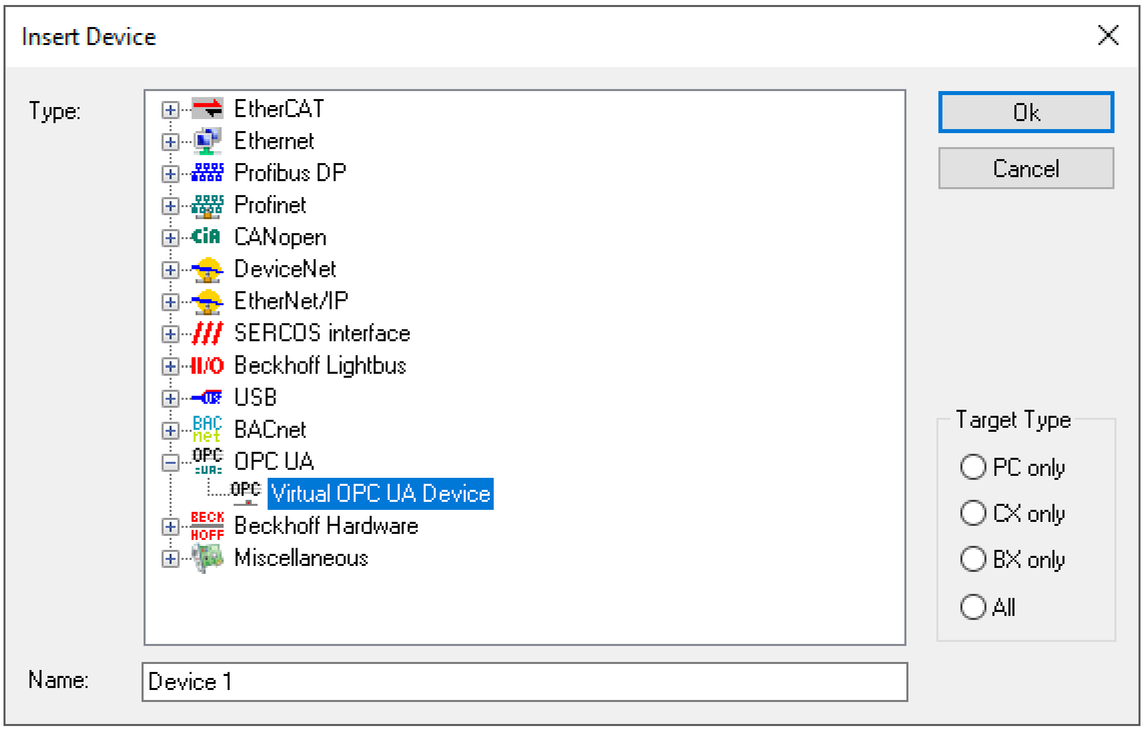
In the above example, the local user name "Administrator" is allowed to access the namespace configuration, whereas the user account "domain\RegularUser" of the Windows domain may not access this namespace because the user level is too low.

4.2 Client I/O

4.2.1 Overview

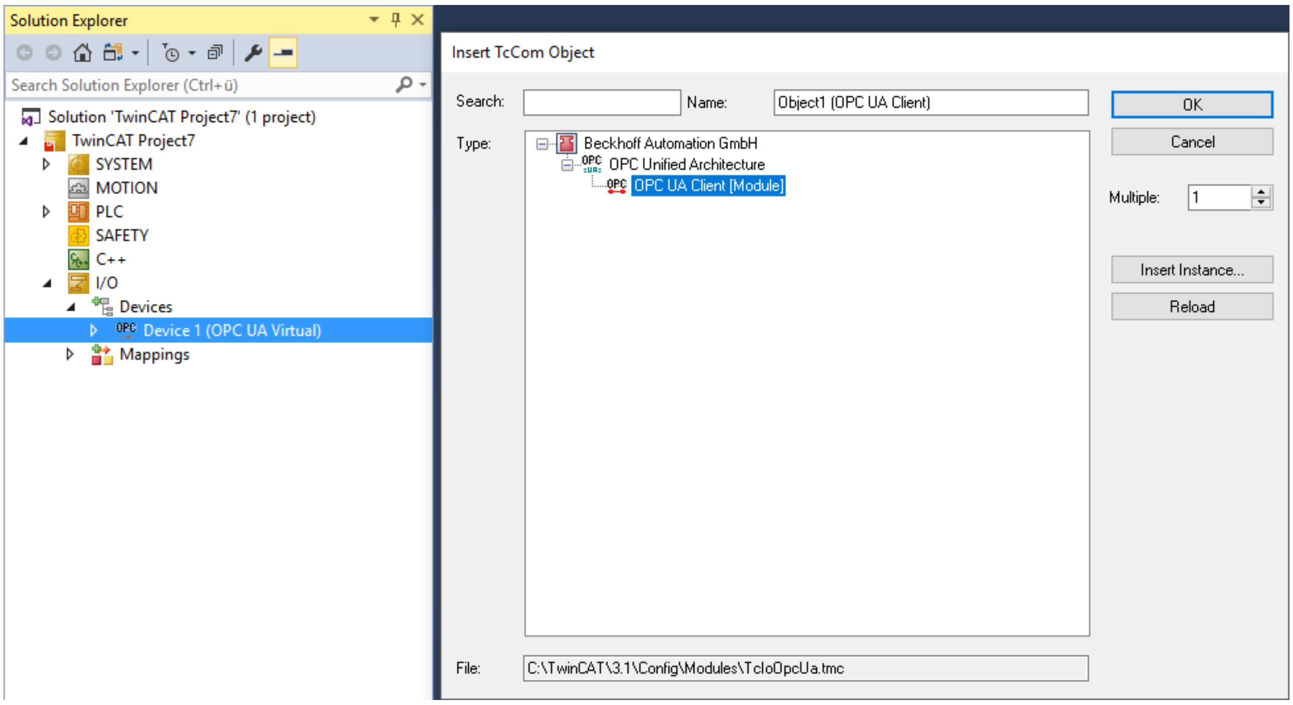
This function requires TwinCAT 3.1 Build 4022.4 or higher.

The TwinCAT OPC UA Client is also integrated as an I/O driver and is available as such in the TwinCAT I/O configuration. This not only shortens the engineering time required to establish the OPC UA connection to a remote server, it also enables the implementation of many application cases, such as the creation of protocol bridges that are simple to configure.



Name: Device 1

First of all, add a virtual device container to the configuration and then an OPC UA Client object.



Name: Object1 (OPC UA Client)

Multiple: 1

File: C:\TwinCAT\3.1\Config\Modules\TcIoOpcUa.tmc

After you have created the OPC UA I/O Client object, you can configure its connection to an OPC UA Target Server and add variables, methods and structures to the process image. The process of data exchange with a target server is reduced to a simple method of mapping process variables.

Requirements

Products	Setup versions	Target platform
TF6100	4.x.x	IPC or CX (x86, x64, ARM)

4.2.2 Quick start

The following instructions enable you to make a quick start with the OPC UA I/O Client. In the instructions a PLC project will be created that enables a variable via the OPC UA Server (for the simulation of a UA Server) and reads the enabled variable in again via the OPC UA I/O Client. This can of course be divided between two projects and two controllers.

Enabling variables for a server

For simulation, the OPC UA Server should be used in this quick start.

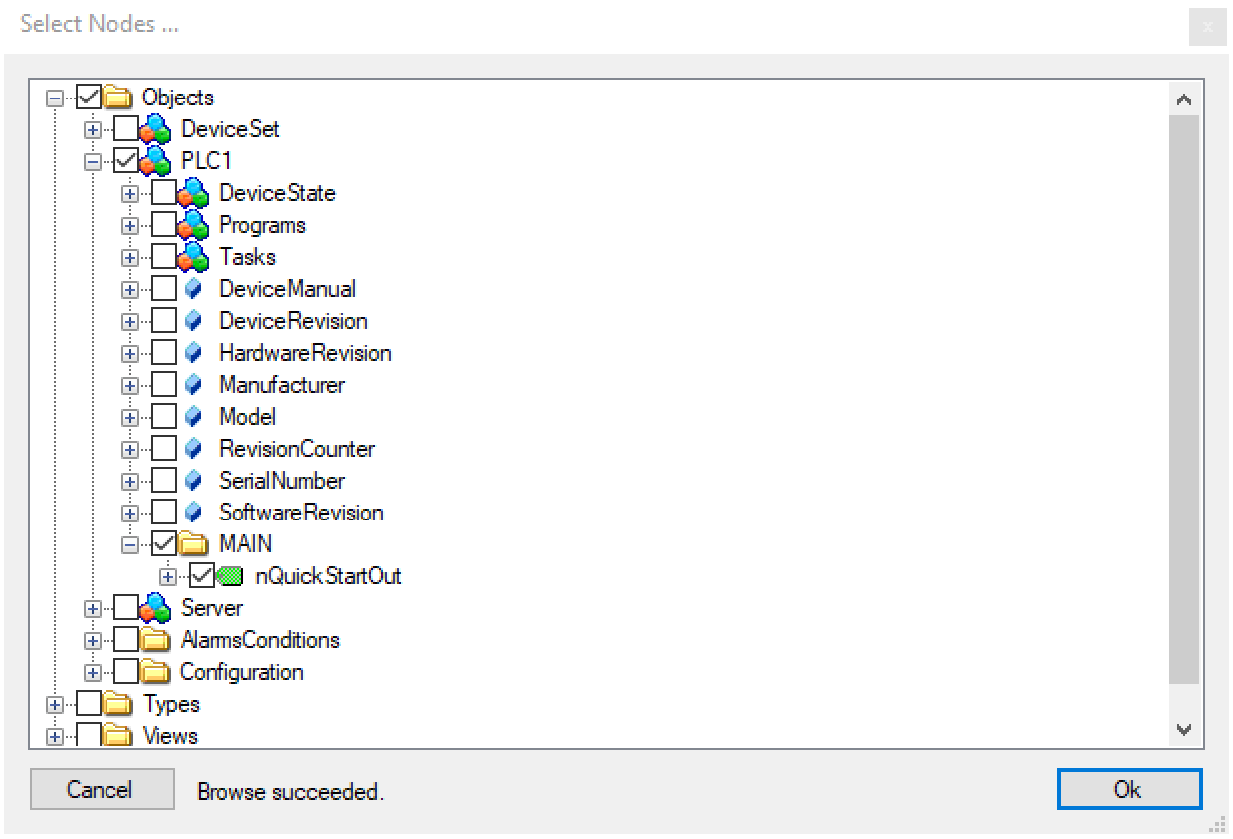
1. Create a new TwinCAT project and add a new PLC project to the project.
2. Create a new variable "nQuickStartOut" of data type INT in the PLC project. This should be the variable that is enabled via the OPC UA Server.
3. Set the OPC UA attribute [▶ 43] for this variable in order to be able to enable the variable via the OPC UA Server.
4. Activate the check box for the download of the TMC file.
5. Activate the configuration.

⇨ The variable is enabled for the server.

Configuring the I/O UA Client

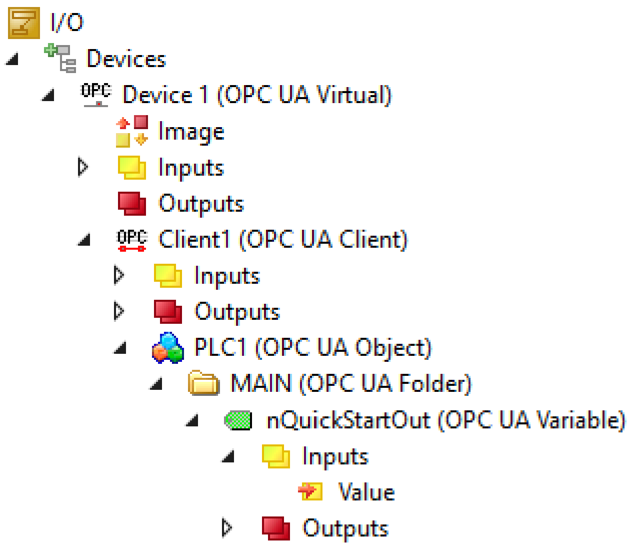
1. In the same PLC project, create another variable "nQuickStartIn" of data type INT and define it as an input variable (AT%I*). This should be the variable that the OPC UA Client I/O device links to the server variable through mapping.
2. Compile the project so that the input variable is available in the PLC process image.
3. Add a new "Virtual OPC UA Device".

4. Add a new "OPC UA Client" to the device and open its settings.
5. Navigate to the **Settings** tab. Enter the server URL of the OPC UA Server. In this sample this is "opc.tcp://localhost:4840".
6. Click **Add Nodes**.
7. Navigate to the shared variable "nQuickStartOut" on the OPC UA Server (located under PLC1 > MAIN) and select it. The variable is now added to the process image of the "Virtual OPC UA Device".


Select Nodes ...

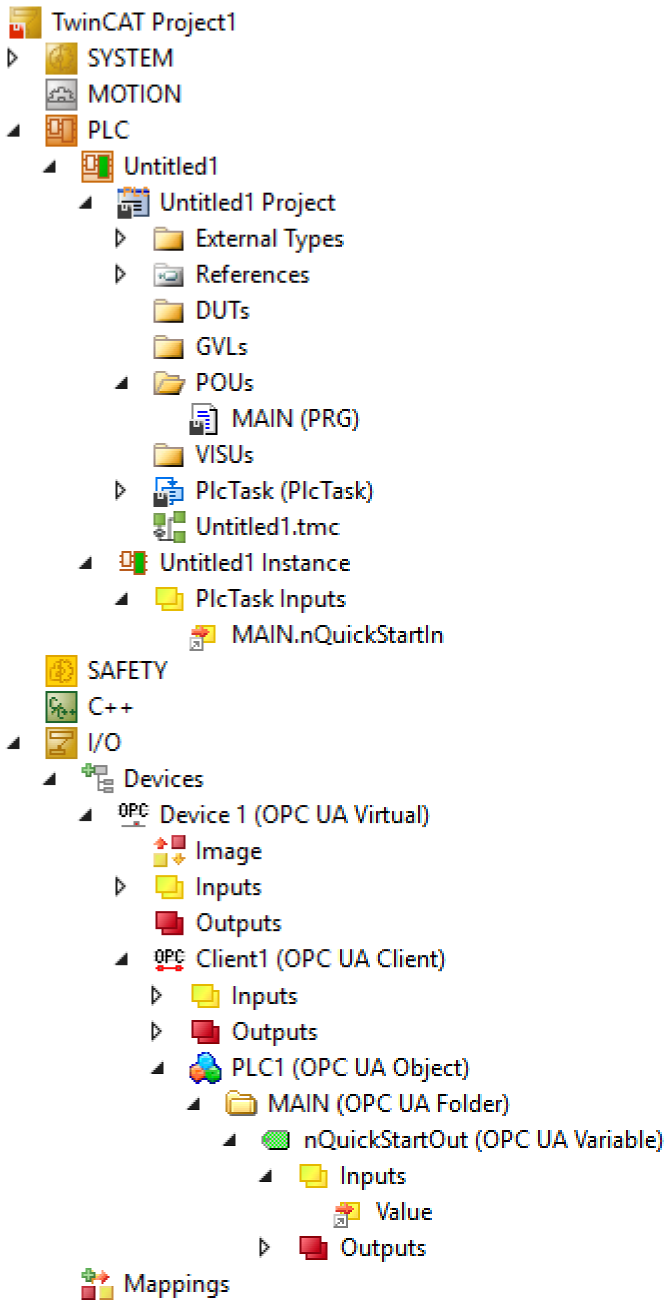
- [✓] Objects
  - [ ] DeviceSet
  - [✓] PLC1
    - [ ] DeviceState
    - [ ] Programs
    - [ ] Tasks
    - [ ] DeviceManual
    - [ ] DeviceRevision
    - [ ] HardwareRevision
    - [ ] Manufacturer
    - [ ] Model
    - [ ] RevisionCounter
    - [ ] SerialNumber
    - [ ] SoftwareRevision
    - [✓] MAIN
      - [✓] nQuickStartOut
  - [ ] Server
  - [ ] AlarmsConditions
  - [ ] Configuration
- [ ] Types
- [ ] Views

Browse succeeded.


- I/O
  - Devices
    - Device 1 (OPC UA Virtual)
      - Image
      - Inputs
      - Outputs
      - Client1 (OPC UA Client)
        - Inputs
        - Outputs
        - PLC1 (OPC UA Object)
          - MAIN (OPC UA Folder)
            - nQuickStartOut (OPC UA Variable)
              - Inputs
                - Value
              - Outputs

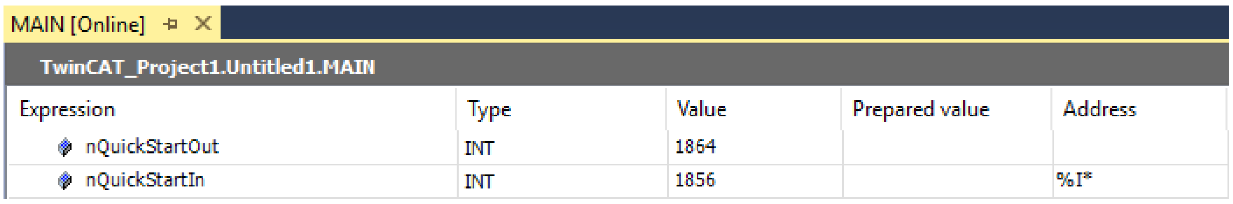
8. Expand the newly added variable in the process image and double-click on **Input > Value**. Link the variable to the variable "nQuickStartIn" from the PLC process image.
9. Activate the configuration.

⇨ You should now find the configuration shown below. The two process image variables nQuickStartIn and nQuickStartOut are linked to each other.



```
TwinCAT Project1
  SYSTEM
  MOTION
  PLC
    Untitled1
      Untitled1 Project
        External Types
        References
        DUTs
        GVLs
        POUs
          MAIN (PRG)
        VISUs
        PlcTask (PlcTask)
        Untitled1.tmc
      Untitled1 Instance
        PlcTask Inputs
          MAIN.nQuickStartIn
  SAFETY
  C++
  I/O
    Devices
      Device 1 (OPC UA Virtual)
        Image
        Inputs
        Outputs
        Client1 (OPC UA Client)
          Inputs
          Outputs
          PLC1 (OPC UA Object)
            MAIN (OPC UA Folder)
              nQuickStartOut (OPC UA Variable)
                Inputs
                  Value
                Outputs
    Mappings
```

10. After activating the configuration, log in to the PLC program.

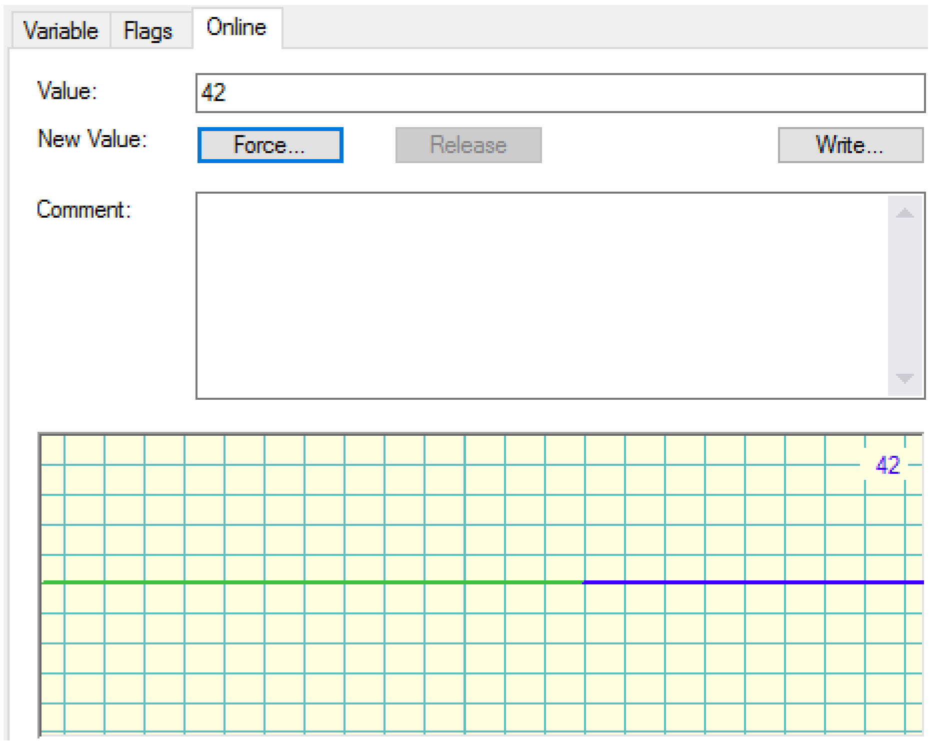


MAIN [Online]

TwinCAT_Project1.Untitled1.MAIN

Expression	Type	Value	Prepared value	Address
nQuickStartOut	INT	1864		
nQuickStartIn	INT	1856		%I*

You can also view the current value in the process image mapping.



Value: 42

11. You can use the settings on the OPC UA Client to modify parameters such as the sampling rate.

Requirements

Products	Setup versions	Target platform
TF6100	4.x.x	IPC or CX (x86, x64, ARM)

4.2.3 Supported data types

The OPC UA Client enables direct access to an OPC UA Server from a real-time logic. For the reading and writing of data, the data type of the OPC UA node must be assigned to the TwinCAT environment (mapping). This assignment is described below.

Basic data types

The assignment of the basic data types is described in PLCopen OPC UA Information Model for IEC 61131-3.

OPC UA data type	PLC data type
Boolean	BOOL
SByte	SINT
Byte	USINT
Int16	INT
Int32	DINT
String	STRING

OPC UA data type	PLC data type
USint	BYTE
Float	REAL
Double	LREAL
UInt16	UINT
UInt32	UDINT
Int64	LINT
UInt64	ULINT
DateTime	DT

You can use the mapping both on the OPC UA Client PLCopen function block and on the OPC UA I/O Client.

Derived data types

OPC UA defines basic data types. Other data types are derived from these.

On the client side, access to all UA data types is only possible with PLC data types. Data types derived from the basic data types are also supported from TcOpcUaClient version 2.1.0.36 or higher.

Currently supported non-base data types on the client side are:

- Counter (use UDINT in PLC)

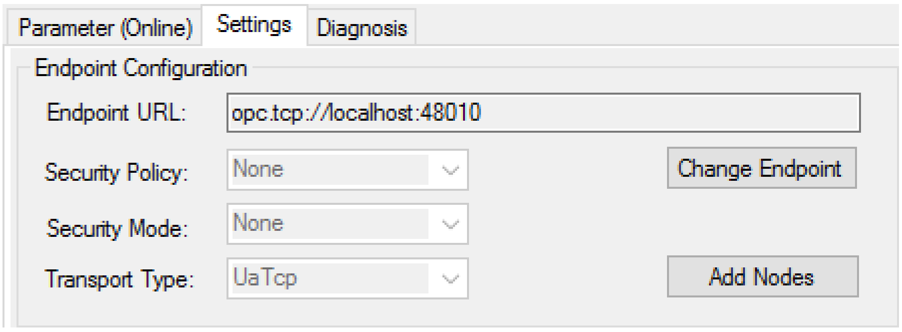
In addition, the OPC UA I/O Client supports structured data types (StructuredTypes) if the structured data type does not incorporate any non-supported data type (e.g. in a member variable). The following list provides an overview of the data types that are not currently supported:

- ByteString
- NodeID
- LocalizedText (and thus AnalogItemTypes, DiscreteItemTypes)

4.2.4 Adding nodes of a Server

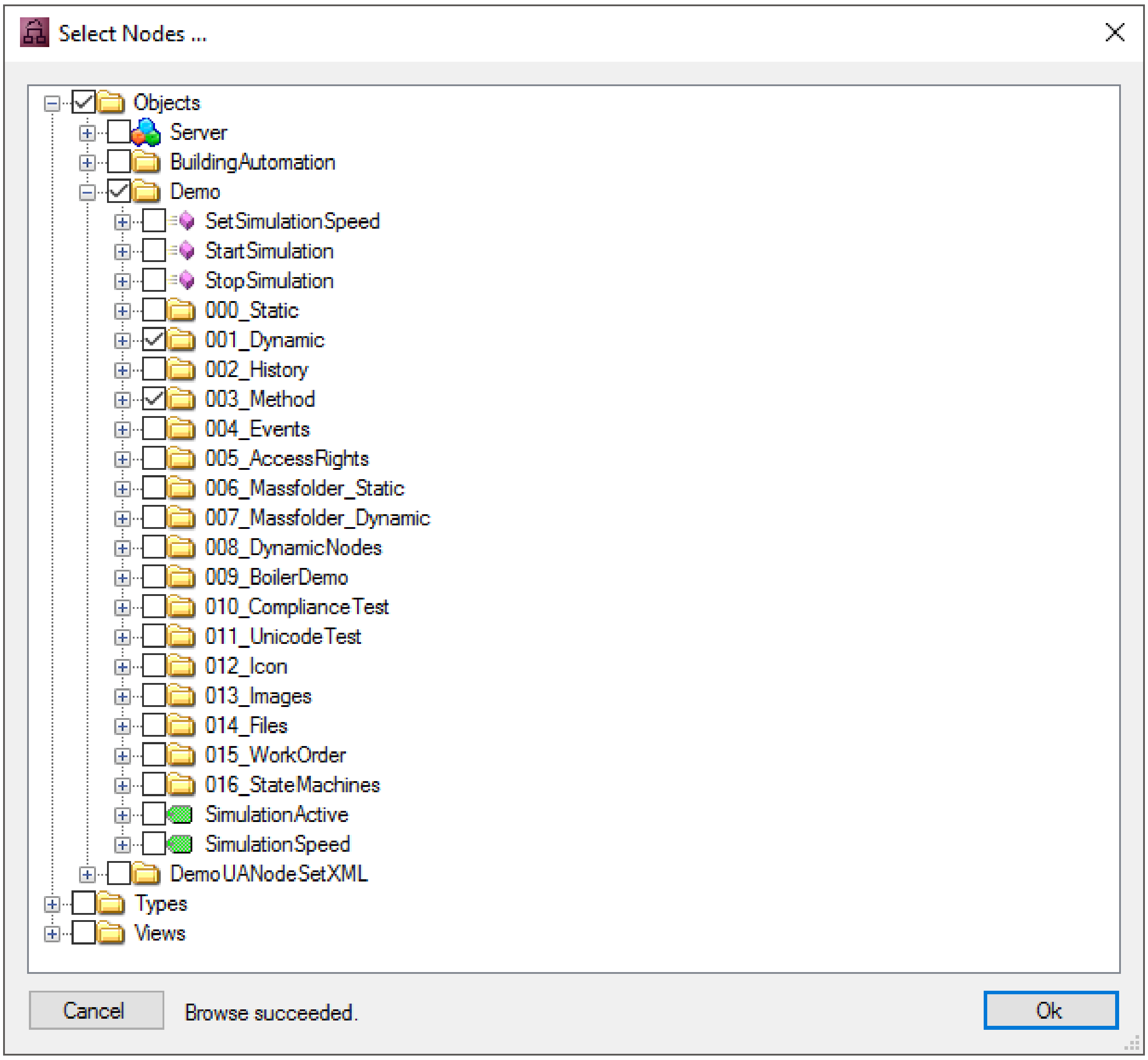
To add nodes of an OPC UA Target Server to the process image of the OPC UA I/O Client, you can use the namespace browser that is integrated in the I/O Client.

Open the **Settings** tab of the client configuration and click on the **Add Nodes** button to open the namespace browser.

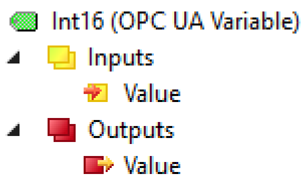


Endpoint URL: opc.tcp://localhost:48010
Security Policy: None
Security Mode: None
Transport Type: UaTcp

The namespace browser dialog automatically establishes a connection with the Target Server by using the specified connection parameters.



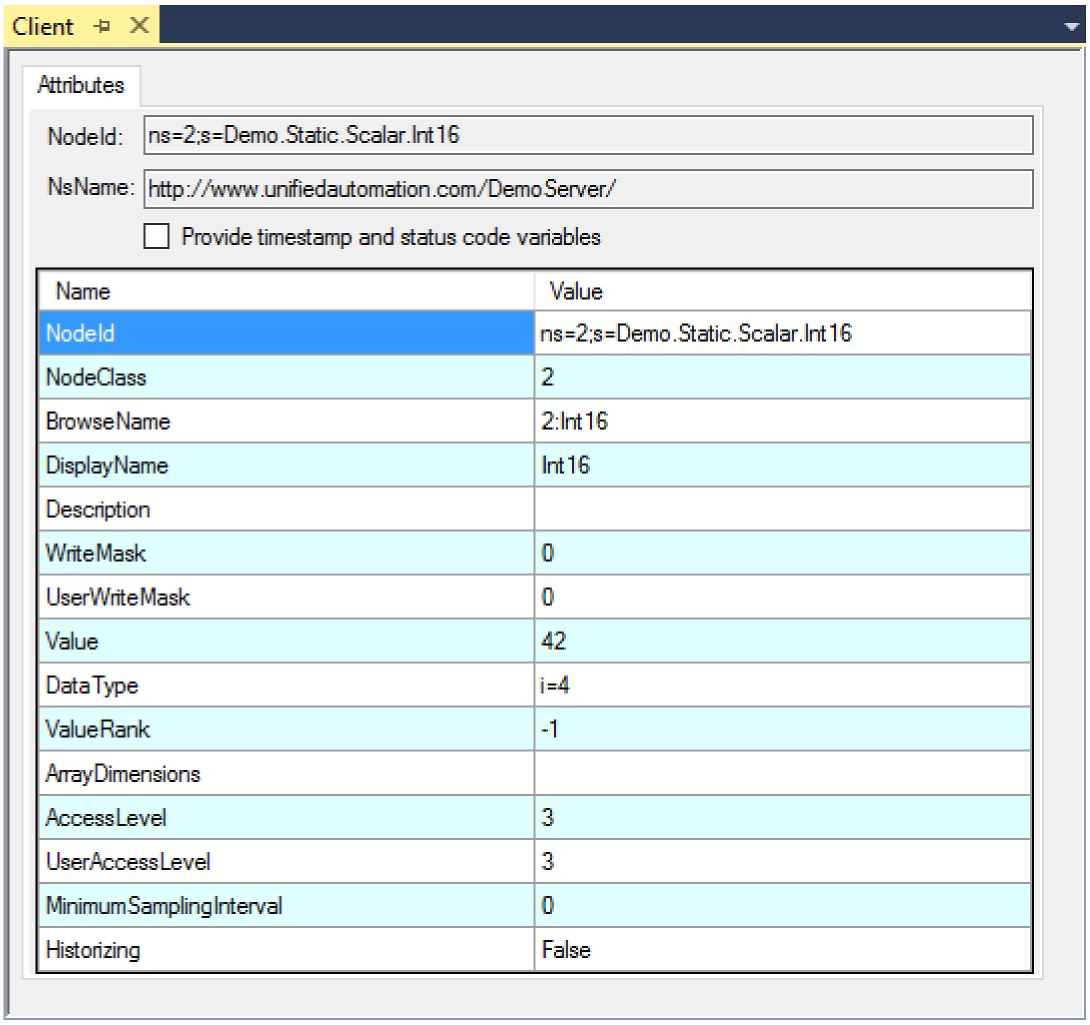
Within the namespace browser you can activate and deactivate the check boxes in order to add nodes to the process image or remove them from it. Each selected node is shown in the process image as an input or output variable.



The input variable reads data from the node and contains the last value received from the node. The output variable writes data to the node if a value has been set.

4.2.5 Node attributes

If you double-click on a node in the process image, you will see the UA attributes with their current values as they were at the time of opening the window.



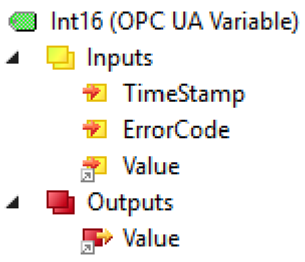
NodeId: ns=2;s=Demo.Static.Scalar.Int16

NsName: http://www.unifiedautomation.com/DemoServer/

☐ Provide timestamp and status code variables

Name	Value
NodeId	ns=2;s=Demo.Static.Scalar.Int16
NodeClass	2
BrowseName	2:Int16
DisplayName	Int16
Description	
WriteMask	0
UserWriteMask	0
Value	42
DataType	i=4
ValueRank	-1
ArrayDimensions	
AccessLevel	3
UserAccessLevel	3
MinimumSamplingInterval	0
Historizing	False

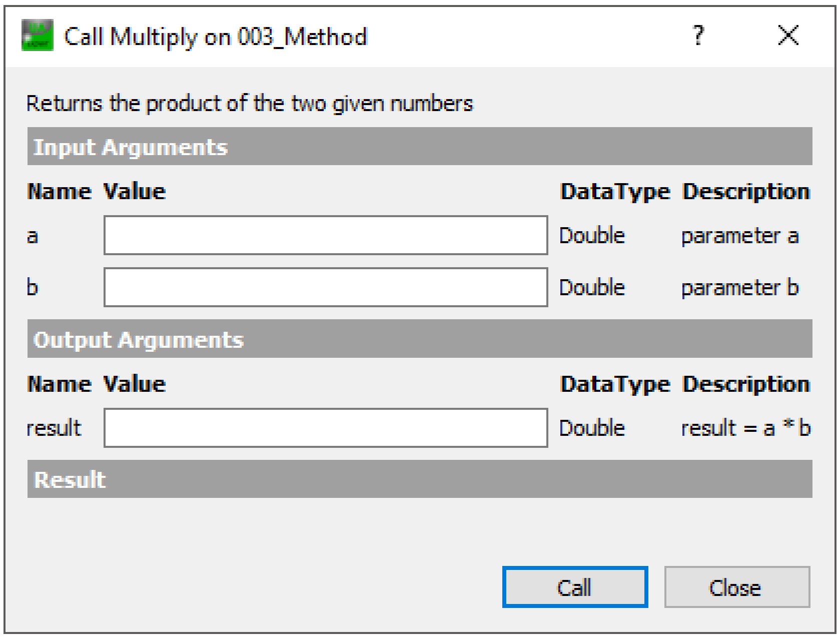
Further variables that can be used for diagnostic purposes are added to the process image using the check box **Provide timestamp and status code variables**.



4.2.6 Method call

The OPC UA I/O Client supports the call of server methods. You can add a method to the process image like any other variable. The "input arguments" of the method are then available as output variables in the process image, whereas the "output arguments" are added as input variables. Additional input and output variables, e.g. bExecute, bBusy, bError, are added to the process image so that the method can be called.

Sample: Method on the server


Call Multiply on 003_Method

Returns the product of the two given numbers

Input Arguments

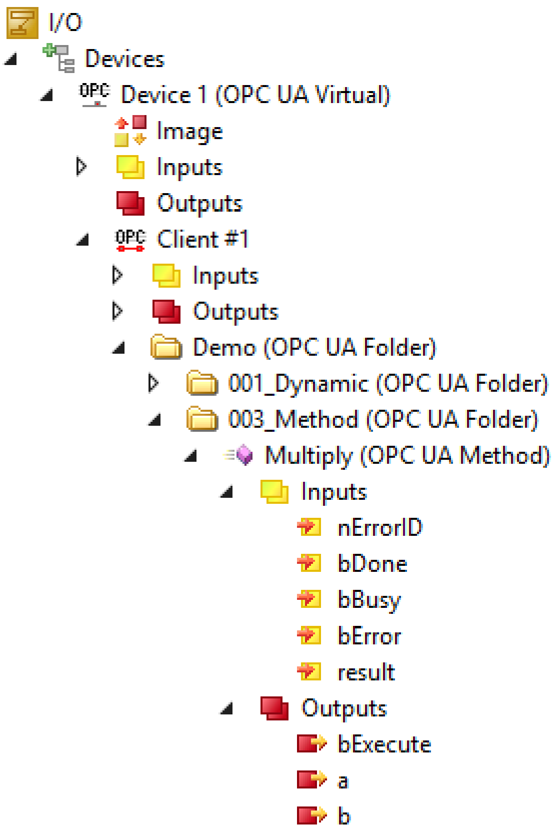
Name	Value	DataType	Description
a		Double	parameter a
b		Double	parameter b

Output Arguments

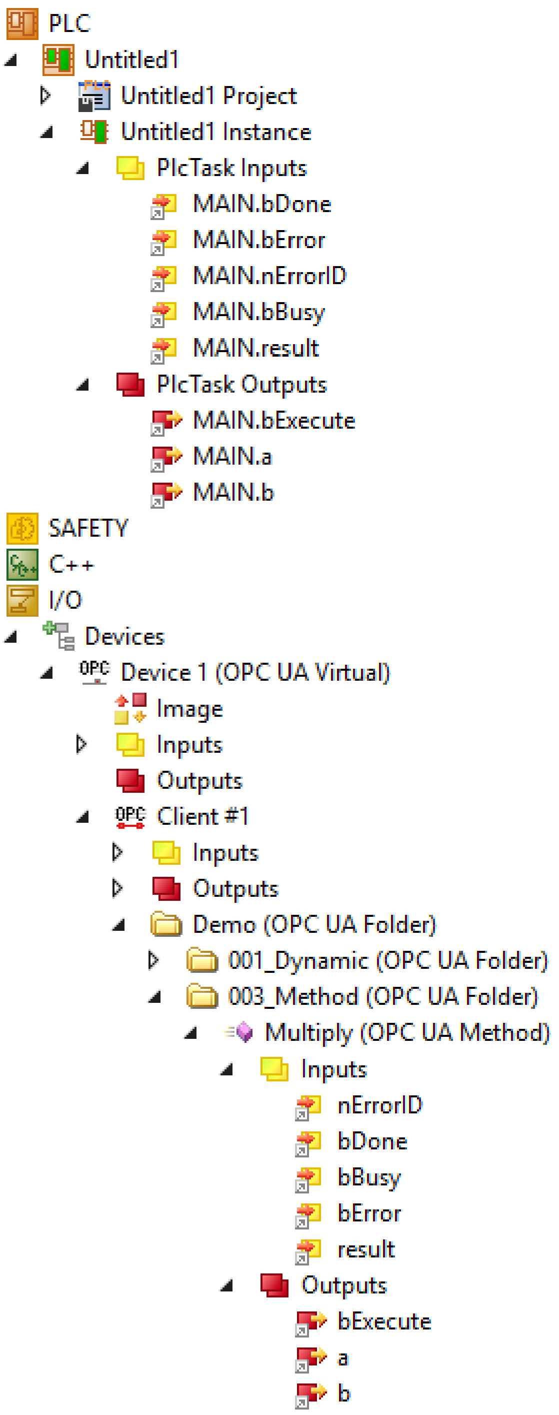
Name	Value	DataType	Description
result		Double	result = a * b

Result

Sample: Method after addition to the process image

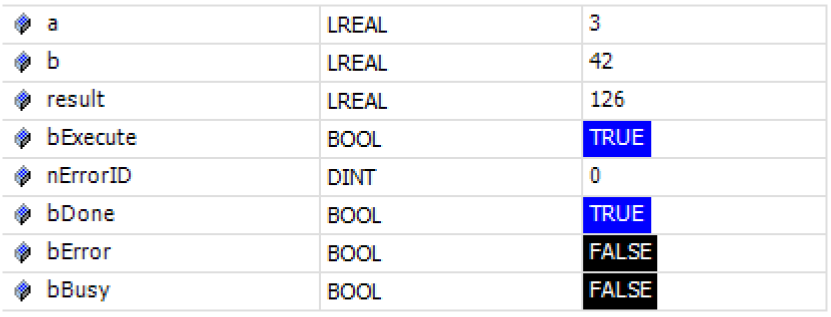

```
I/O
  Devices
    Device 1 (OPC UA Virtual)
      Image
      Inputs
      Outputs
      Client #1
        Inputs
        Outputs
        Demo (OPC UA Folder)
          001_Dynamic (OPC UA Folder)
          003_Method (OPC UA Folder)
            Multiply (OPC UA Method)
              Inputs
                nErrorID
                bDone
                bBusy
                bError
                result
              Outputs
                bExecute
                a
                b
```

You can then create a mapping between the input/output variables and the PLC variables.


PLC
Untitled1
Untitled1 Project
Untitled1 Instance
PlcTask Inputs
MAIN.bDone
MAIN.bError
MAIN.nErrorID
MAIN.bBusy
MAIN.result
PlcTask Outputs
MAIN.bExecute
MAIN.a
MAIN.b
SAFETY
C++
I/O
Devices
Device 1 (OPC UA Virtual)
Image
Inputs
Outputs
Client #1
Inputs
Outputs
Demo (OPC UA Folder)
001_Dynamic (OPC UA Folder)
003_Method (OPC UA Folder)
Multiply (OPC UA Method)
Inputs
nErrorID
bDone
bBusy
bError
result
Outputs
bExecute
a
b

Calling of a method

To call a method, set the output variable bExecute to TRUE. You can check whether the method call has been completed and whether it was successful via the input variables nErrorID, bDone, bBusy and bError.

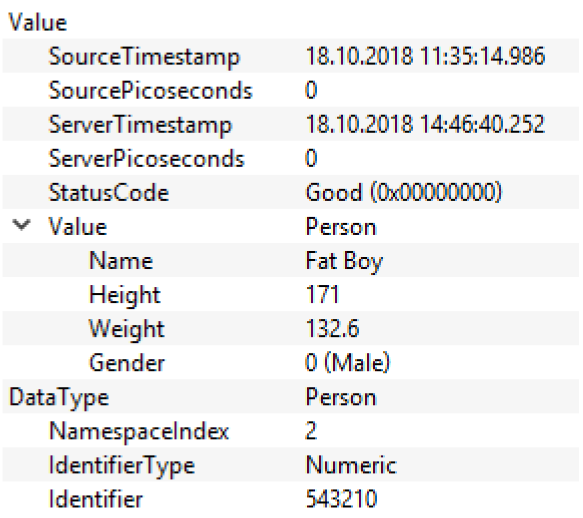


a	LREAL	3
b	LREAL	42
result	LREAL	126
bExecute	BOOL	TRUE
nErrorID	DINT	0
bDone	BOOL	TRUE
bError	BOOL	FALSE
bBusy	BOOL	FALSE

4.2.7 StructuredTypes

The OPC UA I/O Client supports read/write procedures with structured data types (StructuredTypes). You can also add StructuredTypes to the process image like any other variable. When adding a StructuredType to the process image, the type to be parsed is added to the TwinCAT type system so that, for example, it can simply be used by a PLC application.

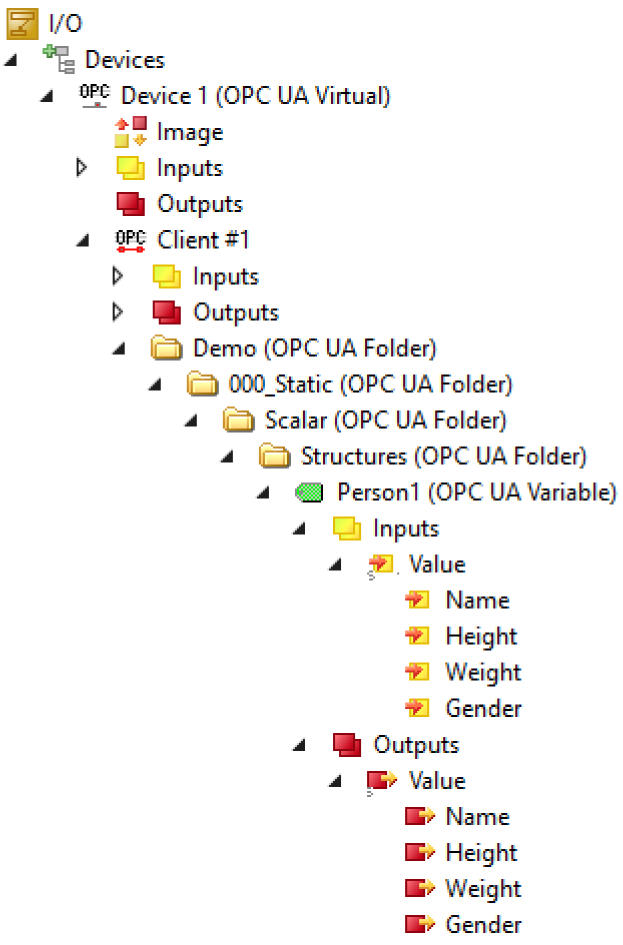
Sample: StructuredType on the server



Value	
SourceTimestamp	18.10.2018 11:35:14.986
SourcePicoseconds	0
ServerTimestamp	18.10.2018 14:46:40.252
ServerPicoseconds	0
StatusCode	Good (0x00000000)
Value	Person
Name	Fat Boy
Height	171
Weight	132.6
Gender	0 (Male)
DataType	Person
NamespaceIndex	2
IdentifierType	Numeric
Identifier	543210

In this sample the server contains a node of the structured data type "Person", which contains various member variables (Name, Height, Weight, Gender).

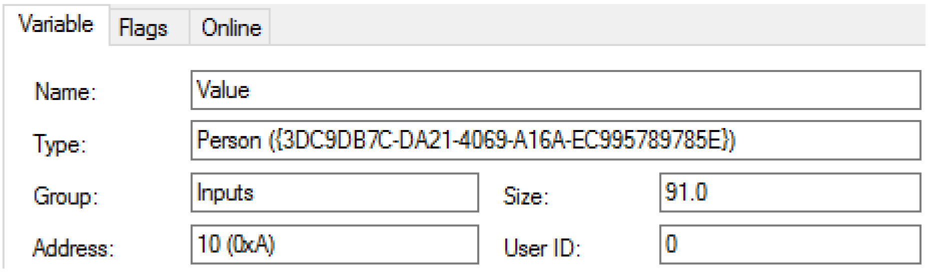
Sample: StructuredTypes in the process image



After you have added a node to the process image, the process image contains the node and also the structural information of the type, e.g. whether individual member variables of the node should be read or written.

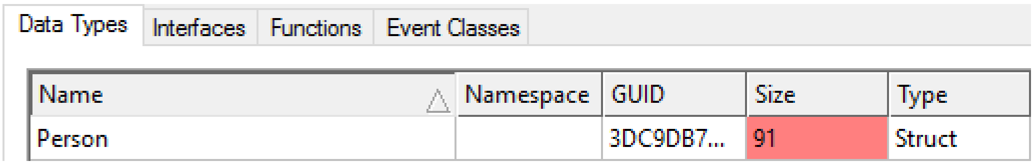
StructuredTypes in the TwinCAT type system

The data type is added to the TwinCAT type system. The "Value" tree items then have this data type.



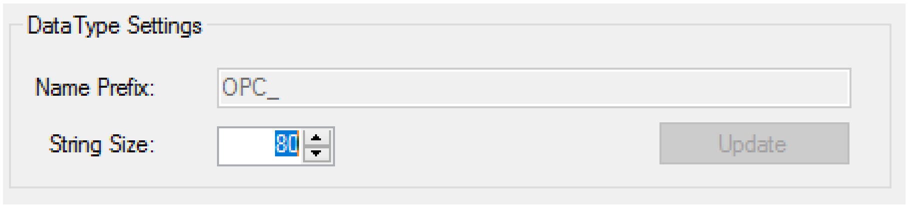
Name:	Value		
Type:	Person ({3DC9DB7C-DA21-4069-A16A-EC995789785E})		
Group:	Inputs	Size:	91.0
Address:	10 (0xA)	User ID:	0

You can also view the data type in the TwinCAT type system under SYSTEM > Type System.



Name	Namespace	GUID	Size	Type
Person		3DC9DB7...	91	Struct

To distinguish the data type from other data types you can add a prefix in the settings of the OPC UA Client.

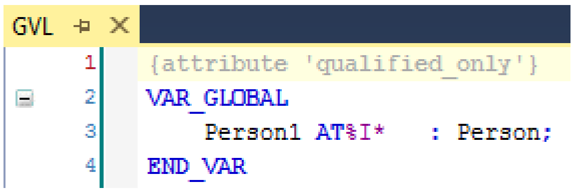


Name Prefix: OPC_

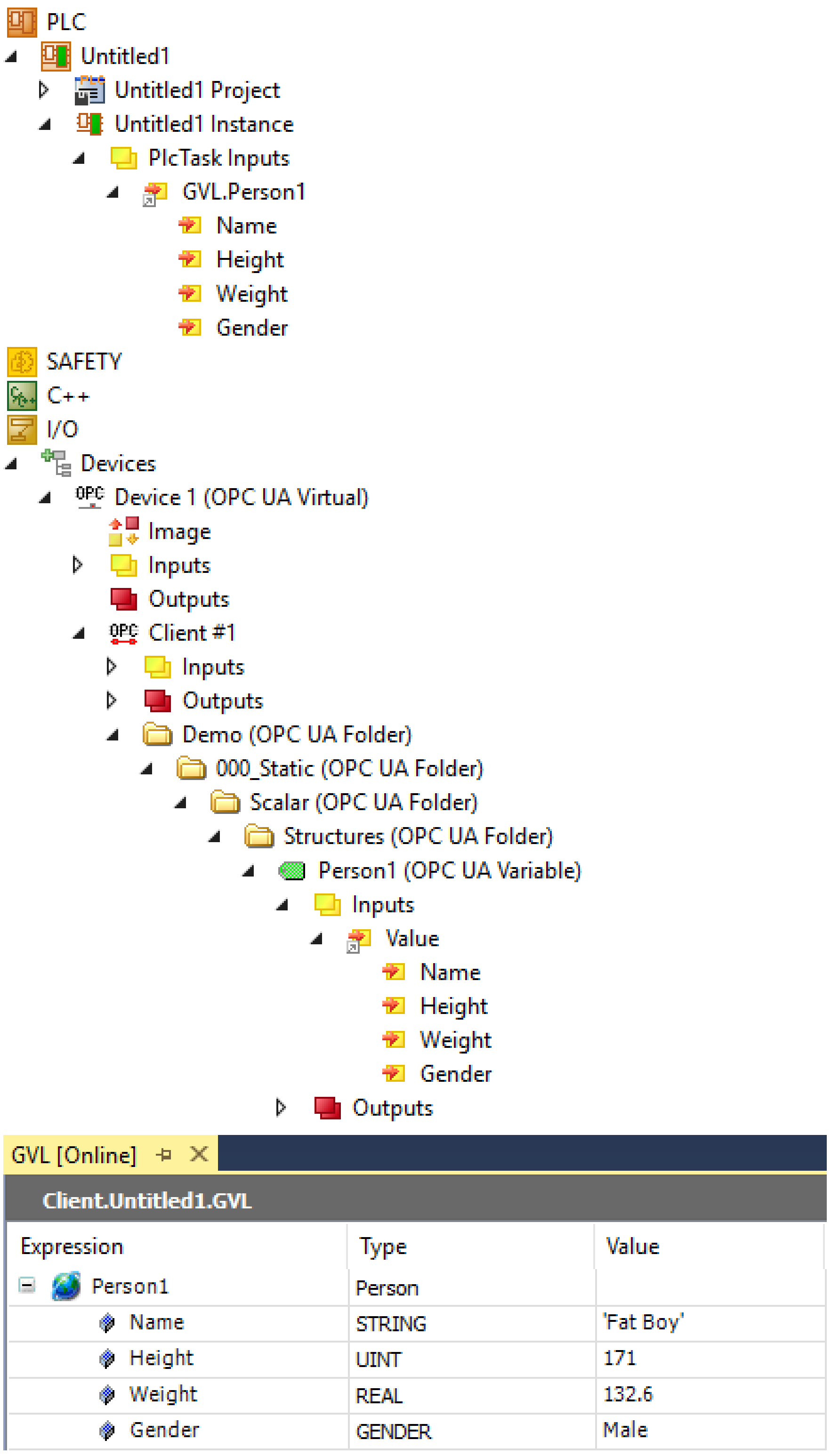
String Size: 80

Mapping a StructuredType

Since every StructuredType is added to the TwinCAT type system, the mapping of the variables is simple. Create an input/output variable of this data type and subsequently a mapping.



```
{attribute 'qualified_only'}
VAR_GLOBAL
    Person1 AT%I*   : Person;
END_VAR
```



```
PLC
◢ Untitled1
    ▷ Untitled1 Project
    ◢ Untitled1 Instance
        ◢ PlcTask Inputs
            ◢ GVL.Person1
                Name
                Height
                Weight
                Gender
SAFETY
C++
I/O
◢ Devices
    ◢ Device 1 (OPC UA Virtual)
        Image
        ▷ Inputs
        Outputs
        ◢ Client #1
            ▷ Inputs
            ▷ Outputs
            ◢ Demo (OPC UA Folder)
                ◢ 000_Static (OPC UA Folder)
                    ◢ Scalar (OPC UA Folder)
                        ◢ Structures (OPC UA Folder)
                            ◢ Person1 (OPC UA Variable)
                                ◢ Inputs
                                    ◢ Value
                                        Name
                                        Height
                                        Weight
                                        Gender
                                ▷ Outputs
```

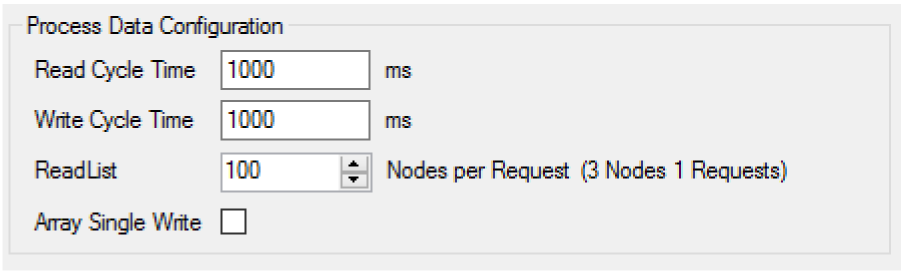
GVL [Online]

Client.Untitled1.GVL

Expression	Type	Value
⊟ Person1	Person	
Name	STRING	'Fat Boy'
Height	UINT	171
Weight	REAL	132.6
Gender	GENDER	Male

4.2.8 Data recording

Sampling rates are configured in each case on a device basis. The configuration is opened by double-clicking on the OPC UA Client device. The **Process Data Configuration** area shows various parameters for influencing the speed of data recording by a server.



Read Cycle Time 1000 ms
Write Cycle Time 1000 ms
ReadList 100 Nodes per Request (3 Nodes 1 Requests)
Array Single Write ☐

Parameter	Description
Read Cycle Time	Specifies how fast variables are cyclically read.
Write Cycle Time	Defines how often a write command is triggered on the OPC UA channel. If a variable value changes several times within a specific cycle time, only the last value is written to the OPC UA channel.
ReadList	Read commands on the OPC UA channel are bundled to save bandwidth. This parameter specifies how many variables are recorded in a single read command on the OPC UA channel. The labeling behind it indicates how many read commands arise from the current configuration.
Array Single Write	On activation, every write process is executed as a single write command on the OPC UA channel.

4.2.9 Security

4.2.9.1 Overview

Security was a central requirement in the development of OPC UA. It is addressed in various areas:

- Encryption
- Integrity
- Authentication

The confidentiality of the exchanged information is secured by the encryption of the exchanged messages. Modern cryptographic algorithms are used for this. In order to be able to cope with future security requirements as well, even stronger and more modern algorithms can subsequently be added to an application without changing the protocol.

Different security levels can be selected according to the requirements of the respective application. In some areas it is sufficient to sign the messages in order to prevent changes being made by third parties, while additional encryption of the messages is necessary in other cases where the data must also not be read by third parties.

TF6100 OPC UA offers the following security levels (security endpoints) that clients can connect to:

- None: No security
- Sign: signed messages
- Sign & Encrypt: signed and encrypted messages

The signing of messages prevents a third party from changing the contents of a message. This prevents, for example, a write statement to open a switch being falsified by a third party and the switch being closed instead.

OPC UA applications identify themselves via so-called software and application instance certificates. With the aid of software certificates it is possible to grant certain client applications extended access to the information on an OPC UA Server, for example for the engineering of an OPC UA Server. Application instance certificates can be used to ensure that an OPC UA Server communicates only with preconfigured clients. A client can ensure by means of the server's application instance certificate that it is speaking to the correct server (similar to the certificates of a Web browser). The taking into account of these certificates is optional, i.e. an OPC UA server can also grant the same access to each client, depending on the user rights.

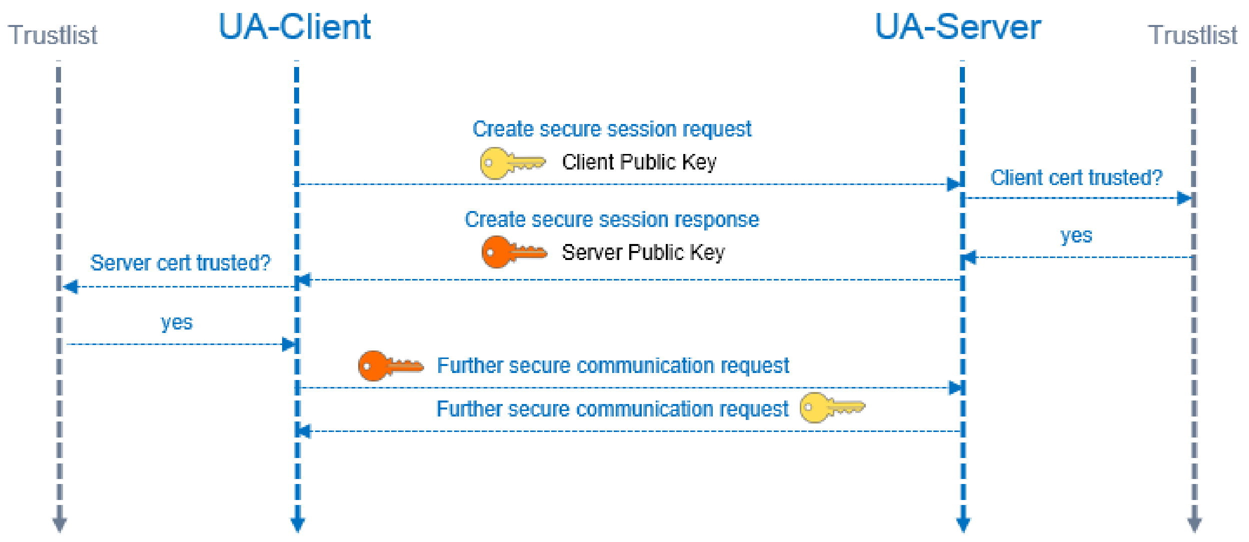
The TwinCAT OPC UA Client and TwinCAT OPC UA Server generate a self-signed certificate during the first startup process. This certificate consists of a private key and a public key. The private key is saved in the *\PKI\CA\private* directory and the corresponding public key in the *\PKI\CA\certs* directory. Any OPC UA Client wishing to establish a secure connection with the Server via one of the security endpoints (Sign, Sign&Encrypt) must know the public key of the OPC UA Server. Conversely the OPC UA Server must know the Client's public key. This so-called key exchange is described in the following sections:

- Authentication [▶ 122]
- Certificate exchange [▶ 148]
- Access rights [▶ 123]

4.2.9.2 Certificate exchange

The communication between an OPC UA Client and an OPC UA Server can optionally be secured by communication with a secure endpoint. By default, both the TwinCAT OPC UA Server and the TwinCAT OPC UA Client generate a machine-specific, self-signed certificate for authentication the first time they are started.

If you want to use such an encrypted communication connection in your environment, you need to establish a trust relationship between OPC UA Server and OPC UA Client.


Trustlist
UA-Client
UA-Server
Trustlist
Create secure session request
Client Public Key
Client cert trusted?
Create secure session response
Server Public Key
yes
Server cert trusted?
yes
Further secure communication request
Further secure communication request

Trust settings on the server

If you want to authenticate one or more OPC UA Clients to the OPC UA Server via certificates, the OPC UA Server must trust the public keys of the clients. This can be done via the file system, for example. The server manages the trust settings for certificates via the file system.

- Trusted certificates: %InstallDir%\Server\PKI\CA\trusted\certs
- Rejected certificates: %InstallDir%\Server\PKI\CA\rejected\certs

By moving client certificates between these directories, the trust settings can be adjusted accordingly.

Reading a client certificate

A simple way of accessing the client certificate is described below. To do this, you establish a connection to the UA Server using a secure endpoint (for example, Basic128Rsa15/Sign&Encrypt) without first copying the client certificate to the UA Server. This connection is naturally rejected by the UA Server, since it does not trust the UA Client at this point in time. In this case, the TwinCAT OPC UA Client would return the error 0xE4DD0102, for example. However, after rejecting the connection request, the UA Server stores a copy of the client certificate in the above-mentioned "Rejected" directory. The name of the certificate corresponds to the "Thumbprint" of the certificate and can thus be clearly assigned to the client certificate. You can now move this file directly to the above-mentioned "Trusted" directory so that the UA Server can trust the UA Client and accept the connection for all other connection requests.

Announcing OPC UA Servers to the OPC UA Client

Depending on the OPC UA Client employed, different steps may need to be taken so that the OPC UA Client trusts the OPC UA Server. Typically, for client applications with a graphical user interface, a warning message is displayed the first time you connect to the server, whereby the server certificate can then be classified as trustworthy.

The following instruction is therefore only valid for the TwinCAT OPC UA Client.

The public key of the OPC UA Server is located in the following directory as a DER file: *%InstallDir%\Server \PKI\CA\own\certs*

In the case of the TwinCAT OPC UA Client, copy the file into the corresponding "Trusted" directory: *%InstallDir%\Client\PKI\CA\certs*

4.3 Client PLCopen

4.3.1 Overview

The TwinCAT OPC UA Client offers several options for communicating directly with one or more OPC UA Servers from the control logic. On the one hand, an OPC UA interface is available directly from the PLC, in which a connection to an OPC UA Server can be initiated via PLCopen standardized function blocks. On the other there is an OPC UA Client I/O device that offers a simple mapping-based interface.

PLC function blocks

The following table provides an overview of the functionalities offered:

Functionality	Description	Relevant function blocks
Connect/Disconnect	Establishment and disconnection of connections to OPC UA Servers.	UA_Connect UA_Disconnect
Polling (Read/Write)	Reading and writing of variables in the UA namespace in the polling mode. Contains no subscriptions.	UA_Connect UA_Disconnect UA_Read UA_Write UA_GetNamespaceIndex UA_NodeGetHandle UA_NodeReleaseHandle
Method Call	Execution of methods in the UA namespace.	UA_Connect UA_Disconnect UA_MethodGetHandle UA_MethodReleaseHandle UA_MethodCall
Security	Establishment of an encrypted connection to an OPC UA Server.	UA_Connect UA_Disconnect

The interfaces of each function block are described in the section PLC API [▸ 218].

I/O device

Further information on the TwinCAT OPC UA Client I/O device can be found in the section I/O > Quick start [▸ 135].

4.3.2 Supported data types

The OPC UA Client enables direct access to OPC UA Servers from the real-time logic. To read and write data, the data types must be assigned to both environments (mapping). This assignment is described below.

Basic data types

The assignment of the basic data types is described in PLCopen OPC UA Information Model for IEC 61131-3.

OPC UA data type	PLC data type
Boolean	BOOL
SByte	SINT
Byte	USINT
Int16	INT
Int32	DINT
String	STRING
USint	BYTE
Float	REAL
Double	LREAL
UInt16	UINT
UInt32	UDINT
Int64	LINT
UInt64	ULINT
DateTime	DT

Derived data types

OPC UA defines basic data types. Other data types are derived from these.

On the client side, access to all UA data types is only possible with PLC data types. Data types derived from the basic data types are also supported from TcOpcUaClient version 2.1.0.36 or higher.

Currently supported non-base data types on the client side are:

- Counter (use UDINT in PLC)

Access to arrays

When creating a node handle, the system checks the possibilities to access the node. Further checks are performed during reading and writing.

To access array nodes and input and output parameters of method calls, certain conditions must be fulfilled.

- For accessing nodes: The array dimension and data length must match the data provided when reading and writing.
- Reading strings: If just one string exceeds the specified length for PLC strings, the read process fails.
- Only array dimensions up to three levels are supported.

4.3.3 Best practice

4.3.3.1 How to determine communication parameters

In general a graphic OPC UA Client is used to determine the attributes of a node or methods that have to be used together with the PLC function blocks, e.g.:

- Node Identifier [▶ 151]

- Node namespace index and corresponding namespace URI [▶ 151]
- Node Data Type [▶ 152]
- Node ID and Object Node ID methods [▶ 153]

The following documentation uses the generic OPC UA Client UA Expert as an example. This client can be purchased from the Unified Automation website: www.unified-automation.com.

Node Identifier

A general task consists of reading or writing variables that are generally referred to as nodes in the context of OPC UA.

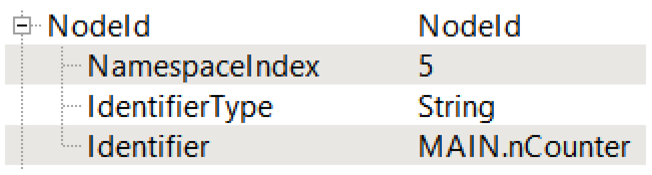
Nodes can be marked with the following three attributes:

- NamespaceIndex: Namespace in which the node is located, e.g. the PLC runtime
- Identifier: Unique identifier of the node within its namespace
- IdentifierType: Type of node: String, Guid and Numeric

These attributes represent the so-called NodeID – the representation of a node on an OPC UA Server – and are required by some function blocks.

With the help of UA Expert you can simply determine the attributes of a node by establishing a connection to the OPC UA Server and browsing to the desired node. The attributes are then visible in the Attributes panel.

Sample:

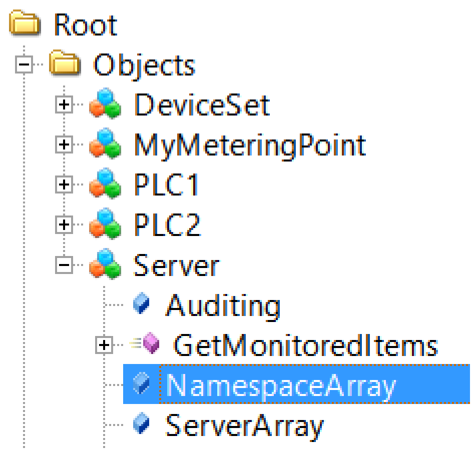


NodeId	NodeId
NamespaceIndex	5
IdentifierType	String
Identifier	MAIN.nCounter

Node NamespaceIndex and corresponding NamespaceURI

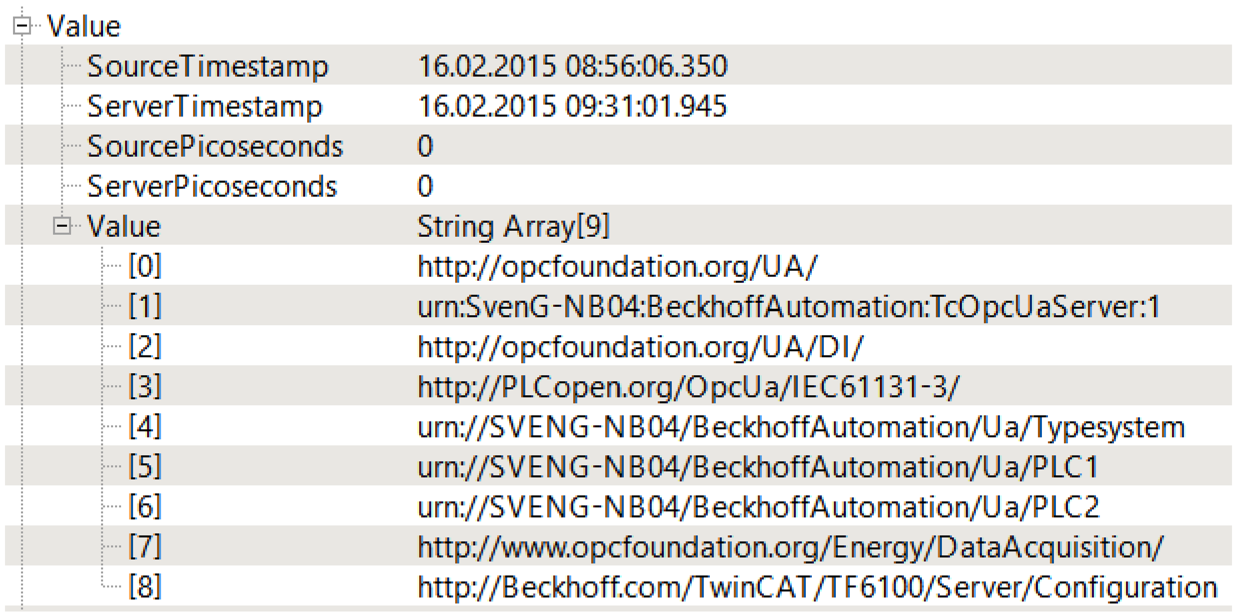
According to the OPC UA specification, the namespace index (as shown above) can be a dynamically generated value. Therefore, OPC UA Clients must always use the corresponding namespace URI to resolve the NamespaceIndex before a node handle is detected.

Use the UA_GetNamespaceIndex [▶ 220] function block to obtain the NamespaceIndex for a NamespaceURI. The NamespaceURI required for this can be determined with the help of UA Expert by establishing a connection to the OPC UA Server and browsing to the NamespaceArray node.



This node contains information about all namespaces registered on the OPC UA Server. The corresponding NamespaceURIs are visible in the Attributes panel.

Sample:



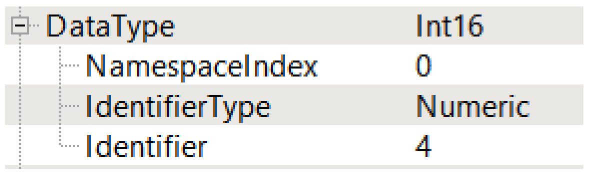
Value	
SourceTimestamp	16.02.2015 08:56:06.350
ServerTimestamp	16.02.2015 09:31:01.945
SourcePicoseconds	0
ServerPicoseconds	0
Value	String Array[9]
[0]	http://opcfoundation.org/UA/
[1]	urn:SvenG-NB04:BeckhoffAutomation:TcOpcUaServer:1
[2]	http://opcfoundation.org/UA/DI/
[3]	http://PLCopen.org/OpcUa/IEC61131-3/
[4]	urn://SVENG-NB04/BeckhoffAutomation/Ua/Typesystem
[5]	urn://SVENG-NB04/BeckhoffAutomation/Ua/PLC1
[6]	urn://SVENG-NB04/BeckhoffAutomation/Ua/PLC2
[7]	http://www.opcfoundation.org/Energy/DataAcquisition/
[8]	http://Beckhoff.com/TwinCAT/TF6100/Server/Configuration

A NodeID in which the NamespaceIndex is 5 is shown in the sample in the section Node Identifier [▶ 151]. According to the NamespaceArray shown in the figure, the corresponding NamespaceURI is urn://SVENG-NB04/BeckhoffAutomation/Ua/PLC1. This URI can now be used for the function block UA_GetNamespaceIndex. The OPC UA Server ensures that the URI always remains the same, even after a restart. As the NamespaceIndex shown can change, however, the NamespaceURI should always be used in combination with the UA_GetNamespaceIndex function block for later use with other function blocks, e.g. UA_Read [▶ 230], UA_Write [▶ 232], to resolve the correct NamespaceIndex.

Node DataType

The data type of a node is required in order to see which PLC data type needs to be used in order to assign a read value or write it to a node. With the help of UA Expert you can simply determine the data type of a node by establishing a connection to the OPC UA Server and browsing to the desired node. The data type is then visible in the Attributes panel.

Sample:



DataType	Int16
NamespaceIndex	0
IdentifierType	Numeric
Identifier	4

In this case the data type (DataType) is "Int16". This must be assigned to an equivalent data type in the PLC, e.g. "INT".

The PLCopen IEC61131 - to - OPC UA specification describes the defined data type mapping. The following table is an excerpt from this specification:

IEC61131 elementary data types	OPC UA built-in data types
BOOL	Boolean
SINT	SByte
USINT	Byte
INT	Int16
UINT	UInt16
DINT	Int32
UDINT	UInt32
LINT	Int64
ULINT	UInt64

IEC61131 elementary data types	OPC UA built-in data types
BYTE	Byte
WORD	UInt16
DWORD	UInt32
LWORD	UInt64
REAL	Float
LREAL	Double
STRING	String
CHAR	Byte
WSTRING	String
WCHAR	UInt16
DT DATE_AND_TIME	DateTime
DATE	DateTime
TOD TIME_OF_DAY	DateTime
TIME	Double

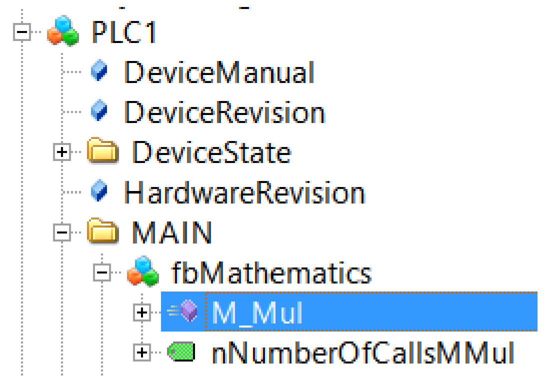
Method NodeID and Object NodeID

When calling methods from the OPC UA namespace, two identifiers are required if the method handle is captured using the function block UA_MethodGetHandle [▸ 224]:

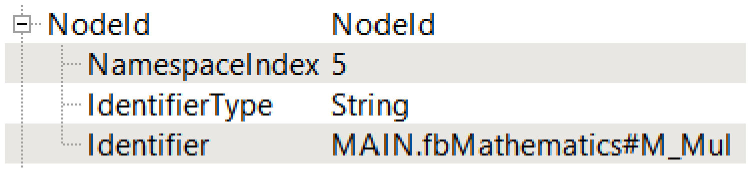
- ObjectNodeID: Identifies the UA object that contains the method
- MethodNodeID: Identifies the method itself

With the help of UA Expert you can simply determine both NodeIDs by establishing a connection to the OPC UA Server and browsing to the desired method or the desired UA object that contains the method.

Sample – M_Mul method:

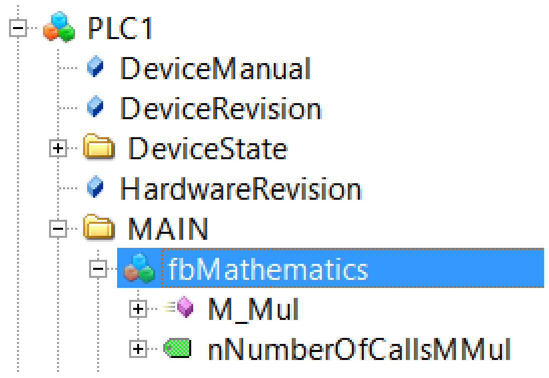


The method identifier is then visible in the Attributes panel.

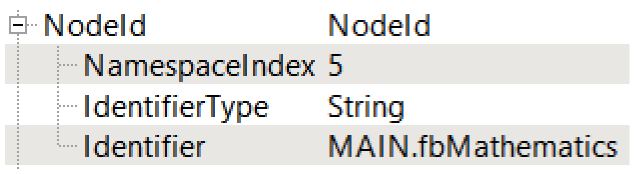


NodeId	NodeId
NamespaceIndex	5
IdentifierType	String
Identifier	MAIN.fbMathematics#M_Mul

Sample – fbMathematics object:



The object identifier is then visible in the Attributes panel.



NodeId	NodeId
NamespaceIndex	5
IdentifierType	String
Identifier	MAIN.fbMathematics

4.3.3.2 How to establish a connection

The following section describes how you use the TcX_PLCopen_OpcUa function block to establish a connection to a local or remote OPC UA Server. This connection can then be used to call other functions, such as read or write nodes, or call methods.

This section contains the following topics:

- Overview [▸ 154]
- Schematic workflow [▸ 154]
- General notes [▸ 155]
- Code snippet [▸ 155]

Overview

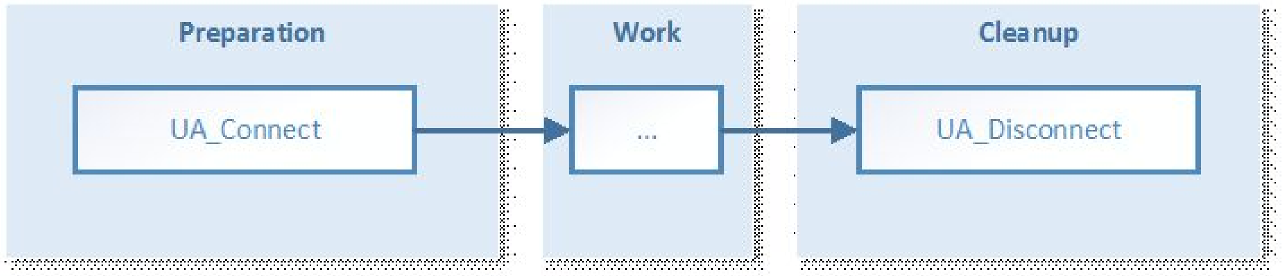
The following function blocks are required to establish a connection to an OPC UA Server and to interrupt the session later: UA_Connect [▸ 218], UA_Disconnect [▸ 219].

First read the section How to determine communication parameters [▸ 150] to better understand certain UA functionalities (e.g. how to determine node identifiers).

Schematic workflow

The schematic workflow of each TwinCAT OPC UA Client can be categorized into three different phases: Preparation, Work and Cleanup.

The use case described in this section can be visualized as follows:



General notes

The UA_Connect function block requires the following information in order to be able to establish a connection to a local or remote OPC UA Server:

- Server URL
- Session Connect Information

The server URL basically consists of a prefix, a hostname and a port. The prefix describes the OPC UA transport protocol that should be used for the connection, e.g."opc.tcp://" for a binary TCP connection (default). The host name or IP address part describes the address information of the OPC UA target server, e.g. "192.168.1.1" or "CX-12345". The port number is the target port of the OPC UA Server, e.g. "4840". Overall the server URL may then look like this: opc.tcp://CX-12345:4840.

Code snippet

Declaration:

```
(* Declarations for UA_Connect *)
fbUA_Connect : UA_Connect;
SessionConnectInfo : ST_UASessionConnectInfo;
nConnectionHdl : DWORD;

(* Declarations for UA_Disconnect *)
fbUA_Disconnect : UA_Disconnect;

(* Declarations for state machine and output handling *)
iState : INT;
bDone : BOOL;
bBusy : BOOL;
bError : BOOL;
nErrorID : DWORD;
```

Implementation:

```
CASE iState OF

  0:
      bError := FALSE;
      nErrorID := 0;
      SessionConnectInfo.tConnectTimeout := T#1M;
      SessionConnectInfo.tSessionTimeout := T#1M;
      SessionConnectInfo.sApplicationName := '';
      SessionConnectInfo.sApplicationUri := '';
      SessionConnectInfo.eSecurityMode := eUASecurityMsgMode_None;
      SessionConnectInfo.eSecurityPolicyUri := eUASecurityPolicy_None;
      SessionConnectInfo.eTransportProfileUri := eUATransportProfileUri_UATcp;
      stNodeAddInfo.nIndexRangeCount := nIndexRangeCount;
      stNodeAddInfo.stIndexRange := stIndexRange;
      iState := iState + 1;

  1:
    fbUA_Connect(
      Execute := TRUE,
      ServerURL := ‘opc.tcp://192.168.1.1:4840’,
      SessionConnectInfo := SessionConnectInfo,
      Timeout := T#5S,
      ConnectionHdl => nConnectionHdl);
    IF NOT fbUA_Connect.Busy THEN
      fbUA_Connect(Execute := FALSE);
      IF NOT fbUA_Connect.Error THEN
        iState := iState + 1;
      ELSE
        bError := TRUE;
        nErrorID := fbUA_Connect.ErrorID;
        nConnectionHdl := 0;
        iState := 0;
      END_IF
    END_IF

  2:
    fbUA_Disconnect(
      Execute := TRUE,
      ConnectionHdl := nConnectionHdl);

    IF NOT fbUA_Disconnect.Busy THEN
      fbUA_Disconnect(Execute := FALSE);
      IF NOT fbUA_Disconnect.Error THEN
        iState := 0;
      ELSE
        bError := TRUE;
        nErrorID := fbUA_Disconnect.ErrorID;
        iState := 0;
        nConnectionHdl := 0;
      END_IF
    END_IF

END_CASE
```

4.3.3.3 How to read nodes

The following section describes how you use the TcX_PLCopen_OpcUa function block to read out an OPC UA node from a local or remote OPC UA Server.

This section contains the following topics:

- Overview [▸ 156]
- Schematic workflow [▸ 156]
- General notes [▸ 156]
- Code snippet [▸ 157]

Overview

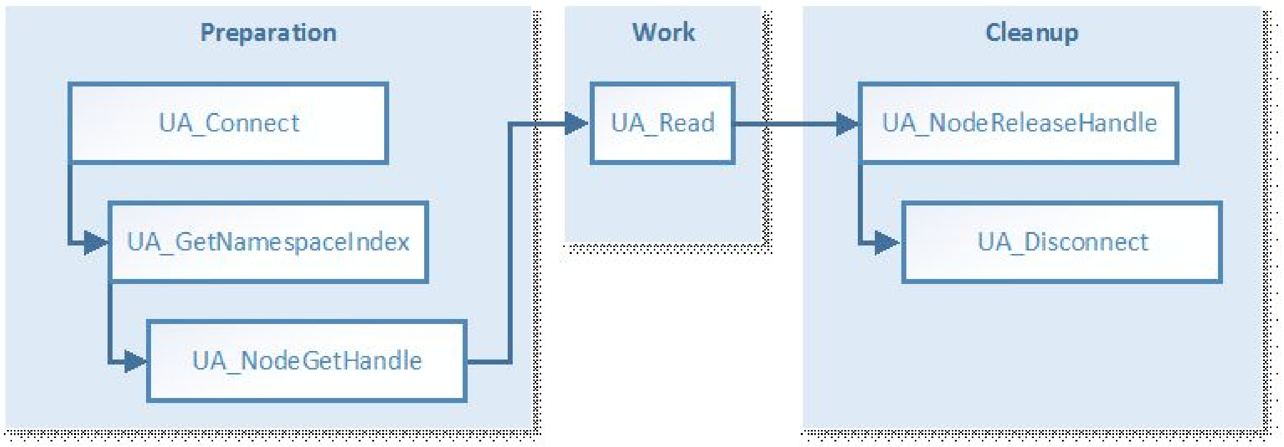
The following function blocks are required to establish a connection to an OPC UA Server, to read UA nodes and to interrupt the session later: UA_Connect [▸ 218], UA_GetNamespaceIndex [▸ 220], UA_NodeGetHandle [▸ 226], UA_Read [▸ 230], UA_NodeReleaseHandle [▸ 228], UA_Disconnect [▸ 219].

First of all, read the section How to determine communication parameters [▸ 150] so as to be able to understand certain UA functions better (e.g. how NodeIdentifiers can be determined) as well as the section How to establish a connection [▸ 154].

Schematic workflow

The schematic workflow of each TwinCAT OPC UA Client can be categorized into three different phases: Preparation, Work and Cleanup.

The use case described in this section can be visualized as follows:



General notes

- The UA_Connect function block requires the following information to establish a connection to a local or remote OPC UA Server (see also How to establish a connection [▸ 154]):
  - Server URL
  - Session Connect Information

- The UA_GetNamespaceIndex function block requires a connection handle (from UA_Connect) and a NamespaceURI for resolution in a NamespaceIndex, which will be used later by UA_NodeGetHandle to obtain a node handle (see also How to determine communication parameters [▶ 150]).
- The UA_NodeGetHandle function block requires a connection handle (from UA_Connect) and the NodeID (from a ST_UANodeID) in order to obtain a node handle (see also How to determine communication parameters [▶ 150]).
- The UA_Read function block requires a connection handle (from UA_Connect), a node handle (from UA_NodeGetHandle) and a pointer to the target variable (where the read value is to be saved). Make sure that the target variable has the correct data type (see How to determine communication parameters [▶ 150]).
- The UA_NodeReleaseHandle function block requires a connection handle (from UA_Connect) and a node handle (from UA_NodeGetHandle).

Code snippet

Declaration:

```
(* Declarations for UA_GetNamespaceIndex *)
fbUA_GetNamespaceIndex : UA_GetNamespaceIndex;
nNamespaceIndex : UINT;

(* Declarations for UA_NodeGetHandle *)
fbUA_NodeGetHandle : UA_NodeGetHandle;
NodeID : ST_UANodeID;
nNodeHdl : DWORD;

(* Declarations for UA_Read *)
fbUA_Read : UA_Read;
stIndexRange : ARRAY [1..nMaxIndexRange] OF ST_UAIndexRange;
nIndexRangeCount : UINT;
stNodeAddInfo : ST_UANodeAdditionalInfo;
sNodeIdentifier : STRING(MAX_STRING_LENGTH) := 'MAIN.nCounter';
nReadData : INT;
cbDataRead : UDINT;

(* Declarations for UA_NodeReleaseHandle *)
fbUA_NodeReleaseHandle : UA_NodeReleaseHandle;
```

Implementation:

```
CASE iState OF
  0:
    [...]

  2: (* GetNS Index *)
    fbUA_GetNamespaceIndex(
      Execute := TRUE,
      ConnectionHdl := nConnectionHdl,
      NamespaceUri := sNamespaceUri,
      NamespaceIndex => nNamespaceIndex
      );
    IF NOT fbUA_GetNamespaceIndex.Busy THEN
      fbUA_GetNamespaceIndex(Execute := FALSE);
      IF NOT fbUA_GetNamespaceIndex.Error THEN
        iState := iState + 1;
      ELSE
        bError := TRUE;
        nErrorID := fbUA_GetNamespaceIndex.ErrorID;
        iState := 6;
      END_IF
    END_IF

  3: (* UA_NodeGetHandle *)
    NodeID.eIdentifierType := eUAIdentifierType_String;
    NodeID.nNamespaceIndex := nNamespaceIndex;
    NodeID.sIdentifier := sNodeIdentifier;
    fbUA_NodeGetHandle(
      Execute := TRUE,
      ConnectionHdl := nConnectionHdl,
      NodeID := NodeID,
      NodeHdl => nNodeHdl);
    IF NOT fbUA_NodeGetHandle.Busy THEN
      fbUA_NodeGetHandle(Execute := FALSE);
      IF NOT fbUA_NodeGetHandle.Error THEN
```

```
        iState := iState + 1;
      ELSE
        bError := TRUE;
        nErrorID := fbUA_NodeGetHandle.ErrorID;
        iState := 6;
      END_IF
    END_IF

  4: (* UA_Read *)
    fbUA_Read(
      Execute := TRUE,
      ConnectionHdl := nConnectionHdl,
      NodeHdl := nNodeHdl,
      cbData := SIZEOF(nReadData),
      stNodeAddInfo := stNodeAddInfo,
      pVariable := ADR(nReadData));
    IF NOT fbUA_Read.Busy THEN
      fbUA_Read( Execute := FALSE, cbData_R => cbDataRead);
      IF NOT fbUA_Read.Error THEN
        iState := iState + 1;
      ELSE
        bError := TRUE;
        nErrorID := fbUA_Read.ErrorID;
        iState := 6;
      END_IF
    END_IF

  5: (* Release Node Handle *)
    fbUA_NodeReleaseHandle(
      Execute := TRUE,
      ConnectionHdl := nConnectionHdl,
      NodeHdl := nNodeHdl);
    IF NOT fbUA_NodeReleaseHandle.Busy THEN
      fbUA_NodeReleaseHandle(Execute := FALSE);
      IF NOT fbUA_NodeReleaseHandle.Error THEN
        iState := iState + 1;
      ELSE
        bError := TRUE;
        nErrorID := fbUA_NodeReleaseHandle.ErrorID;
        iState := 6;
      END_IF
    END_IF

  6:
    [...]

END_CASE
```

4.3.3.4 How to write nodes

The following section describes how you use the TcX_PLCopen_OpcUa function block to write values in an OPC UA node from a local or remote OPC UA Server.

This section contains the following topics:

- Overview [▸ 158]
- Schematic workflow [▸ 159]
- General notes [▸ 159]
- Code snippet [▸ 159]

Overview

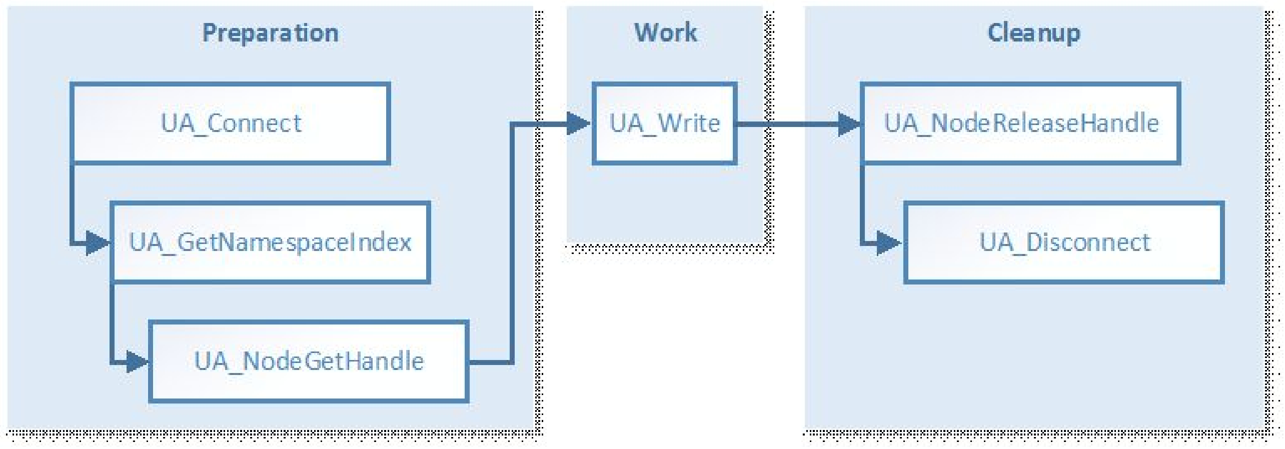
The following function blocks are required to establish a connection to an OPC UA Server, write UA nodes and subsequently interrupt the session: UA_Connect [▸ 218], UA_GetNamespaceIndex [▸ 220], UA_NodeGetHandle [▸ 226], UA_Write [▸ 232], UA_NodeReleaseHandle [▸ 228], UA_Disconnect [▸ 219].

First of all, read the section How to determine communication parameters [▸ 150] so as to be able to understand certain UA functions better (e.g. how NodeIdentifiers can be determined) as well as the section How to establish a connection [▸ 154].

Schematic workflow

The schematic workflow of each TwinCAT OPC UA Client can be categorized into three different phases: Preparation, Work and Cleanup.

The use case described in this section can be visualized as follows:



General notes

- The UA_Connect function block requires the following information to establish a connection to a local or remote OPC UA Server (see also How to establish a connection [▸ 154]):
    - Server URL
    - Session Connect Information
- The UA_GetNamespaceIndex function block requires a connection handle (from UA_Connect) and a NamespaceURI for resolution in a NamespaceIndex, which will be used later by UA_NodeGetHandle to obtain a node handle (see also How to determine communication parameters [▸ 150]).
- The UA_NodeGetHandle function block requires a connection handle (from UA_Connect) and the NodeID (from a ST_UANodeID) in order to obtain a node handle (see also How to determine communication parameters [▸ 150]).
- The UA_Write function block requires a connection handle (from UA_Connect), a node handle (from UA_NodeGetHandle) and a pointer to a variable containing the value that is to be written. Make sure that the target variable has the correct data type (see How to determine communication parameters [▸ 150]).
- The UA_NodeReleaseHandle function block requires a connection handle (from UA_Connect) and a node handle (from UA_NodeGetHandle).

Code snippet

Declaration:

```
(* Declarations for UA_GetNamespaceIndex *)
fbUA_GetNamespaceIndex : UA_GetNamespaceIndex;
nNamespaceIndex : UINT;

(* Declarations for UA_NodeGetHandle *)
fbUA_NodeGetHandle : UA_NodeGetHandle;
NodeID : ST_UANodeID;
nNodeHdl : DWORD;

(* Declarations for UA_Write *)
fbUA_Write : UA_Write;
stIndexRange : ARRAY [1..nMaxIndexRange] OF ST_UAIndexRange;
nIndexRangeCount : UINT;
stNodeAddInfo : ST_UANodeAdditionalInfo;
sNodeIdentifier: STRING(MAX_STRING_LENGTH) := 'MAIN.nNumber';
nWriteData: INT := 42;

(* Declarations for UA_NodeReleaseHandle *)
fbUA_NodeReleaseHandle : UA_NodeReleaseHandle;
```

Implementation:

```
CASE iState OF
  0:
    [...]

  2: (* GetNS Index *)
    fbUA_GetNamespaceIndex(
      Execute := TRUE,
      ConnectionHdl := nConnectionHdl,
      NamespaceUri := sNamespaceUri,
      NamespaceIndex => nNamespaceIndex
      );
    IF NOT fbUA_GetNamespaceIndex.Busy THEN
      fbUA_GetNamespaceIndex(Execute := FALSE);
      IF NOT fbUA_GetNamespaceIndex.Error THEN
        iState := iState + 1;
      ELSE
        bError := TRUE;
        nErrorID := fbUA_GetNamespaceIndex.ErrorID;
        iState := 6;
      END_IF
    END_IF

  3: (* UA_NodeGetHandle *)
    NodeID.eIdentifierType := eUAIdentifierType_String;
    NodeID.nNamespaceIndex := nNamespaceIndex;
    NodeID.sIdentifier := sNodeIdentifier;
    fbUA_NodeGetHandle(
      Execute := TRUE,
      ConnectionHdl := nConnectionHdl,
      NodeID := NodeID,
      NodeHdl => nNodeHdl);
    IF NOT fbUA_NodeGetHandle.Busy THEN
      fbUA_NodeGetHandle(Execute := FALSE);
      IF NOT fbUA_NodeGetHandle.Error THEN
        iState := iState + 1;
      ELSE
        bError := TRUE;
        nErrorID := fbUA_NodeGetHandle.ErrorID;
        iState := 6;
      END_IF
    END_IF

  4: (* UA_Write *)
    fbUA_Write(
      Execute := TRUE,
      ConnectionHdl := nConnectionHdl,
      NodeHdl := nNodeHdl,
      stNodeAddInfo := stNodeAddInfo,
      cbData := SIZEOF(nWriteData),
      pVariable := ADR(nWriteData));
    IF NOT fbUA_Write.Busy THEN
      fbUA_Write(
        Execute := FALSE,
        pVariable := ADR(nWriteData));
      IF NOT fbUA_Write.Error THEN
        iState := iState + 1;
      ELSE
        bError := TRUE;
        nErrorID := fbUA_Write.ErrorID;
        iState := 6;
      END_IF
    END_IF

  5: (* Release Node Handle *)
    fbUA_NodeReleaseHandle(
      Execute := TRUE,
      ConnectionHdl := nConnectionHdl,
      NodeHdl := nNodeHdl);
    IF NOT fbUA_NodeReleaseHandle.Busy THEN
      fbUA_NodeReleaseHandle(Execute := FALSE);
      IF NOT fbUA_NodeReleaseHandle.Error THEN
        iState := iState + 1;
      ELSE
        bError := TRUE;
        nErrorID := fbUA_NodeReleaseHandle.ErrorID;
        iState := 6;
      END_IF
    END_IF
  6:
    [...]
END_CASE
```

4.3.3.5 How to call methods

The following section describes how you use the TcX_PLCopen_OpcUa function block to call methods on a local or remote OPC UA Server.

This section contains the following topics:

- Overview [▸ 161]
- Schematic workflow [▸ 161]
- General notes [▸ 161]
- Code snippet [▸ 162]

Overview

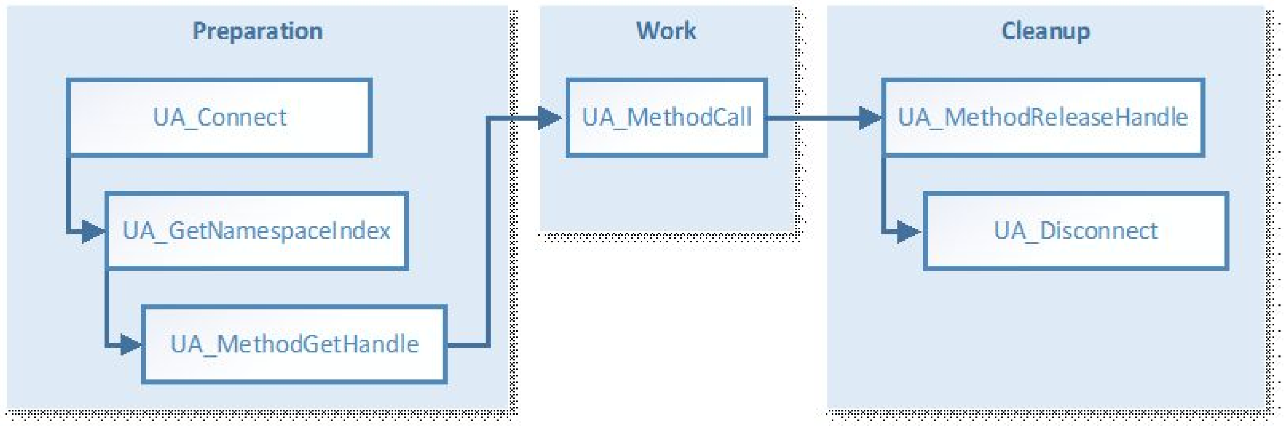
The following function blocks are required to connect to an OPC UA Server, call UA methods, and subsequently interrupt the session: UA_Connect [▸ 218], UA_GetNamespaceIndex [▸ 220], UA_MethodGetHandle [▸ 224], UA_MethodCall [▸ 223], UA_MethodReleaseHandle [▸ 225], UA_Disconnect [▸ 219].

First of all, read the section How to determine communication parameters [▸ 150] so as to be able to understand certain UA functions better (e.g. how MethodIdentifier can be determined) as well as the section How to establish a connection [▸ 154].

Schematic workflow

The schematic workflow of each TwinCAT OPC UA Client can be categorized into three different phases: Preparation, Work and Cleanup.

The use case described in this section can be visualized as follows:



General notes

- The UA_Connect function block requires the following information to establish a connection to a local or remote OPC UA Server (see also How to establish a connection [▸ 154]):
  - Server URL
  - Session Connect Information
- The UA_GetNamespaceIndex function block requires a connection handle (from UA_Connect) and a NamespaceURI for resolution in a NamespaceIndex, which will be used later by UA_NodeGetHandle to obtain a node handle (see also How to determine communication parameters [▸ 150]).

- The UA_MethodGetHandle function block requires a connection handle (from UA_Connect), an ObjectNodeID and a MethodNodeID in order to obtain a method handle (see also How to determine communication parameters [▶ 150]).
- The UA_MethodCall function block requires a connection handle (from UA_Connect), a method handle (from UA_MethodGetHandle) and information about the input and output arguments of the method that is to be called. Information about the input arguments is represented by the input parameters pInputArgInfo and pInputArgData of UA_MethodCall. Information about the output parameters is represented by the pOutputArgInfo and pOutputArgData input parameters of UA_MethodCall. The input parameter pOutputArgInfoAndData then represents a pointer to a structure containing the results of the method call, including all output parameters. In the following code snippet the pInputArgInfo and pInputArgData parameters are calculated and created in the M_Init method.
- The UA_NodeReleaseHandle function block requires a connection handle (from UA_Connect) and a method handle (from UA_MethodGetHandle).

Code snippet

M_Init initialization method of the function block containing the UA method call

```
MEMSET(ADR(nInputData),0,SIZEOF(nInputData));
nArg := 1;

(********** Input parameter 1 **********)
InputArguments[nArg].DataType := eUAType_Int16;
InputArguments[nArg].ValueRank := -1; (* Scalar = -1 or Array *)
InputArguments[nArg].ArrayDimensions[1] := 0; (* Number of Dimension in case its an array *)
InputArguments[nArg].nLenData := SIZEOF(numberIn1); (* Length if its a STRING *)
IF nOffset + SIZEOF(numberIn1) > nInputArgSize THEN
  bInputDataError := TRUE;
  RETURN;
ELSE
  MEMCPY(ADR(nInputData)+nOffset,ADR(numberIn1),SIZEOF(numberIn1)); (* VALUE in BYTES FORM *)
  nOffset := nOffset + SIZEOF(numberIn1);
END_IF
nArg := nArg + 1;

(********** Input parameter 2 **********)
InputArguments[nArg].DataType := eUAType_Int16;
InputArguments[nArg].ValueRank := -1; (* Scalar = -1 or Array *)
InputArguments[nArg].ArrayDimensions[1] := 0; (* Number of Dimension in case its an array *)
InputArguments[nArg].nLenData := SIZEOF(numberIn2); (* Length if its a STRING *)
IF nOffset + SIZEOF(numberIn2) > nInputArgSize THEN
  bInputDataError := TRUE;
  RETURN;
ELSE
  MEMCPY(ADR(nInputData)+nOffset,ADR(numberIn2),SIZEOF(numberIn2));(* VALUE in BYTES FORM *)
  nOffset := nOffset + SIZEOF(numberIn2);
END_IF

cbWriteData := nOffset;
```

Declaration:

```
(* Declarations for UA_GetNamespaceIndex *)
fbUA_GetNamespaceIndex : UA_GetNamespaceIndex;
nNamespaceIndex : UINT;

(* Declarations for UA_MethodGetHandle *)
fbUA_MethodGetHandle: UA_MethodGetHandle;
ObjectNodeID: ST_UANodeID;
MethodNodeID: ST_UANodeID;
nMethodHdl: DWORD;

(* Declarations for UA_MethodCall *)
fbUA_MethodCall: UA_MethodCall;
sObjectNodeIdIdentifier : STRING(MAX_STRING_LENGTH) := 'MAIN.fbMathematics';
sMethodNodeIdIdentifier : STRING(MAX_STRING_LENGTH) := 'MAIN.fbMathematics#M_Mul';
nAdrWriteData: PVOID;
numberIn1: INT := 42; // change according to input value and data type
numberIn2: INT := 42; // change according to input value and data type
numberOutPro: DINT; // result (output parameter of M_Mul())
cbWriteData: UDINT; // calculated automatically by M_Init()
InputArguments: ARRAY[1..2] OF ST_UAMethodArgInfo; // change according to input parameters
stOutputArgInfo: ARRAY[1..1] OF ST_UAMethodArgInfo; // change according to output parameters
stOutputArgInfoAndData: ST_OutputArgInfoAndData;
nInputData: ARRAY[1..4] OF BYTE; // numberIn1(INT16)(2) + numberIn2(INT16)(2)
```

```
nOffset: UDINT; // calculated by M_Init()
nArg: INT; // used by M_Init()

(* Declarations for UA_MethodReleaseHandle *)
fbUA_MethodReleaseHandle: UA_MethodReleaseHandle;
```

Implementation:

```
CASE iState OF
  0:
    [...]

  2: (* GetNS Index *)
    fbUA_GetNamespaceIndex(
      Execute := TRUE,
      ConnectionHdl := nConnectionHdl,
      NamespaceUri := sNamespaceUri,
      NamespaceIndex => nNamespaceIndex);
    IF NOT fbUA_GetNamespaceIndex.Busy THEN
      fbUA_GetNamespaceIndex(Execute := FALSE);
      IF NOT fbUA_GetNamespaceIndex.Error THEN
        iState := iState + 1;
      ELSE
        bError := TRUE;
        nErrorID := fbUA_GetNamespaceIndex.ErrorID;
        iState := 7;
      END_IF
    END_IF

  3: (* Get Method Handle *)
    ObjectNodeID.eIdentifierType := eUAIdentifierType_String;
    ObjectNodeID.nNamespaceIndex := nNamespaceIndex;
    ObjectNodeID.sIdentifier := sObjectNodeIdIdentifier;
    MethodNodeID.eIdentifierType := eUAIdentifierType_String;
    MethodNodeID.nNamespaceIndex := nNamespaceIndex;
    MethodNodeID.sIdentifier := sMethodNodeIdIdentifier;

    M_Init();

    IF bInputDataError = FALSE THEN
      iState := iState + 1;
    ELSE
      bBusy := FALSE;
      bError := TRUE;
      nErrorID := 16#70A; //out of memory
    END_IF

  4: (* Method Get Handle *)
    fbUA_MethodGetHandle(
      Execute := TRUE,
      ConnectionHdl := nConnectionHdl,
      ObjectNodeID := ObjectNodeID,
      MethodNodeID := MethodNodeID,
      MethodHdl => nMethodHdl);
    IF NOT fbUA_MethodGetHandle.Busy THEN
      fbUA_MethodGetHandle(Execute := FALSE);
      IF NOT fbUA_MethodGetHandle.Error THEN
        iState := iState + 1;
      ELSE
        bError := TRUE;
        nErrorID := fbUA_MethodGetHandle.ErrorID;
        iState := 6;
      END_IF
    END_IF

  5: (* Method Call *)
    stOutputArgInfo[1].nLenData := SIZEOF(stOutputArgInfoAndData.pro);
    fbUA_MethodCall(
      Execute := TRUE,
      ConnectionHdl := nConnectionHdl,
      MethodHdl := nMethodHdl,
      nNumberOfInputArguments := nNumberOfInputArguments,
      pInputArgInfo := ADR(InputArguments),
      cbInputArgInfo := SIZEOF(InputArguments),
      pInputArgData := ADR(nInputData),
      cbInputArgData := cbWriteData,
      pInputWriteData := 0,
      cbInputWriteData := 0,
      nNumberOfOutputArguments := nNumberOfOutputArguments,
      pOutputArgInfo := ADR(stOutputArgInfo),
      cbOutputArgInfo := SIZEOF(stOutputArgInfo),
      pOutputArgInfoAndData := ADR(stOutputArgInfoAndData),
      cbOutputArgInfoAndData := SIZEOF(stOutputArgInfoAndData));
    IF NOT fbUA_MethodCall.Busy THEN
      fbUA_MethodCall(Execute := FALSE);
      IF NOT fbUA_MethodCall.Error THEN
        iState := iState + 1;
        numberOutPro := stOutputArgInfoAndData.pro;
      ELSE
        bError := TRUE;
        nErrorID := fbUA_MethodCall.ErrorID;
        iState := 6;
      END_IF
    END_IF

  6: (* Release Method Handle *)
    fbUA_MethodReleaseHandle(
      Execute := TRUE,
      ConnectionHdl := nConnectionHdl,
      MethodHdl := nMethodHdl);
    IF NOT fbUA_MethodReleaseHandle.Busy THEN
      fbUA_MethodReleaseHandle(Execute := FALSE);
      bBusy := FALSE;
      IF NOT fbUA_MethodReleaseHandle.Error THEN
        iState := 7;
      ELSE
        bError := TRUE;
        nErrorID := fbUA_MethodReleaseHandle.ErrorID;
        iState := 7;
      END_IF
    END_IF

  7:
    [...]

END_CASE
```

4.3.4 Security

4.3.4.1 Overview

Security was a central requirement in the development of OPC UA. It is addressed in various areas:

- Encryption
- Integrity
- Authentication

The confidentiality of the exchanged information is secured by the encryption of the exchanged messages. Modern cryptographic algorithms are used for this. In order to be able to cope with future security requirements as well, even stronger and more modern algorithms can subsequently be added to an application without changing the protocol.

Different security levels can be selected according to the requirements of the respective application. In some areas it is sufficient to sign the messages in order to prevent changes being made by third parties, while additional encryption of the messages is necessary in other cases where the data must also not be read by third parties.

TF6100 OPC UA offers the following security levels (security endpoints) that clients can connect to:

- None: No security
- Sign: signed messages
- Sign & Encrypt: signed and encrypted messages

The signing of messages prevents a third party from changing the contents of a message. This prevents, for example, a write statement to open a switch being falsified by a third party and the switch being closed instead.

OPC UA applications identify themselves via so-called software and application instance certificates. With the aid of software certificates it is possible to grant certain client applications extended access to the information on an OPC UA Server, for example for the engineering of an OPC UA Server. Application instance certificates can be used to ensure that an OPC UA Server communicates only with preconfigured clients. A client can ensure by means of the server's application instance certificate that it is speaking to the correct server (similar to the certificates of a Web browser). The taking into account of these certificates is optional, i.e. an OPC UA server can also grant the same access to each client, depending on the user rights.

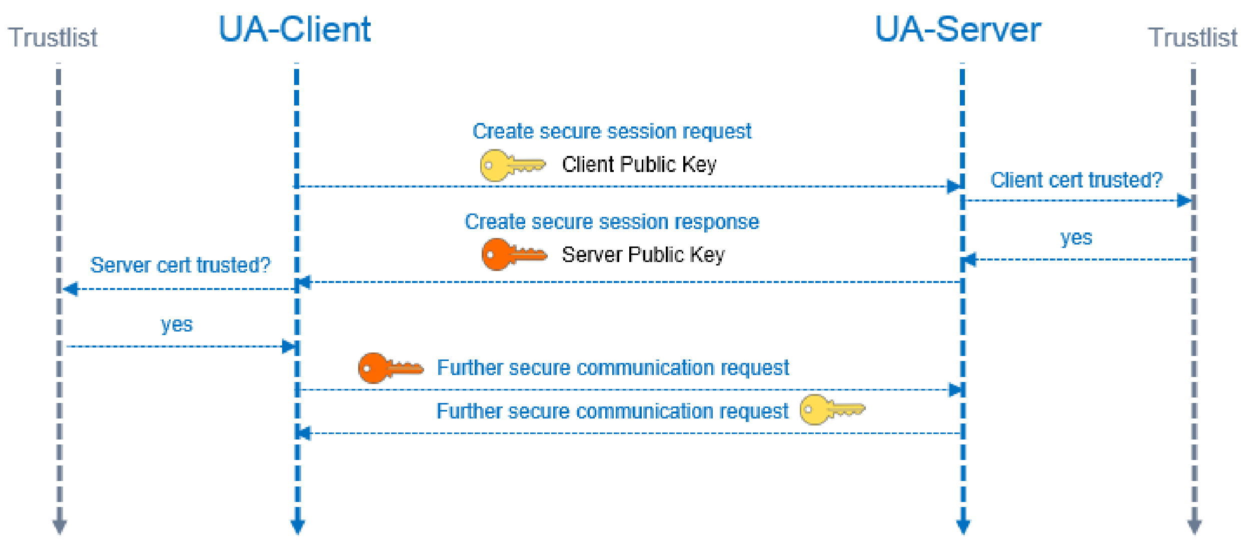
The TwinCAT OPC UA Client and TwinCAT OPC UA Server generate a self-signed certificate during the first startup process. This certificate consists of a private key and a public key. The private key is saved in the *\PKI\CA\private* directory and the corresponding public key in the *\PKI\CA\certs* directory. Any OPC UA Client wishing to establish a secure connection with the Server via one of the security endpoints (Sign, Sign&Encrypt) must know the public key of the OPC UA Server. Conversely the OPC UA Server must know the Client's public key. This so-called key exchange is described in the following sections:

- Authentication [▶ 122]
- Certificate exchange [▶ 165]
- Access rights [▶ 123]

4.3.4.2 Certificate exchange

The communication between an OPC UA Client and an OPC UA Server can optionally be secured by communication with a secure endpoint. By default, both the TwinCAT OPC UA Server and the TwinCAT OPC UA Client generate a machine-specific, self-signed certificate for authentication the first time they are started.

If you want to use such an encrypted communication connection in your environment, you need to establish a trust relationship between OPC UA Server and OPC UA Client.


Trustlist
UA-Client
UA-Server
Trustlist
Create secure session request
Client Public Key
Client cert trusted?
Create secure session response
yes
Server Public Key
Server cert trusted?
yes
Further secure communication request
Further secure communication request

Trust settings on the server

If you want to authenticate one or more OPC UA Clients to the OPC UA Server via certificates, the OPC UA Server must trust the public keys of the clients. This can be done via the file system, for example. The server manages the trust settings for certificates via the file system.

- Trusted certificates: %InstallDir%\Server\PKI\CA\trusted\certs
- Rejected certificates: %InstallDir%\Server\PKI\CA\rejected\certs

By moving client certificates between these directories, the trust settings can be adjusted accordingly.

Reading a client certificate

A simple way of accessing the client certificate is described below. To do this, you establish a connection to the UA Server using a secure endpoint (for example, Basic128Rsa15/Sign&Encrypt) without first copying the client certificate to the UA Server. This connection is naturally rejected by the UA Server, since it does not trust the UA Client at this point in time. In this case, the TwinCAT OPC UA Client would return the error 0xE4DD0102, for example. However, after rejecting the connection request, the UA Server stores a copy of the client certificate in the above-mentioned "Rejected" directory. The name of the certificate corresponds to the "Thumbprint" of the certificate and can thus be clearly assigned to the client certificate. You can now move this file directly to the above-mentioned "Trusted" directory so that the UA Server can trust the UA Client and accept the connection for all other connection requests.

Announcing OPC UA Servers to the OPC UA Client

Depending on the OPC UA Client employed, different steps may need to be taken so that the OPC UA Client trusts the OPC UA Server. Typically, for client applications with a graphical user interface, a warning message is displayed the first time you connect to the server, whereby the server certificate can then be classified as trustworthy.

The following instruction is therefore only valid for the TwinCAT OPC UA Client.

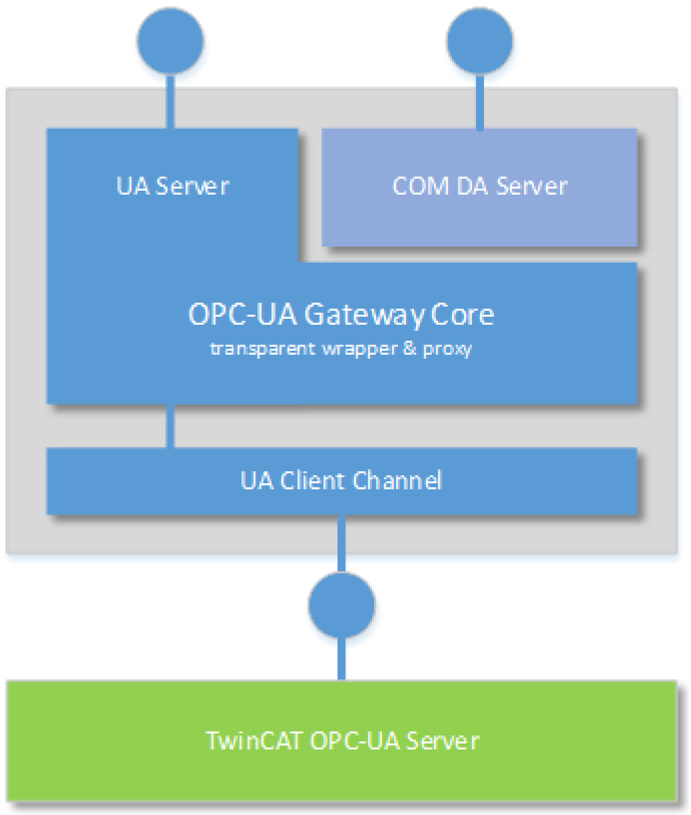
The public key of the OPC UA Server is located in the following directory as a DER file: *%InstallDir%\Server \PKI\CA\own\certs*

In the case of the TwinCAT OPC UA Client, copy the file into the corresponding "Trusted" directory: *%InstallDir%\Client\PKI\CA\certs*

4.4 Gateway

4.4.1 Overview

The TwinCAT OPC UA Gateway is the latest addition to the TS6100/TF6100 software product. It not only includes a conventional OPC DA interface for connecting older OPC COM DA applications to the TwinCAT OPC UA Server and can therefore be regarded as the successor of the old TwinCAT OPC DA Server (TS6120/TF6120), it also offers an OPC UA interface for converting several basic TwinCAT OPC UA Servers to a central OPC UA Server.



4.4.2 Quick start

The TS6100/TF6100 setup automatically installs the TwinCAT OPC UA Gateway and registers the application as a Windows service. The setup automatically configures access to a TwinCAT OPC UA Server running on the same computer as the gateway.

If more than one OPC UA Server is added to the gateway, or if the server is running on a different computer, the standard configuration has to be modified. Use the Configurator [▸ 171] to configure these settings.

Configuration of the TwinCAT OPC UA Server

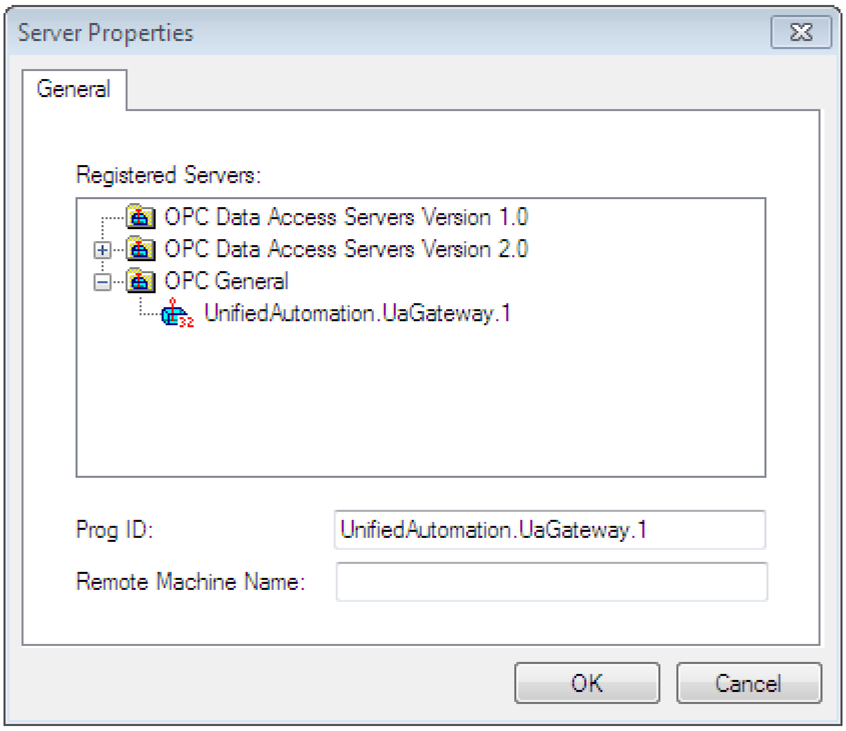
Check the configuration of the OPC UA Server and make sure that it is operating as expected before continuing.

For further information regarding the configuration of the OPC UA Server, read the Quick start [▸ 41] in the section "OPC UA Server".

Quick start – OPC COM DA

To connect an OPC COM DA Client to the gateway, start the client and establish a connection to the following ProgId:

```
UnifiedAutomation.UaGateway.1
```



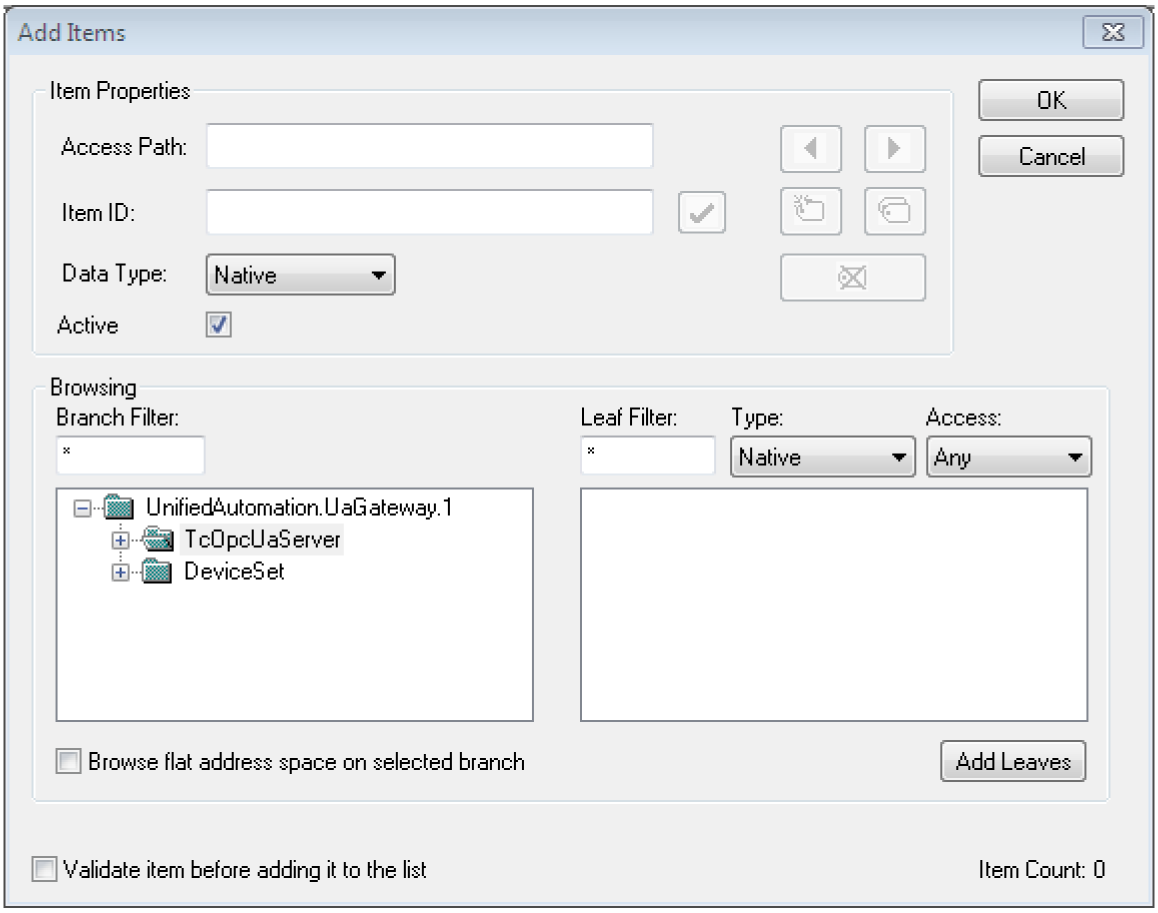
Registered Servers:

- OPC Data Access Servers Version 1.0
- OPC Data Access Servers Version 2.0
- OPC General
  - UnifiedAutomation.UaGateway.1

Prog ID: UnifiedAutomation.UaGateway.1

Remote Machine Name:

When browsing the gateway, one or more OPC UA Servers will be visible in the namespace of the gateway.



Access Path:

Item ID:

Data Type: Native

Active ☑

Branch Filter: *

Leaf Filter: *

Type: Native

Access: Any

- UnifiedAutomation.UaGateway.1
  - TcOpcUaServer
  - DeviceSet

☐ Browse flat address space on selected branch

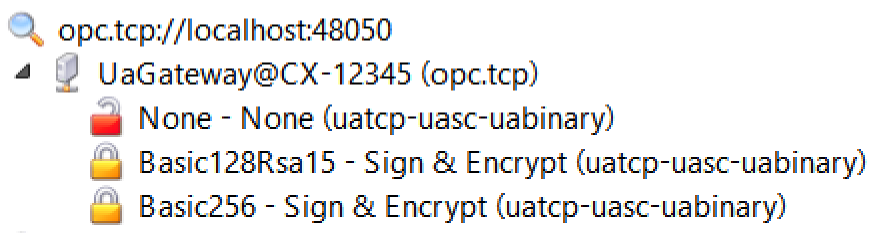
☐ Validate item before adding it to the list

Item Count: 0

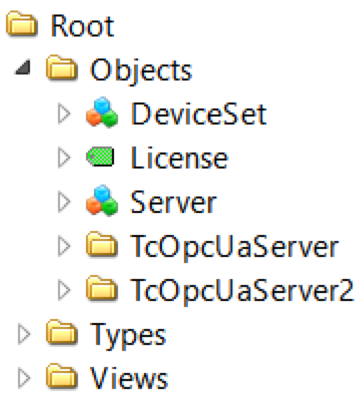
Quick start – OPC UA

The gateway not only offers an OPC COM DA interface, but also allows the aggregation of one or more OPC UA Servers. The gateway also opens an OPC UA interface for this purpose. The gateway can be accessed via the following OPC UA Server URL:

```
opc.tcp://[HostnameOrIpAddressOrLocalhost]:48050
```


opc.tcp://localhost:48050
UaGateway@CX-12345 (opc.tcp)
None - None (uatcp-uasc-uabinary)
Basic128Rsa15 - Sign & Encrypt (uatcp-uasc-uabinary)
Basic256 - Sign & Encrypt (uatcp-uasc-uabinary)

The namespace of the gateway then contains all underlying TwinCAT OPC UA Servers.


Root
Objects
DeviceSet
License
Server
TcOpcUaServer
TcOpcUaServer2
Types
Views

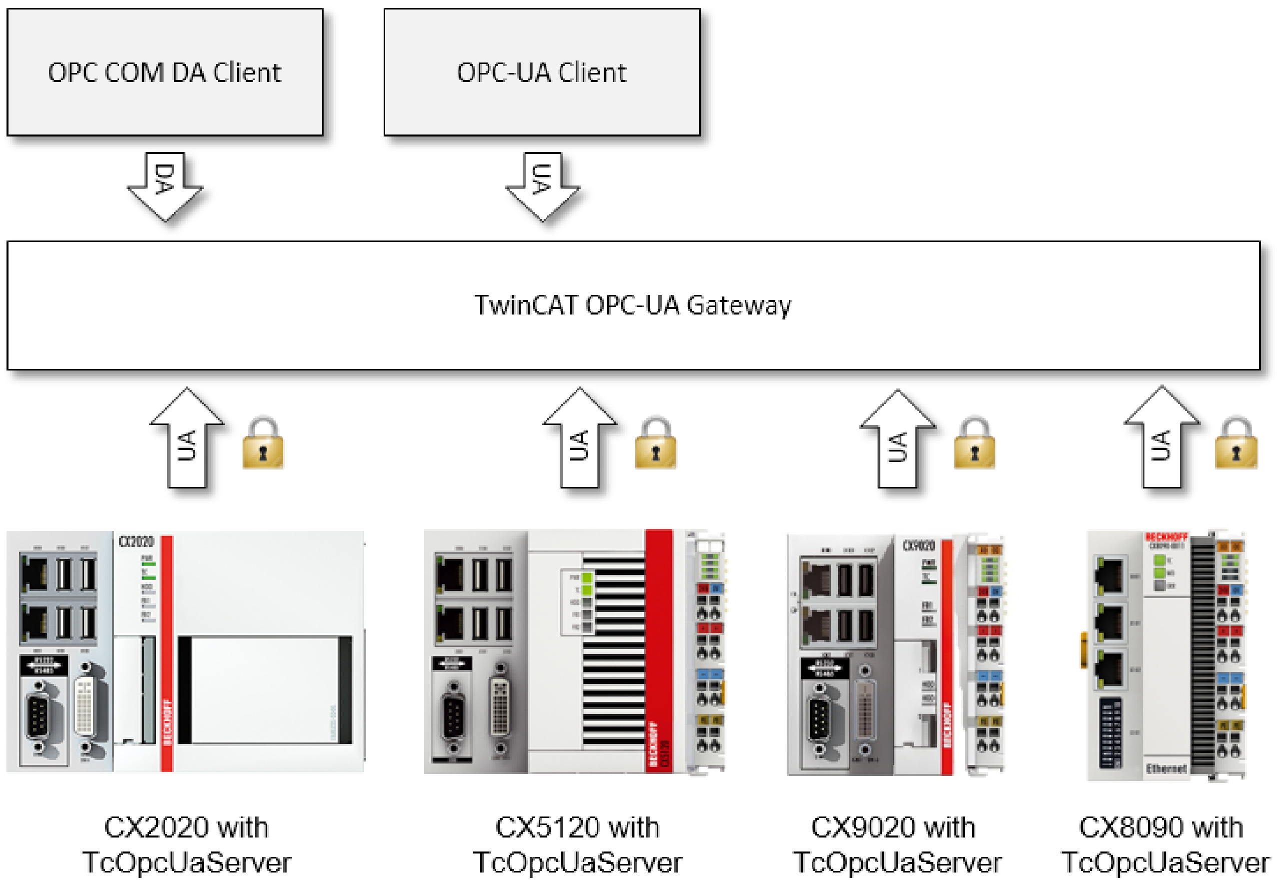
4.4.3 Licensing

The TwinCAT OPC UA Gateway is supplied free of charge. No further license purchase is required.

Note that the gateway component can only be used for connecting to TwinCAT OPC UA Servers. The software prevents connections to third-party UA Servers. The Unified Automation UA Gateway is recommended where the environment necessitates the connection with a UA Server from a third party. It can be purchased from http://www.unified-automation.com.

4.4.4 Scenarios

On account of the open and flexible PC-based automation technology from Beckhoff, the OPC UA Gateway can be operated and installed in different ways. The following section describes the various setup scenarios and explains the advantages and disadvantages of each configuration.


OPC COM DA Client
OPC-UA Client
DA
UA
TwinCAT OPC-UA Gateway
UA
UA
UA
UA
CX2020 with TcOpcUaServer
CX5120 with TcOpcUaServer
CX9020 with TcOpcUaServer
CX8090 with TcOpcUaServer

Gateway and UA Server on the same computer

In this scenario, the gateway and the UA Server are installed on the same computer. The gateway is configured with the standard settings in order to establish a connection with the local OPC UA Server with the following Server URL: opc.tcp://localhost:4840.

This scenario only works on non-Windows CE devices.

The TwinCAT OPC UA Server is also available for Windows CE, but the gateway is only available for "big" Windows platforms.

Gateway and UA Server on different computers

In this scenario, the gateway and the UA Server are installed on different computers. The gateway is configured for the establishment of a connection with the remote OPC UA Server by defining the latter's corresponding Server URL, e.g. opc.tcp://192.168.1.1:4840.

The OPC UA Server may be installed on a small embedded device (e.g. a CX8090 with Windows CE), while the gateway component is installed on a separate big Windows platform, e.g. a central server device.

Gateway connected to multiple UA Server devices

Please note that, regardless of the scenarios described above, it is also possible to connect other TwinCAT OPC UA Servers to the gateway. For example, the gateway can be configured to access the following servers:

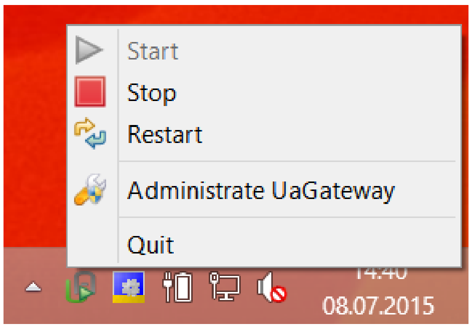
- the local TwinCAT OPC UA Server (e.g. opc.tcp://localhost:4840)
- a remote TwinCAT OPC UA Server (e.g. opc.tcp://192.168.1.1:4840)
- another remote TwinCAT OPC UA Server (e.g. opc.tcp://192.168.1.21:4841)
- ...

4.4.5 Configurator

4.4.5.1 Overview

The UA Gateway Administration Tool is a graphic user interface for the configuration of the gateway.

The tool is opened via the context menu of the gateway symbol in the Windows taskbar. After starting the administration tool via the command **Administrate UaGateway**, the graphic user interface appears.

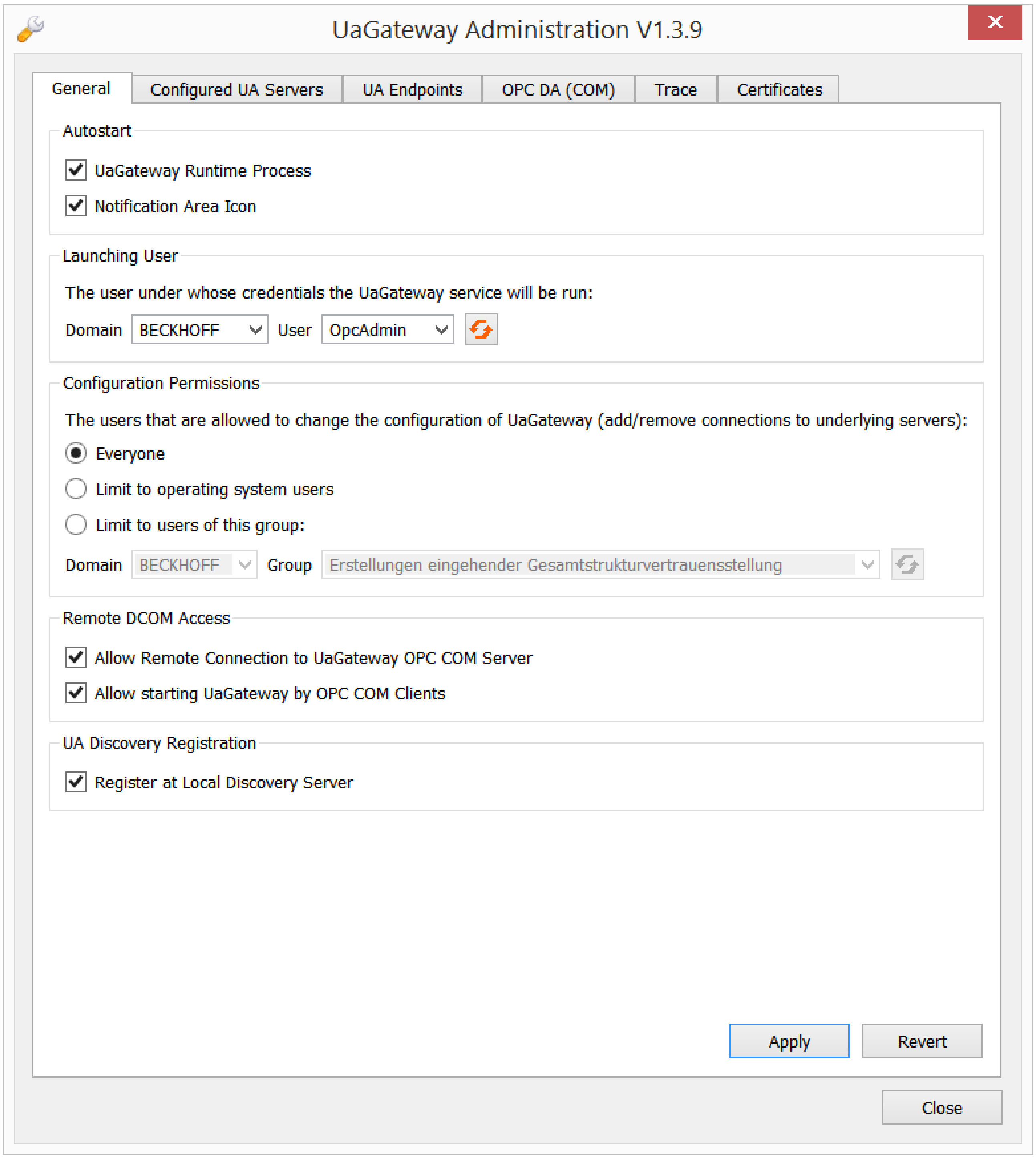


The interface offers several configuration options:

- General settings [▶ 171]
- Additional UA Servers [▶ 173]
- Additional endpoints [▶ 174]
- OPC COM DA settings [▶ 175]

4.4.5.2 General settings

The General tab displays general settings of the UA Gateway.


UaGateway Administration V1.3.9

General | Configured UA Servers | UA Endpoints | OPC DA (COM) | Trace | Certificates

Autostart

- [✓] UaGateway Runtime Process
- [✓] Notification Area Icon

Launching User

The user under whose credentials the UaGateway service will be run:

Domain BECKHOFF User OpcAdmin

Configuration Permissions

The users that are allowed to change the configuration of UaGateway (add/remove connections to underlying servers):

- [✓ Everyone]
- Limit to operating system users
- Limit to users of this group:

Domain BECKHOFF Group Erstellungen eingehender Gesamtstrukturvertrauensstellung

Remote DCOM Access

- [✓] Allow Remote Connection to UaGateway OPC COM Server
- [✓] Allow starting UaGateway by OPC COM Clients

UA Discovery Registration

- [✓] Register at Local Discovery Server

Autostart

In this area you can configure the autostart behavior of the UA Gateway.

Activate **UaGateway Runtime Process** to start the UA Gateway Service automatically when the computer is switched on.

Activate the **Notification Area Icon** to start the symbol of the notification area when a user logs on.

Launching User

The UA Gateway is executed as a Windows NT service. This service is assigned a specific user context so that COM/DCOM can be properly configured. The user you select is assigned to the UA Gateway service. In addition, the user is granted a LogOnAsService right (so he/she can start the service) and is added to a local user group ("UaGatewayUsers"). This group is added to the Access Control List (ACL) of the local machine. For proper COM/DCOM configuration you must add to this group all the users who are permitted to start and access the UA Gateway.

Configuration Permissions

It is possible to allow only certain users to change the configuration of the UA Gateway, i.e. to add or remove connections to basic servers. You can choose from the following settings:

Everyone	Any user (including users anonymously logged on to UA) who can contact the UA Gateway can change the configuration.
Limit to operating system users	Only local users and users within the same domain can change the configuration.
Limit to users of this group	Only users within a specific group to change the configuration. If not all available groups are displayed in the **Group** drop-down list (or a newly created group is missing), use the **Refresh** button to read this group again.

Remote DCOM Access

When **Allow Remote Connection to UaGateway OPC COM Server** is enabled, DCOM port 135 and the executable UA Gateway are added to the firewall exception list.

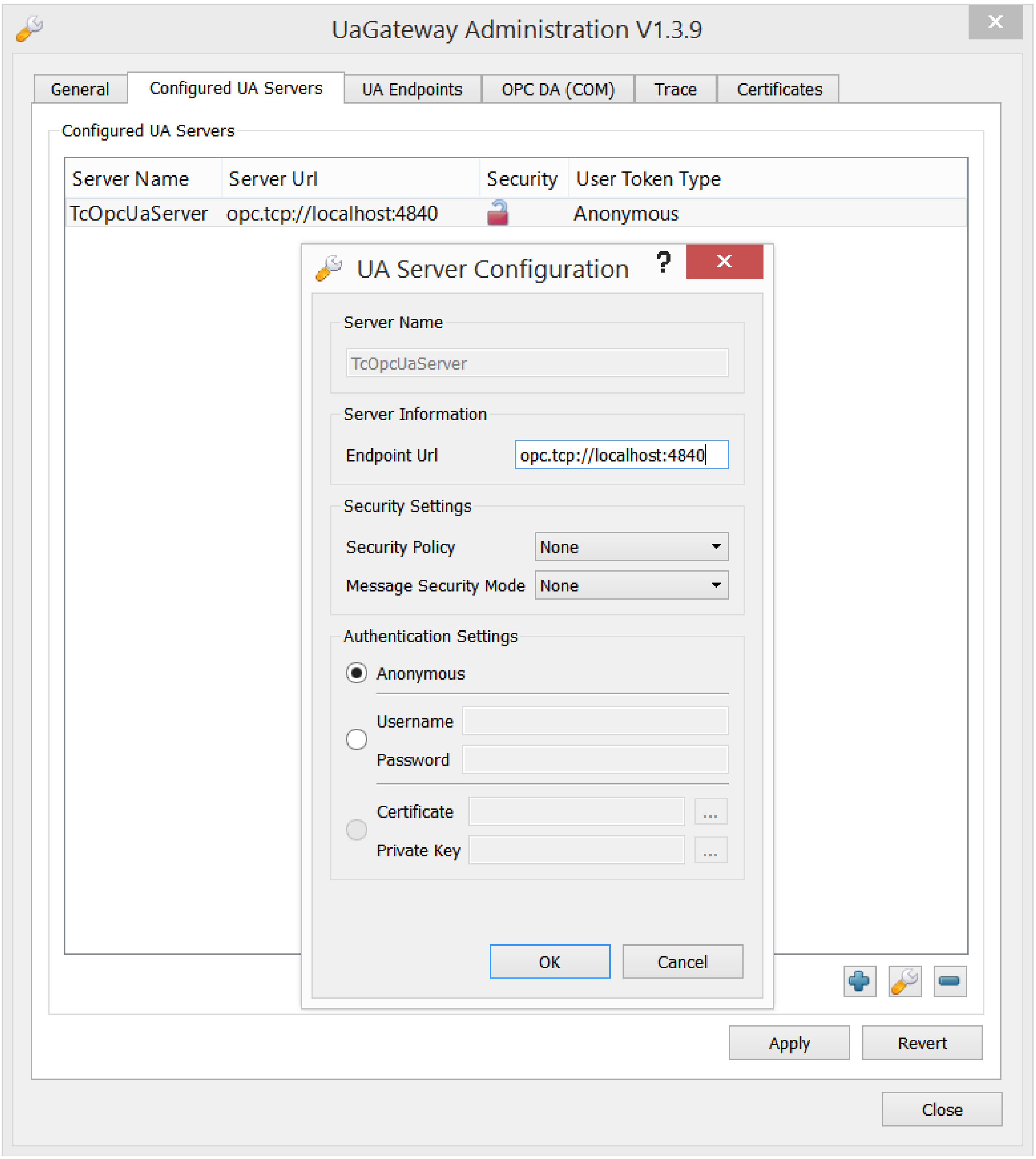
If **Allow starting UaGateway by DCOM Clients** is disabled, DCOM clients cannot start the UA Gateway. In this case, UA Gateway can still be started or stopped using the Notification Area Icon or the Start menu entries.

UA Discovery Registration (UA Local Discovery Server)

Activate **Register at Local Discovery Server** if the UA Gateway is to be registered with the OPC UA LDS (Local Discovery Server), if one is installed.

4.4.5.3 Additional UA Servers

The **Configured UA Servers** tab offers options for the configuration of the underlying OPC UA Servers. By default the gateway already establishes a connection with the local OPC UA Server (which is running on the same computer).


UaGateway Administration V1.3.9

General | Configured UA Servers | UA Endpoints | OPC DA (COM) | Trace | Certificates

Configured UA Servers

Server Name	Server Url	Security	User Token Type
TcOpcUaServer	opc.tcp://localhost:4840	🔓	Anonymous

UA Server Configuration

Server Name: TcOpcUaServer

Server Information
Endpoint Url: opc.tcp://localhost:4840

Security Settings
Security Policy: None
Message Security Mode: None

Authentication Settings
● Anonymous
○ Username / Password
○ Certificate / Private Key

OK | Cancel

Apply | Revert | Close

To configure or remove further OPC UA Servers from the configuration, click on the Plus and Minus buttons in the lower right-hand corner and then **Apply** to save the changes.

4.4.5.4 Additional endpoints

The **UA Endpoints** tab shows the settings for the UA endpoint configuration.

The UA endpoint is the connection information that a UA Client requires to connect to the gateway.

General

Use the checkboxes to specify the logon methods that a client can use to connect to your UA Gateway.

Endpoints

Here you can define all settings required for different UA endpoints. The endpoint is configured with default settings as standard. These represent a single UA endpoint offering two security options: None and Basic128RSsa15.

The None security option allows every UA Client to connect to the UA Gateway. This configuration is only recommended during commissioning and testing. In a production environment this configuration should be switched off.

The various configuration elements are described in the following sections.

Network configuration

Endpoint URL	This is the endpoint URL of the UA Gateway as seen in FindServers and GetEndpoint calls.
Protocol	This is the protocol used for this endpoint.
Host name/IP	This is the host name of the UA Gateway (it can also be the IP address of the PC running the UA Gateway).
Network adapter	This is the network adapter to be used for binding. The available options are:
All	Binding is to be applied to all IP addresses of the computer. The endpoint will be accessible via the given port on all IP addresses.
Network adapter	Select a network adapter and an IP address (below) to bind only to that address. The endpoint will only be accessible to clients that establish a connection with the selected IP address.
Local only	With this selection, the UA Gateway only establishes a binding with the loopback adapter. The endpoint can only be reached by clients running on the same machine as the UA Gateway.
Port	This is the TCP port of the endpoint (normally 48050).

Security

In this area you can configure the supported security settings of the endpoint. Select the checkboxes for the security options you want to apply to a specific endpoint. For options other than "None", the available message security mode(s) must be specified. Signing ensures that messages cannot be changed and that they are exchanged between applications that have established a connection. Encryption guarantees that no one can read the messages.

4.4.5.5 OPC COM DA settings

The **OPC DA (COM)** tab shows the settings for the configuration of the COM DA Server of the UA Gateway.

The following section describes how to configure the COM DA Server of the UA Gateway using the administration tool.

General

ItemIDs of the COM DA Server are formed from the URI namespace and the identifier of the variable node in the OPC UA address space. The namespace part can be omitted in the case of a single namespace.

In the **Default Name Space** drop-down field you can specify the default namespace with the namespace of a basic OPC server. The ItemIDs of this particular namespace can then be reached by specifying the identifier only, because the default namespace is automatically added internally when an element is accessed. This feature can be used to reconfigure all ItemIDs in the client that accesses the UA Gateway server, if the latter serves as a tunnel solution for a basic COM DA Server, while maintaining the ItemIDs of the original COM DA Server.

In the second drop-down field the **Timestamp Source** can be defined. The following options are available:

Internal	The time stamps are generated by the OPC COM DA Server.

SourceTimestamp	The SourceTimestamps are used as time stamps provided by the OPC COM DA Server.
ServerTimestamp	The ServerTimestamps are used as time stamps provided by the OPC COM DA Server.

Properties mapping from UA to COM DA

When connecting to the UA Gateway's OPC COM DA Server, all six standard properties (DataType, Value, Quality, TimeStamp, AccessRights and ScanRate) are automatically assigned. Underlying OPC Servers can provide further properties (e.g. user-defined properties, DI properties, etc.). These properties can be assigned to vendor-specific properties (PropertyID ≧ 5000) in the COM DA Server of the UA Gateway.

These vendor-specific PropertyIDs are automatically assigned when the properties are requested for the first time. This dialog allows you to change the assigned PropertyIDs or configure how the OPC UA properties in the UA Gateway address space are assigned to the vendor-specific COM DA properties. You have to define the property name on the UA side and the namespace of the property in the UA Gateway and assign it to the COM DA PropertyIDs. When connecting to the UA Gateway's COM DA server, you can navigate through the available properties (QueryAvailableProperties) of a single OPCItem and then you will be able to see the associated properties as they have been configured (in the range of vendor-specific PropertyIDs above 5000).

Press the **[+]** or **[-]** key respectively to add or remove a certain property. To change the contents of a particular field, double-click it and enter the required values. Double-clicking a value in the **UA Property NameSpace URI** column displays a drop-down menu where you can make a selection.

If you add a new property by pressing **[+]**, the values of the last entry are copied to the new line and the PropertyID is automatically incremented.

4.4.6 Migrating from Tx6120

One of the primary purposes of the UA Gateway is to provide a sustainable connectivity in order to replace the Tx6120 OPC DA supplement/function. Observe the following notes if you wish to migrate Tx6120 OPC DA to UA Gateway.

Standard configuration

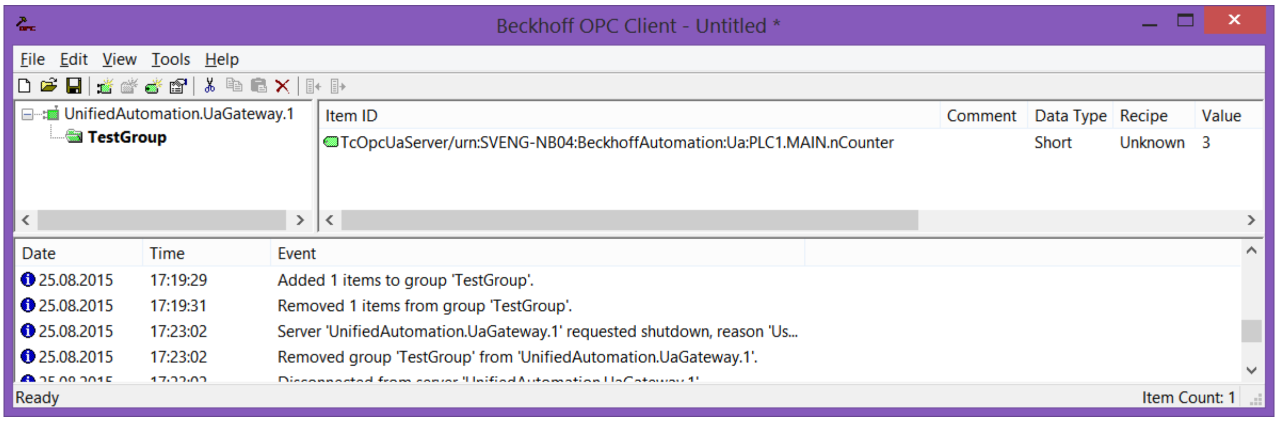
The standard configuration of the UA Gateway automatically establishes a connection with the local OPC UA Server and offers the OPC DA Clients an OPC DA interface. For a connection based on this standard configuration, the OPC DA clients must consider the following points:

- The default ProgID of the UA Gateway is "UnifiedAutomation.Gateway.1". The TwinCAT OPC DA Server uses a different ProgID ("Beckhoff.TwinCATOpcServerDA").
- The UA Gateway always uses a ProgID instead of multiple clones.
- The ItemIdentifier of an OPC symbol is generated differently in the UA Gateway. This behavior can be changed.

Changing the syntax of an ItemIdentifier

The syntax used by the UA Gateway for ItemIdentifier can be changed so that the latter corresponds more to the type of the TwinCAT OPC DA Server. By default, the UA Gateway uses a different syntax to that of the TwinCAT OPC DA Server when creating its identifiers.

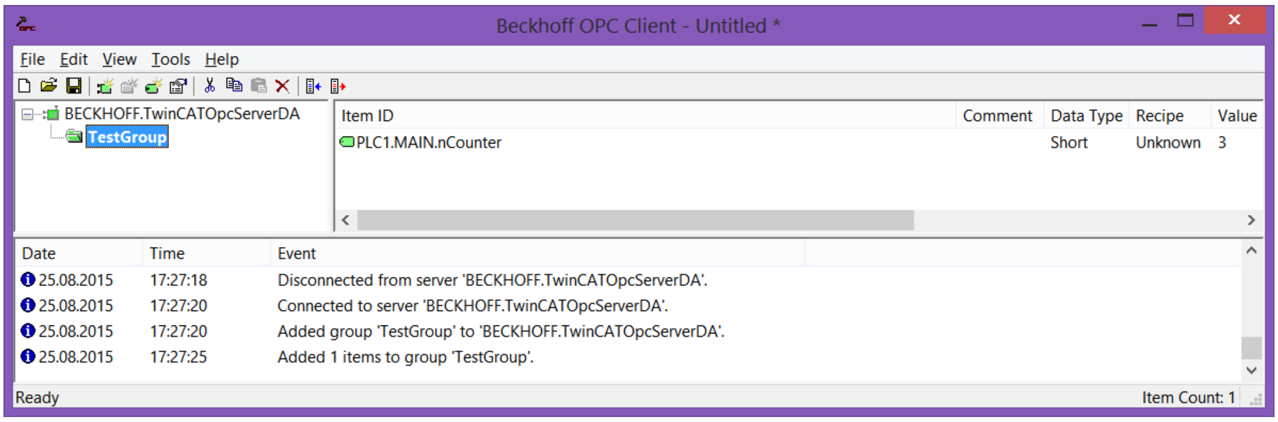
UA Gateway sample:


Beckhoff OPC Client - Untitled *
File Edit View Tools Help
UnifiedAutomation.UaGateway.1
TestGroup
Ready
Item Count: 1

Item ID	Comment	Data Type	Recipe	Value
TcOpcUaServer/urn:SVENG-NB04:BeckhoffAutomation:Ua:PLC1.MAIN.nCounter		Short	Unknown	3

Date	Time	Event
25.08.2015	17:19:29	Added 1 items to group 'TestGroup'.
25.08.2015	17:19:31	Removed 1 items from group 'TestGroup'.
25.08.2015	17:23:02	Server 'UnifiedAutomation.UaGateway.1' requested shutdown, reason 'Us...
25.08.2015	17:23:02	Removed group 'TestGroup' from 'UnifiedAutomation.UaGateway.1'.
25.08.2015	17:23:02	Disconnected from server 'UnifiedAutomation.UaGateway.1'.

Sample TwinCAT OPC DA Server:


Beckhoff OPC Client - Untitled *
File Edit View Tools Help
BECKHOFF.TwinCATOpcServerDA
TestGroup
Ready
Item Count: 1

Item ID	Comment	Data Type	Recipe	Value
PLC1.MAIN.nCounter		Short	Unknown	3

Date	Time	Event
25.08.2015	17:27:18	Disconnected from server 'BECKHOFF.TwinCATOpcServerDA'.
25.08.2015	17:27:20	Connected to server 'BECKHOFF.TwinCATOpcServerDA'.
25.08.2015	17:27:20	Added group 'TestGroup' to 'BECKHOFF.TwinCATOpcServerDA'.
25.08.2015	17:27:25	Added 1 items to group 'TestGroup'.

The UA Gateway uses a prefix so that the underlying OPC UA Client from which the variable originates can be clearly identified.

The following steps are required to configure the UA Gateway so that it forms its identifiers in roughly the same way as the TwinCAT OPC DA Server. The functionality has been implemented to simplify the migration process.

1. Open the UA Gateway configuration file
   C:\Program Files (x86)\UnifiedAutomation\UaGateway\bin\uagateway.config.xml
2. Look for the following XML tags in the XML file:

```
<OpcServerConfig>
  <ComDaServerConfig>
    <ComDaNamespaceUseAlias>false</ComDaNamespaceUseAlias>
  </ComDaServerConfig>
</OpcServerConfig>
```

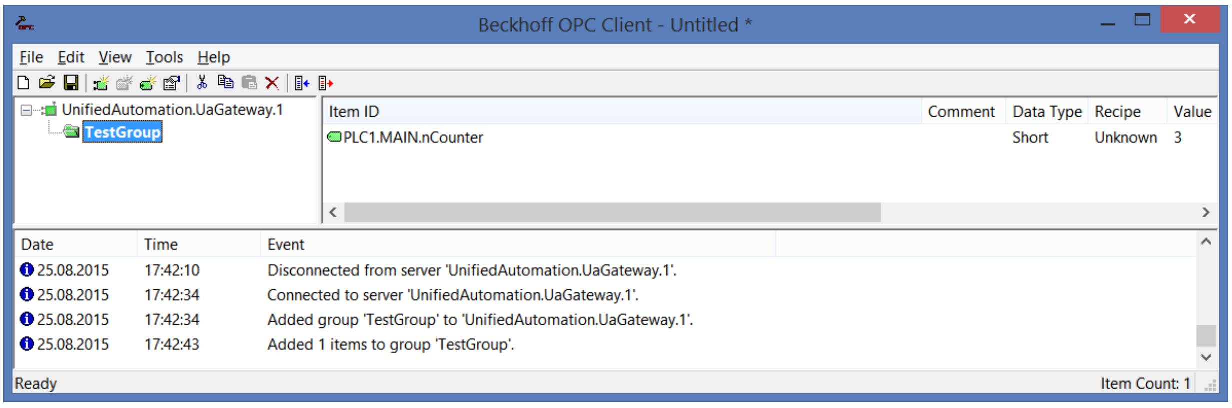
3. If the XML tag ComDaNamespaceUseAlias is set to "true", user-defined prefixes can be specified. To do this, look for the following XML tag in the same XML file:

```
<OpcServerConfig>
  <UaServerConfig>
    <ConfiguredNamespaces>
      ...
    </ConfiguredNamespaces
  </UaServerConfig>
</OpcServerConfig>
```

4. In this XML structure, identify the TwinCAT OPC UA Server namespace. By default, it should read as follows:

```
<OpcServerConfig>
  <UaServerConfig>
    <ConfiguredNamespaces>
      ...
      <Namespace>
        <Index>...</Index>
        <Uri>TcOpcUaServer/urn:Hostname:BeckhoffAutomation:Ua:PLC1</Uri>
        <AllowRenameUri>false</AllowRenameUri>
        <UniqueId>TcOpcUaServer#TcOpcUaServer/urn:Hostname:BeckhoffAutomation:Ua:PLC1</UniqueId>
        <ComAlias>...</ComAlias>
      </Namespace>
      ...
    </ConfiguredNamespaces
  </UaServerConfig>
</OpcServerConfig>
```

5. On your computer, the placeholder "..." may look different. Set <ComAlias> to your preferred prefix, for example "PLC1". The identifiers are then created with the prefix "PLC1".


Beckhoff OPC Client - Untitled *
File Edit View Tools Help
UnifiedAutomation.UaGateway.1
TestGroup
Ready
Item Count: 1

Item ID	Comment	Data Type	Recipe	Value
PLC1.MAIN.nCounter		Short	Unknown	3

Date	Time	Event
25.08.2015	17:42:10	Disconnected from server 'UnifiedAutomation.UaGateway.1'.
25.08.2015	17:42:34	Connected to server 'UnifiedAutomation.UaGateway.1'.
25.08.2015	17:42:34	Added group 'TestGroup' to 'UnifiedAutomation.UaGateway.1'.
25.08.2015	17:42:43	Added 1 items to group 'TestGroup'.

4.4.7 Security

4.4.7.1 Overview

Security was a central requirement in the development of OPC UA. It is addressed in various areas:

- Encryption
- Integrity
- Authentication

The confidentiality of the exchanged information is secured by the encryption of the exchanged messages. Modern cryptographic algorithms are used for this. In order to be able to cope with future security requirements as well, even stronger and more modern algorithms can subsequently be added to an application without changing the protocol.

Different security levels can be selected according to the requirements of the respective application. In some areas it is sufficient to sign the messages in order to prevent changes being made by third parties, while additional encryption of the messages is necessary in other cases where the data must also not be read by third parties.

TF6100 OPC UA offers the following security levels (security endpoints) that clients can connect to:

- None: No security
- Sign: signed messages
- Sign & Encrypt: signed and encrypted messages

The signing of messages prevents a third party from changing the contents of a message. This prevents, for example, a write statement to open a switch being falsified by a third party and the switch being closed instead.

OPC UA applications identify themselves via so-called software and application instance certificates. With the aid of software certificates it is possible to grant certain client applications extended access to the information on an OPC UA Server, for example for the engineering of an OPC UA Server. Application instance certificates can be used to ensure that an OPC UA Server communicates only with preconfigured clients. A client can ensure by means of the server's application instance certificate that it is speaking to the correct server (similar to the certificates of a Web browser). The taking into account of these certificates is optional, i.e. an OPC UA server can also grant the same access to each client, depending on the user rights.

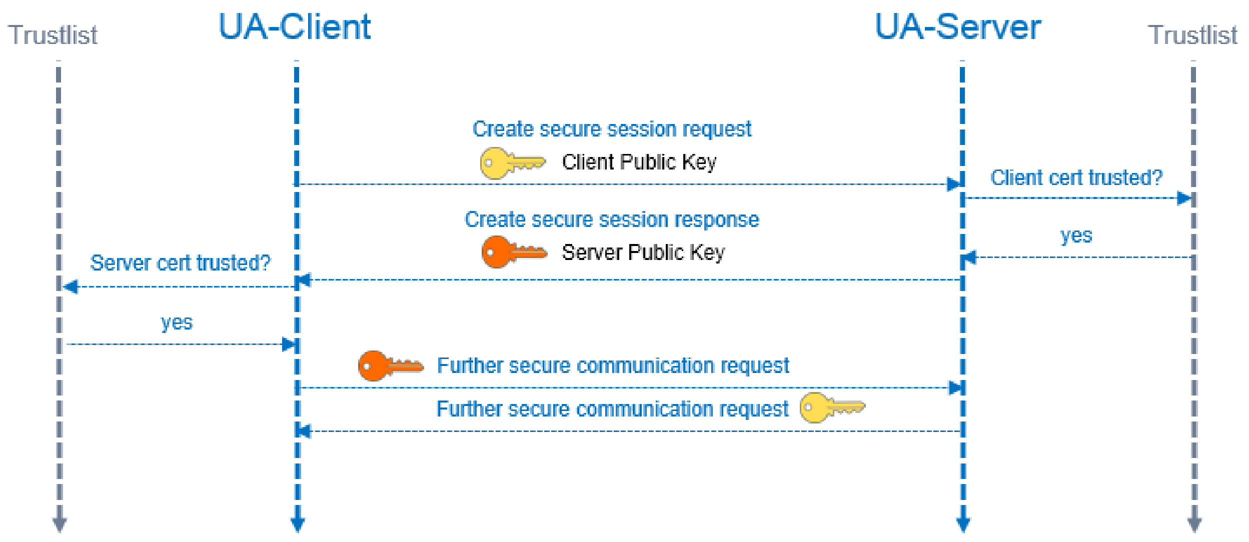
The TwinCAT OPC UA Client and TwinCAT OPC UA Server generate a self-signed certificate during the first startup process. This certificate consists of a private key and a public key. The private key is saved in the *\PKI\CA\private* directory and the corresponding public key in the *\PKI\CA\certs* directory. Any OPC UA Client wishing to establish a secure connection with the Server via one of the security endpoints (Sign, Sign&Encrypt) must know the public key of the OPC UA Server. Conversely the OPC UA Server must know the Client’s public key. This so-called key exchange is described in the following sections:

- Authentication [▶ 122]
- Certificate exchange [▶ 179]
- Access rights [▶ 123]

4.4.7.2 Certificate exchange

The communication between an OPC UA Client and an OPC UA Server can optionally be secured by communication with a secure endpoint. By default, both the TwinCAT OPC UA Server and the TwinCAT OPC UA Client generate a machine-specific, self-signed certificate for authentication the first time they are started.

If you want to use such an encrypted communication connection in your environment, you need to establish a trust relationship between OPC UA Server and OPC UA Client.


Trustlist
UA-Client
UA-Server
Trustlist
Create secure session request
Client Public Key
Client cert trusted?
yes
Create secure session response
Server Public Key
Server cert trusted?
yes
Further secure communication request
Further secure communication request

Trust settings on the server

If you want to authenticate one or more OPC UA Clients to the OPC UA Server via certificates, the OPC UA Server must trust the public keys of the clients. This can be done via the file system, for example. The server manages the trust settings for certificates via the file system.

- Trusted certificates: %InstallDir%\Server\PKI\CA\trusted\certs
- Rejected certificates: %InstallDir%\Server\PKI\CA\rejected\certs

By moving client certificates between these directories, the trust settings can be adjusted accordingly.

Reading a client certificate

A simple way of accessing the client certificate is described below. To do this, you establish a connection to the UA Server using a secure endpoint (for example, Basic128Rsa15/Sign&Encrypt) without first copying the client certificate to the UA Server. This connection is naturally rejected by the UA Server, since it does not trust the UA Client at this point in time. In this case, the TwinCAT OPC UA Client would return the error 0xE4DD0102, for example. However, after rejecting the connection request, the UA Server stores a copy of the client certificate in the above-mentioned "Rejected" directory. The name of the certificate corresponds to the "Thumbprint" of the certificate and can thus be clearly assigned to the client certificate. You can now move this file directly to the above-mentioned "Trusted" directory so that the UA Server can trust the UA Client and accept the connection for all other connection requests.

Announcing OPC UA Servers to the OPC UA Client

Depending on the OPC UA Client employed, different steps may need to be taken so that the OPC UA Client trusts the OPC UA Server. Typically, for client applications with a graphical user interface, a warning message is displayed the first time you connect to the server, whereby the server certificate can then be classified as trustworthy.

The following instruction is therefore only valid for the TwinCAT OPC UA Client.

The public key of the OPC UA Server is located in the following directory as a DER file: *%InstallDir%\Server \PKI\CA\own\certs*

In the case of the TwinCAT OPC UA Client, copy the file into the corresponding "Trusted" directory: *%InstallDir%\Client\PKI\CA\certs*

4.5 Sample Client

4.5.1 Overview

As of version 1.6.80 of the TwinCAT OPC UA Server, a small "UA Sample Client" program is automatically installed. The program enables you to browse the OPC UA namespace and to test the UA Server installation. It is located in the Windows Start menu and in the installation directory of the Supplement/ Function. You can run the program both directly on the UA Server and on a computer in your network.

The UA Sample Client currently offers the following features:

- Connecting to the OPC UA Server
- Establishing a secure connection with OPC UA Server (see Establishing a secure connection to OPC UA Server [▸ 181])
- Browsing the UA namespace of an OPC UA Server (see Browsing the UA namespace [▸ 184])
- Adding a UA node from the namespace to the watchlist, which reads the value of the node regularly (see Using the Watchlist [▸ 185])

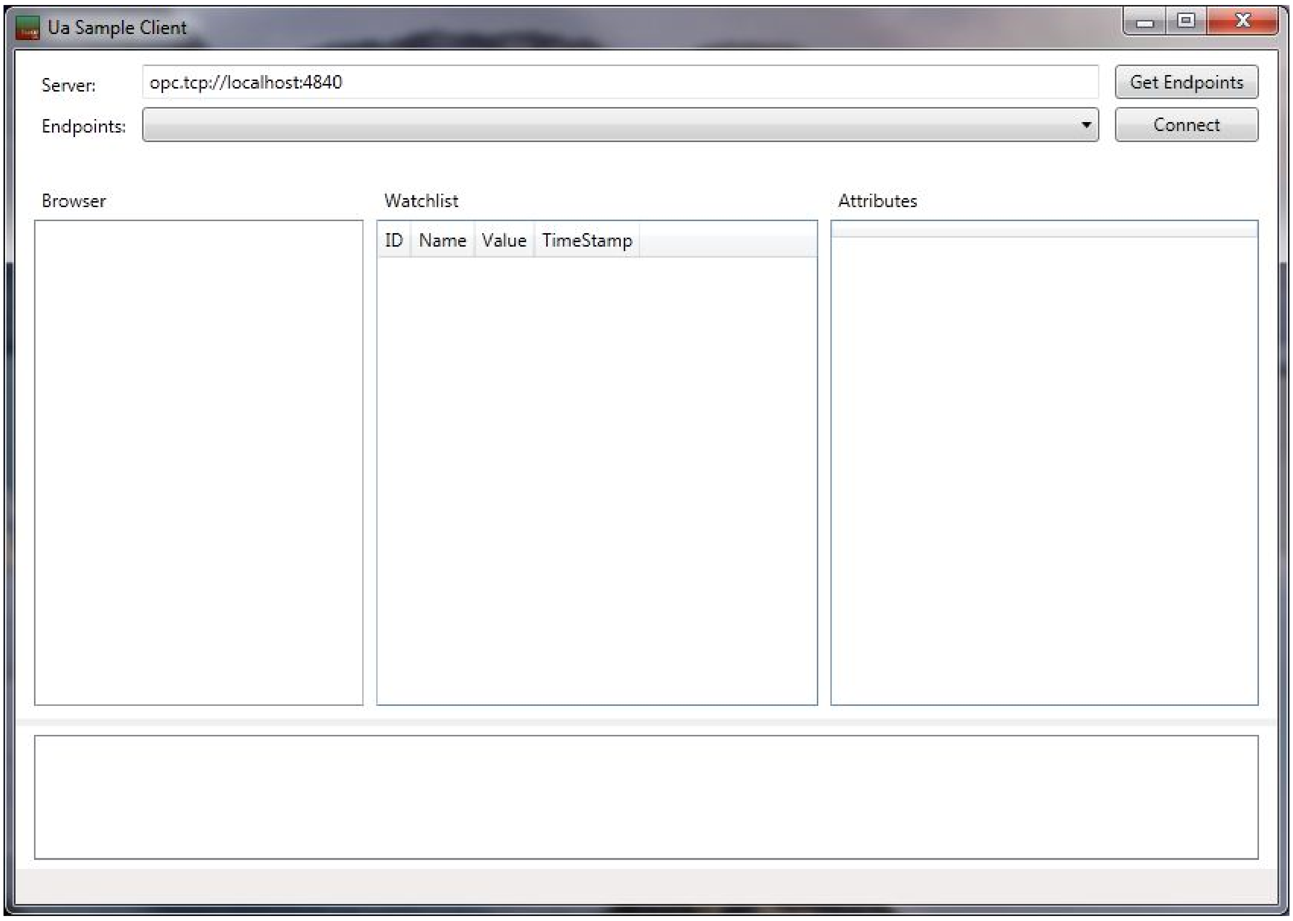
This application is only an OPC UA Sample Client. It does not offer any sophisticated functionalities, but has been developed to provide users with an easy-to-use interface for carrying out initial tests on the OPC UA Server

To start the application, run the *UA SampleClient.exe* file with the **Run as Admin** option.

Endpoints of the OPC UA Server

The UA Sample Client first connects itself to a specified server URL. The client acquires all endpoints of the OPC UA Server (see **Endpoints** drop-down list). The list returned to the server then contains more information about all the available endpoints the client can connect to. Each endpoint can contain the host name of the OPC UA Server instead of the IP address. The client then uses the information from the endpoint to connect to the server.

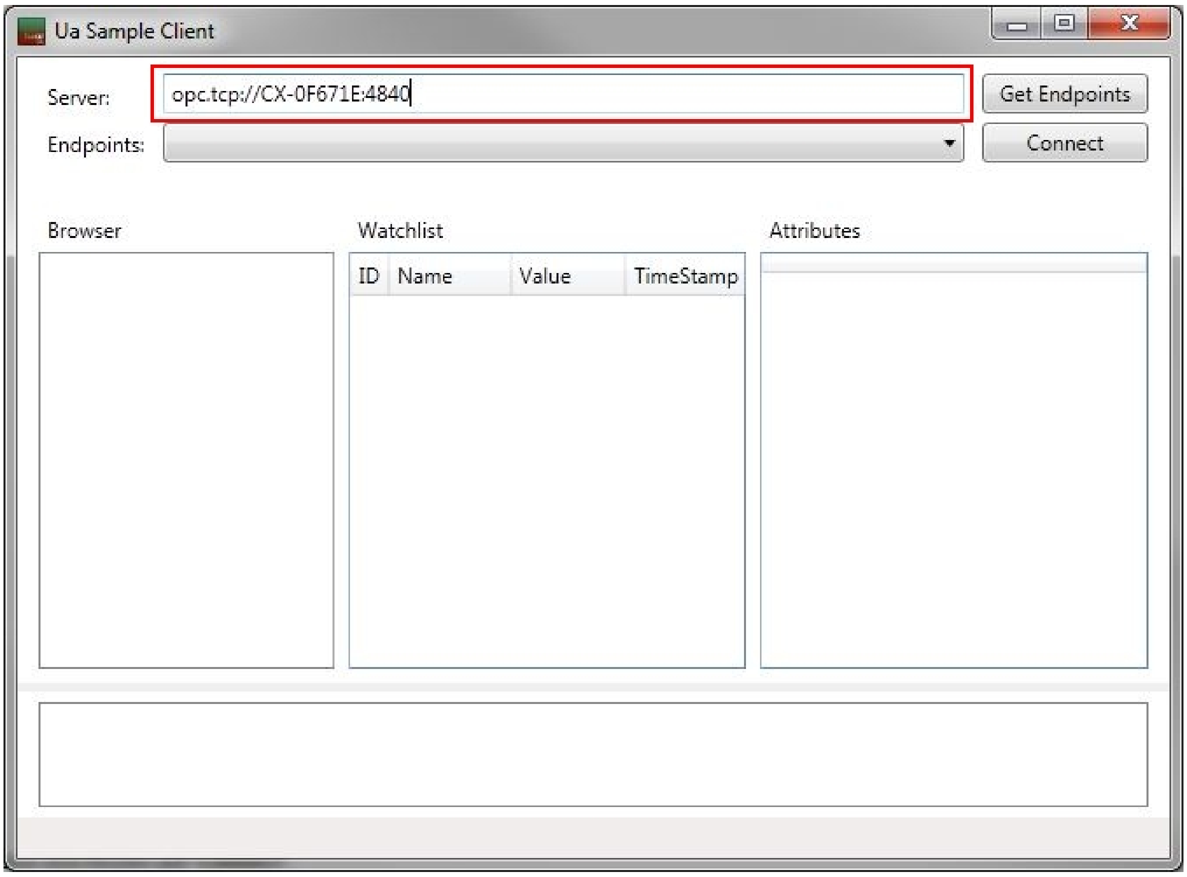
If the name solution does not work on the user's network, the client cannot connect. If the endpoint to which you want the client to connect contains the host name of the server, make sure that the name solution works on your network and that the host name is accessible on the server.



4.5.2 Establishing a secure connection to OPC UA Server

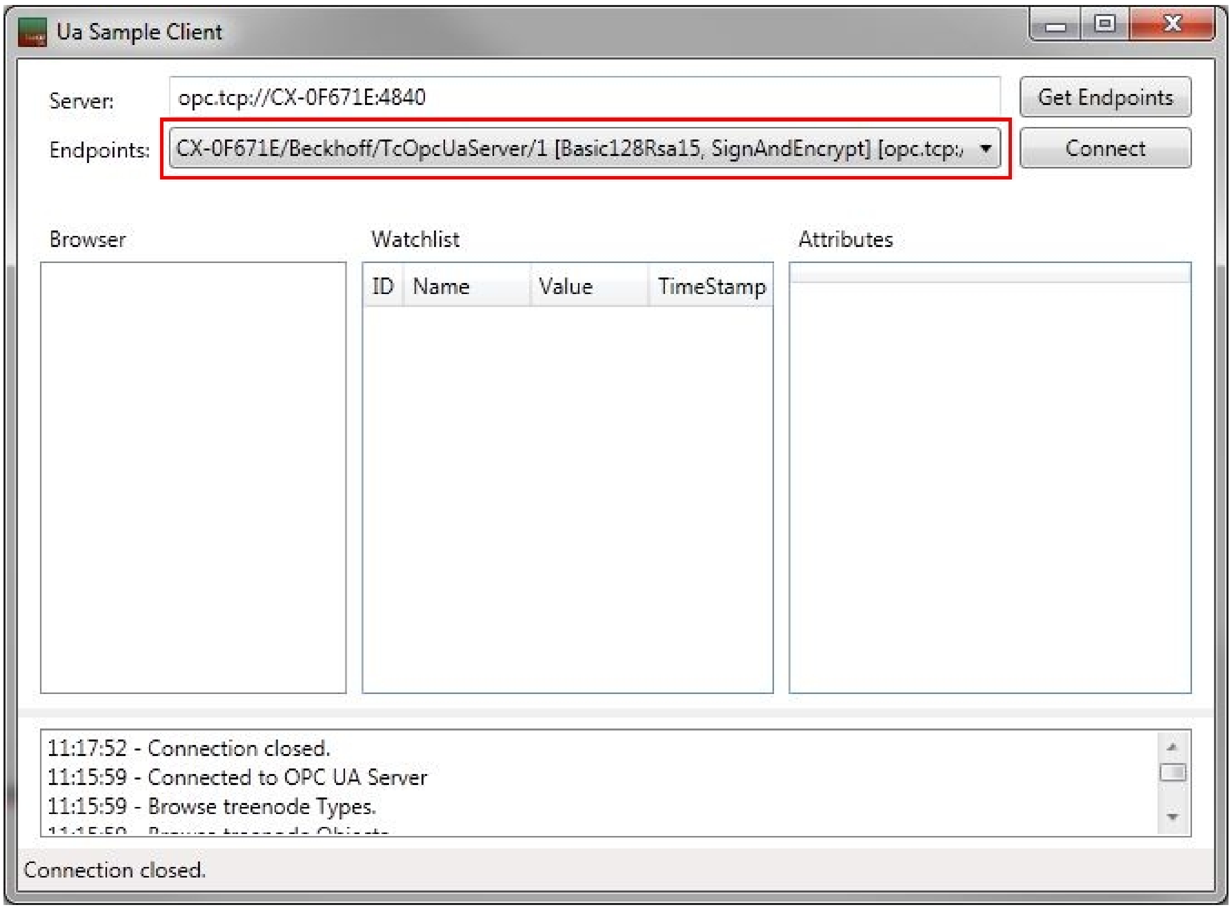
1. Enter the URL of an OPC UA Server in the upper text field of the UA Sample Client.
2. Click the **Get Endpoints** button.

⇨ The endpoints provided by the UA Server are then displayed in the **Endpoints** drop-down list.



3. In this sample, select the entry "<SomeName>/Beckhoff/TcOpcUaServer/1[Basic128Rsa15, SignAndEncrypt] [opc.tcp://<SomeName>:4840]" and click **Connect**.
   You must copy the public key from the certificate of the UA Sample Client to the UA Server so that it "trusts" the Sample Client. Otherwise, the connection attempt is rejected by the UA Server via the secure

channel ("BadSecureChannelClosed"). Further information on the certificate management with the OPC UA Server can be found in the section Certificate exchange [▶ 179].



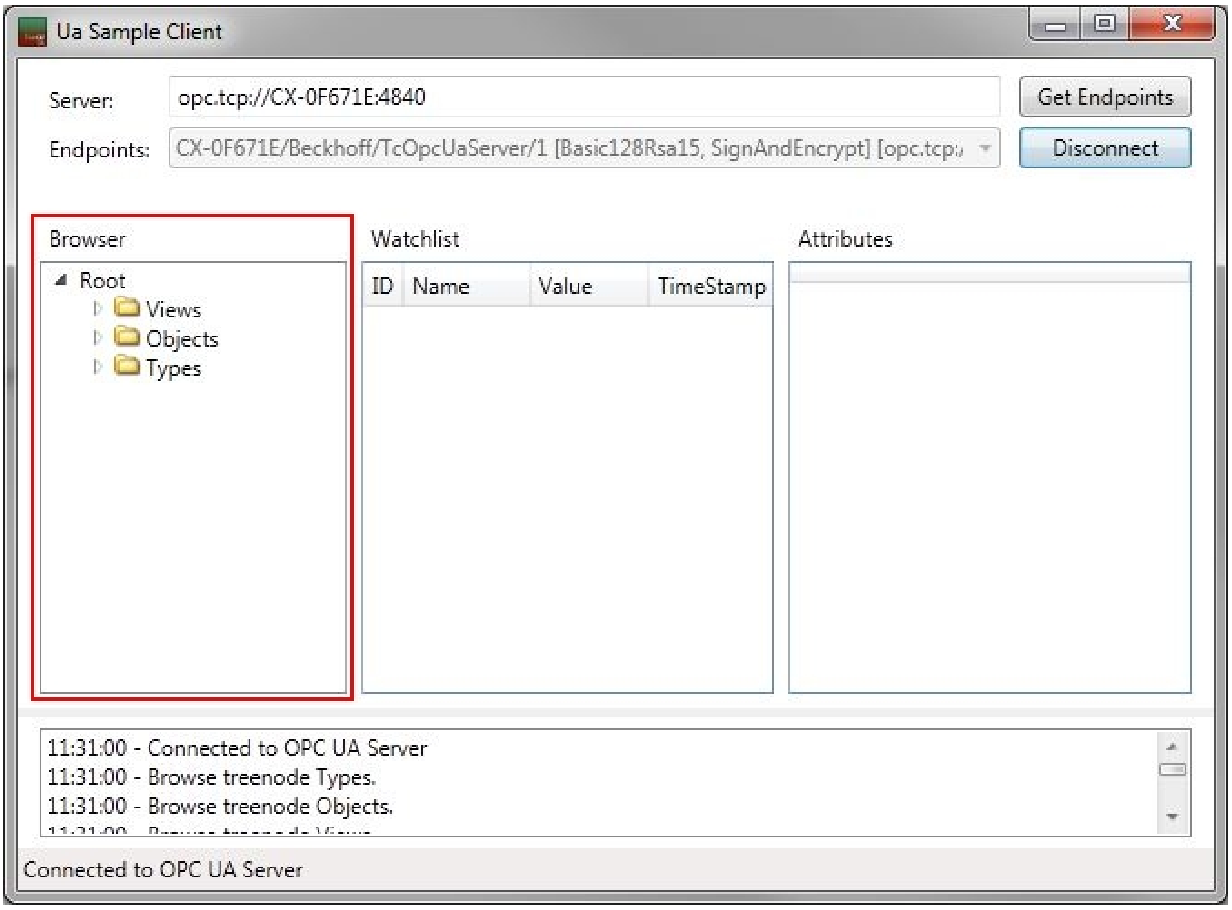
Server: opc.tcp://CX-0F671E:4840

Endpoints: CX-0F671E/Beckhoff/TcOpcUaServer/1 [Basic128Rsa15, SignAndEncrypt] [opc.tcp:/

```
11:17:52 - Connection closed.
11:15:59 - Connected to OPC UA Server
11:15:59 - Browse treenode Types.
```

Connection closed.

⇨ You can now use the **Browser** in the left half of the window to navigate through the UA namespace.



Server: opc.tcp://CX-0F671E:4840

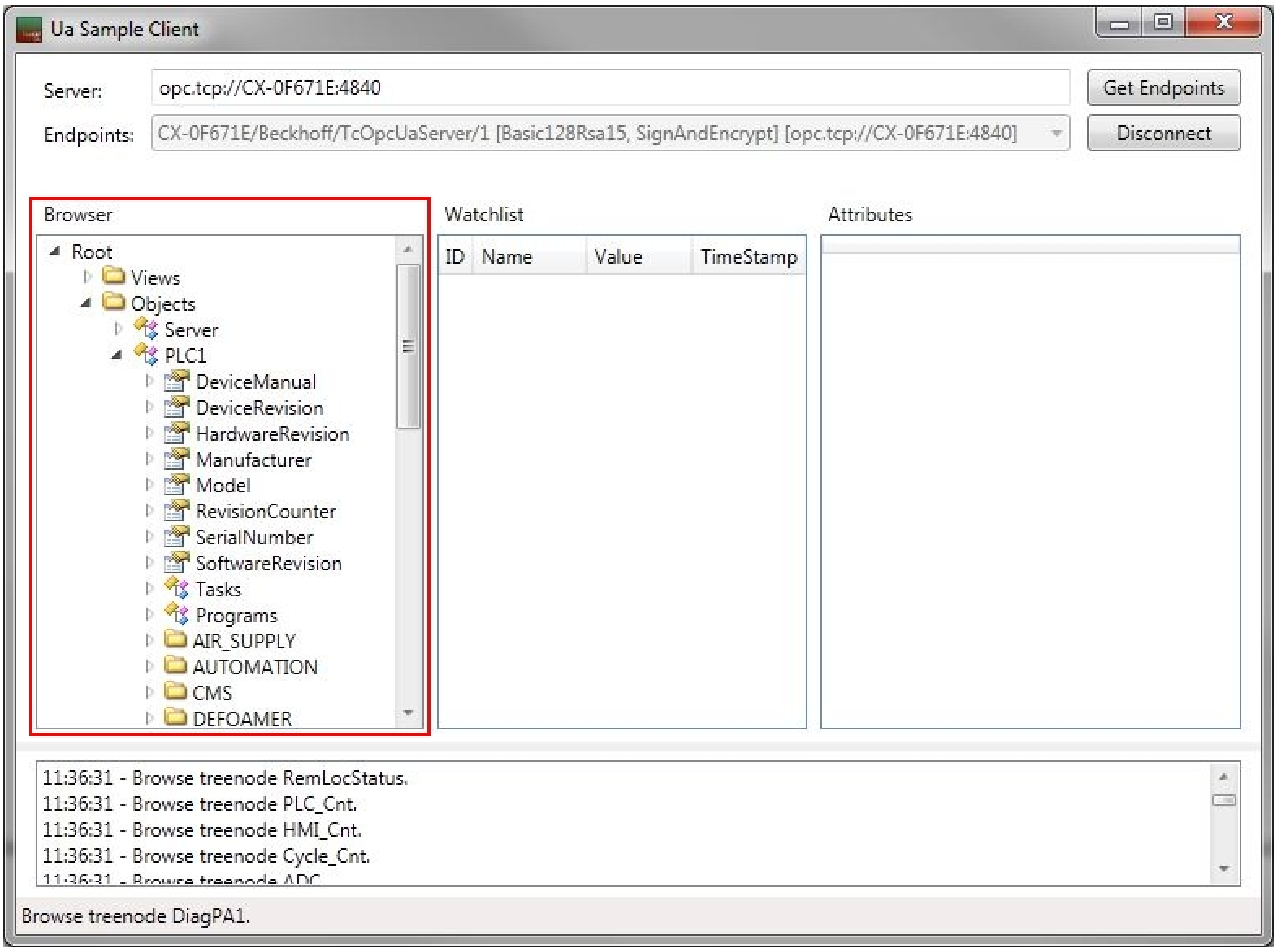
Endpoints: CX-0F671E/Beckhoff/TcOpcUaServer/1 [Basic128Rsa15, SignAndEncrypt] [opc.tcp:/

```
11:31:00 - Connected to OPC UA Server
11:31:00 - Browse treenode Types.
11:31:00 - Browse treenode Objects.
```

Connected to OPC UA Server

4.5.3 Browsing the UA namespace

When a successful connection has been established you can use the **Browser** in the left half of the UA Sample Client to navigate through the UA namespace. Below the node **PLC1** you will find the currently running PLC program, and you can display the variables declared there and released for UA.



Server: opc.tcp://CX-0F671E:4840

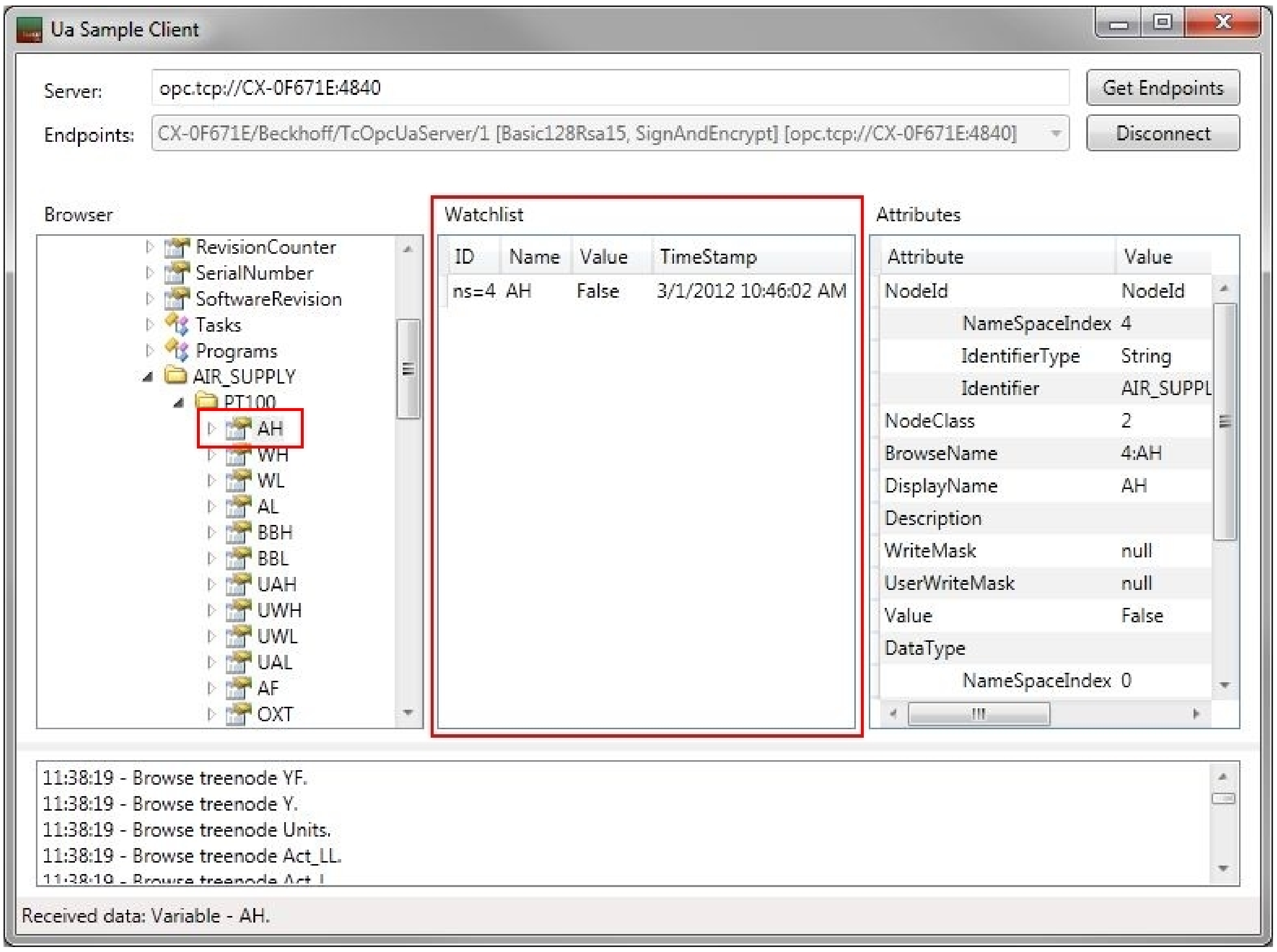
Endpoints: CX-0F671E/Beckhoff/TcOpcUaServer/1 [Basic128Rsa15, SignAndEncrypt] [opc.tcp://CX-0F671E:4840]

```
11:36:31 - Browse treenode RemLocStatus.
11:36:31 - Browse treenode PLC_Cnt.
11:36:31 - Browse treenode HMI_Cnt.
11:36:31 - Browse treenode Cycle_Cnt.
11:36:31 - Browse treenode ADC
```

Browse treenode DiagPA1.

4.5.4 Using the Watchlist

You can insert PLC variables from the UA namespace into a watchlist, for example to have their values read cyclically by the UA Sample Client. To do this, open the context menu of a variable and select **Add to Watchlist**. The variable is then transferred to the watchlist and its values are automatically read out cyclically from the PLC.



Server: opc.tcp://CX-0F671E:4840

Endpoints: CX-0F671E/Beckhoff/TcOpcUaServer/1 [Basic128Rsa15, SignAndEncrypt] [opc.tcp://CX-0F671E:4840]

Browser: RevisionCounter, SerialNumber, SoftwareRevision, Tasks, Programs, AIR_SUPPLY, PT100, AH, WH, WL, AL, BBH, BBL, UAH, UWH, UWL, UAL, AF, OXT

ID	Name	Value	TimeStamp
ns=4	AH	False	3/1/2012 10:46:02 AM

Attribute	Value
NodeId	NodeId
NameSpaceIndex	4
IdentifierType	String
Identifier	AIR_SUPPL
NodeClass	2
BrowseName	4:AH
DisplayName	AH
Description	
WriteMask	null
UserWriteMask	null
Value	False
DataType	
NameSpaceIndex	0

```
11:38:19 - Browse treenode YF.
11:38:19 - Browse treenode Y.
11:38:19 - Browse treenode Units.
11:38:19 - Browse treenode Act_LL.
```

Received data: Variable - AH.

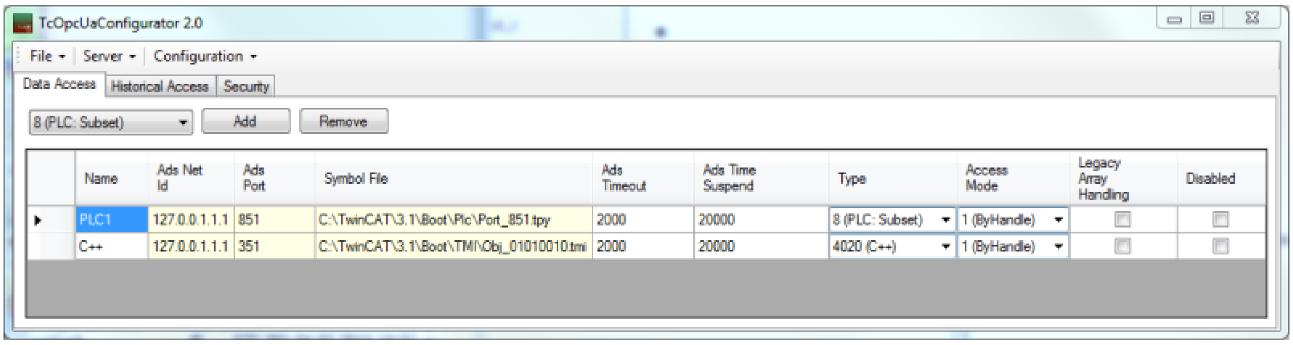
5 Configuration

5.1 Setup Version 3.x.x

5.1.1 Overview

This section describes the use of the TwinCAT OPC UA Configurator. The OPC UA Configurator parameterizes the OPC UA Server, or rather is a graphical user interface for configuring the *ServerConfig.xml* file. It provides the following configuration options:

- Adding/removing devices for data access (PLC runtimes, TwinCAT 3 C++ instances, I/O tasks)
- Activation/deactivation of Historical Access
- Activation/deactivation of anonymous accesses or user name/password authentication
- Restart or shutdown of the OPC UA Server

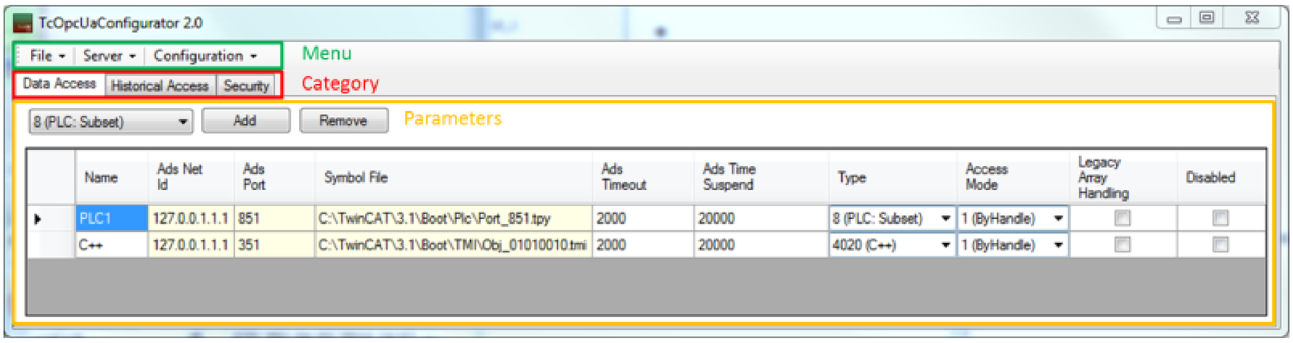


	Name	Ads Net Id	Ads Port	Symbol File	Ads Timeout	Ads Time Suspend	Type	Access Mode	Legacy Array Handling	Disabled
▶	PLC1	127.0.0.1.1.1	851	C:\TwinCAT\3.1\Boot\Plc\Port_851.tpy	2000	20000	8 (PLC: Subset)	1 (ByHandle)	☐	☐
	C++	127.0.0.1.1.1	351	C:\TwinCAT\3.1\Boot\TMI\Obj_01010010.tmi	2000	20000	4020 (C++)	1 (ByHandle)	☐	☐

On starting the OPC UA Configurator, the currently active configuration of the OPC UA Server is automatically loaded, which is located as standard in the \%InstallDir%\Server\ directory as the *ServerConfig.xml* file. However, you can also load, save and activate other ServerConfig.xml files via the menu.

The user interface of the OPC UA Configurator consists of the following sections:

- Menu section (Menu): Saving and activation of configurations
- Categories section (Category): Grouping of the individual parameters into various categories, e.g. Data Access
- Parameters section (Parameters): Individual parameters for activating/deactivating certain features



	Name	Ads Net Id	Ads Port	Symbol File	Ads Timeout	Ads Time Suspend	Type	Access Mode	Legacy Array Handling	Disabled
▶	PLC1	127.0.0.1.1.1	851	C:\TwinCAT\3.1\Boot\Plc\Port_851.tpy	2000	20000	8 (PLC: Subset)	1 (ByHandle)	☐	☐
	C++	127.0.0.1.1.1	351	C:\TwinCAT\3.1\Boot\TMI\Obj_01010010.tmi	2000	20000	4020 (C++)	1 (ByHandle)	☐	☐

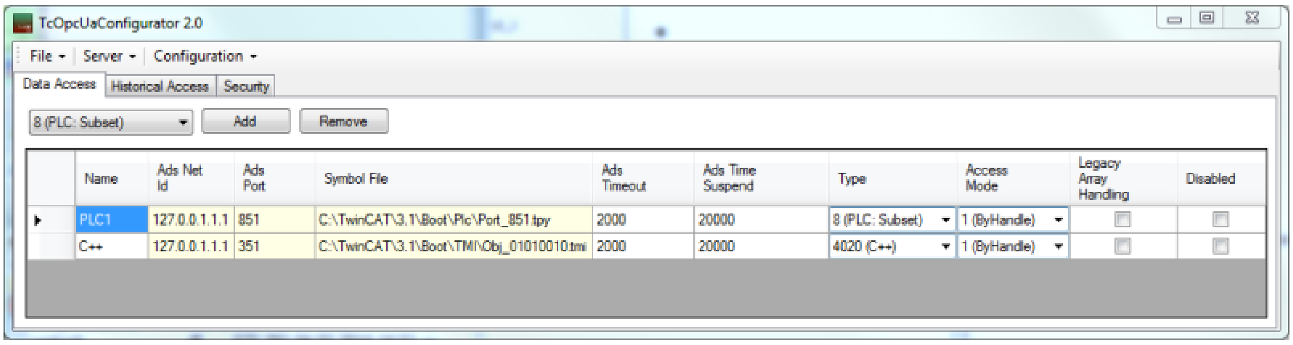
Further information on the individual configuration parameters can be found in the descriptions of the individual categories:

- Data Access [▶ 188]
- Historical Access [▶ 189]
- Server Security [▶ 189]
- Alarms and Conditions [▶ 190]

- Online Panel [▸ 191]

5.1.2 Data Access

The **Data Access** tab shows the configuration setting for TwinCAT runtime devices. This is where you configure a device if it is to be available via OPC UA.



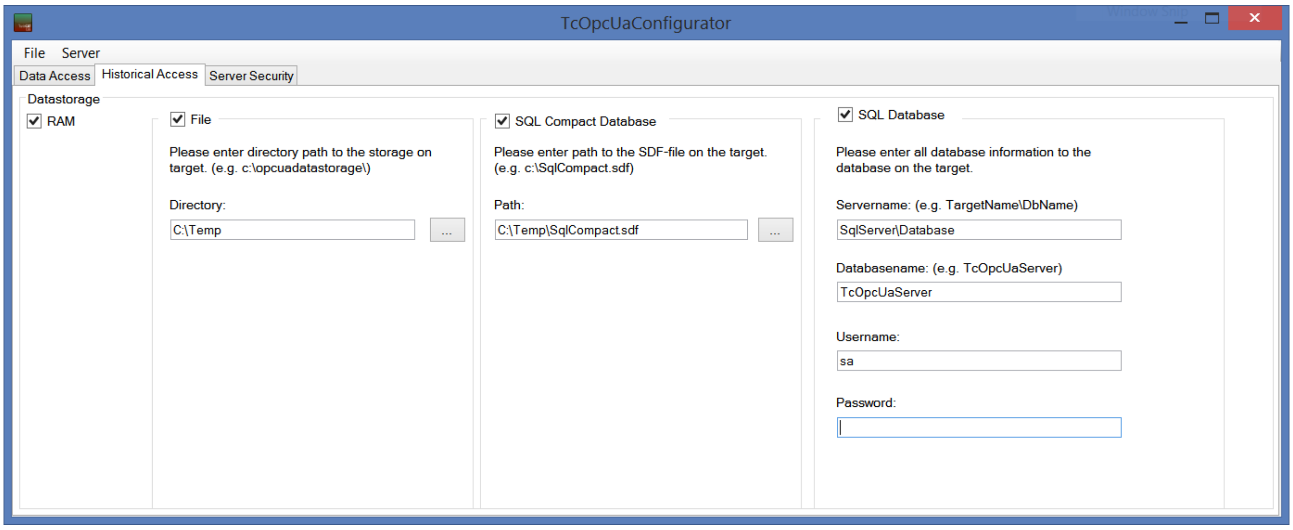
	Name	Ads Net Id	Ads Port	Symbol File	Ads Timeout	Ads Time Suspend	Type	Access Mode	Legacy Array Handling	Disabled
▸	PLC1	127.0.0.1.1.1	851	C:\TwinCAT\3.1\Boot\Plc\Port_851.tpy	2000	20000	8 (PLC: Subset)	1 (ByHandle)	☐	☐
	C++	127.0.0.1.1.1	351	C:\TwinCAT\3.1\Boot\TMI\Obj_01010010.tmi	2000	20000	4020 (C++)	1 (ByHandle)	☐	☐

Setting	Description
Name	Display name of the TwinCAT runtime in the OPC UA namespace
AdsNetId	AmsNetId of the TwinCAT runtime. 127.0.0.1.1.1 for a local runtime or the corresponding AmsNetId of a remote device. ADS routes must be configured when communicating with a remote device.
AdsPort	AdsPort of the TwinCAT runtime, e.g. 851 for the first TwinCAT 3 PLC runtime.
SymbolFile	Symbol file containing information about all available variables and object instances of the runtime.
AdsTimeout	Timeout setting when communicating with the target runtime. It is recommended to leave this setting at the default value.
AdsTimeSuspend	TimeSuspend setting when communicating with the target runtime. It is recommended to leave this setting at the default value. TimeSuspend timeout is triggered when a target runtime is temporarily unavailable. The OPC UA Server starts a new communication attempt after TimeSuspend.
Type	Type of the target runtime with which the OPC UA Server is to communicate, e.g. TwinCAT 3 PLC, TwinCAT 2 PLC,...
AccessMode	The access mode is normally only required if the OPC UA Server is to communicate with a runtime hosted on a Beckhoff BC/BX device. In this case the corresponding setting must be activated. Otherwise the default setting can be adopted.
LegacyArrayHandling	Enable this setting if obsolete OPC UA Clients require array elements to be available as separate UA nodes.
LegacyUriFormat	Activates various naming schemes for the generation of NamespaceUri as described in the section Generating NamespaceURI [▸ 125].
ReleaseHandles	Enables the release of handles for TwinCAT runtime symbols when UaClients no longer "need" a node. This is the case when a node is removed from the monitoring (DeleteMonitoredItem) or when the monitoring mode of an element is disabled.
ForceIdentifierCase	Applies only to types TPY and SYM. Forces the names of all nodes to be case-sensitive.
Disabled	Click the checkbox to disable the runtime so that it does not appear in the OPC UA namespace.

Further information on runtime-specific parameters can be found in the section Server [▸ 40].

5.1.3 Historical Access

The **Historical Access** tab shows the general settings for the configuration of historical access. This is where you activate and configure the storage media that are to be used for historical access.


TcOpcUaConfigurator

File Server

Data Access | Historical Access | Server Security

Datastorage

☑ RAM

☑ File

Please enter directory path to the storage on target. (e.g. c:\opcuadatastorage\)

Directory: C:\Temp

☑ SQL Compact Database

Please enter path to the SDF-file on the target. (e.g. c:\SqlCompact.sdf)

Path: C:\Temp\SqlCompact.sdf

☑ SQL Database

Please enter all database information to the database on the target.

Servername: (e.g. TargetName\DbName) SqlServer\Database

Databasename: (e.g. TcOpcUaServer) TcOpcUaServer

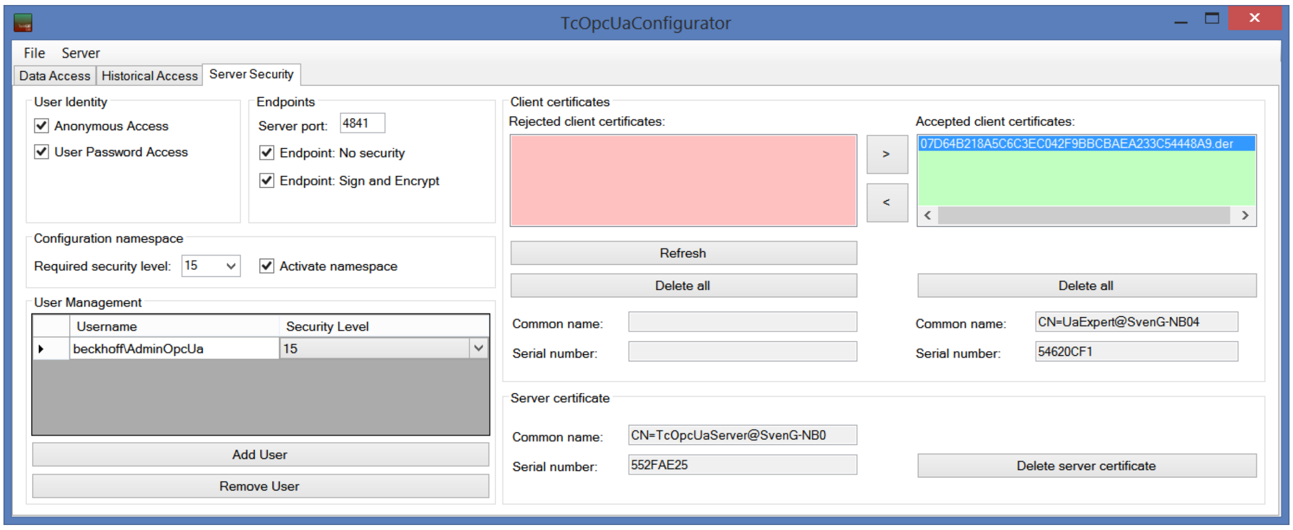
Username: sa

Password:

Further information on historical access can be found in the section Server [▸ 40].

5.1.4 Server Security

The **Server Security** tab shows the general settings for the security and endpoint mechanisms of the OPC UA Server.


TcOpcUaConfigurator

File Server

Data Access | Historical Access | Server Security

User Identity

☑ Anonymous Access

☑ User Password Access

Endpoints

Server port: 4841

☑ Endpoint: No security

☑ Endpoint: Sign and Encrypt

Configuration namespace

Required security level: 15 ☑ Activate namespace

User Management

Username	Security Level
beckhoff\AdminOpcUa	15

Add User

Remove User

Client certificates

Rejected client certificates:

Accepted client certificates: 07D64B218A5C6C3EC042F9BBCBAEA233C54448A9.der

Refresh

Delete all

Delete all

Common name:

Serial number:

Common name: CN=UaExpert@SvenG-NB04

Serial number: 54620CF1

Server certificate

Common name: CN=TcOpcUaServer@SvenG-NB0

Serial number: 552FAE25

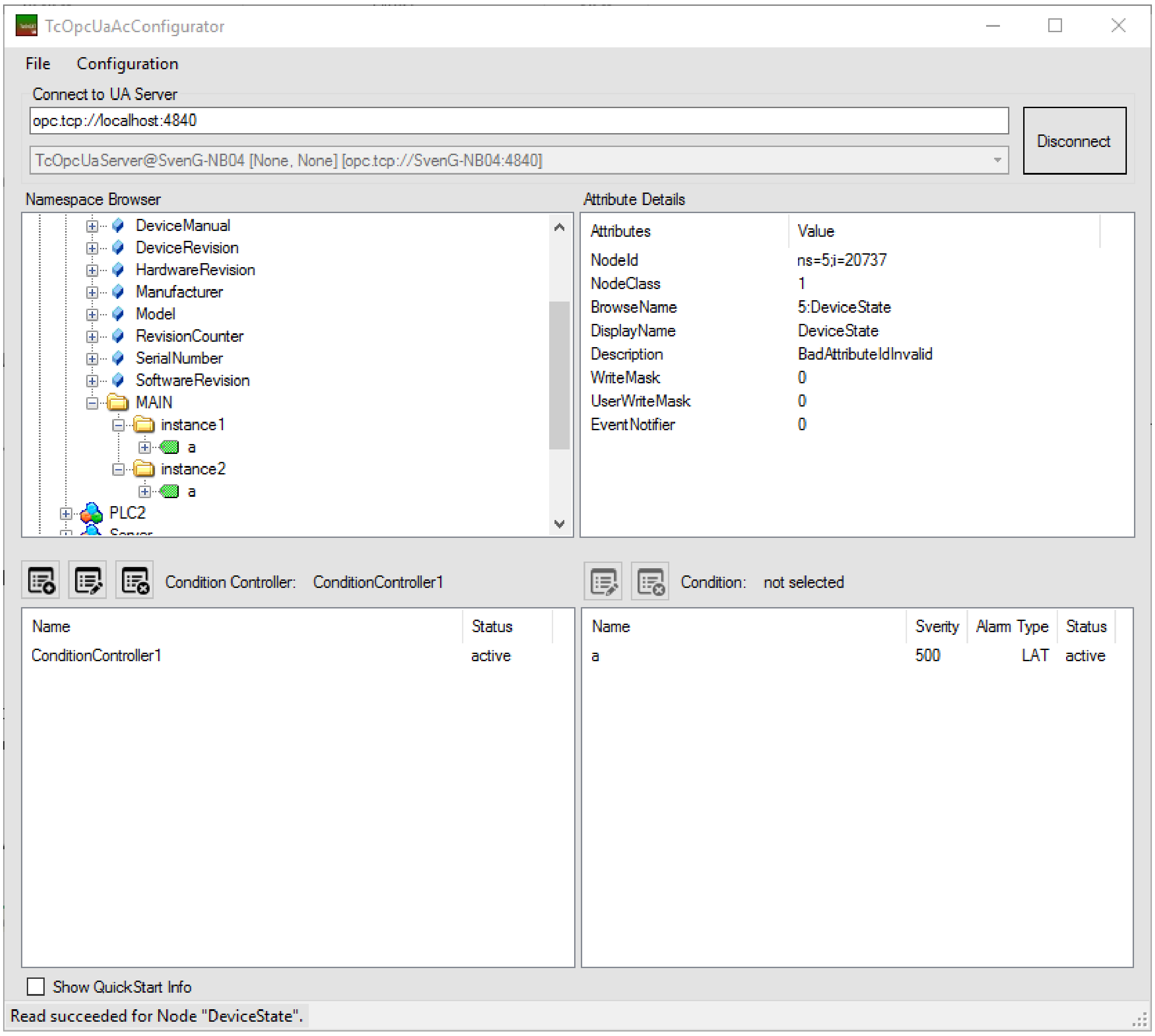
Delete server certificate

Settings	Description
User Identity	Enable or disable a specific user identity, such as disabling the Anonymous Access checkbox to force user authentication.
Endpoints	Enables or disables certain endpoints, such as disabling the "No Security" endpoint to force UA Clients to connect to the "Sign and Encrypt" endpoint.
Configuration namespace	Facilitates enabling or disabling of the Configuration namespace [▸ 132] and setting of a specific security level, which is required to access this namespace (user authentication required).

Settings	Description
User Management	Enables configuration of existing user accounts and sets an access level for each account. Access levels are currently used only for configuring access to the Configuration namespace [▶ 132]. Note that the user account - if the computer on which the OPC UA Server is running is a member of a Windows domain - may also be an account in that domain.
Client certificates	Allows you to move OPC UA Client certificates between the trust lists "Rejected" and "Accepted". Client certificates can be easily identified, trusted, or rejected when the clients are connected to one of the secure endpoints.

5.1.5 Alarms and Conditions

The **Alarms and Conditions (A&C)** configurator is a separate graphic tool. It creates an XML-based configuration file with which the OPC UA Server creates its A&C configuration.



opc.tcp://localhost:4840

TcOpcUaServer@SvenG-NB04 [None, None] [opc.tcp://SvenG-NB04:4840]

Attributes	Value
NodeId	ns=5;i=20737
NodeClass	1
BrowseName	5:DeviceState
DisplayName	DeviceState
Description	BadAttributeIdInvalid
WriteMask	0
UserWriteMask	0
EventNotifier	0

Condition Controller: ConditionController1

Name	Status
ConditionController1	active

Condition: not selected

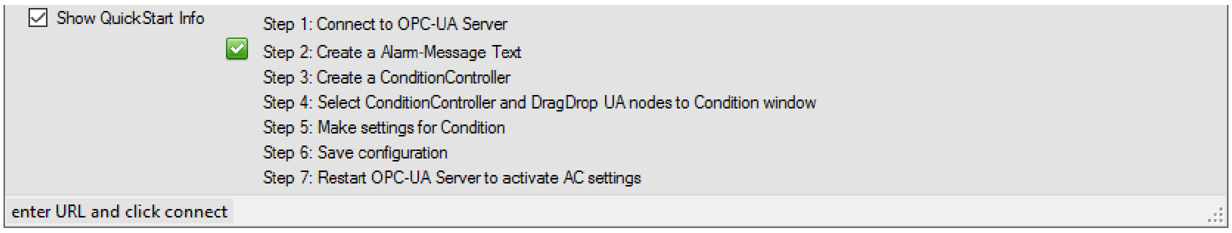
Name	Sverity	Alarm Type	Status
a	500	LAT	active

Read succeeded for Node "DeviceState".

The A&C Configurator is located in the installation directory of TwinCAT OPC UA: *C:\TwinCAT\Functions \TF6100-OPC-UA\Win32\ConfiguratorAc*.

Use of the A&C Configurator

1. Connect to the OPC UA Server whose Alarms and Conditions are to be configured. To do this, enter the server URL and select an endpoint to which you want the Configurator to connect.
2. When the connection is established and the namespace of the server is visible, create one or more ConditionControllers. A ConditionController is an administrative unit that contains the current alarm configuration in the form of one or more conditions.
3. Select a ConditionController and create one or more conditions by dragging variables to be monitored into the Conditions window in the lower right corner and storing them there.
4. Follow the instructions in the Quick Start Guide for the Configurator step-by-step.

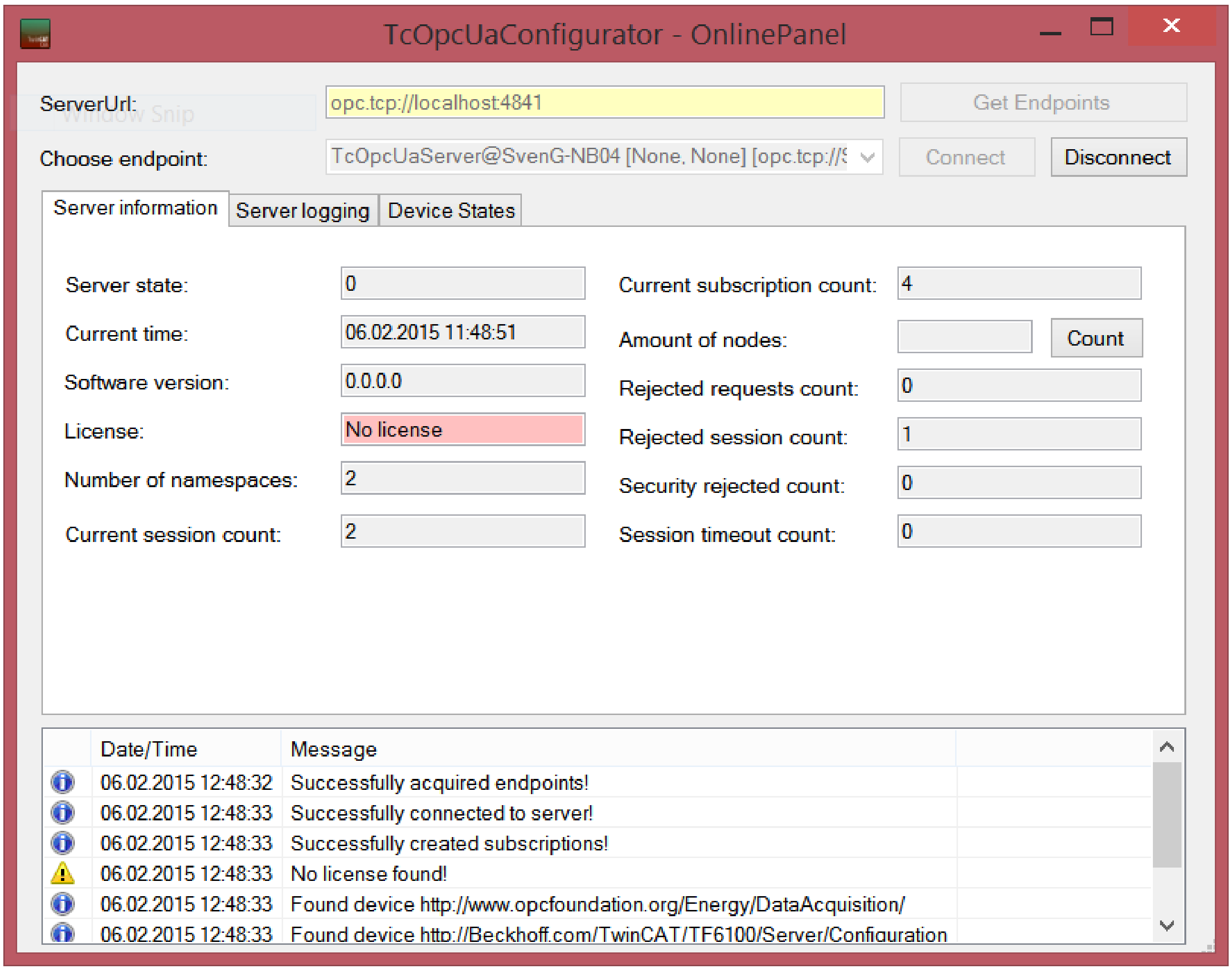

☑ Show QuickStart Info

Step 1: Connect to OPC-UA Server
✓ Step 2: Create a Alarm-Message Text
Step 3: Create a ConditionController
Step 4: Select ConditionController and DragDrop UA nodes to Condition window
Step 5: Make settings for Condition
Step 6: Save configuration
Step 7: Restart OPC-UA Server to activate AC settings

enter URL and click connect

5.1.6 Online Panel

The **Online Panel** extends the OPC UA Configurator by OPC UA Client functions. It enables you to connect locally or remotely to an OPC UA Server for the purpose of obtaining diagnostic or management functions.



ServerUrl: opc.tcp://localhost:4841

Choose endpoint: TcOpcUaServer@SvenG-NB04 [None, None] [opc.tcp://S

Server state:	0	Current subscription count:	4
Current time:	06.02.2015 11:48:51	Amount of nodes:	
Software version:	0.0.0.0	Rejected requests count:	0
License:	No license	Rejected session count:	1
Number of namespaces:	2	Security rejected count:	0
Current session count:	2	Session timeout count:	0

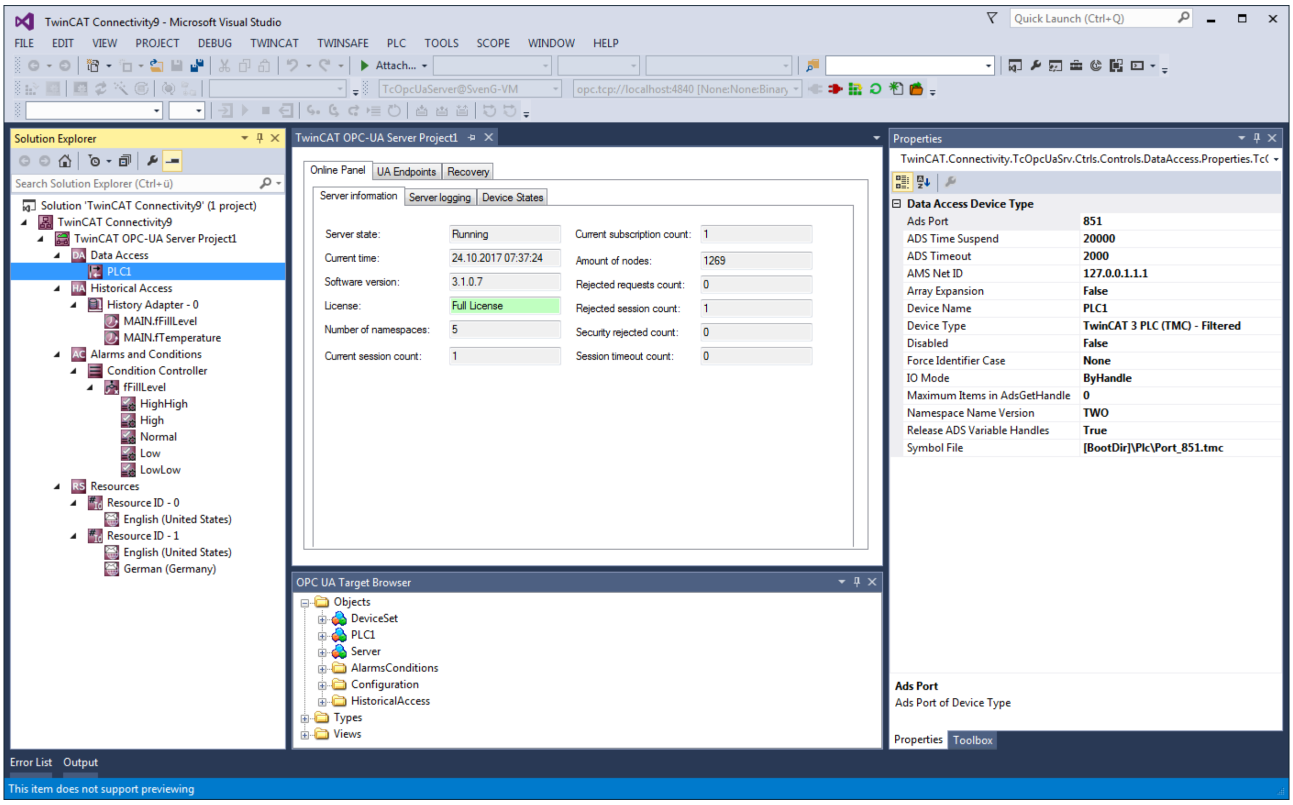
```
Date/Time            Message
06.02.2015 12:48:32  Successfully acquired endpoints!
06.02.2015 12:48:33  Successfully connected to server!
06.02.2015 12:48:33  Successfully created subscriptions!
06.02.2015 12:48:33  No license found!
06.02.2015 12:48:33  Found device http://www.opcfoundation.org/Energy/DataAcquisition/
06.02.2015 12:48:33  Found device http://Beckhoff.com/TwinCAT/TF6100/Server/Configuration
```

Area/tab	Description
Server information	Provides general diagnostic information on the device currently connected to the OPC UA Server. Possible useful information: • License status • Number of currently created OPC UA Client sessions • Number of available nodes (calculation may take some time) • Server version
Server logging	Provides functions for activating a log file on the OPC UA Server. However, this should only be used if it has been recommended by Beckhoff support, as the creation of the log file can impair general system performance.
Device States	Provides diagnostic information on every ADS device configured on the OPC UA Server. This information can be very helpful if you need to check the connection to an ADS device. A common configuration problem can be that the corresponding symbol file (TPY, TMC) cannot be loaded by the OPC UA Server. In this case, the Device States panel issues a corresponding error message.

5.2 Setup Version 4.x.x

5.2.1 Overview

The TF6100 setup (version 4.x.x and higher) contains the latest version of the OPC UA Server Configurator. This was integrated in Microsoft Visual Studio as a separate project type to provide an integrated and consistent engineering concept. You can configure all the different facets of the OPC UA Server and in doing so also use source control mechanisms such as Team Foundation Server or Subversion Integrations.



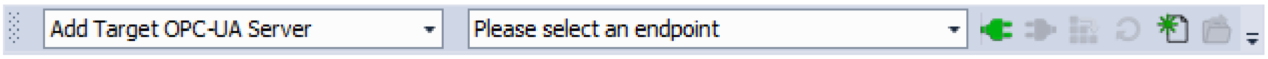
Server state:	Running	Current subscription count:	1
Current time:	24.10.2017 07:37:24	Amount of nodes:	1269
Software version:	3.1.0.7	Rejected requests count:	0
License:	Full License	Rejected session count:	1
Number of namespaces:	5	Security rejected count:	0
Current session count:	1	Session timeout count:	0

Data Access Device Type	
Ads Port	851
ADS Time Suspend	20000
ADS Timeout	2000
AMS Net ID	127.0.0.1.1.1
Array Expansion	False
Device Name	PLC1
Device Type	TwinCAT 3 PLC (TMC) - Filtered
Disabled	False
Force Identifier Case	None
IO Mode	ByHandle
Maximum Items in AdsGetHandle	0
Namespace Name Version	TWO
Release ADS Variable Handles	True
Symbol File	[BootDir]\Plc\Port_851.tmc

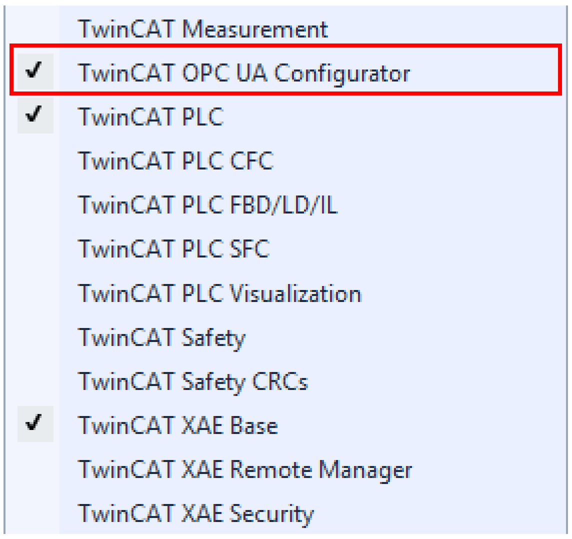
Ads Port
Ads Port of Device Type

Toolbar

The toolbar of the OPC UA Configurator can be used to establish a connection with a local or remote TwinCAT OPC UA Server. All configuration settings can then be implemented for the selected server.



If the toolbar is not displayed, you can add it to the user interface via the menu **View > Toolbars**.



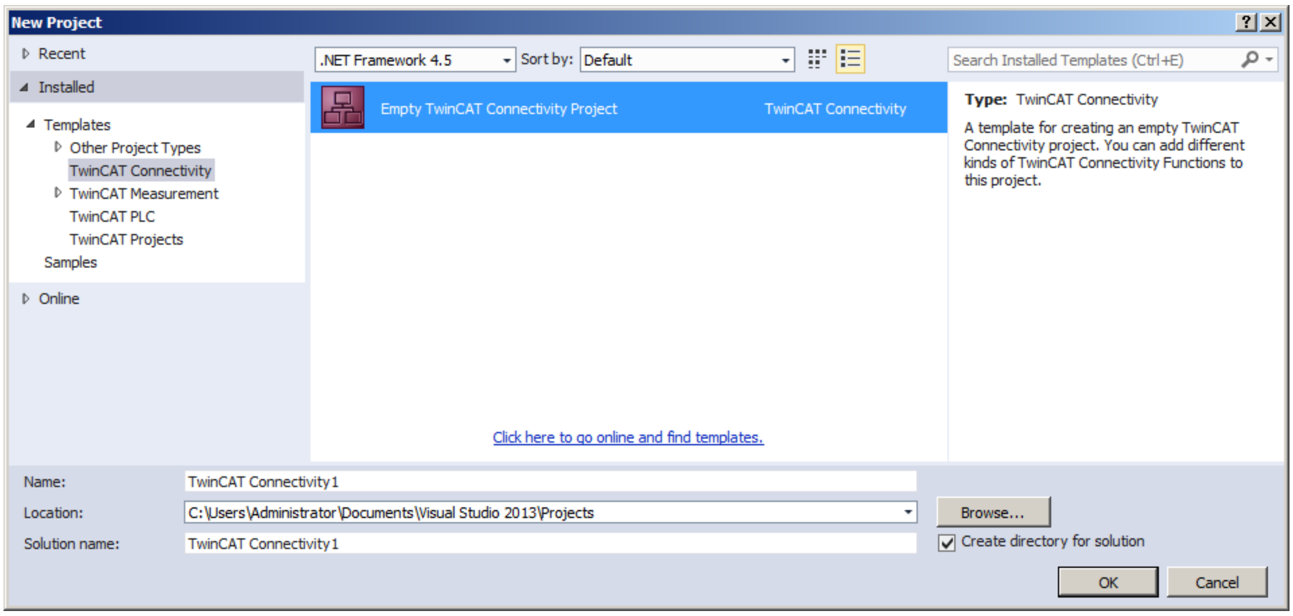
Requirements

Products	Setup versions	Target platform
TF6100	4.x.x	IPC or CX (x86, x64, ARM)

5.2.2 Creating a new project

The project package of the OPC UA Configurator integrates itself in the so-called connectivity package. You can select this when creating a new Visual Studio project.

Project template "TwinCAT Connectivity Project":



Type: TwinCAT Connectivity

A template for creating an empty TwinCAT Connectivity project. You can add different kinds of TwinCAT Connectivity Functions to this project.

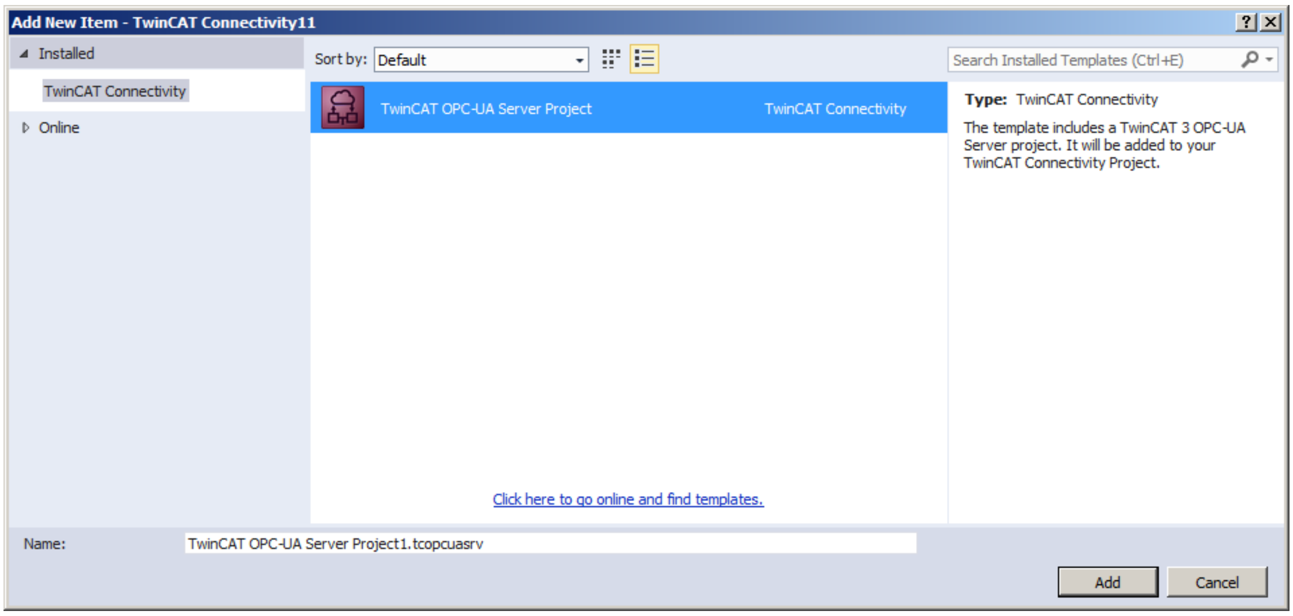
Name: TwinCAT Connectivity1

Location: C:\Users\Administrator\Documents\Visual Studio 2013\Projects

Solution name: TwinCAT Connectivity1

✓ Create directory for solution

Project template "TwinCAT OPC-UA Server Project":


Add New Item - TwinCAT Connectivity11
Installed
TwinCAT Connectivity
Online
Sort by: Default
Search Installed Templates (Ctrl+E)
TwinCAT OPC-UA Server Project
TwinCAT Connectivity
Click here to go online and find templates.
Add
Cancel

Type: TwinCAT Connectivity

The template includes a TwinCAT 3 OPC-UA Server project. It will be added to your TwinCAT Connectivity Project.

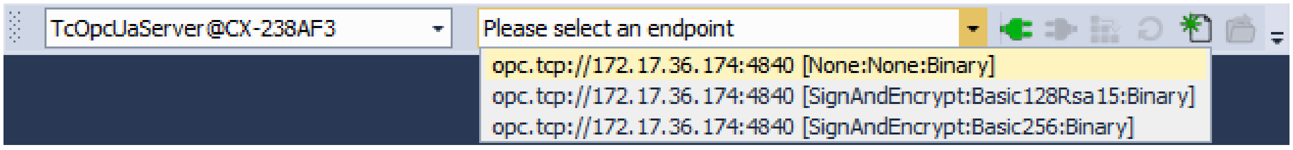
Name: TwinCAT OPC-UA Server Project1.tcopcuasrv

Requirements

Products	Setup versions	Target platform
TF6100	4.x.x	IPC or CX (x86, x64, ARM)

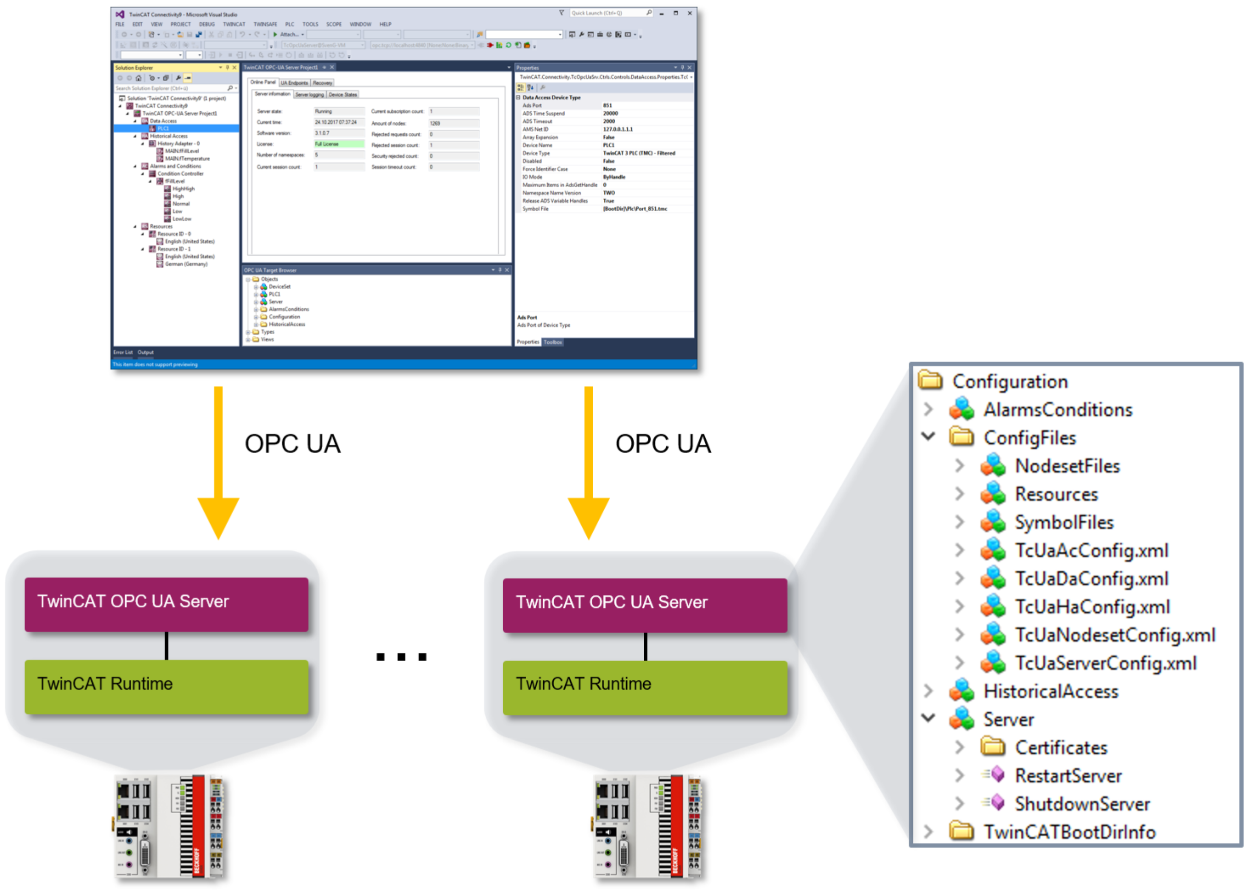
5.2.3 Select target device

The OPC UA Configurator enables the complete parameterization of the Server via OPC UA. In much the same way as in the TwinCAT XAE system, you can select a target OPC UA Server via the toolbar.



opc.tcp://172.17.36.174:4840 [None:None:Binary]
opc.tcp://172.17.36.174:4840 [SignAndEncrypt:Basic128Rsa15:Binary]
opc.tcp://172.17.36.174:4840 [SignAndEncrypt:Basic256:Binary]

All settings in the configuration are then implemented for this OPC UA Server. The basis for this is the so-called configuration namespace of the server (see Configuring the namespace [▶ 132]).



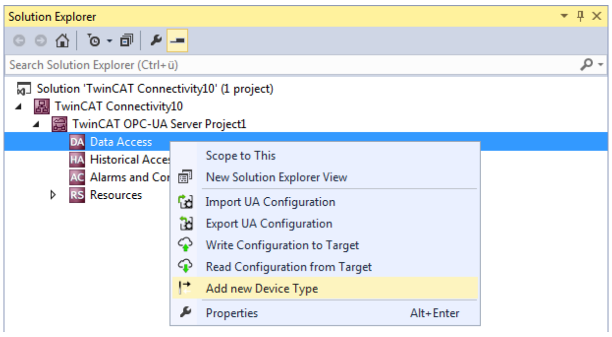
Requirements

Products	Setup versions	Target platform
TF6100	4.x.x	IPC or CX (x86, x64, ARM)

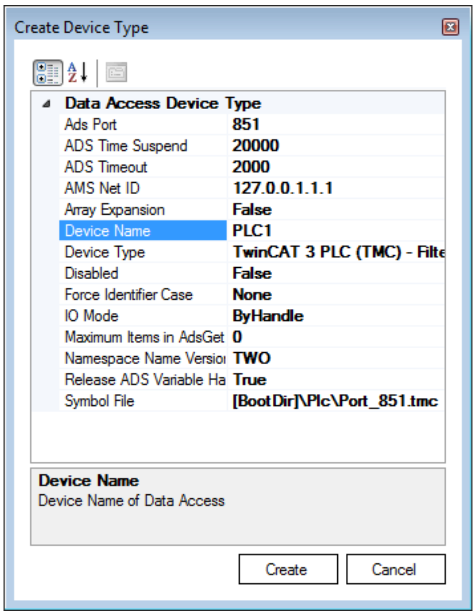
5.2.4 Adding ADS devices

The OPC UA Server can "talk" to one of more ADS devices. To establish a connection, a route to the respective ADS device is required. In the OPC UA Configurator, ADS devices are created, configured and thus announced to the OPC UA Server in the **Data Access** facet.

New ADS devices are added to the configuration via the context menu command **Add new Device Type**.


Solution Explorer
Search Solution Explorer (Ctrl+ü)
Solution 'TwinCAT Connectivity10' (1 project)
TwinCAT Connectivity10
TwinCAT OPC-UA Server Project1
Data Access
Historical Acce
Alarms and Co
Resources
Scope to This
New Solution Explorer View
Import UA Configuration
Export UA Configuration
Write Configuration to Target
Read Configuration from Target
Add new Device Type
Properties
Alt+Enter

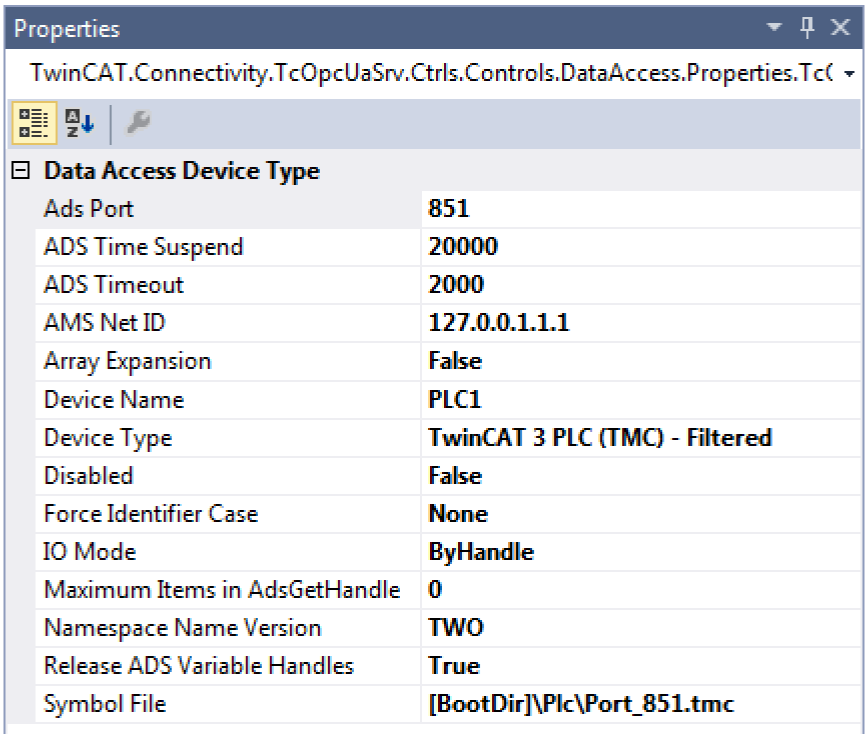
When the command is executed, a dialog box opens in which connection parameters can be configured for this device, e.g. AMS Net ID, ADS port or the symbol file.



Data Access Device Type	
Ads Port	851
ADS Time Suspend	20000
ADS Timeout	2000
AMS Net ID	127.0.0.1.1.1
Array Expansion	False
Device Name	PLC1
Device Type	TwinCAT 3 PLC (TMC) - Filte
Disabled	False
Force Identifier Case	None
IO Mode	ByHandle
Maximum Items in AdsGet	0
Namespace Name Versio	TWO
Release ADS Variable Ha	True
Symbol File	[BootDir]\Plc\Port_851.tmc

Device Name
Device Name of Data Access

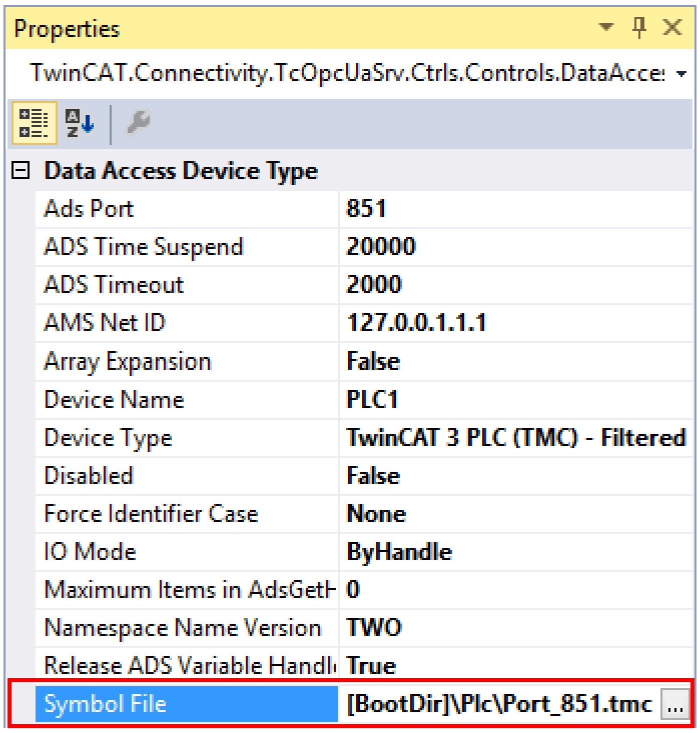
You can subsequently modify the connection parameters if necessary via the Properties window in Visual Studio.


Properties
TwinCAT.Connectivity.TcOpcUaSrv.Ctrls.Controls.DataAccess.Properties.Tc(

Data Access Device Type	
Ads Port	851
ADS Time Suspend	20000
ADS Timeout	2000
AMS Net ID	127.0.0.1.1.1
Array Expansion	False
Device Name	PLC1
Device Type	TwinCAT 3 PLC (TMC) - Filtered
Disabled	False
Force Identifier Case	None
IO Mode	ByHandle
Maximum Items in AdsGetHandle	0
Namespace Name Version	TWO
Release ADS Variable Handles	True
Symbol File	[BootDir]\Plc\Port_851.tmc

Selecting the symbol file

Symbol files that are present on the selected target device can be imported directly. These symbol files can be stored either in the TwinCAT boot directory or in the symbol directory of the OPC UA Server. You can select the files via the corresponding dialog during the symbol file configuration.

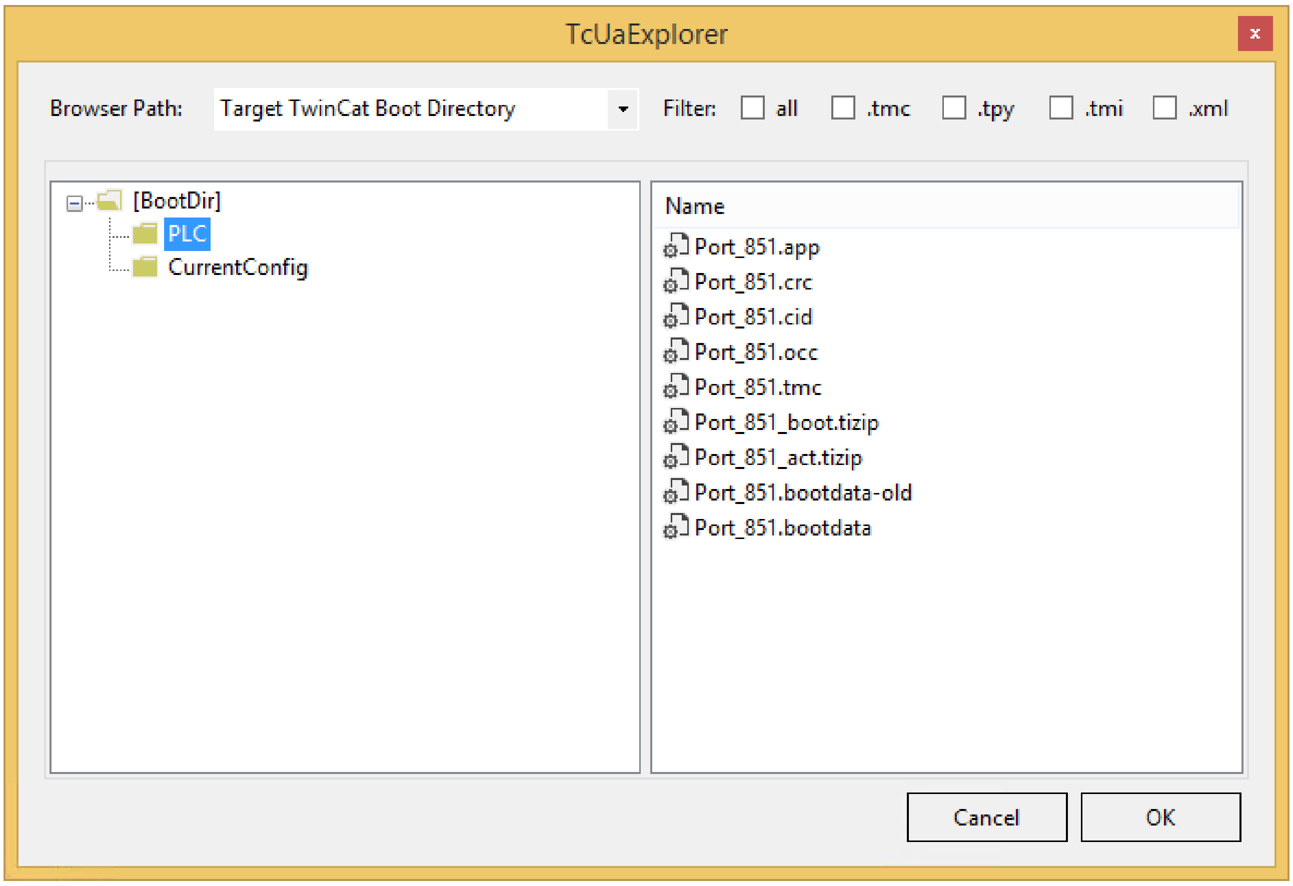

Properties

TwinCAT.Connectivity.TcOpcUaSrv.Ctrls.Controls.DataAcce:

Data Access Device Type

Ads Port	851
ADS Time Suspend	20000
ADS Timeout	2000
AMS Net ID	127.0.0.1.1.1
Array Expansion	False
Device Name	PLC1
Device Type	TwinCAT 3 PLC (TMC) - Filtered
Disabled	False
Force Identifier Case	None
IO Mode	ByHandle
Maximum Items in AdsGetH	0
Namespace Name Version	TWO
Release ADS Variable Handl	True
Symbol File	[BootDir]\Plc\Port_851.tmc

The TwinCAT OPC UA File Explorer can be connected to either the local TwinCAT directory or the remote boot directory. The latter can be read in via the configuration namespace of the server (see Configuring the namespace [▸ 132]).


TcUaExplorer

Browser Path: Target TwinCat Boot Directory

Filter: ☐ all ☐ .tmc ☐ .tpy ☐ .tmi ☐ .xml

[BootDir]
- PLC
- CurrentConfig

Name
- Port_851.app
- Port_851.crc
- Port_851.cid
- Port_851.occ
- Port_851.tmc
- Port_851_boot.tizip
- Port_851_act.tizip
- Port_851.bootdata-old
- Port_851.bootdata

Requirements

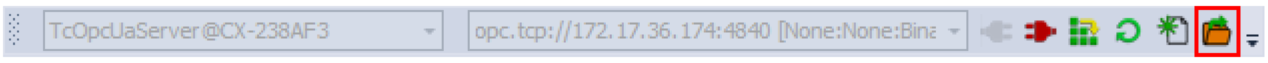
Products	Setup versions	Target platform
TF6100	4.x.x	IPC or CX (x86, x64, ARM)

5.2.5 Downloading and uploading configurations

Via the configurator you can initiate the download/upload of complete server configurations as well as loading every single facet (data access, historical access, etc.) individually to the target device and opening it there. The functions necessary for this are integrated both in the toolbar and in the context menu of the respective facet.

Opening a configuration from the target device

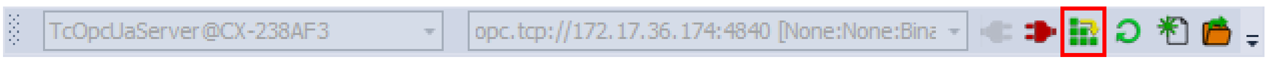
You can open the configuration of the selected target device via the corresponding button in the toolbar.



See also: Select target device [▶ 194]

Activating the configuration on a target device

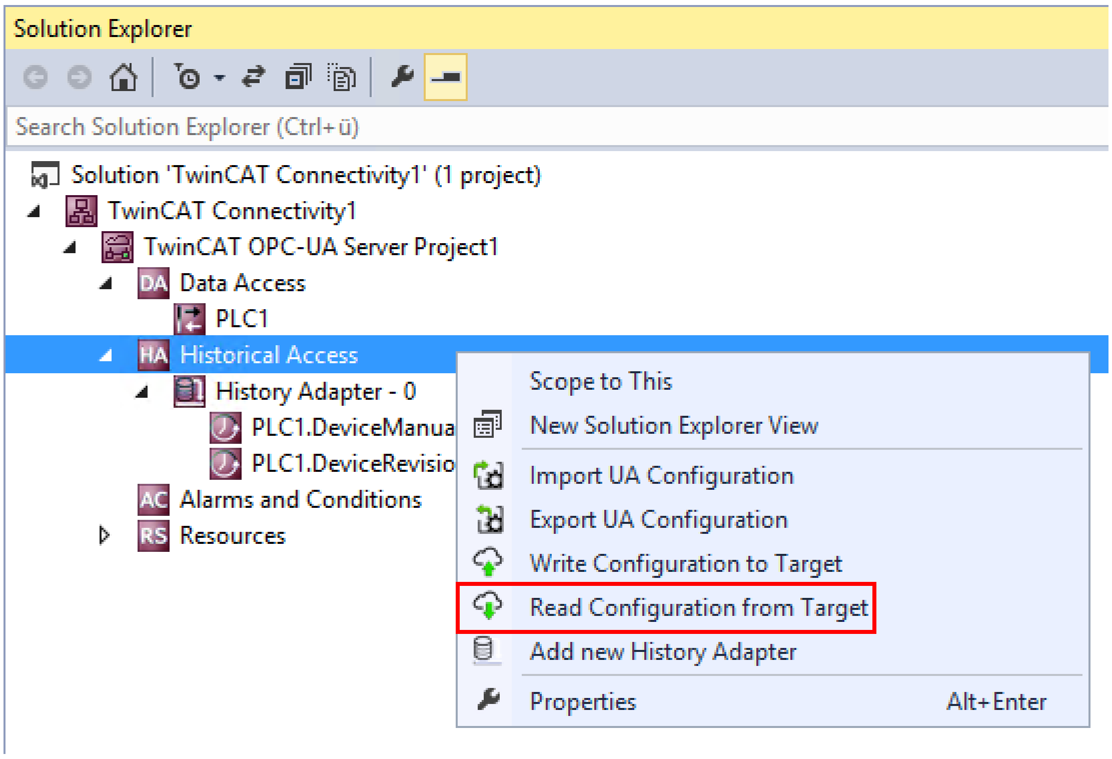
You can download the currently opened configuration to the selected target device using the corresponding button in the toolbar.



See also: Select target device [▶ 194]

Opening a partial configuration

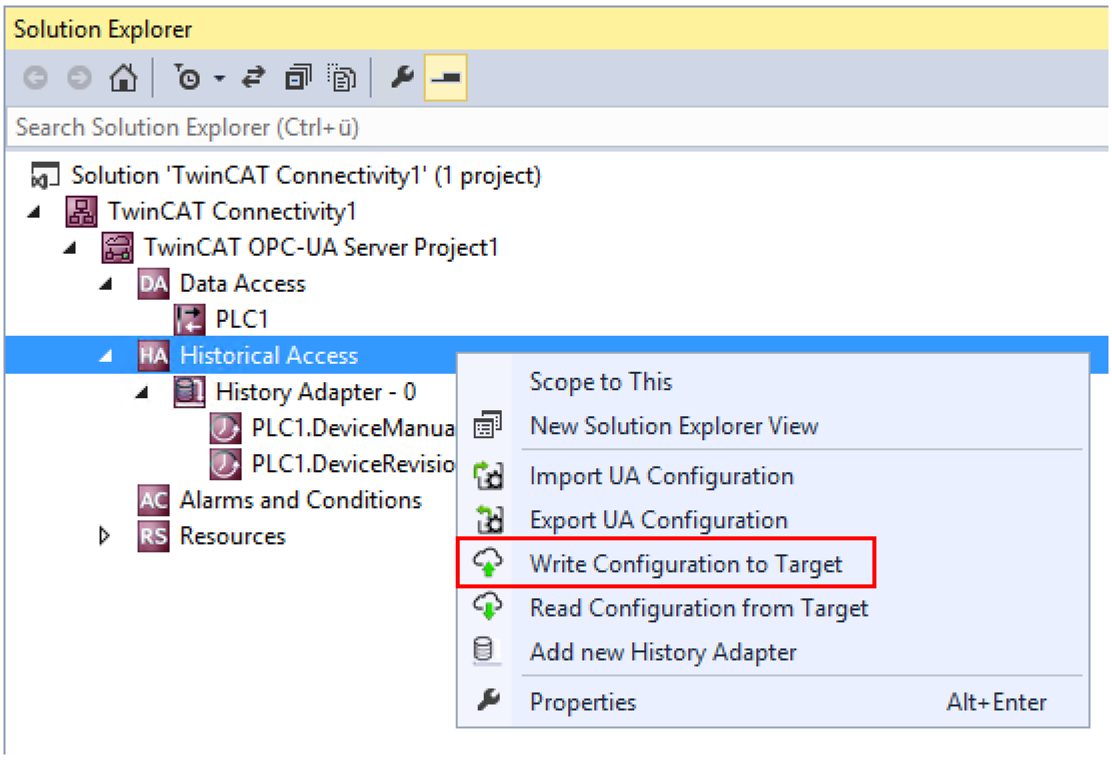
You can open the partial configuration of the selected target device using the command **Read Configuration from Target** in the context menu of a certain facet of the configuration.



See also: Select target device [▶ 194]

Downloading a partial configuration

You can download the partial configuration to the selected target device using the command **Write Configuration to Target** in the context menu of a certain facet of the configuration.



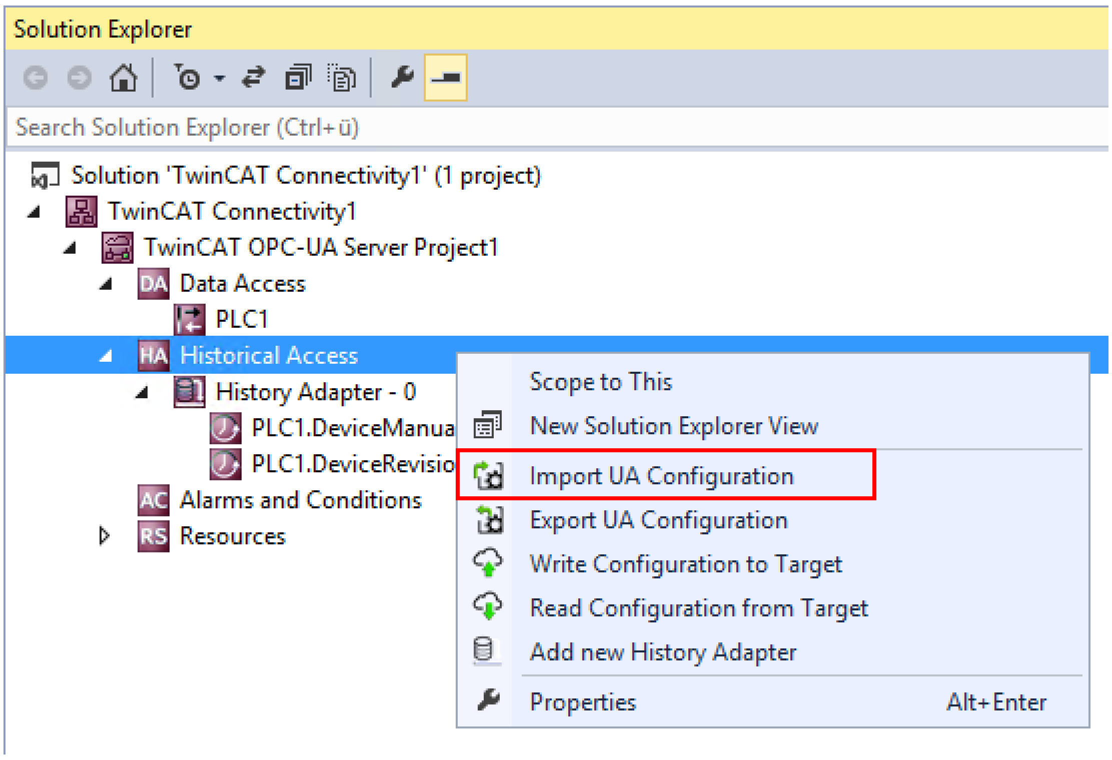
See also: Select target device [▶ 194]

5.2.6 Importing and exporting configuration files

The context menu commands enable the import/export of configuration files of the OPC UA Server.

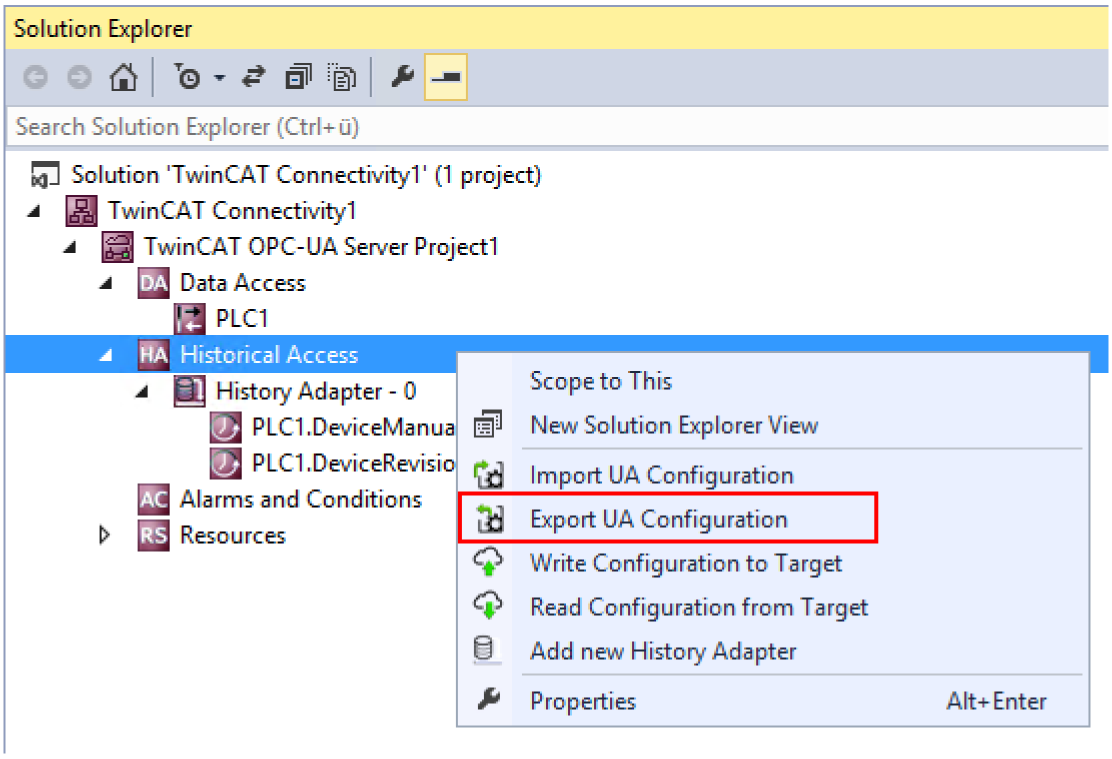
Importing a partial configuration

You can import the partial configuration (e.g. historical access) from an XML configuration file using the command **Import UA Configuration** in the context menu of a certain facet of the configuration.



Exporting a partial configuration

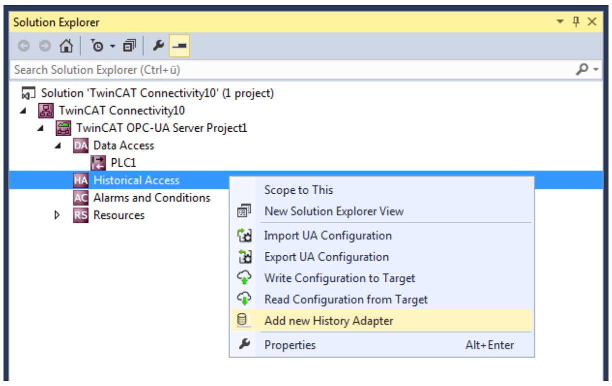
You can export the partial configuration (e.g. historical access) to an XML configuration file using the command **Export UA Configuration** in the context menu of a certain facet of the configuration.



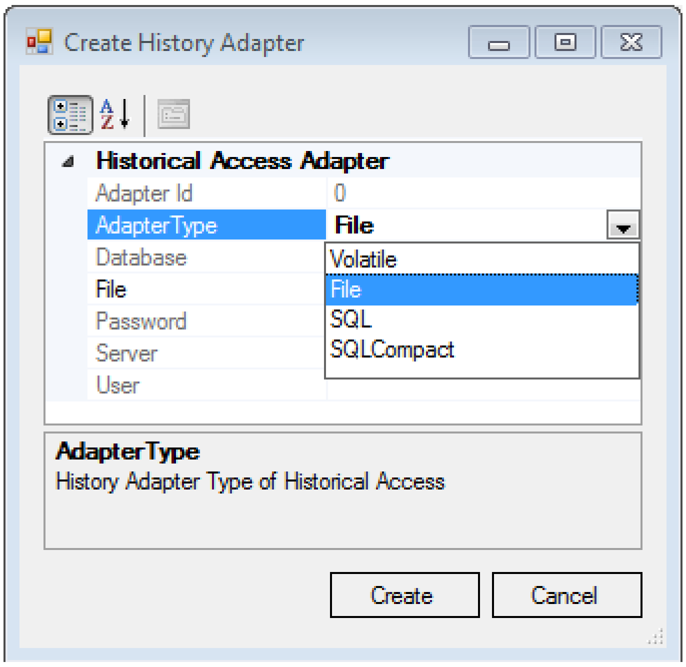
5.2.7 Configuring historical access

To configure Historical Access, you must first set up the History Adapters. These are the different locations for storing historical data, such as RAM, file, SQL Server.

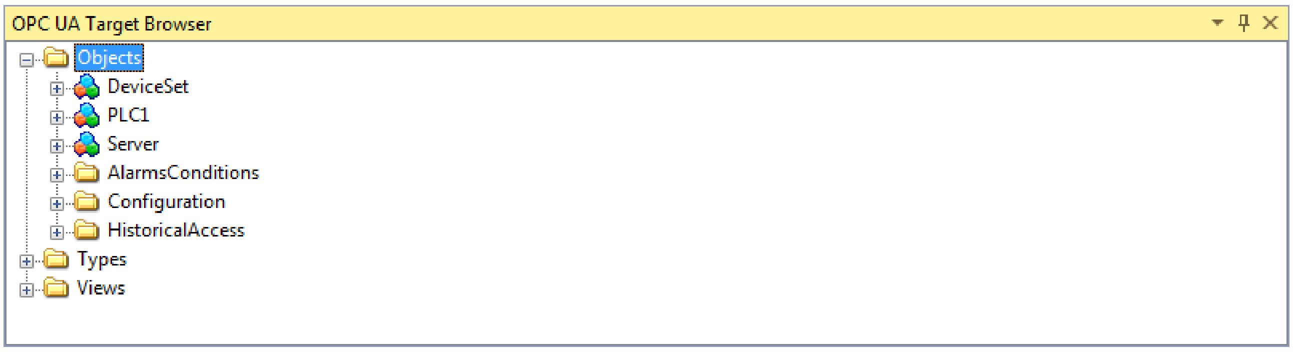
History adapters are added to the configuration using the context menu command **Add new History Adapter**.



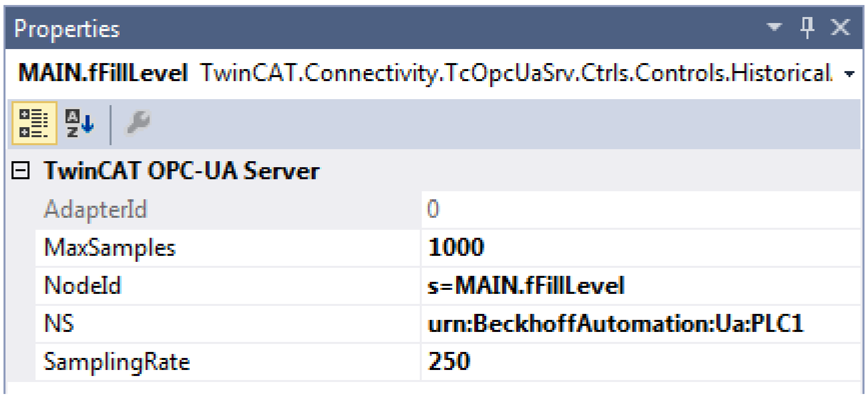
Depending on the adapter type you have to specify further parameters, e.g. the desired file storage path or the access data for the SQL Server.



After you have created a history adapter you can add the desired variables to the adapter. These variables must already exist on the selected OPC UA Server when the engineering is implemented. You can use the integrated **OPC UA Target Browser** to select the variables and then add the variables from the target browser to the history adapter by drag & drop.



Additional parameters can be specified in the properties window of the newly added variable, e.g. the desired SamplingRate or the size of the ring buffer to be used in the History Adapter.



Properties

MAIN.fFillLevel TwinCAT.Connectivity.TcOpcUaSrv.Ctrls.Controls.Historical.

TwinCAT OPC-UA Server

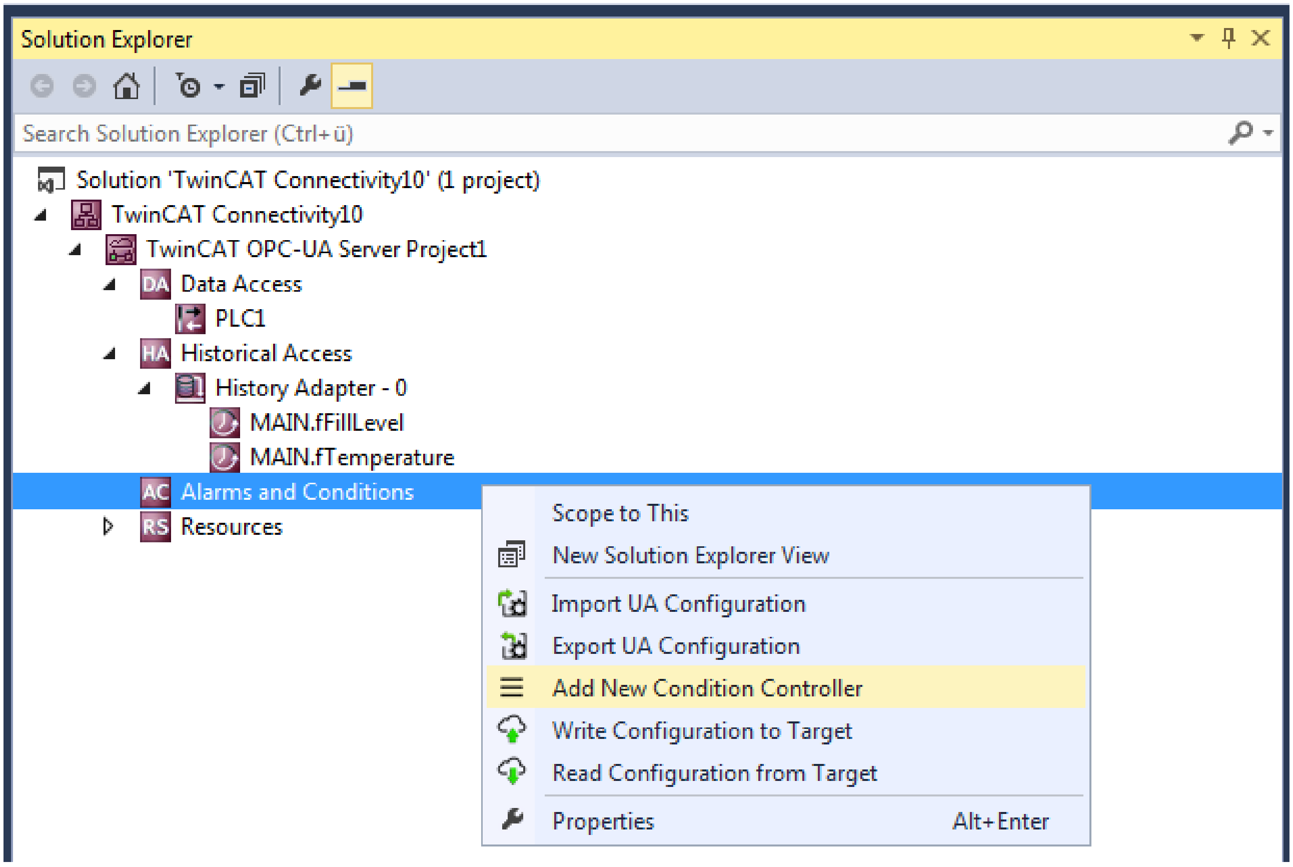
AdapterId	0
MaxSamples	1000
NodeId	s=MAIN.fFillLevel
NS	urn:BeckhoffAutomation:Ua:PLC1
SamplingRate	250

See also: Select target device [▶ 194]

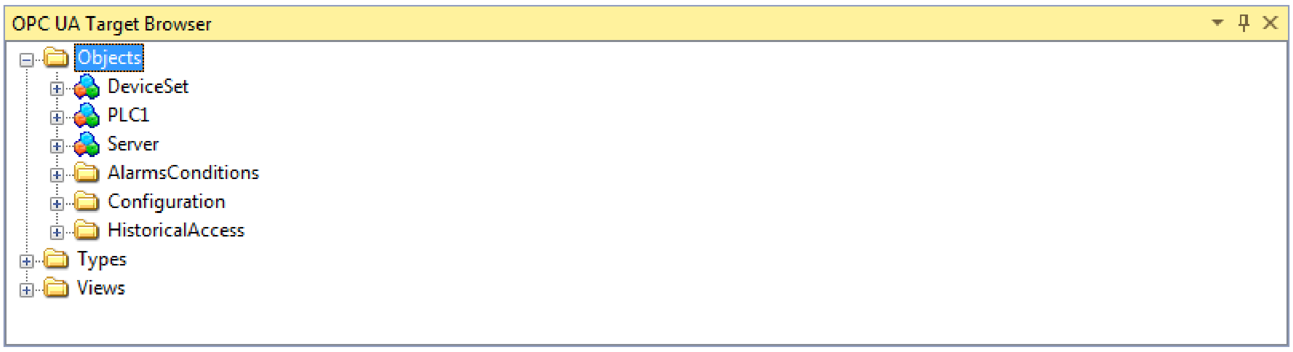
5.2.8 Configuring Alarms and Conditions

In order to configure Alarms and Conditions (A&C) you must first set up the Condition Controllers. These are container units that group together alarms.

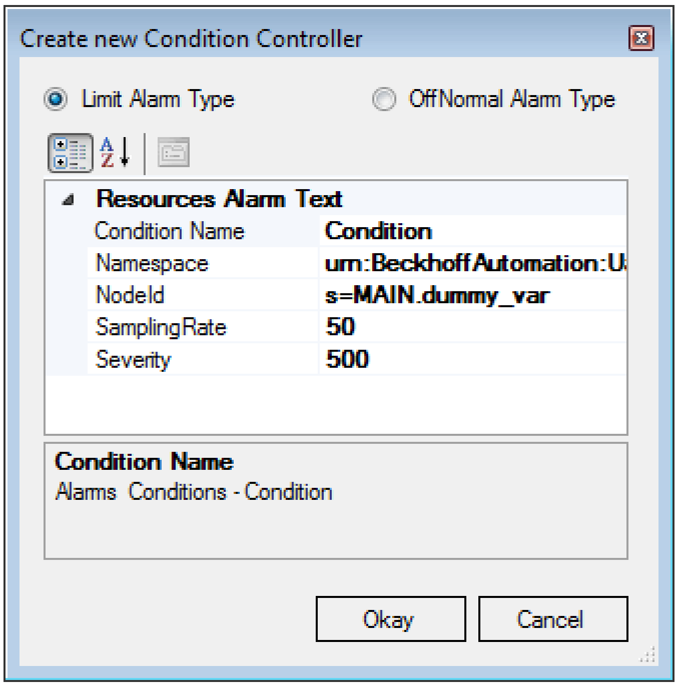
Condition Controllers are added to the configuration using the context menu command **Add New Condition Controller**.



After you have created a Condition Controller, you can add the desired variables to the controller and monitor them in the sense of alarms and conditions. A condition is created for each variable, which specifies the parameters for monitoring. These variables must already exist on the selected OPC UA Server when the engineering is implemented. You can use the integrated **OPC UA Target Browser** to select the variables and then add the variables from the target browser to the Condition Controller by drag & drop.



In the dialog window which then opens you can define the condition type and further parameters for the monitoring, e.g. SamplingRate and Severity.



Create new Condition Controller

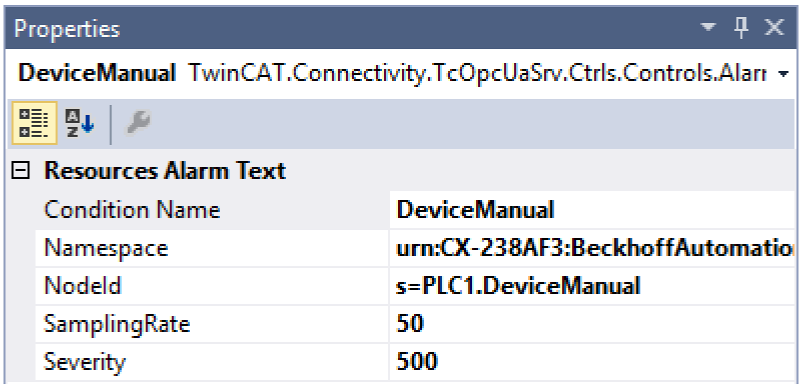
(● Limit Alarm Type) ○ OffNormal Alarm Type

Resources Alarm Text

Condition Name	Condition
Namespace	urn:BeckhoffAutomation:U
NodeId	s=MAIN.dummy_var
SamplingRate	50
Severity	500

Condition Name
Alarms Conditions - Condition

Depending on the selected condition type you can specify additional parameters in the properties window of the condition. The threshold values for the respective condition type are displayed as individual entries in the tree view of the configuration. Here too, you can configure the corresponding parameters in the properties window.

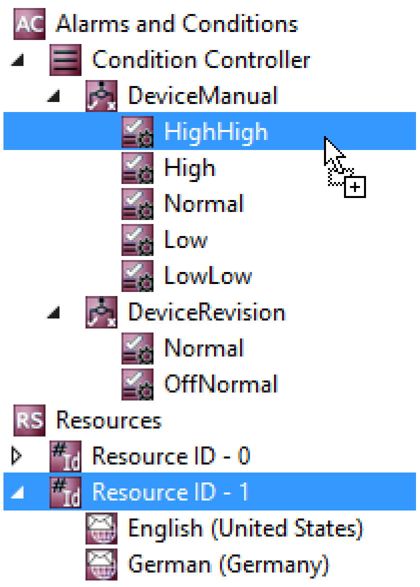


DeviceManual TwinCAT.Connectivity.TcOpcUaSrv.Ctrls.Controls.Aları

Resources Alarm Text

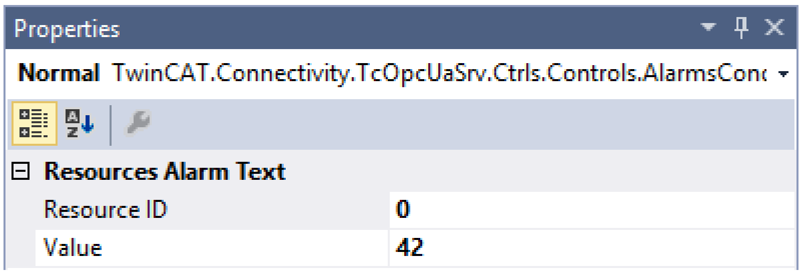
Condition Name	DeviceManual
Namespace	urn:CX-238AF3:BeckhoffAutomatio
NodeId	s=PLC1.DeviceManual
SamplingRate	50
Severity	500

Subsequently you have to define the alarm texts that are to be sent to the OPC UA Client when a condition is triggered. The section Configuring alarm texts [▸ 205] describes how alarm texts are created. You can drag and drop the alarm texts onto the respective threshold value of a condition.



Alarm type OffNormal

With an alarm type OffNormal you define a normal value that a variable should usually have. An alarm is triggered if the variable value deviates from this.


Properties
Normal TwinCAT.Connectivity.TcOpcUaSrv.Ctrls.Controls.AlarmsCon

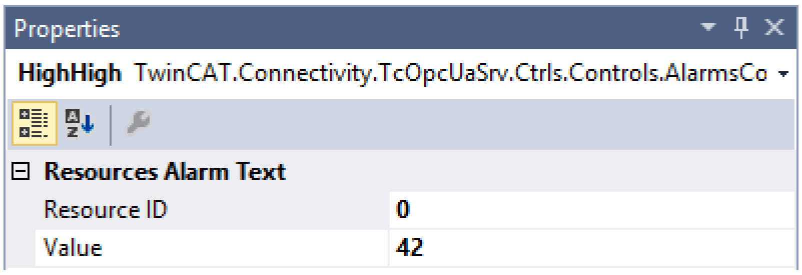
Resources Alarm Text

Resource ID	0
Value	42

After configuring the normal value you must select the alarm texts to be sent in the Resources area and drag and drop them onto the respective threshold value of the condition.

Alarm type Limit

With an alarm type Limit you define different threshold values upon whose reaching an alarm is to be sent.


Properties
HighHigh TwinCAT.Connectivity.TcOpcUaSrv.Ctrls.Controls.AlarmsCo

Resources Alarm Text

Resource ID	0
Value	42

After configuring the threshold values you must select the alarm texts to be sent in the Resources area and drag and drop them onto the respective threshold value of the condition.

Requirements

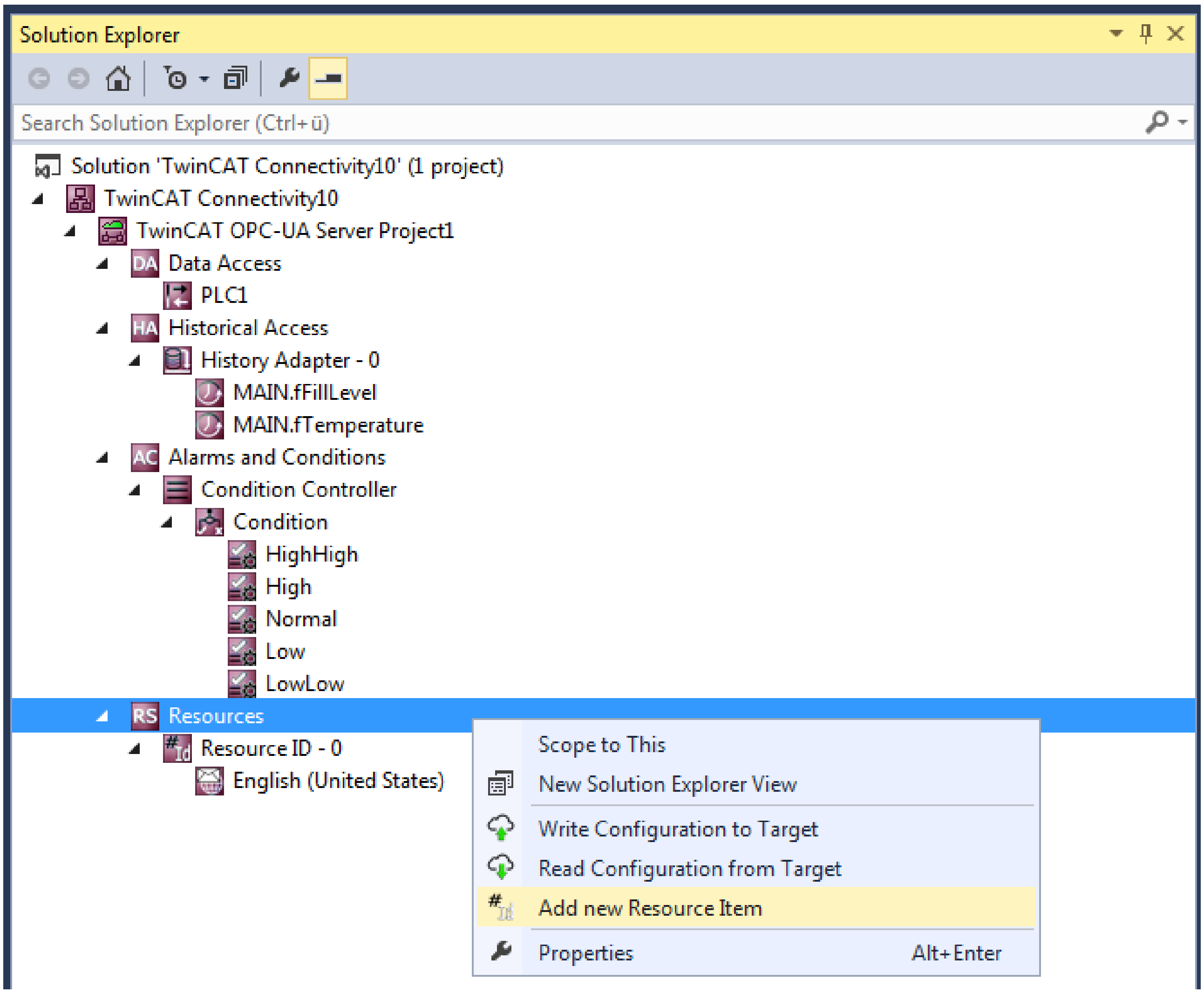
Products	Setup versions	Target platform
TF6100	4.x.x	IPC or CX (x86, x64, ARM)

See also: Select target device [▶ 194]

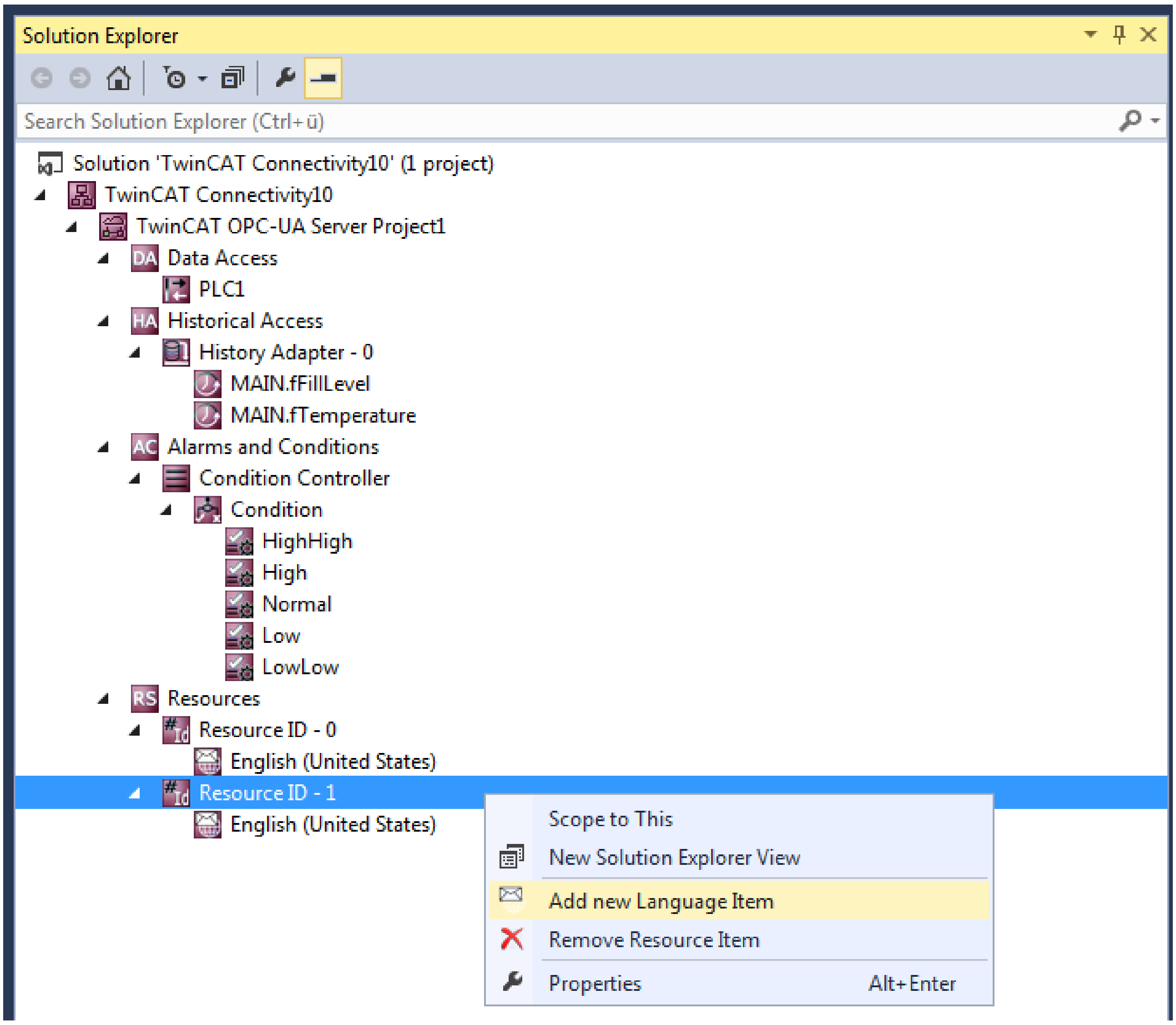
5.2.9 Configuring alarm texts

The OPC UA Configurator enables the (multilingual) management of alarm texts that are used, for example, with Alarms and Conditions [▶ 202]. The configuration of the alarm texts takes place in the **Resources** facet. Each alarm text is identified by a unique ID. Multiple language texts can then be assigned to this ID.

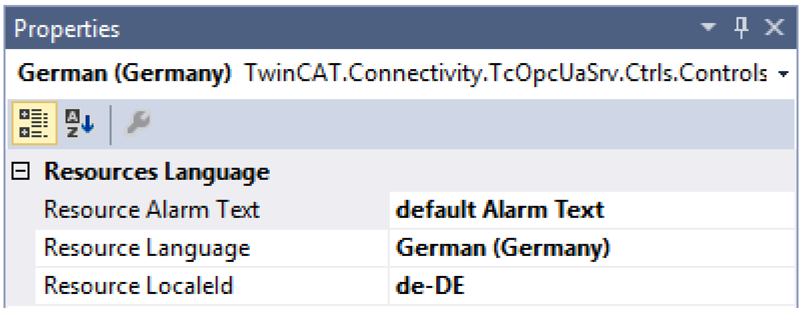
You can create so-called "resource items" using the context menu command **Add new Resource Item**.



You add new language items to a resource item using the command **Add new Language Item** in the resource item's context menu.



You can further parameterize a language item, e.g. the language text and the assigned language, in the properties window. When you define the language the associated LocaleID is automatically set. The LocaleID is requested by the OPC UA Client to indicate in which language it expects alarm texts.


Properties

German (Germany) TwinCAT.Connectivity.TcOpcUaSrv.Ctrls.Controls

Resources Language

Resource Alarm Text	default Alarm Text
Resource Language	German (Germany)
Resource LocaleId	de-DE

Requirements

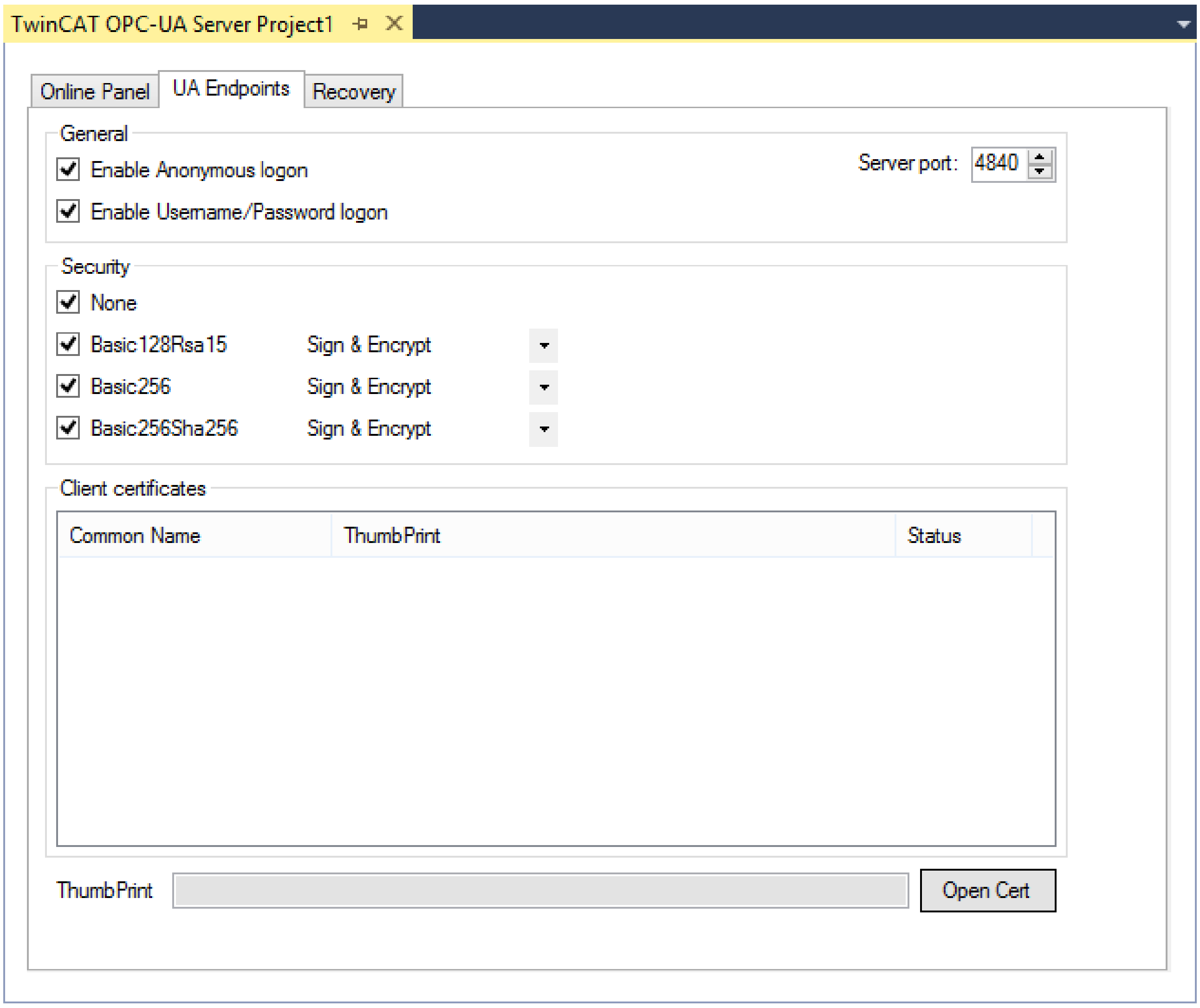
Products	Setup versions	Target platform
TF6100	4.x.x	IPC or CX (x86, x64, ARM)

5.2.10 Configuring endpoints

The endpoints of the OPC UA Server indicate which security mechanisms are to be used during the connection establishment of a client. These range from "unencrypted" to "encrypted and signed", based on different key strengths.

The endpoints can be activated and deactivated using the configurator. It may be useful to deactivate the unencrypted endpoint so that all clients can only connect themselves with valid certificates that are classified as trustworthy.

The endpoints are configured directly at the level of the OPC UA Server project. By double-clicking on the project you can make the corresponding settings on the **UA Endpoints** tab. The settings become effective after an activation of the configuration and a subsequent restart of the server (see Downloading and uploading configurations [▶ 198] and Restarting the server [▶ 216]).



General
- [✓] Enable Anonymous logon
- [✓] Enable Username/Password logon

Server port: 4840

Security
- [✓] None
- [✓] Basic128Rsa15 Sign & Encrypt
- [✓] Basic256 Sign & Encrypt
- [✓] Basic256Sha256 Sign & Encrypt

Client certificates

Common Name	ThumbPrint	Status

ThumbPrint

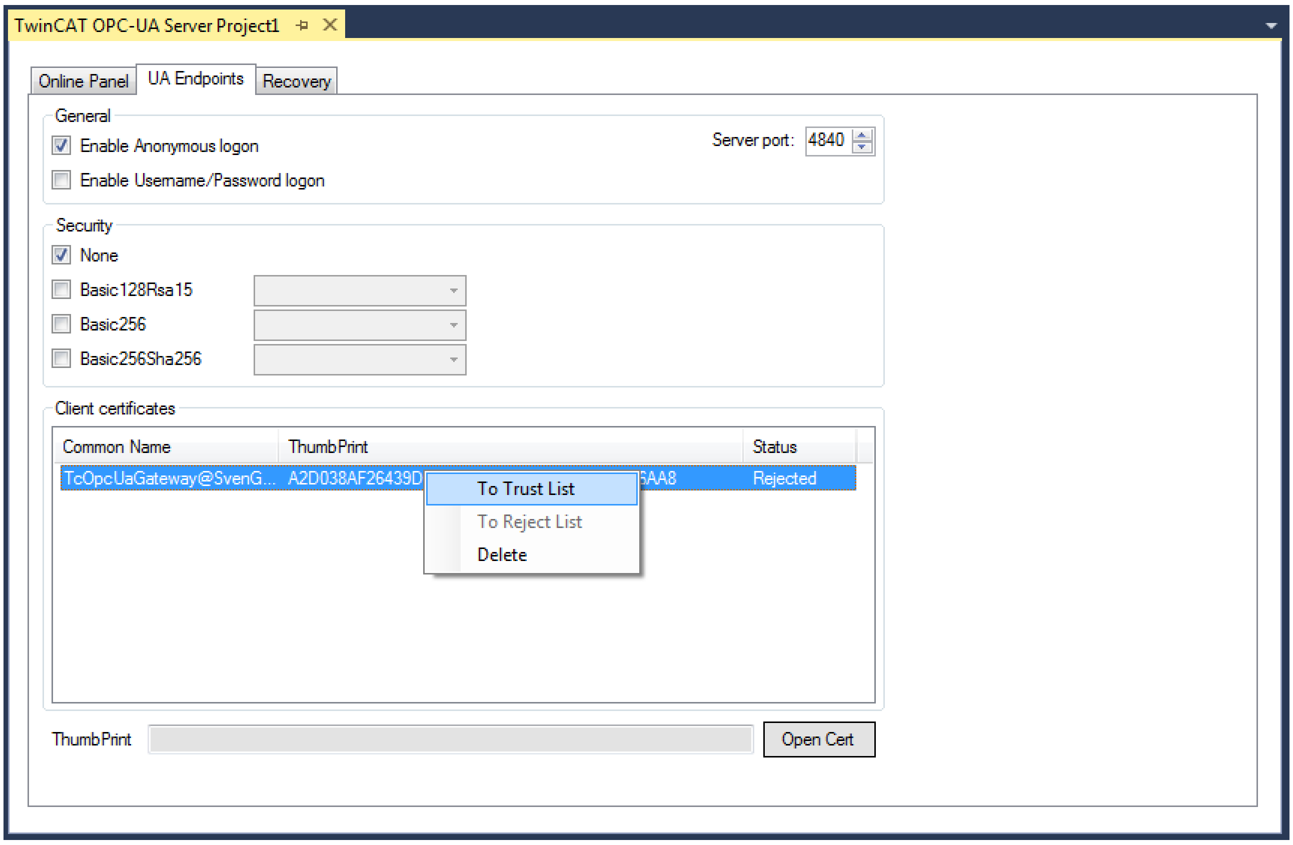
Requirements

Products	Setup versions	Target platform
TF6100	4.x.x	IPC or CX (x86, x64, ARM)

5.2.11 Configuring certificate trust settings

The Configurator facilitates management of the client certificates on the server. In the project settings you can classify the certificates as trustworthy or refuse them on the **UA Endpoints** tab in the **Client certificates** area.

After an OPC UA Client has attempted to connect to a secure server endpoint for the first time, the client certificate is deposited on the server and declared "rejected". The server administrator can subsequently enable the certificate. A subsequent connection attempt of the client with a secured endpoint will then be successful.


TwinCAT OPC-UA Server Project1

Online Panel | UA Endpoints | Recovery

General
- [✓] Enable Anonymous logon
- [ ] Enable Username/Password logon

Server port: 4840

Security
- [✓] None
- [ ] Basic128Rsa15
- [ ] Basic256
- [ ] Basic256Sha256

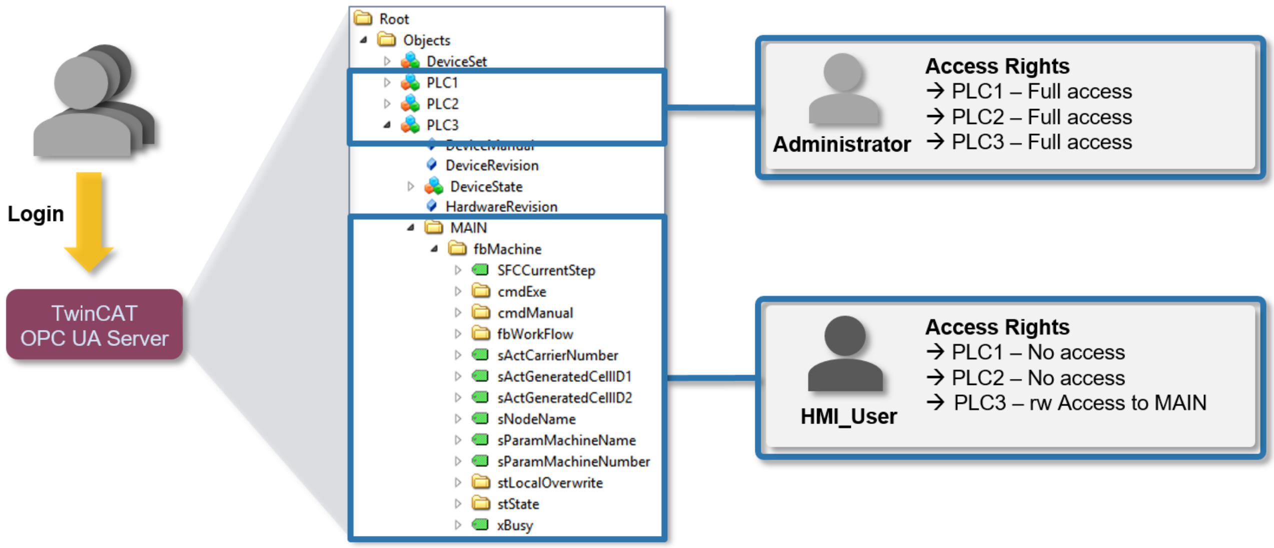
Client certificates

Common Name	ThumbPrint	Status
TcOpcUaGateway@SvenG...	A2D038AF26439D...5AA8	Rejected

ThumbPrint

5.2.12 Configuring security settings

The OPC UA Server enables the configuration of permissions at namespace and node level. This allows you to fine-granulate the access to ADS devices (for example, to different PLC runtimes) as well as variables. These security settings are available for all ADS devices that can be displayed in the server namespace.



Configuration

The permissions are configured on the basis of an XML-based configuration file (*TcUaSecurityConfig.xml*), which is located in the same directory as the server. The configuration file consists of the three areas "Users", "Groups" and "AccessInfos".

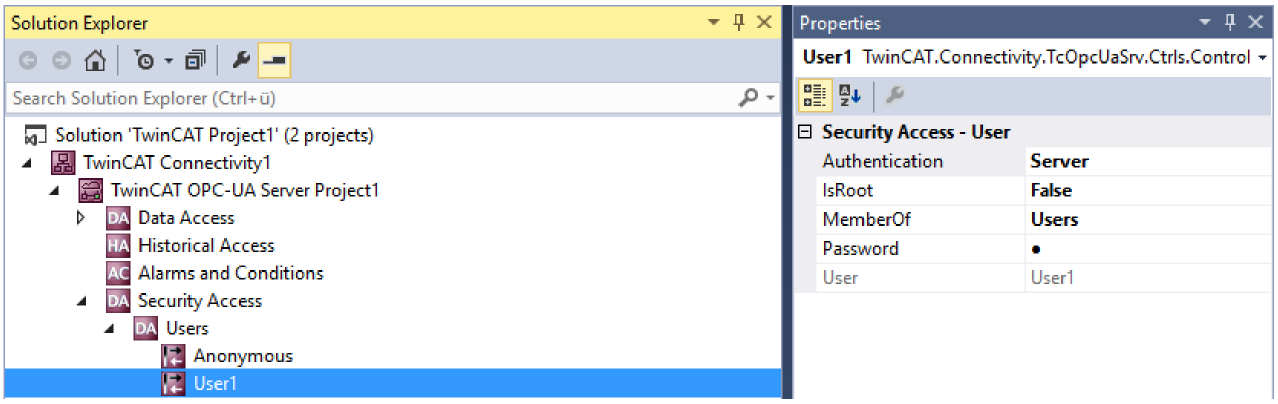
Users

In the "Users" area you can configure user accounts that are to be accepted by the OPC UA Server as logins. There are three different authentication methods:

OS (recommended authentication method)	The mechanisms of the operating system are used to validate user name and password. The user account is subject completely to the control of the operating system and/or domain.
Server (not recommended)	User name and password are known only to the OPC UA Server. Both pieces of information are stored in plain text in the XML file.
None	Only the user name of the server is evaluated, the password is ignored.

Users can be configured with a tag <DefaultAccess> that specifies the standard access of the user to a certain namespace.

Users can be members of one or more groups. You can specify this using the **MemberOf** attribute. In case of memberships of several groups, separate the groups by a semicolon.


Solution Explorer
Search Solution Explorer (Ctrl+ü)
Solution 'TwinCAT Project1' (2 projects)
TwinCAT Connectivity1
TwinCAT OPC-UA Server Project1
Data Access
Historical Access
Alarms and Conditions
Security Access
Users
Anonymous
User1
Properties
User1 TwinCAT.Connectivity.TcOpcUaSrv.Ctrls.Control
Security Access - User

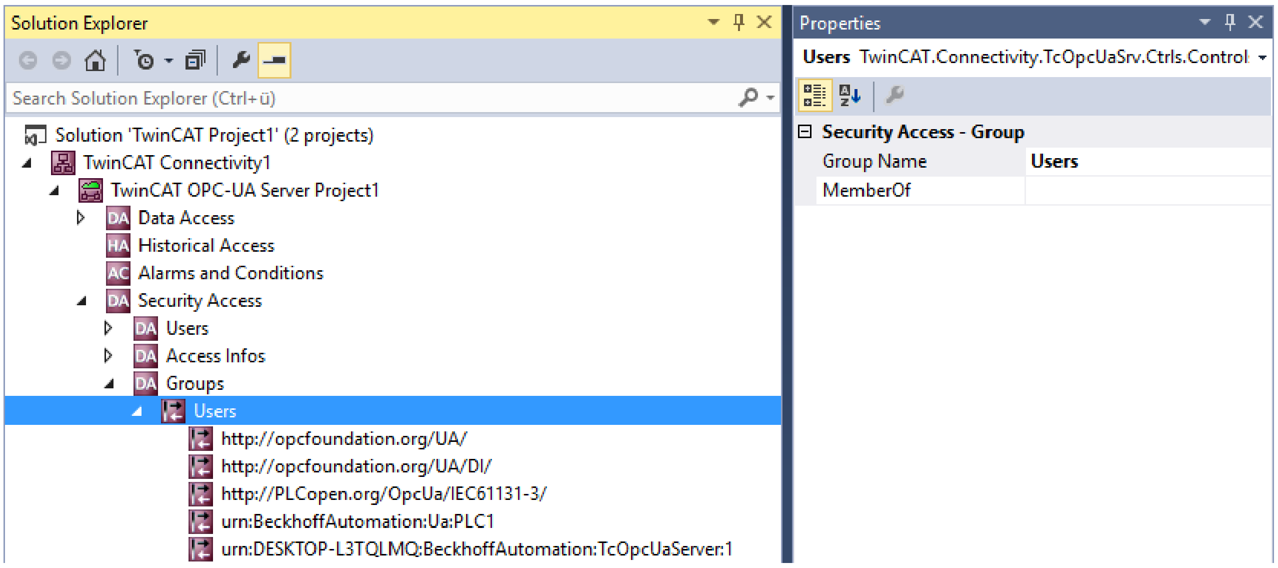
Authentication	Server
IsRoot	False
MemberOf	Users
Password	●
User	User1

Groups

In order to enable a simpler configuration with several user accounts, you can combine the users into groups.

Groups can also be configured with a tag <DefaultAccess>.

You can nest groups using the **MemberOf** attribute. In case of memberships of several groups, separate the groups by a semicolon.


Solution Explorer
Search Solution Explorer (Ctrl+ü)
Solution 'TwinCAT Project1' (2 projects)
TwinCAT Connectivity1
TwinCAT OPC-UA Server Project1
Data Access
Historical Access
Alarms and Conditions
Security Access
Users
Access Infos
Groups
Users
http://opcfoundation.org/UA/
http://opcfoundation.org/UA/DI/
http://PLCopen.org/OpcUa/IEC61131-3/
urn:BeckhoffAutomation:Ua:PLC1
urn:DESKTOP-L3TQLMQ:BeckhoffAutomation:TcOpcUaServer:1
Properties
Users TwinCAT.Connectivity.TcOpcUaSrv.Ctrls.Control
Security Access - Group

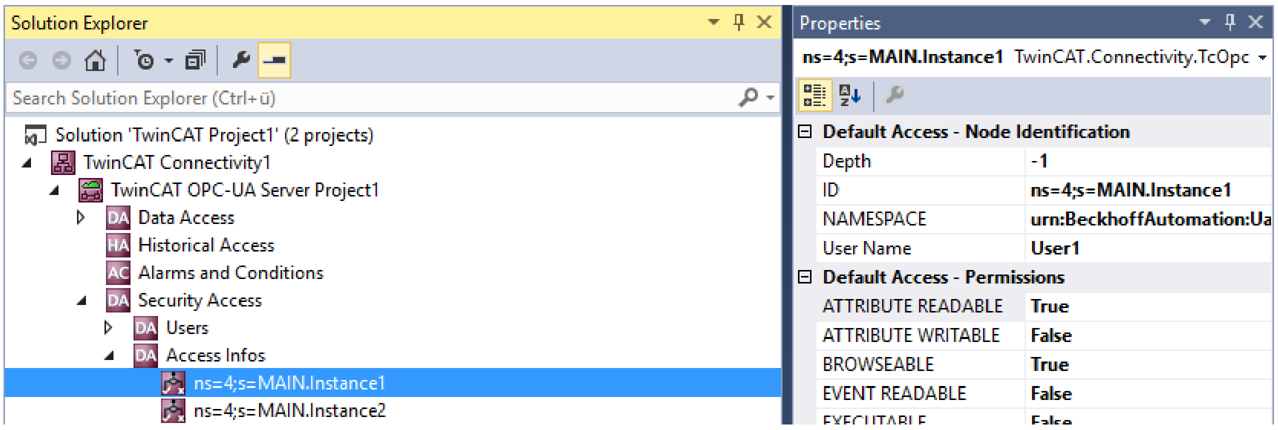
Group Name	Users
MemberOf	

AccessInfos

If a fine-granular setting of permissions at the node level is to be implemented, then <AccessInfos> can be configured additionally, which specify the access permissions on nodes. Access rights can be passed on to subelements. Although AccessInfos allow the most fine-grained configuration of permissions, such a configuration can quickly become confusing. Therefore, check whether configuring access rights at the namespace level (see above) is not sufficient.

The AccessInfo for a node contains the following settings:

NS	configures the NamespaceName in which the node is localized
Id	configures the identifier of the node, including the IdentifierType (e.g. s = String)
Depth	inheritance level of permissions (-1 for infinite)
User/Group	user or group that is to be given access to this node, including the AccessLevels

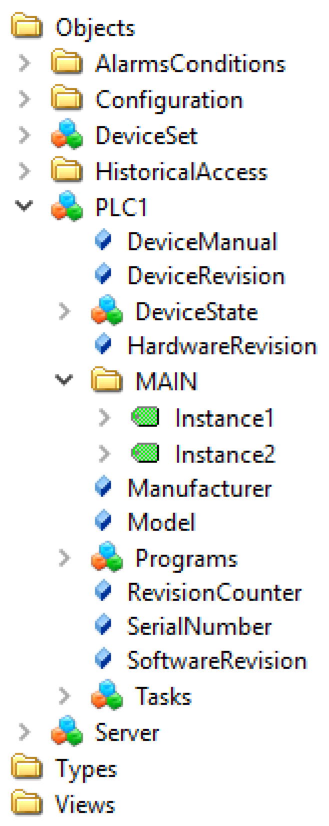


Default Access - Node Identification	
Depth	-1
ID	ns=4;s=MAIN.Instance1
NAMESPACE	urn:BeckhoffAutomation:Ua
User Name	User1
Default Access - Permissions	
ATTRIBUTE READABLE	True
ATTRIBUTE WRITABLE	False
BROWSEABLE	True
EVENT READABLE	False
EXECUTABLE	False

AccessInfos can be configured by dragging & dropping variables from the Target Browser. The configurable permissions are cumulative.

Example configuration

Let's take the following simple control program. The variables are already published in the OPC UA namespace of the server. The OPC UA Server is initially in the delivery state.



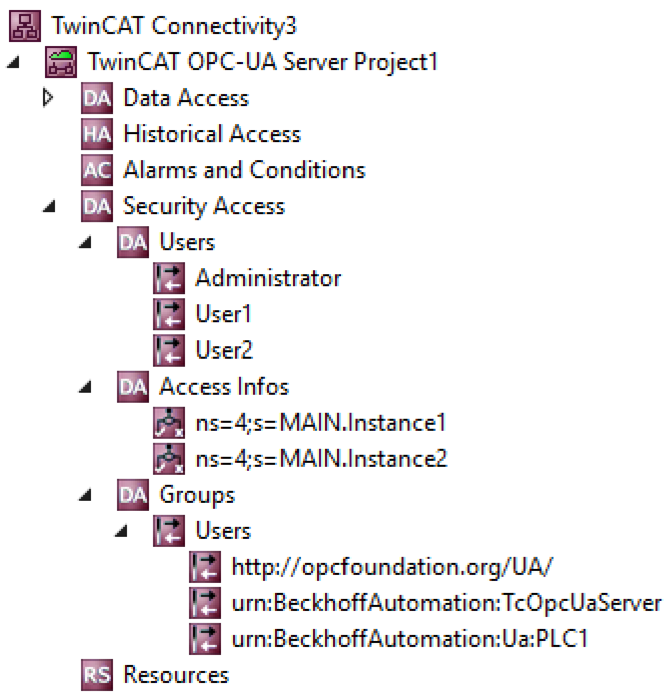
Access restrictions

Access to the server is to be restricted for clients as follows:

- Anonymous access is to be deactivated.
- There is to be a user - "Administrator" - who has full access to the complete server.
- There is to be a user - "User1" - who only has read access to MAIN.Instance1. The user should not come from the operating system here, but should only be used internally in the server.
- There is to be a user - "User2" - who only has read access to MAIN.Instance2. The user should not come from the operating system here, but should only be used internally in the server.
- General access permissions are to be configured for all users via a group called "Users".

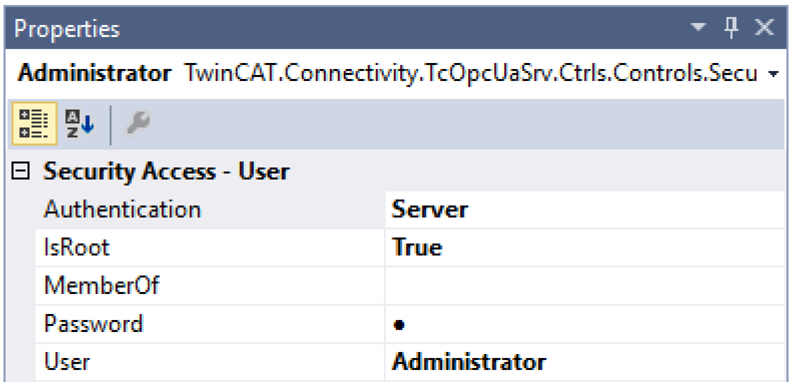
Settings

The configuration of the OPC UA Server is set as follows:



```
TwinCAT Connectivity3
◢ TwinCAT OPC-UA Server Project1
  ▷ Data Access
    Historical Access
    Alarms and Conditions
  ◢ Security Access
    ◢ Users
        Administrator
        User1
        User2
    ◢ Access Infos
        ns=4;s=MAIN.Instance1
        ns=4;s=MAIN.Instance2
    ◢ Groups
      ◢ Users
          http://opcfoundation.org/UA/
          urn:BeckhoffAutomation:TcOpcUaServer
          urn:BeckhoffAutomation:Ua:PLC1
    Resources
```

Settings for the user "Administrator":



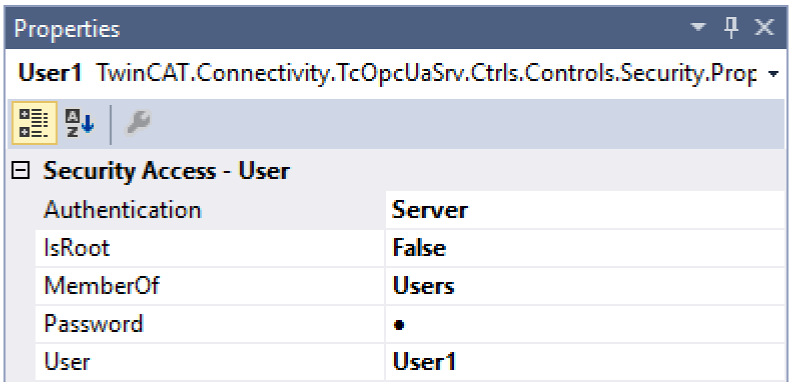
Properties

Administrator TwinCAT.Connectivity.TcOpcUaSrv.Ctrls.Controls.Secu

Security Access - User

Authentication	**Server**
IsRoot	**True**
MemberOf	
Password	•
User	**Administrator**

Settings for the user "User1":



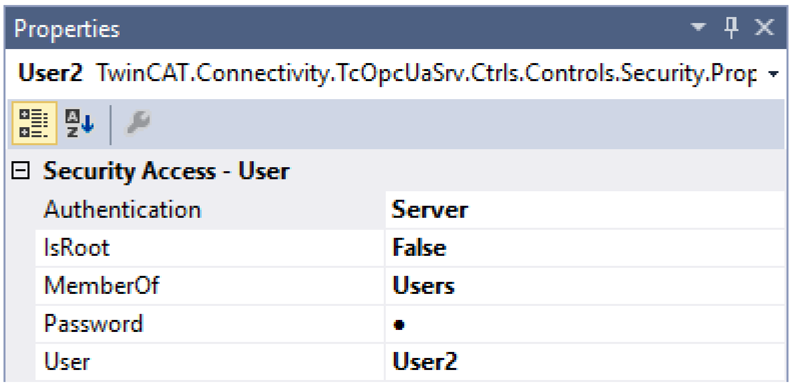
Properties

User1 TwinCAT.Connectivity.TcOpcUaSrv.Ctrls.Controls.Security.Prop

Security Access - User

Authentication	**Server**
IsRoot	**False**
MemberOf	**Users**
Password	•
User	**User1**

Settings for the user "User2":

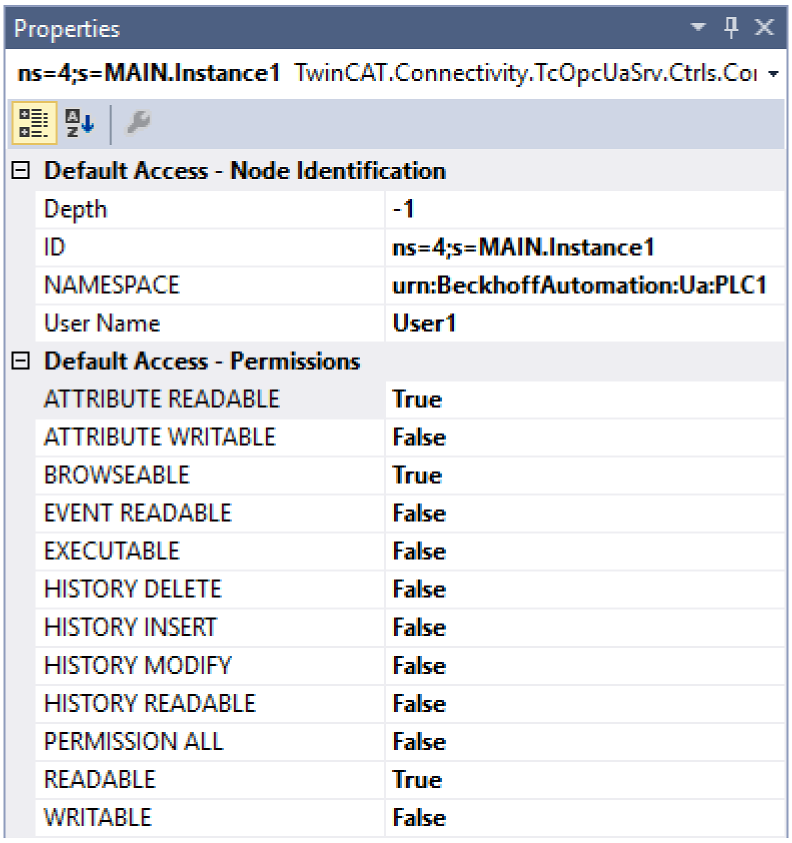


Properties

User2 TwinCAT.Connectivity.TcOpcUaSrv.Ctrls.Controls.Security.Prop

Security Access - User	
Authentication	Server
IsRoot	False
MemberOf	Users
Password	●
User	User2

Settings for AccessInfos "MAIN.Instance1":

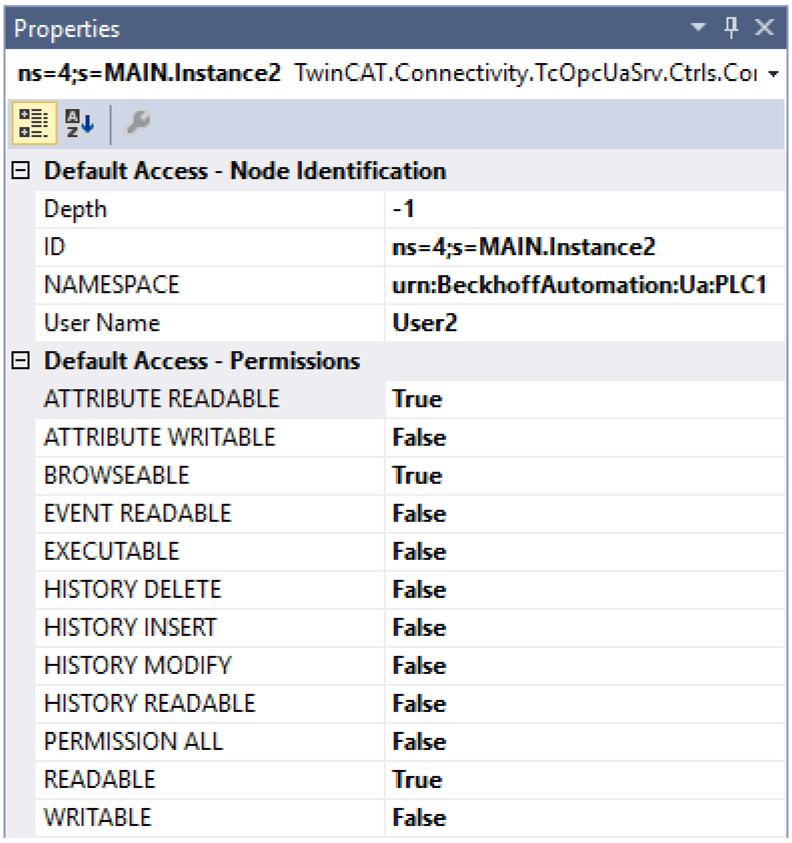


Properties

ns=4;s=MAIN.Instance1 TwinCAT.Connectivity.TcOpcUaSrv.Ctrls.Co

Default Access - Node Identification	
Depth	-1
ID	ns=4;s=MAIN.Instance1
NAMESPACE	urn:BeckhoffAutomation:Ua:PLC1
User Name	User1
Default Access - Permissions	
ATTRIBUTE READABLE	True
ATTRIBUTE WRITABLE	False
BROWSEABLE	True
EVENT READABLE	False
EXECUTABLE	False
HISTORY DELETE	False
HISTORY INSERT	False
HISTORY MODIFY	False
HISTORY READABLE	False
PERMISSION ALL	False
READABLE	True
WRITABLE	False

Settings for AccessInfos "MAIN.Instance2":



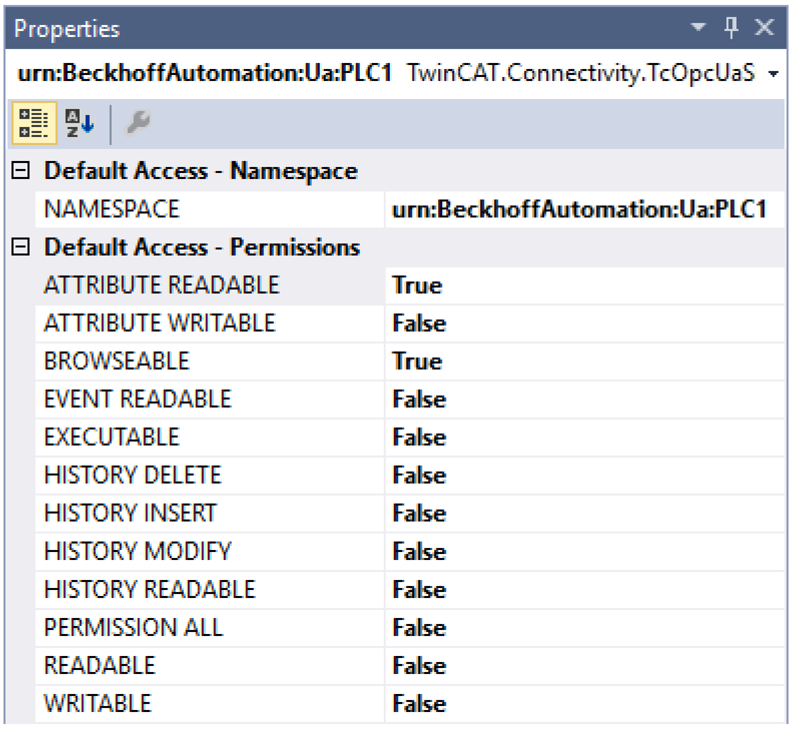
Properties

ns=4;s=MAIN.Instance2 TwinCAT.Connectivity.TcOpcUaSrv.Ctrls.Co

Default Access - Node Identification	
Depth	-1
ID	ns=4;s=MAIN.Instance2
NAMESPACE	urn:BeckhoffAutomation:Ua:PLC1
User Name	User2
Default Access - Permissions	
ATTRIBUTE READABLE	True
ATTRIBUTE WRITABLE	False
BROWSEABLE	True
EVENT READABLE	False
EXECUTABLE	False
HISTORY DELETE	False
HISTORY INSERT	False
HISTORY MODIFY	False
HISTORY READABLE	False
PERMISSION ALL	False
READABLE	True
WRITABLE	False

Settings for the group "Users":

The user group is equipped both with basic access to required server and type system namespaces and with read and browse permissions to the PLC1 namespace.



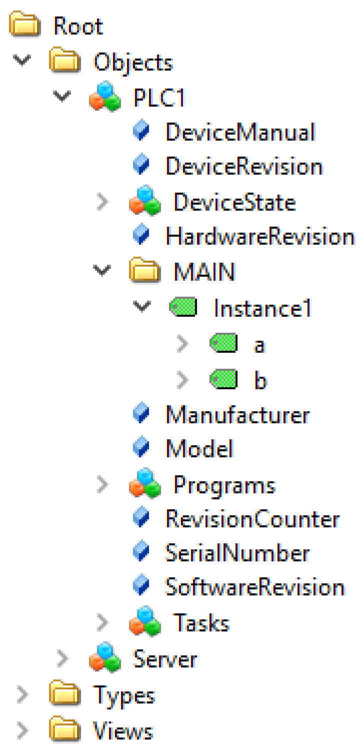
Properties

urn:BeckhoffAutomation:Ua:PLC1 TwinCAT.Connectivity.TcOpcUaS

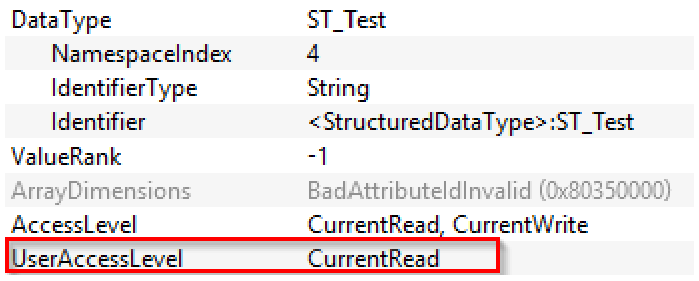
Default Access - Namespace	
NAMESPACE	urn:BeckhoffAutomation:Ua:PLC1
Default Access - Permissions	
ATTRIBUTE READABLE	True
ATTRIBUTE WRITABLE	False
BROWSEABLE	True
EVENT READABLE	False
EXECUTABLE	False
HISTORY DELETE	False
HISTORY INSERT	False
HISTORY MODIFY	False
HISTORY READABLE	False
PERMISSION ALL	False
READABLE	False
WRITABLE	False

Result

Following activation of the configuration, the namespace of the server for "User1" looks like the following after establishment of a connection:

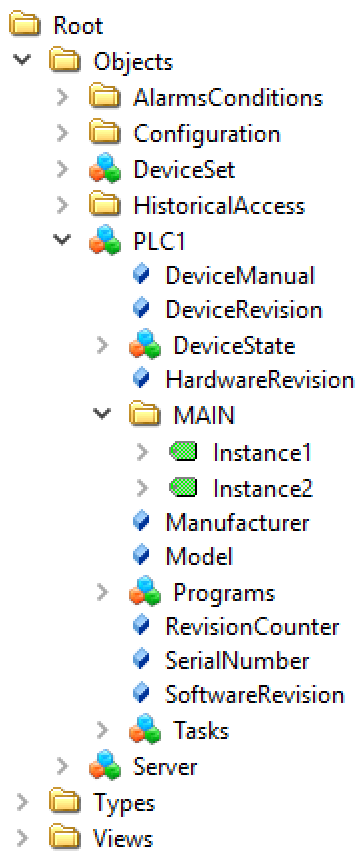


The user has only read rights to the node "Instance1", which is clear from the attribute UserAccessLevel:



DataType	ST_Test
NamespaceIndex	4
IdentifierType	String
Identifier	<StructuredDataType>:ST_Test
ValueRank	-1
ArrayDimensions	BadAttributeIdInvalid (0x80350000)
AccessLevel	CurrentRead, CurrentWrite
UserAccessLevel	CurrentRead

The user "Administrator", conversely, has full access rights to all elements of the namespace:


Root
Objects
AlarmsConditions
Configuration
DeviceSet
HistoricalAccess
PLC1
DeviceManual
DeviceRevision
DeviceState
HardwareRevision
MAIN
Instance1
Instance2
Manufacturer
Model
Programs
RevisionCounter
SerialNumber
SoftwareRevision
Tasks
Server
Types
Views

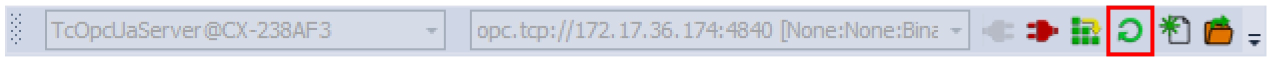
5.2.13 Restarting the server

The OPC UA Configurator enables the triggering of a restart of the OPC UA Server. This can be done locally or remotely and refers to the selected target device.

Loss of connection

A restart of the OPC UA Server always leads to a loss of the connection of all connected clients.

The restart is triggered via the toolbar.


TcOpcUaServer@CX-238AF3
opc.tcp://172.17.36.174:4840 [None:None:Bina

Requirements

Products	Setup versions	Target platform
TF6100	4.x.x	IPC or CX (x86, x64, ARM)

See also: Select target device [▶ 194]

5.2.14 Troubleshooting and logging

For an advanced diagnosis you can activate the logging function of the OPC UA Server.

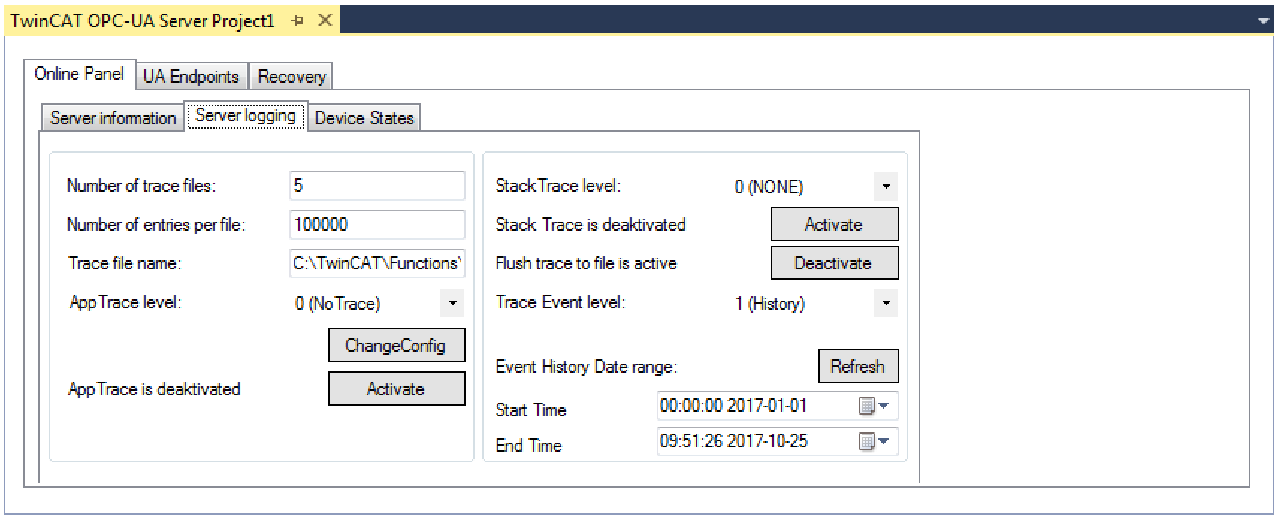
Writing the log file

Activating the logging function on the server causes a log file to be written on the file system. Make sure that there is sufficient memory space available and set the logging parameters accordingly (number of log files, size per log file).

Performance and timing behavior

Activation of the logging function will change the timing behavior of the OPC UA Server. As a result there may be losses of speed, depending on the platform and project.

The logging function is activated using the **Activate** button on the **Online Panel** tab in the project configurator. You can activate the function locally or remotely depending on the selected target device. The logging function remains active until it is deactivated again via the configurator or until the OPC UA Server is restarted.


TwinCAT OPC-UA Server Project1

Online Panel | UA Endpoints | Recovery

Server information | Server logging | Device States

Number of trace files: 5
Number of entries per file: 100000
Trace file name: C:\TwinCAT\Functions\
AppTrace level: 0 (NoTrace)
ChangeConfig
AppTrace is deaktivated — Activate

StackTrace level: 0 (NONE)
Stack Trace is deaktivated — Activate
Flush trace to file is active — Deactivate
Trace Event level: 1 (History)
Event History Date range: Refresh
Start Time 00:00:00 2017-01-01
End Time 09:51:26 2017-10-25

Activate App Trace

In most cases it is sufficient to create a so-called "AppTrace". The following applies: The higher the "trace level", the more detailed (and more) data is written.

Activate Stack Trace

In a few cases it is also necessary to create a so-called "StackTrace". The same applies here: The higher the "trace level", the more detailed (and more) data is written.

Requirements

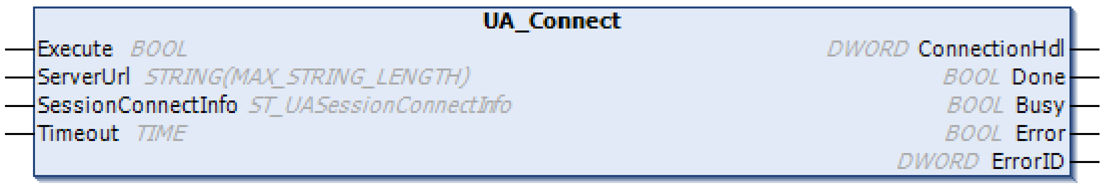
Products	Setup versions	Target platform
TF6100	4.x.x	IPC or CX (x86, x64, ARM)

See also: Selecting a target device [▶ 194]

6 PLC API

6.1 Function blocks

6.1.1 UA_Connect



This function block establishes an OPC UA Remote connection to another OPC UA Server, which is specified via ServerUrl and SessionConnectInfo. The function block returns a connection handle that can be used for other function blocks, such as UA_Read.

VAR_INPUT

```
VAR_INPUT
    Execute            : BOOL;
    ServerUrl          : STRING(MAX_STRING_LENGTH);
    SessionConnectInfo : ST_UASessionConnectInfo;
    Timeout            : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

Execute: The command is triggered by a positive edge at this input.

ServerUrl: OPC UA Server URL, i.e. 'opc.tcp://172.16.3.207:4840' or 'opc.tcp://CX_0193BF:4840'.

SessionConnectInfo: Connection information (see ST_UASessionConnectInfo [▶ 235])

Timeout: Time until the function is aborted. DEFAULT_ADS_TIMEOUT is a global constant, set to 5 seconds. ST_UASessionConnectInfo.tSessionTimeout must be shorter than this timeout.

VAR_OUTPUT

```
VAR_OUTPUT
    ConnectionHdl : DWORD;
    Done          : BOOL;
    Busy          : BOOL;
    Error         : BOOL;
    ErrorID       : DWORD;
END_VAR
```

ConnectionHdl: OPC UA connection handle.

Done: Switches to TRUE if the function block was executed successfully.

Busy: TRUE until the function block has executed a command, at the most for the duration of the "Timeout" at the input. The inputs accept no new command as long as Busy = TRUE. It is not the connection time that is monitored but the reception time.

Error: Switches to TRUE if an error occurs while executing a command. The command-specific error code is included in ErrorID.

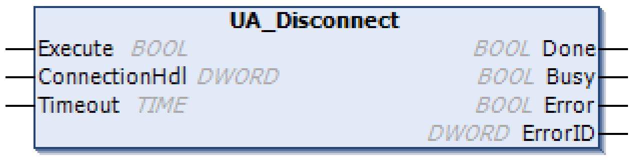
ErrorID: Contains the command-specific error code of the most recently executed command.

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT 3.1	Win32, Win64, CE-X86, CE-ARM	Tc3_PLCopen_OpcUa

Development environment	Target platform	PLC libraries to include
TwinCAT 2.11 R3 Build 2245	Win32, CE-X86, CE-ARM	Tc2_PLCopen_OpcUa

6.1.2 UA_Disconnect



This function block closes an OPC UA Remote connection to another OPC UA Server. The connection is specified via its connection handle.

VAR_INPUT

```
VAR_INPUT
    Execute            : BOOL;
    ConnectionHdl      : DWORD;
    Timeout            : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

Execute: The command is triggered by a positive edge at this input.

ConnectionHdl: Connection handle previously output by the function block UA_Connect [▶ 218].

Timeout: Time until the function is aborted. DEFAULT_ADS_TIMEOUT is a global constant, set to 5 seconds.

VAR_OUTPUT

```
VAR_OUTPUT
    Done        : BOOL;
    Busy        : BOOL;
    Error       : BOOL;
    ErrorID     : DWORD;
END_VAR
```

Done: Switches to TRUE if the function block was executed successfully.

Busy: TRUE until the function block has executed a command, at the most for the duration of the "Timeout" at the input. The inputs accept no new command as long as Busy = TRUE. It is not the connection time that is monitored but the reception time.

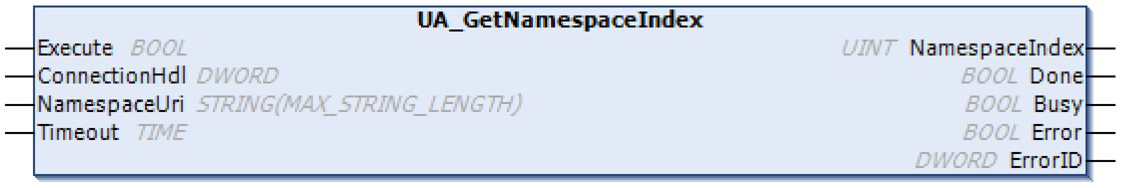
Error: Switches to TRUE if an error occurs while executing a command. The command-specific error code is contained in nErrID.

ErrorID: Contains the command-specific error code of the most recently executed command.

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT 3.1	Win32, Win64, CE-X86, CE-ARM	Tc3_PLCopen_OpcUa
TwinCAT 2.11 R3 Build 2245	Win32, CE-X86, CE-ARM	Tc2_PLCopen_OpcUa

6.1.3 UA_GetNamespaceIndex



This function block collects the namespace index for a namespace URI. The namespace index is required for identifying symbols, for example, if the function blocks UA_Read [▸ 230]or UA_Write [▸ 232]are used.

VAR_INPUT

```
VAR_INPUT
    Execute       : BOOL;
    ConnectionHdl : DWORD;
    NamespaceUri  : STRING(MAX_STRING_LENGTH);
    Timeout       : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

Execute: The command is triggered by a positive edge at this input.

ConnectionHdl: Connection handle previously output by the function block UA_Connect [▸ 218].

NamespaceUri: Namespace URI to be resolved. For the TwinCAT OPC UA Server, this is "PLC1" for the first PLC runtime.

Timeout: Time until the function is aborted. DEFAULT_ADS_TIMEOUT is a global constant, set to 5 seconds.

VAR_OUTPUT

```
VAR_OUTPUT
    NamespaceIndex : UINT;
    Done           : BOOL;
    Busy           : BOOL;
    Error          : BOOL;
    ErrorID        : DWORD;
END_VAR
```

NamespaceIndex: Namespace Index of the given namespace URI. This can be used in other function blocks, e.g. UA_NodeGetHandle or UA_MethodGetHandle.

Done: Switches to TRUE if the function block was executed successfully.

Busy: TRUE until the function block has executed a command, at the most for the duration of the "Timeout" at the input. The inputs accept no new command as long as Busy = TRUE. It is not the connection time that is monitored but the reception time.

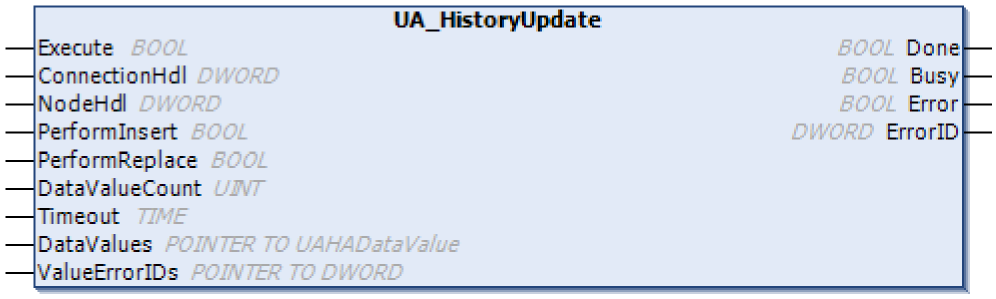
Error: Switches to TRUE if an error occurs while executing a command. The command-specific error code is included in ErrorID.

ErrorID: Contains the command-specific error code of the most recently executed command.

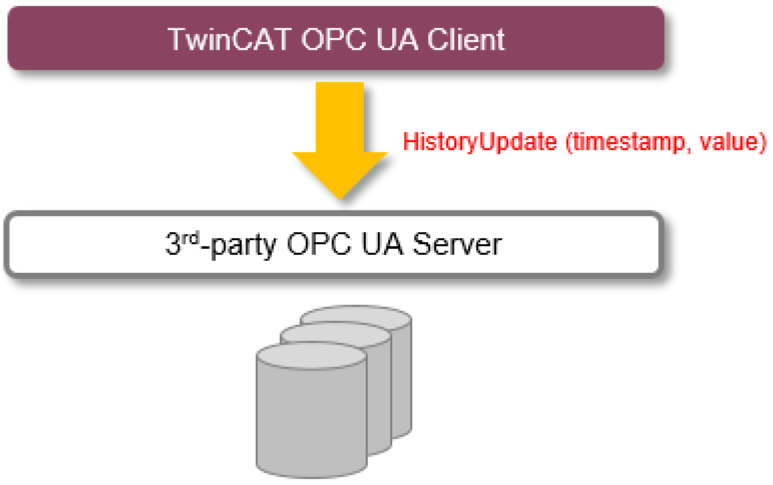
Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT 3.1	Win32, Win64, CE-X86, CE-ARM	Tc3_PLCopen_OpcUa
TwinCAT 2.11 R3 Build 2245	Win32, CE-X86, CE-ARM	Tc2_PLCopen_OpcUa

6.1.4 UA_HistoryUpdate



This function block sends historical data via OPC UA to a server that supports the OPC UA HistoryUpdate function, e.g. the TwinCAT OPC UA Server. With one call you can transfer a large number of values including time stamps to the server for a node handle. The server ensures that the values transmitted are saved in a data memory and are available via Historical Access.



The function block can be instanced several times if values of several node handles (different variables) are to be transmitted.

Operation with TwinCAT OPC UA Server

The function block is well suited if you use Historical Access in the TwinCAT OPC UA Server and want to make data available from a certain time interval in which, for example, a special machine state prevailed. Values for the desired period can be purposefully transmitted.

If on the other hand values are sent cyclically and are to be made available in the server via Historical Access, then the Historical Access function on the server side [▸ 54] is better suited, as in this case you only have to configure the recording node in the configurator and set the desired sampling rate.

See also: Program sample TF6100_OPCUA_HASample [▸ 240]

VAR_INPUT

```
VAR_INPUT
    Execute        : BOOL;
    ConnectionHdl  : DWORD;
    NodeHdl        : DWORD;
    PerformInsert  : BOOL;
    PerformReplace : BOOL;
    DataValueCount : UINT;
    Timeout        : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

Execute: The command is triggered by a positive edge at this input.

ConnectionHdl: Connection handle previously output by the function block UA_Connect [▸ 218].

NodeHdl: Node handle that was previously output by the function block UA_NodeGetHandle [▶ 226].

PerformInsert: The default is TRUE.

PerformReplace: The default is FALSE. If a value for the given time stamp already exists in the history, it is to be replaced if the option PerformReplace is set (= TRUE). Currently this option can only be selected for SQL adapters. Other adapters do not support the option.

DataValueCount: Defines the number of values transferred. A maximum number of 1000 values is supported.

Timeout: Time until the function is aborted. DEFAULT_ADS_TIMEOUT is a global constant, set to 5 seconds.

VAR_IN_OUT

```
VAR_IN_OUT
    DataValues        : ARRAY[*] OF UAHADataValue;
    ValueErrorIDs     : ARRAY[*] OF DWORD;
END_VAR
```

DataValues (read-only): All collected values are transferred in the form of a field of the type UAHADataValue [▶ 238]. The length of the field is not prescribed, but it must correspond at least to the specification of DataValueCount. Internally the values are accessed only for reading.

ValueErrorIDs (write-only): After execution of the command this field contains an error code for each value. The length of the field must correspond at least to the specification of DataValueCount. If one or more values report an error, it is also signaled via the outputs Error and ErrorID of the function block. With the help of this field you can then determine which error has occurred for which value. The error code 16#80000000, for example, signalizes a failed operation, meaning that the value could not be written.

VAR_OUTPUT

```
VAR_OUTPUT
    Done      : BOOL;
    Busy      : BOOL;
    Error     : BOOL;
    ErrorID   : DWORD;
END_VAR
```

Done: Switches to TRUE if the function block was executed successfully.

Busy: TRUE until the function block has executed a command, at the most for the duration of the "Timeout" at the input. The inputs accept no new command as long as Busy = TRUE. It is not the connection time that is monitored but the reception time.

Error: Switches to TRUE if an error occurs while executing a command. The command-specific error code is included in ErrorID.

ErrorID: Contains the command-specific ADS error code of the most recently executed command.

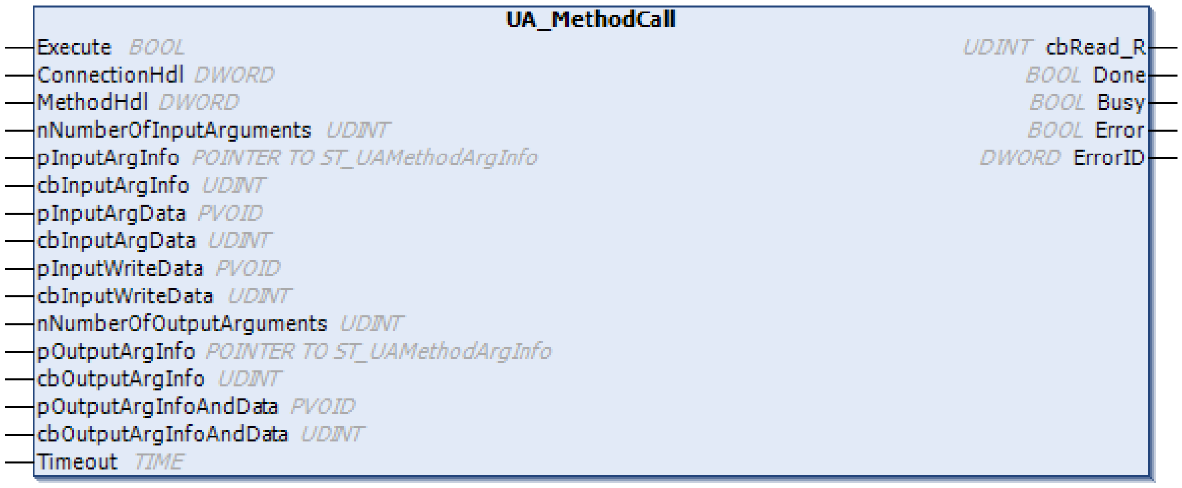
Number of values transferred

The larger the number, the greater the required computing effort and thus the longer the PLC execution time when executing the command.

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT 3.1.4022.27	Win32, Win64, WinCE-x86	Tc3_PLCopen_OpcUa >= v3.1.8.0

6.1.5 UA_MethodCall



This function block calls a method on a remote UA Server. The method is determined by a connection and a method handle. The former can be queried by UA_Connect [▶ 218], the latter by UA_MethodGetHandle [▶ 224].

VAR_INPUT

```
VAR_INPUT
    Execute                         : BOOL;
    ConnectionHdl                   : DWORD;
    MethodHdl                       : DWORD;
    nNumberOfInputArguments         : UDINT;
    pInputArgInfo                   : POINTER TO ST_UAMethodArgInfo;
    cbInputArgInfo                  : UDINT;
    pInputArgData                   : PVOID;
    cbInputArgData                  : UDINT;
    pInputWriteData                 : PVOID;
    cbInputWriteData                : UDINT;
    nNumberOfOutputArguments        : UDINT;
    pOutputArgInfo                  : POINTER TO ST_UAMethodArgInfo;
    cbOutputArgInfo                 : UDINT;
    pOutputArgInfoAndData           : PVOID;
    cbOutputArgInfoAndData          : UDINT;
    Timeout                         : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

Execute: The command is triggered by a positive edge at this input.

ConnectionHdl: Connection handle previously output by the function block UA_Connect [▶ 218].

MethodHdl: Method handle, previously output by the function block UA_MethodGetHandle [▶ 224].

nNumberOfInputArguments: Number of input parameters.

pInputArgInfo: Points to the buffer address where input parameter information is stored in the form of an array ST_UAMethodArgInfo.

cbInputArgInfo: Size of the buffer where the input parameter information is stored.

pInputArgData: Points to the buffer address where input parameters (constant length) are stored.

cbInputArgData: Size of the input buffer where input parameters (with constant length) are stored.

pInputWriteData: Pointer to buffer address where input parameters (dynamic length) are stored.

cbInputWriteData: Size of the input buffer where input parameters (with dynamic length) are stored.

nNumberOfOutputArguments: Number of output parameters.

pOutputArgInfo: Points to the buffer address where output parameter information is stored as array ST_UAMethodArgInfo.
nLenData is required to determine the target memory of the individual output parameters. The other elements can be set in such a way that a type check of the returned parameters takes place or remains undefined.

cbOutputArgInfo: Size of the buffer where the input parameter information is stored.

pOutputArgInfoAndData: Points to the buffer address where the output parameters are to be saved as a BYTE array. The BYTE array contains the number of output parameters as DINT, four reserved bytes and parameter information as ARRAY OF ST_UAMethodArgInfo [▸ 237] (with the length of the output parameters), followed by pure data. Note that the data is packed as 1-byte alignment.

cbOutputArgInfoAndData: Size of the buffer in which the output parameters are to be saved as a BYTE array.

Timeout: Time until the function is aborted. DEFAULT_ADS_TIMEOUT is a global constant, set to 5 seconds.

VAR_OUTPUT

```
VAR_OUTPUT
    cbRead_R      : UDINT;
    Done          : BOOL;
    Busy          : BOOL;
    Error         : BOOL;
    ErrorID       : UDINT;
END_VAR
```

cbRead_R: Counts all the bytes received.

Done: Switches to TRUE if the function block was executed successfully.

Busy: TRUE until the function block has executed a command, at the most for the duration of the "Timeout" at the input. The inputs accept no new command as long as Busy = TRUE. It is not the connection time that is monitored but the reception time.

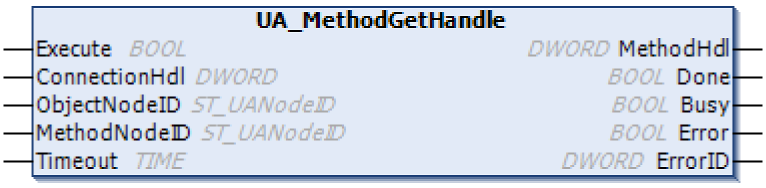
Error: Switches to TRUE if an error occurs while executing a command. The command-specific error code is contained in nErrID.

ErrorID: Contains the command-specific error code of the most recently executed command.

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT 3.1	Win32, Win64, CE-X86, CE-ARM	Tc3_PLCopen_OpcUa

6.1.6 UA_MethodGetHandle



This function block collects a handle for a UA method, which can then be used to call a method using UA_MethodCall [▸ 223].

VAR_INPUT

```
VAR_INPUT
    Execute           : BOOL;
    ConnectionHdl     : DWORD;
    ObjectNodeID      : ST_UANodeID;
    MethodNodeID      : ST_UANodeID;
    Timeout           : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

Execute: The command is triggered by a positive edge at this input.

ConnectionHdl: Connection handle previously output by the function block UA_Connect [▶ 218].

ObjectNodeID: Object node ID of the method to be called. (Type: ST_UANodeID [▶ 237])

MethodNodeID: Method node ID of the method to be called. Corresponds to the ID attribute in the UA namespace. (Type: UA_Connect [▶ 218])

Timeout: Time until the function is aborted. DEFAULT_ADS_TIMEOUT is a global constant, set to 5 seconds.

VAR_OUTPUT

```
VAR_OUTPUT
    MethodHdl    : DWORD;
    Done         : BOOL;
    Busy         : BOOL;
    Error        : BOOL;
    ErrorID      : UDINT;
END_VAR
```

MethodHdl: Returns a method handle that can be used to call a method via UA_MethodCall [▶ 223].

Done: Switches to TRUE if the function block was executed successfully.

Busy: TRUE until the function block has executed a command, at the most for the duration of the "Timeout" at the input. The inputs accept no new command as long as Busy = TRUE. It is not the connection time that is monitored but the reception time.

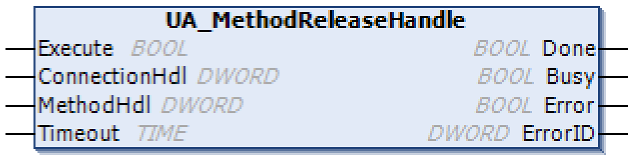
Error: Switches to TRUE if an error occurs while executing a command. The command-specific error code is contained in nErrID.

ErrorID: Contains the command-specific error code of the most recently executed command.

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT 3.1	Win32, Win64, CE-X86, CE-ARM	Tc3_PLCopen_OpcUa

6.1.7 UA_MethodReleaseHandle



This function block releases the specified method handle.

VAR_INPUT

```
VAR_INPUT
    Execute            : BOOL;
    ConnectionHdl      : DWORD;
    MethodHdl          : DWORD;
    Timeout            : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

Execute: The command is triggered by a positive edge at this input.

ConnectionHdl: Connection handle previously output by the function block UA_Connect [▸ 218].

MethodHdl: Method handle previously output by the function block UA_MethodGetHandle [▸ 224].

Timeout: Time until the function is aborted. DEFAULT_ADS_TIMEOUT is a global constant, set to 5 seconds.

VAR_OUTPUT

```
VAR_OUTPUT
    Done      : BOOL;
    Busy      : BOOL;
    Error     : BOOL;
    ErrorID   : UDINT;
END_VAR
```

Done: Switches to TRUE if the function block was executed successfully.

Busy: TRUE until the function block has executed a command, at the most for the duration of the "Timeout" at the input. The inputs accept no new command as long as Busy = TRUE. It is not the connection time that is monitored but the reception time.

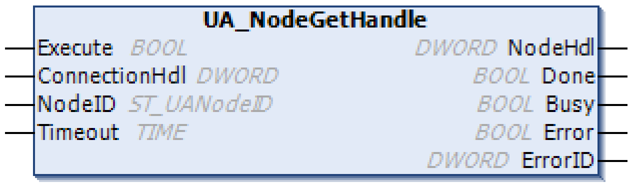
Error: Switches to TRUE if an error occurs while executing a command. The command-specific error code is contained in nErrID.

ErrorID: Contains the command-specific ADS error code of the most recently executed command.

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT 3.1	Win32, Win64, CE-X86, CE-ARM	Tc3_PLCopen_OpcUa

6.1.8 UA_NodeGetHandle



This function block retrieves a node handle for a given symbol in UA namespace. The symbol will be specified by a connection handle and its node ID.

VAR_INPUT

```
VAR_INPUT
    Execute        : BOOL;
    ConnectionHdl : DWORD;
    NodeID         : ST_UANodeID;
    Timeout        : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

Execute: The command is triggered by a positive edge at this input.

ConnectionHdl: Connection handle previously output by the function block UA_Connect [▸ 218].

Node ID: Unique addressing of the UA node, consisting of Identifier, IdentifierType and NamespaceIndex, which are resolved from a NamespaceName, e.g. by means of the method UA_GetNamespaceIndex [▸ 220].

Timeout: Time until the function is aborted. DEFAULT_ADS_TIMEOUT is a global constant, set to 5 seconds.

VAR_OUTPUT

```
VAR_OUTPUT
    NodeHdl      : DWORD;
    Done         : BOOL;
    Busy         : BOOL;
    Error        : BOOL;
    ErrorID      : DWORD;
END_VAR
```

NodeHdl: Node handle that can be used for other function blocks, such as UA_Read or UA_Write.

Done: Switches to TRUE if the function block was executed successfully.

Busy: TRUE until the function block has executed a command, at the most for the duration of the "Timeout" at the input. The inputs do not accept new commands as long as Busy is TRUE. It is not the connection time that is monitored but the reception time.

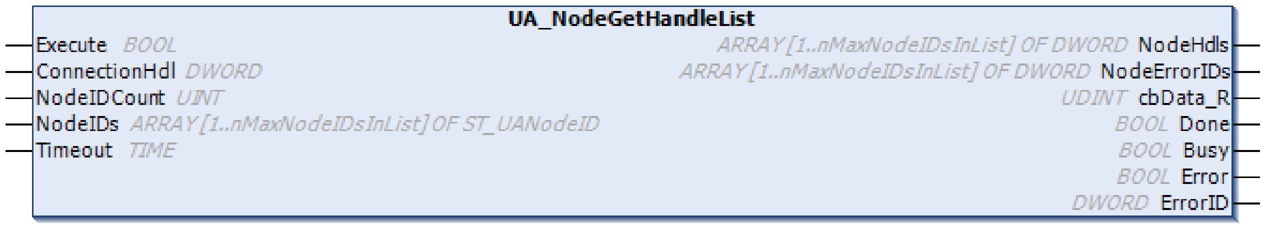
Error: Switches to TRUE if an error occurs while executing a command. The command-specific error code is included in ErrorID.

ErrorID: Contains the command-specific ADS error code of the most recently executed command.

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT 3.1	Win32, Win64, CE-X86, CE-ARM	Tc3_PLCopen_OpcUa
TwinCAT 2.11 R3 Build 2245	Win32, CE-X86, CE-ARM	Tc2_PLCopen_OpcUa

6.1.9 UA_NodeGetHandleList


UA_NodeGetHandleList
Execute BOOL
ConnectionHdl DWORD
NodeIDCount UINT
NodeIDs ARRAY [1..nMaxNodeIDsInList] OF ST_UANodeID
Timeout TIME
ARRAY [1..nMaxNodeIDsInList] OF DWORD NodeHdls
ARRAY [1..nMaxNodeIDsInList] OF DWORD NodeErrorIDs
UDINT cbData_R
BOOL Done
BOOL Busy
BOOL Error
DWORD ErrorID

This function block queries node handles for nodes in the UA namespace.

VAR_INPUT

```
VAR_INPUT
    Execute            : BOOL;
    ConnectionHdl      : DWORD;
    NodeIDCount        : UINT;
    NodeIDs            : ARRAY[1..nMaxNodeIDsInList] OF ST_UANodeID;
    Timeout            : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

Execute: The command is triggered by a positive edge at this input.

ConnectionHdl: Connection handle previously output by the function block UA_Connect [▶ 218].

NodeIDCount: Number of nodes for which a node handle is required.

NodeIDs: Array of NodeIDs created with struct ST_UANodeID [▶ 237].

Timeout: Time until the function is aborted. DEFAULT_ADS_TIMEOUT is a global constant, set to 5 seconds.

VAR_OUTPUT

```
VAR_OUTPUT
    NodeHdls      : ARRAY[1..nMaxNodeIDsInList] OF DWORD;
    NodeErrorIDs : ARRAY[1..nMaxNodeIDsInList] OF DWORD;
    cbData_R      : UDINT;
    Done          : BOOL;
    Busy          : BOOL;
    Error         : BOOL;
    ErrorID       : DWORD;
END_VAR
```

NodeHdls: Array of requested node handles.

NodeErrorIDs: Array of error IDs if no node handles are available.

cbData_R: Size of the data read.

Done: Switches to TRUE if the function block was executed successfully.

Busy: TRUE until the function block has executed a command, at the most for the duration of the "Timeout" at the input. The inputs accept no new command as long as Busy = TRUE. It is not the connection time that is monitored but the reception time.

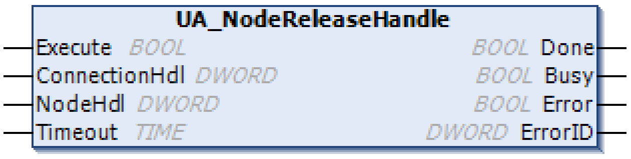
Error: Switches to TRUE if an error occurs while executing a command. The command-specific error code is in nErrID.

ErrorID: Contains the error ID if an error occurs.

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT 3.1	Win32, Win64, CE-X86, CE-ARM	Tc3_PLCopen_OpcUa
TwinCAT 2.11 R3 Build 2245	Win32, CE-X86, CE-ARM	Tc2_PLCopen_OpcUa

6.1.10 UA_NodeReleaseHandle



This function block releases a node handle.

VAR_INPUT

```
VAR_INPUT
    Execute       : BOOL;
    ConnectionHdl : DWORD;
    NodeHdl       : DWORD;
    Timeout       : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

Execute: The command is triggered by a positive edge at this input.

ConnectionHdl: Connection handle previously output by the function block UA_Connect [▸ 218].

NodeHdl: Node handle to be released.

Timeout: Time until the function is aborted. DEFAULT_ADS_TIMEOUT is a global constant, set to 5 seconds.

VAR_OUTPUT

```
VAR_OUTPUT
    Done        : BOOL;
    Busy        : BOOL;
    Error       : BOOL;
    ErrorID     : DWORD;
END_VAR
```

Done: Switches to TRUE if the function block was executed successfully.

Busy: TRUE until the function block has executed a command, at the most for the duration of the "Timeout" at the input. The inputs accept no new command as long as Busy = TRUE. It is not the connection time that is monitored but the reception time.

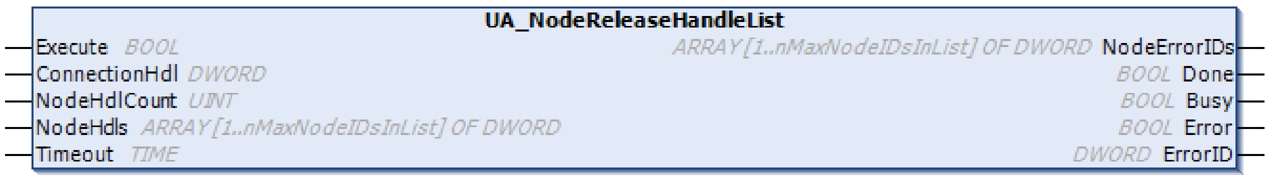
Error: Switches to TRUE if an error occurs while executing a command. The command-specific error code is included in ErrorID.

ErrorID: Contains the command-specific ADS error code of the most recently executed command.

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT 3.1	Win32, Win64, CE-X86, CE-ARM	Tc3_PLCopen_OpcUa
TwinCAT 2.11 R3 Build 2245	Win32, CE-X86, CE-ARM	Tc2_PLCopen_OpcUa

6.1.11 UA_NodeReleaseHandleList



This function block releases several node handles.

VAR_INPUT

```
VAR_INPUT
    Execute           : BOOL;
    ConnectionHdl     : DWORD;
    NodeHdlCount      : UINT;
    NodeHdls          : ARRAY[1..nMaxNodeIDsInList] OF DWORD;
    Timeout           : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

Execute: The command is triggered by a positive edge at this input.

ConnectionHdl: Connection handle previously output by the function block UA_Connect [▶ 218].

NodeHdlCount: Number of node handles.

NodeHdls: Array of node handles to be released.

Timeout: Time until the function is aborted. DEFAULT_ADS_TIMEOUT is a global constant, set to 5 seconds.

VAR_OUTPUT

```
VAR_OUTPUT
    NodeErrorIDs : ARRAY[1..nMaxNodeIDsInList] OF DWORD;
    Done         : BOOL;
    Busy         : BOOL;
    Error        : BOOL;
    ErrorID      : DWORD;
END_VAR
```

NodeErrorIDs: Array of error IDs if a node handle could not be released.

Done: Switches to TRUE if the function block was executed successfully.

Busy: TRUE until the function block has executed a command, at the most for the duration of the "Timeout" at the input. The inputs accept no new command as long as Busy = TRUE. It is not the connection time that is monitored but the reception time.

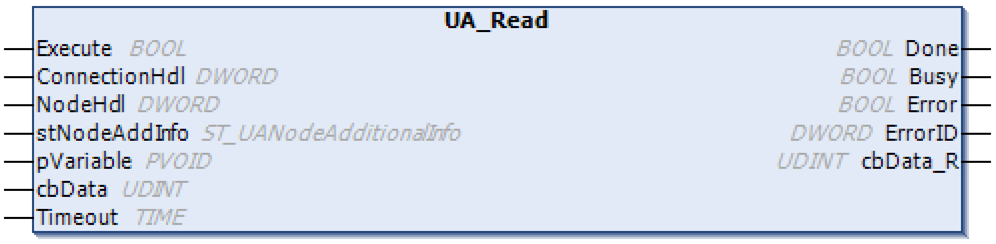
Error: Switches to TRUE if an error occurs while executing a command. The command-specific error code is in nErrID.

ErrorID: Contains the error ID if an error occurs.

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT 3.1	Win32, Win64, CE-X86, CE-ARM	Tc3_PLCopen_OpcUa
TwinCAT 2.11 R3 Build 2245	Win32, CE-X86, CE-ARM	Tc2_PLCopen_OpcUa

6.1.12 UA_Read



This function block reads values from a given node and connection handle.

VAR_INPUT

```
VAR_INPUT
    Execute             : BOOL;
    ConnectionHdl       : DWORD;
    NodeHdl             : DWORD;
    stNodeAddInfo       : ST_UANodeAdditionalInfo;
    pVariable           : PVOID;
    cbData              : UDINT;
    Timeout             : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

Execute: The command is triggered by a positive edge at this input.

ConnectionHdl: Connection handle previously output by the function block UA_Connect [▶ 218].

NodeHdl: Node handle previously output by the function block UA_NodeGetHandle [▶ 226].

stNodeAddInfo: Defines additional information, such as which attribute is read from the UA namespace (default: 'Value' attribute) or which index range is to be used. Specified by STRUCT ST_UANodeAdditionalInfo [▶ 237].

pVariable: Pointer to data memory, where the read data should be stored.

cbData: Determines the size of the data to be read.

Timeout: Time until the function is aborted. DEFAULT_ADS_TIMEOUT is a global constant, set to 5 seconds.

VAR_OUTPUT

```
VAR_OUTPUT
    Done        : BOOL;
    Busy        : BOOL;
    Error       : BOOL;
    ErrorID     : UDINT;
    cbData_R    : UDINT;
END_VAR
```

Done: Switches to TRUE if the function block was executed successfully.

Busy: TRUE until the function block has executed a command, at the most for the duration of the "Timeout" at the input. The inputs accept no new command as long as Busy = TRUE. It is not the connection time that is monitored but the reception time.

Error: Switches to TRUE if an error occurs while executing a command. The command-specific error code is contained in nErrID.

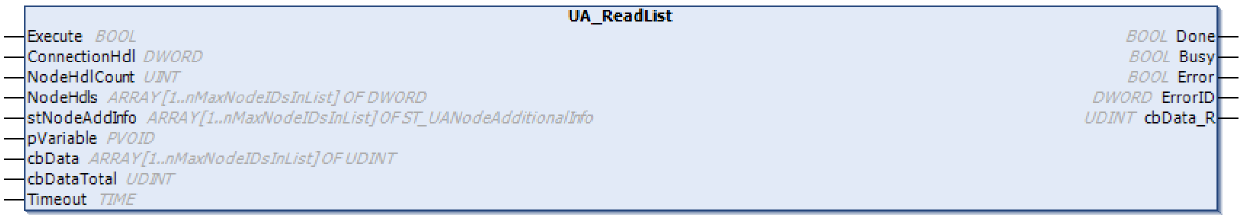
ErrorID: Contains the command-specific ADS error code of the most recently executed command.

cbData_R: Number of bytes to be read.

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT 3.1	Win32, Win64, CE-X86, CE-ARM	Tc3_PLCopen_OpcUa
TwinCAT 2.11 R3 Build 2245	Win32, CE-X86, CE-ARM	Tc2_PLCopen_OpcUa

6.1.13 UA_ReadList



This function block reads values from several given node and connection handles.

VAR_INPUT

```
VAR_INPUT
    Execute           : BOOL;
    ConnectionHdl     : DWORD;
    NodeHdlCount      : UINT;
    NodeHdls          : ARRAY[1..nMaxNodeIDsInList] OF DWORD;
    stNodeAddInfo     : ARRAY[1..nMaxNodeIDsInList] OF ST_UANodeAdditionalInfo;
    pVariable         : PVOID;
    cbData            : ARRAY[1..nMaxNodeIDsInList] UDINT;
    cbDataTotal       : UDINT;
    Timeout           : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

Execute: The command is triggered by a positive edge at this input.

ConnectionHdl: Connection handle previously output by the function block UA_Connect [▶ 218].

NodeHdlCount: Number of node handles stored in the input variable NodeHdls.

NodeHdls: Array of node handles previously received by the function block UA_NodeGetHandle [▶ 226] or UA_NodeGetHandleList [▶ 227].

stNodeAddInfo: Defines additional information, such as which attribute is read from the UA namespace (default: 'Value' attribute) or which index range is to be used. Specified by STRUCT ST_UANodeAdditionalInfo [▶ 237].

pVariable: Pointer to data memory, where the read data should be stored.

cbData: Determines the size of the data to be read.

cbDataTotal: Total size of the data to be received.

Timeout: Time until the function is aborted. DEFAULT_ADS_TIMEOUT is a global constant, set to 5 seconds.

VAR_OUTPUT

```
VAR_OUTPUT
    Done      : BOOL;
    Busy      : BOOL;
    Error     : BOOL;
    ErrorID   : UDINT;
    cbData_R  : UDINT;
END_VAR
```

Done: Switches to TRUE if the function block was executed successfully.

Busy: TRUE until the function block has executed a command, at the most for the duration of the "Timeout" at the input. The inputs accept no new command as long as Busy = TRUE. It is not the connection time that is monitored but the reception time.

Error: Switches to TRUE if an error occurs while executing a command. The command-specific error code is in nErrID.

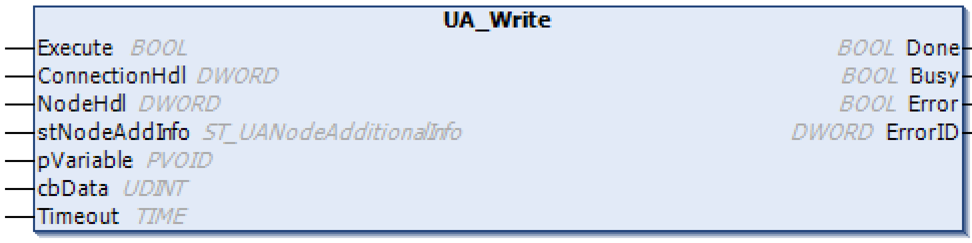
ErrorID: Contains the command-specific ADS error code of the most recently executed command.

cbData_R: Number of bytes read

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT 3.1	Win32, Win64, CE-X86, CE-ARM	Tc3_PLCopen_OpcUa
TwinCAT 2.11 R3 Build 2245	Win32, CE-X86, CE-ARM	Tc2_PLCopen_OpcUa

6.1.14 UA_Write



This function block writes values to a given node and connection handle.

VAR_INPUT

```
VAR_INPUT
    Execute        : BOOL;
    ConnectionHdl  : DWORD;
    NodeHdl        : DWORD;
    stNodeAddInfo  : ST_UANodeAdditionalInfo;
    pVariable      : PVOID;
    cbData          : UDINT;
    Timeout         : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

Execute: The command is triggered by a positive edge at this input.

ConnectionHdl: Connection handle previously output by the function block UA_Connect [▶ 218].

NodeHdl: Node handle previously output by the function block UA_NodeGetHandle [▶ 226].

stNodeAddInfo: Defines additional information, e.g. which IndexRange or which attribute is to be written (by default, the' Value' attribute is used). Specified by STRUCT ST_UANodeAdditionalInfo [▶ 237].

pVariable: Pointer to data to be written.

cbData: Sets the size of the values to be written.

Timeout: Time until the function is aborted. DEFAULT_ADS_TIMEOUT is a global constant, set to 5 seconds.

VAR_OUTPUT

```
VAR_OUTPUT
    Done       : BOOL;
    Busy       : BOOL;
    Error      : BOOL;
    ErrorID    : DWORD;
END_VAR
```

Done: Switches to TRUE if the function block was executed successfully.

Busy: TRUE until the function block has executed a command, at the most for the duration of the "Timeout" at the input. The inputs accept no new command as long as Busy = TRUE. It is not the connection time that is monitored but the reception time.

Error: Switches to TRUE if an error occurs while executing a command. The command-specific error code is included in ErrorID.

ErrorID: Contains the command-specific ADS error code of the most recently executed command.

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT 3.1	Win32, Win64, CE-X86, CE-ARM	Tc3_PLCopen_OpcUa
TwinCAT 2.11 R3 Build 2245	Win32, CE-X86, CE-ARM	Tc2_PLCopen_OpcUa

6.2 Data types

6.2.1 E_UAAttributeID

```
TYPE E_UAAttributeID:
(
    eUAAI_NodeID           := 1,
    eUAAI_NodeClass        := 2,
    eUAAI_BrowseName       := 3,
    eUAAI_DisplayName      := 4,
    eUAAI_Description      := 5,
    eUAAI_WriteMask        := 6,
    eUAAI_UserWriteMask    := 7,
    eUAAI_IsAbstract       := 8,
    eUAAI_Symmetric        := 9,
    eUAAI_InverseName      := 10,
    eUAAI_ContainsNoLoops  := 11,
    eUAAI_EventNotifier    := 12,
    eUAAI_Value            := 13,
    eUAAI_DataType         := 14,
    eUAAI_ValueRank        := 15,
    eUAAI_ArrayDimensions := 16
)DINT;
END_TYPE
```

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT 3.1	Win32, Win64, CE-X86, CE-ARM	Tc3_PLCopen_OpcUa
TwinCAT 2.11 R3 Build 2245	Win32, CE-X86, CE-ARM	Tc2_PLCopen_OpcUa

6.2.2 E_UADataType

```
TYPE E_UADataType:
(
    eUAType_Undefinied      := -1,
    eUAType_Null            := 0,
    eUAType_Boolean         := 1,
    eUAType_SByte           := 2,
    eUAType_Byte            := 3,
    eUAType_Int16           := 4,
    eUAType_UInt16          := 5,
    eUAType_Int32           := 6,
    eUAType_UInt32          := 7,
    eUAType_Int64           := 8,
    eUAType_UInt64          := 9,
    eUAType_Float           := 10,
    eUAType_Double          := 11,
    eUAType_String          := 12,
    eUAType_DateTime        := 13,
    eUAType_Guid            := 14,
    eUAType_ByteString      := 15,
    eUAType_XmlElement      := 16,
    eUAType_NodeId          := 17,
    eUAType_ExpandedNodeId  := 18,
    eUAType_StatusCode      := 19,
    eUAType_QualifiedName   := 20,
    eUAType_LocalizedText   := 21,
    eUAType_ExtensionObject := 22,
    eUAType_DataValue       := 23,
    eUAType_Variant         := 24,
    eUAType_DiagnosticInfo  := 25
)DINT;
END_TYPE
```

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT 3.1	Win32, Win64, CE-X86, CE-ARM	Tc3_PLCopen_OpcUa
TwinCAT 2.11 R3 Build 2245	Win32, CE-X86, CE-ARM	Tc2_PLCopen_OpcUa

6.2.3 E_UAIdentifierType

```
TYPE E_UAIdentifierType:
(
    eUAIdentifierType_String  := 1,
    eUAIdentifierType_Numeric := 2,
    eUAIdentifierType_GUID    := 3,
    eUAIdentifierType_Opaque  := 4
)DINT;
END_TYPE
```

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT 3.1	Win32, Win64, CE-X86, CE-ARM	Tc3_PLCopen_OpcUa
TwinCAT 2.11 R3 Build 2245	Win32, CE-X86, CE-ARM	Tc2_PLCopen_OpcUa

6.2.4 E_UASecurityMsgMode

```
TYPE E_UASecurityMsgMode:
(
    eUASecurityMsgMode_BestAvailable := 0,
    eUASecurityMsgMode_None          := 1,
    eUASecurityMsgMode_Sign          := 2,
    eUASecurityMsgMode_Sign_Encrypt  := 3
)DINT;
END_TYPE
```

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT 3.1	Win32, Win64, CE-X86, CE-ARM	Tc3_PLCopen_OpcUa
TwinCAT 2.11 R3 Build 2245	Win32, CE-X86, CE-ARM	Tc2_PLCopen_OpcUa

6.2.5 E_UASecurityPolicy

```
TYPE E_UASecurityPolicy:
(
    eUASecurityPolicy_BestAvailable := 0
    eUASecurityPolicy_None := 1,
    eUASecurityPolicy_Basic128      := 2,
    eUASecurityPolicy_Basic128Rsa15 := 3,
    eUASecurityPolicy_Basic256      := 4
)DINT;
END_TYPE
```

None: Guideline for configurations with minimal security requirements.

Basic128: Guideline for configurations with low to medium security requirements.

Basic128Rsa15: Defines a security guideline for configurations with moderate to high security requirements.

Basic256: Defines a security policy for configurations with high security requirements.

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT 3.1	Win32, Win64, CE-X86, CE-ARM	Tc3_PLCopen_OpcUa
TwinCAT 2.11 R3 Build 2245	Win32, CE-X86, CE-ARM	Tc2_PLCopen_OpcUa

6.2.6 E_UATransportProfile

```
TYPE E_UATransportProfile:
(
    eUATransportProfileUri_UATcp             := 1,
    eUATransportProfileUri_WSHttpBinary      := 2,
    eUATransportProfileUri_WSHttpXmlOrBinary := 3,
    eUATransportProfileUri_WSHttpXml         := 4
)DINT;
END_TYPE
```

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT 3.1	Win32, Win64, CE-X86, CE-ARM	Tc3_PLCopen_OpcUa
TwinCAT 2.11 R3 Build 2245	Win32, CE-X86, CE-ARM	Tc2_PLCopen_OpcUa

6.2.7 ST_UASessionConnectInfo

```
TYPE ST_UASessionConnectInfo:
STRUCT
    sApplicationUri       : STRING(MAX_STRING_LENGTH);
    sApplicationName      : STRING(MAX_STRING_LENGTH);

    eSecurityMode         : E_UASecurityMsgMode;
    eSecurityPolicyUri    : E_UASecurityPolicy;
    eTransportProfileUri : E_UATransportProfile;

    tSessionTimeout       : TIME;
    tConnectTimeout       : TIME;
END_STRUCT
END_TYPE
```

sApplicationUri: Application Uri maximum string length 255.
From TcUAClient 2.0.0.14 this will automatically be specified by the certificate as defined in the PLCOpen specification.

sApplicationName: Application name with a maximum string length of 255.

eSecurityMode: Security message mode. For available modes see E_UASecurityMsgMode [▸ 235].

eSecurityPolicyUri: Security policy Uri. For available security policy Uri see E_UASecurityPolicy [▸ 235].

eTransportProfileUri: Transport profile Uri. For available transport profile Uri see E_UATransportProfile [▸ 235];

nSessionTimeout: Session timeout value.

nConnectTimeout: Connection timeout value.

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT 3.1	Win32, Win64, CE-X86, CE-ARM	Tc3_PLCopen_OpcUa
TwinCAT 2.11 R3 Build 2245	Win32, CE-X86, CE-ARM	Tc2_PLCopen_OpcUa

6.2.8 ST_UAIndexRange

```
TYPE ST_UAIndexRange:
STRUCT
    nStartIndex : UDINT;
    nEndIndex   : UDINT;
END_STRUCT
END_TYPE
```

nStartIndex: Start index of the data.

nEndIndex: End index of the data.

For all dimensions:

- StartinIndex and EndIndex must be assigned.
- StartIndex must be smaller than EndIndex.
- To be able to access all elements in a dimension, StartIndex and EndIndex must be assigned in the dimension depending on the total number of elements.
- Individual elements of a dimension can be selected by specifying the same StartIndex and EndIndex.

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT 3.1	Win32, Win64, CE-X86, CE-ARM	Tc3_PLCopen_OpcUa
TwinCAT 2.11 R3 Build 2245	Win32, CE-X86, CE-ARM	Tc2_PLCopen_OpcUa

6.2.9 ST_UAMethodArgInfo

```
TYPE ST_UAMethodArgInfo:
STRUCT
    DataType        : E_UADataType := -1;
    ValueRank       : DINT := 2147483647;
    ArrayDimensions : ARRAY[1..3] OF UDINT := [0,0,0];
    nLenData        : DINT;
END_STRUCT
END_TYPE
```

DataType: Defines the UA data type for the method parameter. (Type: E_UADataType [▶ 234])

ValueRank: Determines whether the parameter is scalar (-1) or array.

ArrayDimensions: If the parameter is an array, it specifies the dimensions of the array. Each element determines the length per dimension.

nLenData: Specifies the length of the argument. For output information STRUCT only requests this element.

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT 3.1	Win32, Win64, CE-X86, CE-ARM	Tc3_PLCopen_OpcUa
TwinCAT 2.11 R3 Build 2245	Win32, CE-X86, CE-ARM	Tc2_PLCopen_OpcUa

6.2.10 ST_UANodeID

```
TYPE ST_UANodeID:
STRUCT
    nNamespaceIndex  : UINT;
    nReserved        : ARRAY [1..2] OF BYTE; //fill bytes
    sIdentifier      : STRING(MAX_STRING_LENGTH);
    eIdentifierType  : E_UAIdentifierType;
END_STRUCT
END_TYPE
```

nNamespaceIndex: Namespace index under which the node is available. Can be determined with the function block UA_GetNamespaceIndex [▶ 220].

sIdentifier: Identifier as shown in the UA namespace (attribute' Identifier').

eIdentifierType: Variable type, described by E_UAIdentifierType [▶ 234].

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT 3.1	Win32, Win64, CE-X86, CE-ARM	Tc3_PLCopen_OpcUa
TwinCAT 2.11 R3 Build 2245	Win32, CE-X86, CE-ARM	Tc2_PLCopen_OpcUa

6.2.11 ST_UANodeAdditionalInfo

```
TYPE ST_UANodeAdditionalInfo:
STRUCT
    eAttributeID      : E_UAAttributeID;
    nIndexRangeCount : UINT;
    nReserved         : ARRAY[1..2] OF BYTE; // fill bytes
    stIndexRange      : ARRAY[1..nMaxIndexRange] OF ST_UAIndexRange;
END_STRUCT
END_TYPE
```

eAttributeID: Specifies the ID of the OPC UA attribute. eUAAI_Value is used by default. (Type: E_UAAttributeID [▶ 233])

nIndexRangeCount: Determines how many index ranges are used in stIndexRange.

stIndexRange: Specifies an index range for reading values from an array. (Type: ST_UAIndexRange [▶ 236])

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT 3.1	Win32, Win64, CE-X86, CE-ARM	Tc3_PLCopen_OpcUa
TwinCAT 2.11 R3 Build 2245	Win32, CE-X86, CE-ARM	Tc2_PLCopen_OpcUa

6.2.12 UAHADataValue

This function block acts as a data object. An instance represents a value for the OPC UA Historical Access function. A whole field of these values is transferred to the UA_HistoryUpdate [▶ 221] function block on calling.

Declaration

```
aDataValues : ARRAY [1..50] OF UAHADataValue(ValueSize:=SIZEOF(LREAL));
```

Each data object is initialized with the expected size (in bytes) of the value.

Properties

Value (PVOID): Specifies the address of a variable containing the desired value. This is usually assigned with the help of the operator ADR(). The value itself is hereby assigned at the same time and copied into the data object.

StatusCode (UAHAUpdateStatusCode [▶ 239]**)**: Indicates the status code of the value.

SourceTimeStamp (ULINT): Indicates the time stamp of the source in UTC format. This can be determined with the help of the function F_GetSystemTime (Tc2_System PLC library).

ServerTimeStamp (ULINT): Indicates the time stamp of the OPC UA Server in UTC format. This function is not currently supported.

Data type size of the value

The size of the data type used is already indicated and thus defined in the declaration of the data object. This size is taken as the basis when assigning a value later.

Values of the type STRING are accordingly also saved and transmitted with a fixed initialized size. An indication of the current text length cannot be made.

Sample

```
{attribute 'OPC.UA.DA' := '1'}
fMyValue    : LREAL;       // Variable for HistorcalAccess
aDataValues : ARRAY [1..50] OF UAHADataValue(ValueSize:=SIZEOF(LREAL));
```

```
fMyValue := 27.75;
aDataValues[1].Value           := ADR(fMyValue);
aDataValues[1].StatusCode      := UAHAUpdateStatusCode.HistorianRaw;
aDataValues[1].SourceTimeStamp := F_GetSystemTime();
```

In this sample a field of 50 values is defined, of which each is represented by a data object. The current content of the variable fMyValue (= 27.75) is assigned to the first value.

The field can now be filled by means of further assignments in subsequent PLC cycles.

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT 3.1.4022.27	Win32, Win64, WinCE-x86	Tc3_PLCopen_OpcUa >= v3.1.8.0

6.2.13 UAHAUpdateStatusCode

A status code is assigned to each data value transferred using the OPC UA Historical Access function. This is a property of the object UAHADataValue [▸ 238].

Enumeration

```
{attribute 'qualified_only'}
TYPE UAHAUpdateStatusCode :
(
    HistorianRaw := 0,           // A raw data value.
    HistorianCalculated := 1,    // A data value which was calculated.
    HistorianInterpolated := 2, // A data value which was interpolated.
    Reserved := 3,               // Undefined.
    HistorianPartial := 4,       // A data value which was calculated with an incomplete interval.
    HistorianExtraData := 8,     // A raw data value that hides other data at the same timestamp.
    HistorianMultiValue := 16   // Multiple values match the Aggregate criteria (i.e. multiple minim
um values at different timestamps within the same interval).
) UDINT;
END_TYPE
```

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT 3.1.4022.27	Win32, Win64, WinCE-x86	Tc3_PLCopen_OpcUa >= v3.1.8.0

7 Samples

Overview

Sample	Cate-gory	TwinCAT Version	Description	Download
TF6100 Server and Client sample	Server Client	TC3 (>=3.1.4020.0)	The sample includes a PLC for providing variables for the TwinCAT OPC UA Server (Data Access). Furthermore, it includes samples of the use of the function blocks UA_Read, UA_Write, UA_ReadList and UA_MethodCall (Tc3_PLCopen_OpcUa).	https://infosys.beckhoff.com/content/1033/TF6100_Tc3_OpcUa/Resources/zip/5757068683.zip
TF6100 Client function block UA_HistoryUpdate	Client	TC3	The sample includes a PLC for the use of the function block UA_HistoryUpdate (Tc3_PLCopen_OpcUa)	https://infosys.beckhoff.com/content/1033/TF6100_Tc3_OpcUa/Resources/zip/5972050315.zip
TS6100 Client function blocks	Client	TC2	Samples of the use of the function blocks UA_Read and UA_Write	https://infosys.beckhoff.com/content/1033/TF6100_Tc3_OpcUa/Resources/zip/4695969163.zip
TS6100 Server Historical Access	Server	TC2	Historical Access: PLC sample program	https://infosys.beckhoff.com/content/1033/TF6100_Tc3_OpcUa/Resources/zip/4695967499.zip

8 Appendix

8.1 Error diagnosis

The following table provides some helpful information for troubleshooting.

TwinCAT OPC UA Server

Behavior	Notes
An OPC UA Client does not see the PLC namespace	TF6100 Setup version 3.x and lower: This status indicates a missing license. Check whether you have activated a valid TF6100 license.
An OPC UA Client is assigned the StatusCode 0x810e0000 when reading nodes.	TF6100 Setup version 4.x: This status indicates a missing license. Check whether you have activated a valid TF6100 license.
The variables released via comments/attributes are not displayed in the OPC UA Server.	Check whether the symbol file has been correctly transferred to the controller (e.g. check boxes in the PLC project), verify that it exists in the boot directory and that the path to the symbol file in the configuration file of the server refers to the correct symbol file. You can also use the DeviceState Node in the respective namespace to check any error messages that may have occurred. An entry is made here if the symbol file was not found. Also check that the comments/attributes are spelled correctly.
The server does not comply with the sampling rate/publishing interval required by the OPC UA Client.	OPC UA Client/Server is not a real-time protocol, i.e. there is no guarantee that the server will always meet 100% of the sampling rate or publishing interval required by the client. The available sampling rates and publishing intervals can be viewed in the server configuration file and modified if required (<AvailableSamplingRates> and <MinPublishingInterval>).
An OPC UA Client cannot connect to the server although the server is displayed in the Windows Task Manager. The error message "Host unreachable" (or similar) appears.	Check whether firewall settings prevent communication with the server. The server port must be open for incoming TCP communication so that a client can connect.
An OPC UA Client sees the server's endpoints, but a connection with them fails with the error message "Host unreachable".	Check that the name resolution in your network is working properly and that the server is accessible under its host name. Even if the OPC UA Client apparently connects to the IP address of the server (e.g. opc.tcp://192.168.0.1:4840) to access the server's endpoints, the server always returns its own host name in its endpoints. If the client connects directly to one of the endpoints, it will use the host name of the server again. If the name resolution does not work, the connection fails.
An OPC UA Client sees the endpoints of the server, but a connection to a secure endpoint fails. The error message "BadSecurityChecksFailed" appears.	Check whether the server trusts the client certificate. The required configuration steps can be found in section Certificate exchange [▶ 122].
When using an SQL server to store Historical Access information, the values are not added to the SQL database.	Check the access data to the SQL Server and verify that the SQL Server is also accessible in the network. Also make sure that you are using a "Big Windows" operating system on the TwinCAT OPC UA Server, since SQL Server cannot be used for Historical Access under Windows CE (although SQL Compact is OK).

Behavior	Notes
When reading variables, an OPC UA Client receives the error message "BadDeviceFailure".	This is an indication that the associated ADS device cannot be reached, for example if no PLC program has been started. Check the connectivity with the ADS device and make sure that the appropriate runtime is active.
Arrays are not displayed in the namespace with full resolution	By default, arrays of simple data types are not displayed in expanded form in the namespace. However, individual array indices can still be addressed using the IndexRange function of OPC UA. An OPC UA Client should therefore support this function. If this is not the case, an option switch in the configuration file of the server can be used to display an array in expanded form, so that each individual array element can be addressed as a separate node. This option switch is described in Use of arrays [▶ 71].

8.2 OPC UA Client error codes

The function blocks of the TwinCAT OPC UA Client have their own error codes, which indicate the occurrence of an error and use an ErrorID to display further information about the problem that has occurred. TwinCAT ADS error messages (ADS Return Codes [▶ 243]) with the HighWord 0x0000 and custom error messages from the client or the PLC library with the HighWord 0xE4DD can occur.

Possible TwinCAT ADS errors include the following:

Hex	Name	Description
0x 0000 0705	DEVICE_INVALIDSIZE	Parameter size not correct
0x 0000 0706	DEVICE_INVALIDDATA	Invalid parameter values
0x 0000 070A	DEVICE_NOMEMORY	Not enough memory

This error code list shows the possible custom error values:

Hex	Name	Description
0x E4DD 0001	UAC_E_FAIL	UA service call failed
0x E4DD 0100	UAC_E_CONNECTED	Server already connected
0x E4DD 0101	UAC_E_CONNECT	General error when establishing a connection
0x E4DD 0102	UAC_E_UASECURITY	UA security could not be set up
0x E4DD 0103	UAC_E_ITEMEXISTS	Element ID already exists
0x E4DD 0104	UAC_E_ITEMNOTFOUND	Element does not exist
0x E4DD 0105	UAC_E_ITEMTYPE	Invalid or unsupported item type
0x E4DD 0106	UAC_E_CONVERSION	Variable types cannot be converted
0x E4DD 0107	UAC_E_SUSPENDED	Device hangs. Please try again later...
0x E4DD 0108	UAC_E_TYPE_NOT_SUPPORTED	Conversion variable type is not supported.
0x E4DD 0109	UAC_E_NSNAME_NOTFOUND	No namespace with the specified name found.
0x E4DD 0110	UAC_E_CONNECT_NOTFOUND	Connection failed: Target host could not be found.
0x E4DD 0111	UAC_E_TIMEOUT	Timeout: i.e. target host does not respond
0x E4DD 0112	UAC_E_INVALIDHDL	Session handle invalid
0x E4DD 0113	UAC_E_INVALIDNODEID	UA node ID unknown
0x E4DD 0114	UAC_E_INVAL_IDENTIFIER_TYPE	Identifier type of UaNodeId invalid
0x E4DD 0115	UAC_E_IDENTIFIER_NOTSUPP	Identifier type UaNodeId is not supported
0x E4DD 0116	UAC_E_INVAL_NODE_HDL	Invalid node handle
0x E4DD 0117	UAC_E_UAREADFAILED	UA read failed for unknown reasons
0x E4DD 0118	UAC_E_UAWRITEFAILED	UA write failed for unknown reasons
0x E4DD 0119	UAC_E_INVAL_NODEMETHOD_HDL	Invalid method handle

Hex	Name	Description
0x E4DD 011A	UAC_E_CALL_FAILED	Call failed, cause unknown
0x E4DD 011B	UAC_E_CALLDECODE_FAILED	Successful call, decoding return value failed
0x E4DD 011C	UAC_E_NOTMAPPEDTYPE	Unassigned data type in return value
0x E4DD 011D	UAC_E_CALL_FAILED_BADINTERNAL	Call failed with UA_BadInternal
0x E4DD 011E	UAC_E_METHODIDINVALID	Unknown MethodID (returned on call, even if provided by GetMethodHdl)
0x E4DD 011F	UAC_E_TOOMUCHDIM	Method call has returned parameters with more than 3 dimensions; not supported.
0x E4DD 0120	UAC_E_CALL_FAILED_INVALIDARG	Call failed with OpcUa_BadInvalidArgument
0x E4DD 0121	UAC_E_CALL_FAILED_TYPEMISMATC H	Call failed with UAC_E_CALL_FAILED_TYPEMISMATCH
0x E4DD 0122	UAC_E_CALL_FAILED_OUTOFRANGE	Call failed with UAC_E_CALL_FAILED_OUTOFRANGE
0x E4DD 0123	UAC_E_CALL_FAILED_BADSTRUCTU RE	Call failed with OpcUa_BadStructureMissing
0x E4DD 0124	UAC_E_CALL_TYPEMISMATCH_OUTP ARAM	Call successful, but type of output information provided does not match
0x E4DD 0125	UAC_E_NONVALIDTYPEINFO	Node has insufficient type information
0x E4DD 0126	UAC_E_INVALIDATTRIBID	Access to invalid node attribute
0x E4DD 0128	UAC_E_NOTSUPPORTED	The command is not supported by the connected UaServer, e.g. when calling UA_HistoryUpdate.
0x E4DE 0100	UAC_E_INVALID_ARRAY_LENGTH	An invalid array length not matching DataValueCount was assigned to UA_HistoryUpdate.
0x E4DE 0101	UAC_E_INVALID_DATASIZE	A data value with an invalid data type size was assigned to UA_HistoryUpdate. All assigned DataValues must be of the same data type.
0x E4DE 0102	UAC_E_SUBERROR	A lower-level error was output for at least one of the transferred data values. See ValueErrorIDs at UA_HistoryUpdate.

8.3 ADS Return Codes

Error codes: 0x000 [▸ 243]..., 0x500 [▸ 244]..., 0x700 [▸ 244]..., 0x1000 [▸ 245]...

Global Error Codes

Hex	Dec	Description
0x0	0	no error
0x1	1	Internal error
0x2	2	No Rtime
0x3	3	Allocation locked memory error
0x4	4	Insert mailbox error
0x5	5	Wrong receive HMSG
0x6	6	target port not found
0x7	7	target machine not found
0x8	8	Unknown command ID
0x9	9	Bad task ID
0xA	10	No IO
0xB	11	Unknown ADS command
0xC	12	Win 32 error

Hex	Dec	Description
0xD	13	Port not connected
0xE	14	Invalid ADS length
0xF	15	Invalid AMS Net ID
0x10	16	Low Installation level
0x11	17	No debug available
0x12	18	Port disabled
0x13	19	Port already connected
0x14	20	ADS Sync Win32 error
0x15	21	ADS Sync Timeout
0x16	22	ADS Sync AMS error
0x17	23	ADS Sync no index map
0x18	24	Invalid ADS port
0x19	25	No memory
0x1A	26	TCP send error
0x1B	27	Host unreachable
0x1C	28	Invalid AMS fragment

Router Error Codes

Hex	Dec	Name	Description
0x500	1280	ROUTERERR_NOLOCKEDMEMORY	No locked memory can be allocated
0x501	1281	ROUTERERR_RESIZEMEMORY	The size of the router memory could not be changed
0x502	1282	ROUTERERR_MAILBOXFULL	The mailbox has reached the maximum number of possible messages. The current sent message was rejected
0x503	1283	ROUTERERR_DEBUGBOXFULL	The mailbox has reached the maximum number of possible messages. The sent message will not be displayed in the debug monitor
0x504	1284	ROUTERERR_UNKNOWNPORTTYPE	Unknown port type
0x505	1285	ROUTERERR_NOTINITIALIZED	Router is not initialized
0x506	1286	ROUTERERR_PORTALREADYINUSE	The desired port number is already assigned
0x507	1287	ROUTERERR_NOTREGISTERED	Port not registered
0x508	1288	ROUTERERR_NOMOREQUEUES	The maximum number of Ports reached
0x509	1289	ROUTERERR_INVALIDPORT	Invalid port
0x50A	1290	ROUTERERR_NOTACTIVATED	TwinCAT Router not active

General ADS Error Codes

Hex	Dec	Name	Description
0x700	1792	ADSERR_DEVICE_ERROR	error class <device error>
0x701	1793	ADSERR_DEVICE_SRVNOTSUPP	Service is not supported by server
0x702	1794	ADSERR_DEVICE_INVALIDGRP	invalid index group
0x703	1795	ADSERR_DEVICE_INVALIDOFFSET	invalid index offset
0x704	1796	ADSERR_DEVICE_INVALIDACCESS	reading/writing not permitted
0x705	1797	ADSERR_DEVICE_INVALIDSIZE	parameter size not correct
0x706	1798	ADSERR_DEVICE_INVALIDDATA	invalid parameter value(s)
0x707	1799	ADSERR_DEVICE_NOTREADY	device is not in a ready state
0x708	1800	ADSERR_DEVICE_BUSY	device is busy
0x709	1801	ADSERR_DEVICE_INVALIDCONTEXT	invalid context (must be in Windows)
0x70A	1802	ADSERR_DEVICE_NOMEMORY	out of memory
0x70B	1803	ADSERR_DEVICE_INVALIDPARM	invalid parameter value(s)
0x70C	1804	ADSERR_DEVICE_NOTFOUND	not found (files, ...)
0x70D	1805	ADSERR_DEVICE_SYNTAX	syntax error in command or file
0x70E	1806	ADSERR_DEVICE_INCOMPATIBLE	objects do not match
0x70F	1807	ADSERR_DEVICE_EXISTS	object already exists
0x710	1808	ADSERR_DEVICE_SYMBOLNOTFOUND	symbol not found
0x711	1809	ADSERR_DEVICE_SYMBOLVERSIONINVAL	symbol version invalid
0x712	1810	ADSERR_DEVICE_INVALIDSTATE	server is in invalid state
0x713	1811	ADSERR_DEVICE_TRANSMODENOTSUPP	AdsTransMode not supported

Hex	Dec	Name	Description
0x714	1812	ADSERR_DEVICE_NOTIFYHNDINVALID	Notification handle is invalid
0x715	1813	ADSERR_DEVICE_CLIENTUNKNOWN	Notification client not registered
0x716	1814	ADSERR_DEVICE_NOMOREHDLS	no more notification handles
0x717	1815	ADSERR_DEVICE_INVALIDWATCHSIZE	size for watch too big
0x718	1816	ADSERR_DEVICE_NOTINIT	device not initialized
0x719	1817	ADSERR_DEVICE_TIMEOUT	device has a timeout
0x71A	1818	ADSERR_DEVICE_NOINTERFACE	query interface failed
0x71B	1819	ADSERR_DEVICE_INVALIDINTERFACE	wrong interface required
0x71C	1820	ADSERR_DEVICE_INVALIDCLSID	class ID is invalid
0x71D	1821	ADSERR_DEVICE_INVALIDOBJID	object ID is invalid
0x71E	1822	ADSERR_DEVICE_PENDING	request is pending
0x71F	1823	ADSERR_DEVICE_ABORTED	request is aborted
0x720	1824	ADSERR_DEVICE_WARNING	signal warning
0x721	1825	ADSERR_DEVICE_INVALIDARRAYIDX	invalid array index
0x722	1826	ADSERR_DEVICE_SYMBOLNOTACTIVE	symbol not active
0x723	1827	ADSERR_DEVICE_ACCESSDENIED	access denied
0x724	1828	ADSERR_DEVICE_LICENSENOTFOUND	missing license
0x725	1829	ADSERR_DEVICE_LICENSEEXPIRED	license expired
0x726	1830	ADSERR_DEVICE_LICENSEEXCEEDED	license exceeded
0x727	1831	ADSERR_DEVICE_LICENSEINVALID	license invalid
0x728	1832	ADSERR_DEVICE_LICENSESYSTEMID	license invalid system id
0x729	1833	ADSERR_DEVICE_LICENSENOTIMELIMIT	license not time limited
0x72A	1834	ADSERR_DEVICE_LICENSEFUTUREISSUE	license issue time in the future
0x72B	1835	ADSERR_DEVICE_LICENSETIMETOLONG	license time period to long
0x72c	1836	ADSERR_DEVICE_EXCEPTION	exception occured during system start
0x72D	1837	ADSERR_DEVICE_LICENSEDUPLICATED	License file read twice
0x72E	1838	ADSERR_DEVICE_SIGNATUREINVALID	invalid signature
0x72F	1839	ADSERR_DEVICE_CERTIFICATEINVALID	public key certificate
0x740	1856	ADSERR_CLIENT_ERROR	Error class <client error>
0x741	1857	ADSERR_CLIENT_INVALIDPARM	invalid parameter at service
0x742	1858	ADSERR_CLIENT_LISTEMPTY	polling list is empty
0x743	1859	ADSERR_CLIENT_VARUSED	var connection already in use
0x744	1860	ADSERR_CLIENT_DUPLINVOKEID	invoke ID in use
0x745	1861	ADSERR_CLIENT_SYNCTIMEOUT	timeout elapsed
0x746	1862	ADSERR_CLIENT_W32ERROR	error in win32 subsystem
0x747	1863	ADSERR_CLIENT_TIMEOUTINVALID	Invalid client timeout value
0x748	1864	ADSERR_CLIENT_PORTNOTOPEN	ads-port not opened
0x750	1872	ADSERR_CLIENT_NOAMSADDR	internal error in ads sync
0x751	1873	ADSERR_CLIENT_SYNCINTERNAL	hash table overflow
0x752	1874	ADSERR_CLIENT_ADDHASH	key not found in hash
0x753	1875	ADSERR_CLIENT_REMOVEHASH	no more symbols in cache
0x754	1876	ADSERR_CLIENT_NOMORESYM	invalid response received
0x755	1877	ADSERR_CLIENT_SYNCRESINVALID	sync port is locked

RTime Error Codes

Hex	Dec	Name	Description
0x1000	4096	RTERR_INTERNAL	Internal fatal error in the TwinCAT real-time system
0x1001	4097	RTERR_BADTIMERPERIODS	Timer value not vaild
0x1002	4098	RTERR_INVALIDTASKPTR	Task pointer has the invalid value ZERO
0x1003	4099	RTERR_INVALIDSTACKPTR	Task stack pointer has the invalid value ZERO
0x1004	4100	RTERR_PRIOEXISTS	The demand task priority is already assigned
0x1005	4101	RTERR_NOMORETCB	No more free TCB (Task Control Block) available. Maximum number of TCBs is 64
0x1006	4102	RTERR_NOMORESEMAS	No more free semaphores available. Maximum number of semaphores is 64
0x1007	4103	RTERR_NOMOREQUEUES	No more free queue available. Maximum number of queue is 64

Hex	Dec	Name	Description
0x100D	4109	RTERR_EXTIRQALREADYDEF	An external synchronization interrupt is already applied
0x100E	4110	RTERR_EXTIRQNOTDEF	No external synchronization interrupt applied
0x100F	4111	RTERR_EXTIRQINSTALLFAILED	The apply of the external synchronization interrupt failed
0x1010	4112	RTERR_IRQLNOTLESSOREQUAL	Call of a service function in the wrong context
0x1017	4119	RTERR_VMXNOTSUPPORTED	Intel VT-x extension is not supported
0x1018	4120	RTERR_VMXDISABLED	Intel VT-x extension is not enabled in system BIOS
0x1019	4121	RTERR_VMXCONTROLSMISSING	Missing function in Intel VT-x extension
0x101A	4122	RTERR_VMXENABLEFAILS	Enabling Intel VT-x fails

TCP Winsock Error Codes

Hex	Dec	Description
0x274d	10061	A connection attempt failed because the connected party did not properly respond after a period of time, or established connection failed because connected host has failed to respond.
0x2751	10065	No connection could be made because the target machine actively refused it. This error normally occurs when you try to connect to a service which is inactive on a different host - a service without a server application.
0x274c	10060	No route to a host. A socket operation was attempted to an unreachable host
		Further Winsock error codes: Win32 Error Codes

8.4 Support and Service

Beckhoff and their partners around the world offer comprehensive support and service, making available fast and competent assistance with all questions related to Beckhoff products and system solutions.

Beckhoff's branch offices and representatives

Please contact your Beckhoff branch office or representative for local support and service on Beckhoff products!

The addresses of Beckhoff's branch offices and representatives round the world can be found on her internet pages:
http://www.beckhoff.com

You will also find further documentation for Beckhoff components there.

Beckhoff Headquarters

Beckhoff Automation GmbH & Co. KG

Huelshorstweg 20
33415 Verl
Germany

Phone:	+49(0)5246/963-0
Fax:	+49(0)5246/963-198
e-mail:	info@beckhoff.com

Beckhoff Support

Support offers you comprehensive technical assistance, helping you not only with the application of individual Beckhoff products, but also with other, wide-ranging services:

- support
- design, programming and commissioning of complex automation systems
- and extensive training program for Beckhoff system components

Hotline:	+49(0)5246/963-157
Fax:	+49(0)5246/963-9157

e-mail:	support@beckhoff.com

Beckhoff Service

The Beckhoff Service Center supports you in all matters of after-sales service:

- on-site service
- repair service
- spare parts service
- hotline service

Hotline:	+49(0)5246/963-460
Fax:	+49(0)5246/963-479
e-mail:	service@beckhoff.com