Manual

TC3 XML Server

TwinCAT 3

Version: 1.2
Date: 2017-07-17
Order No.: TF6421

BECKHOFF

Table of contents

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

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.
❗ **Attention**	**Damage to the environment or devices** Failure to follow the instructions associated with this symbol can lead to damage to the environment or equipment.
ℹ **Note**	**Tip or pointer** This symbol indicates information that contributes to better understanding.

2 Overview

The TwinCAT XML Server provides a PLC library wherewith write-/read-access for XML data can be realized. The XML Server is characteristic for its easy handling. It is especially suitable for loading initialization data, as it is often needed to start up a machine.

Besides easy initializing from an XML file, the XML Server allows formatted saving of PLC variables in an XML file. The structure of a variable in the XML document matches the structure of the variable in the PLC. This gives you direct access to individual subelements of a variable in the XML file. Only subelements (elements of a structure or an array), which are defined in the XML file will be transfered. When the PLC variables are written, missing elements can optionally be added.

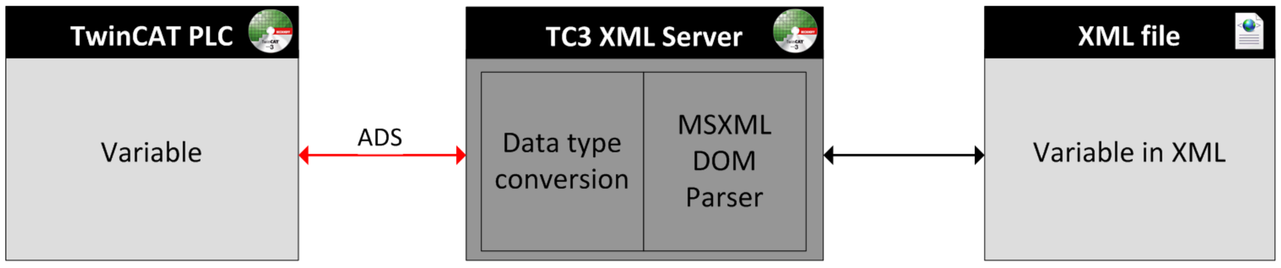
Components

- TwinCAT XML Server: A service that starts along with TwinCAT. It is the connection between TwinCAT and the XML file.
- PLC library [▶ 18]: The PLC library offers four function blocks for read and write operations. They allow formatted saving of the PLC variables in an XML file and initializing TwinCAT variables with data from an XML file.

Functional principle

The XML Server communicates with the TwinCAT PLC via ADS. During a writing process the variables will be converted into text and stored in the XML file through the MSXML DOM Parser. The XML file does not include any information about datatypes, but just the name of the variable and its value: <name_of_variable> value </name_of_variable>

During a reading process the datatype will be determined for each variable via ADS. The corresponding XML text will be converted properly.



3 Installation

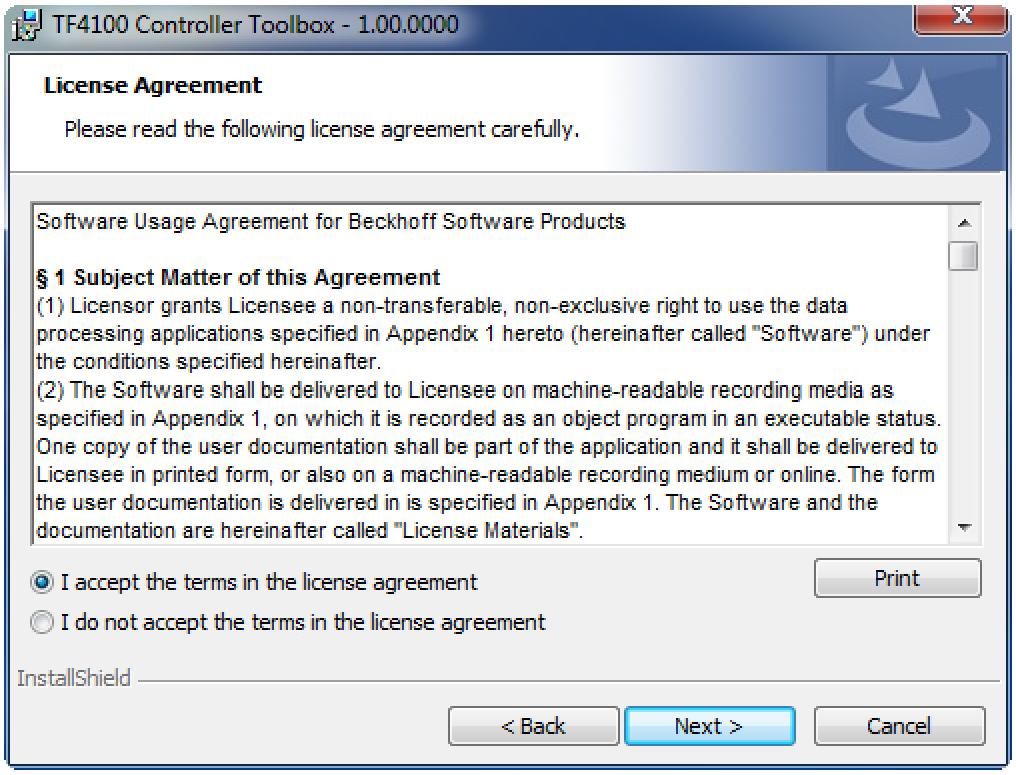
3.1 System Requirements

Technical data	TF6421 XML Server
Target system	Windows XP, Windows 7/8, Windows CE PC (x86-compatible), ARM
Min. TwinCAT version	3.1 Build 4011
Min. TwinCAT-Level	TwinCAT PLC

3.2 Installation

Description of the installation procedure of a TwinCAT 3 Function for Windows-based operating Systems.

1. Double-click the downloaded setup file *TFxxxx*.
   Please note: Under Windows 32-bit/64-bit, please start the installation with "Run as Administrator" by right-clicking the setup file and selecting the corresponding option in the context menu.
2. Click **Next** and accept the license agreement.


TF4100 Controller Toolbox - 1.00.0000

License Agreement

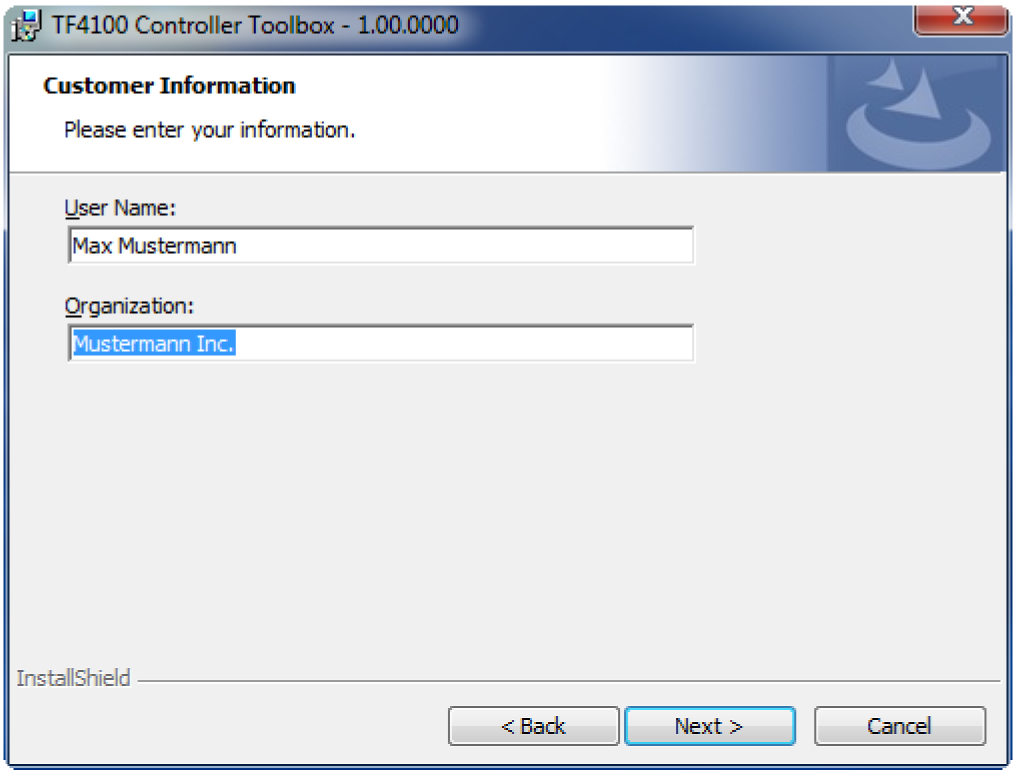
Please read the following license agreement carefully.

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 information in the specified area.



TF4100 Controller Toolbox - 1.00.0000

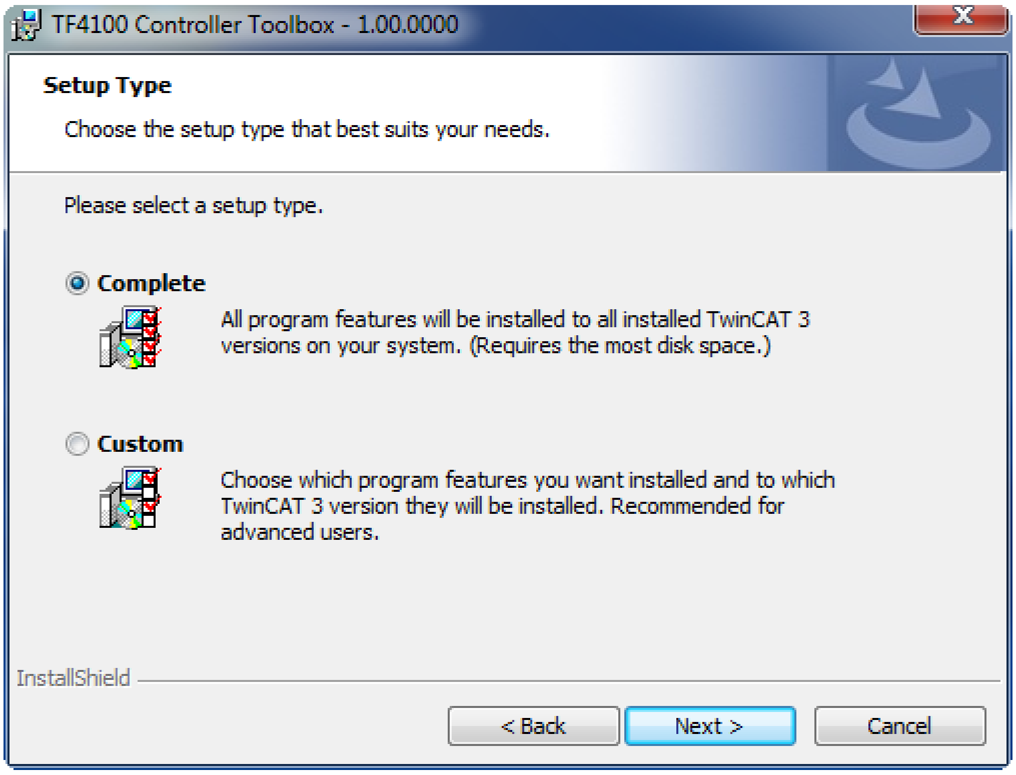
Customer Information

Please enter your information.

User Name: Max Mustermann

Organization: Mustermann Inc.

4. To install the full product, including all sub-components, please choose **Complete** as the Setup Type. Alternatively, you can also install each component separately by choosing **Custom**.



TF4100 Controller Toolbox - 1.00.0000

Setup Type

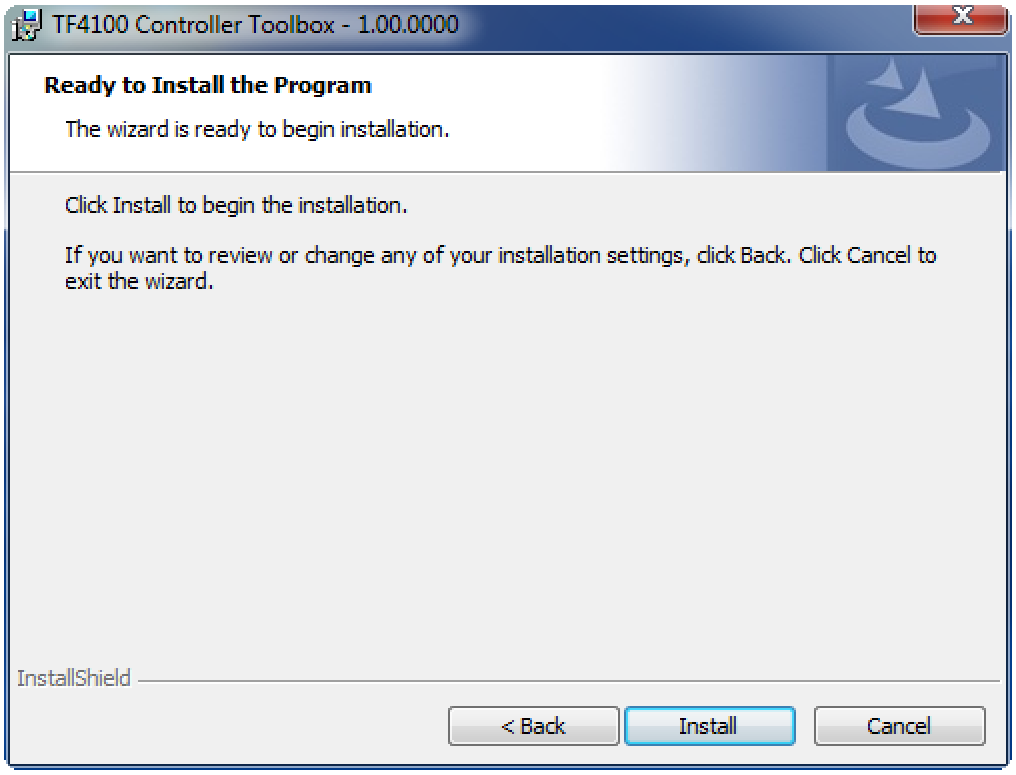
Choose the setup type that best suits your needs.

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.

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



TF4100 Controller Toolbox - 1.00.0000

Ready to Install the Program

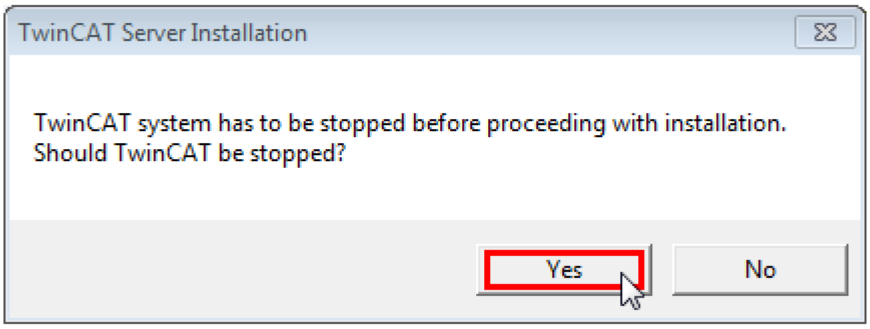
The wizard is ready to begin installation.

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.

The TwinCAT system must be stopped before proceeding with installation.

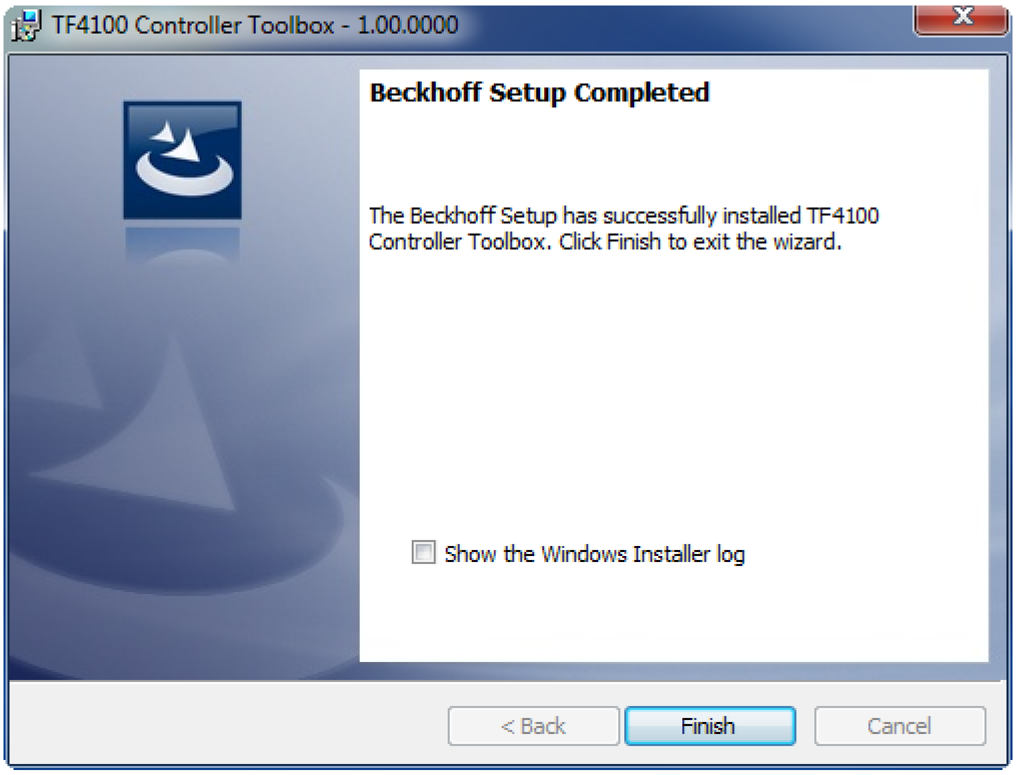
6. Confirm the Dialog with **Yes**.



TwinCAT Server Installation

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

7. Select **Finish** to end the installation process.



Beckhoff Setup Completed

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

☐ Show the Windows Installer log

⇨ The installation is now complete.

After a successful installation, the TC 3 Function needs to be licensed. [▶ 11]

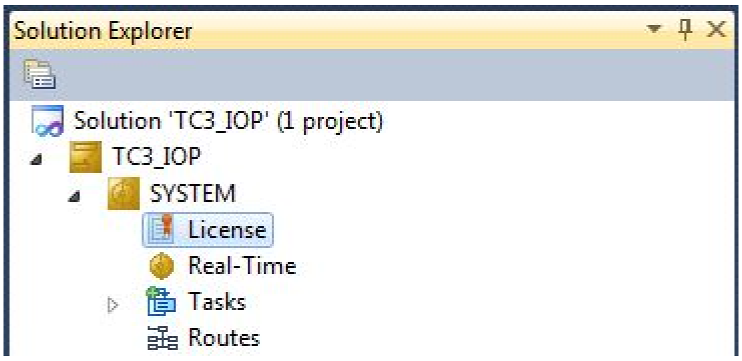
3.3 Licensing

The TwinCAT 3 functions are available both as a full and as a 7-Day trial version. Both license types can be activated via TwinCAT XAE.For more information about TwinCAT 3 licensing, please consult the TwinCAT 3 Help System.The following document describes both licensing scenarios for a TwinCAT 3 function on TwinCAT 3 and is divided into the following sections:

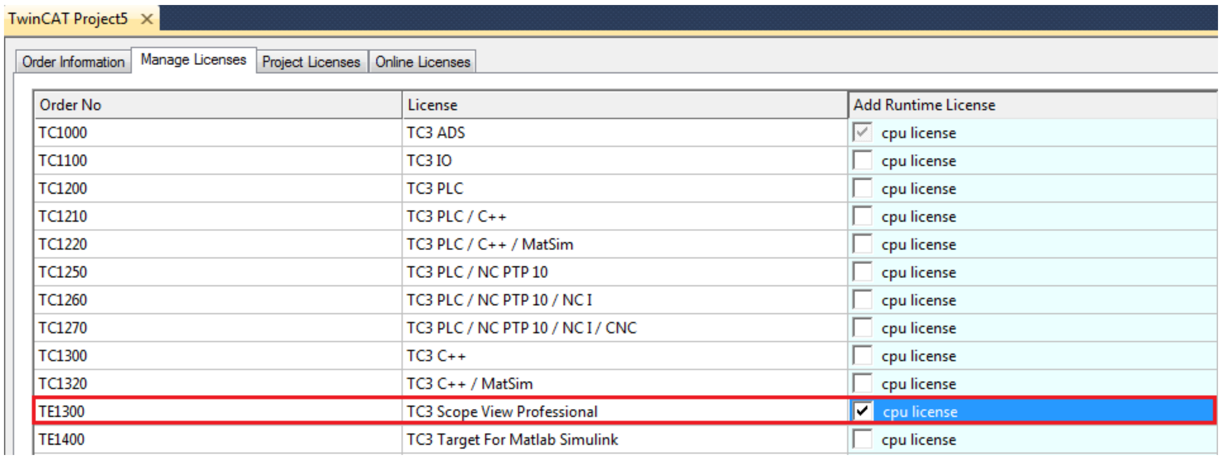
- Licensing a 7-Day trial version [▶ 11]
- Licensing a full version [▶ 12]

Licensing a 7-Day trial version

1. Start TwinCAT XAE
2. Open an existing TwinCAT 3 project or create a new project
3. In **Solution Explorer**, please navigate to the entry **System\License**



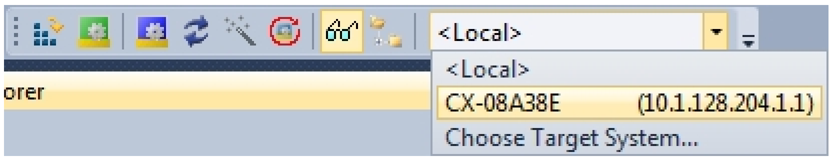
4. Open the tab **Manage Licenses** and add a **Runtime License** for your product (in this screenshot **TE1300: TC3 Scope View Professional**)



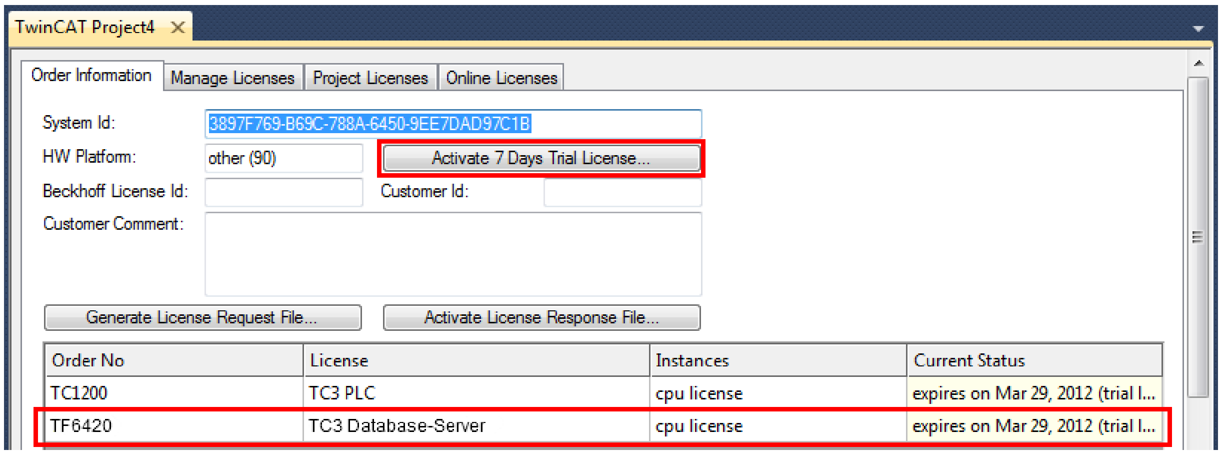
TwinCAT Project5

Order No	License	Add Runtime License
TC1000	TC3 ADS	✓ cpu license
TC1100	TC3 IO	cpu license
TC1200	TC3 PLC	cpu license
TC1210	TC3 PLC / C++	cpu license
TC1220	TC3 PLC / C++ / MatSim	cpu license
TC1250	TC3 PLC / NC PTP 10	cpu license
TC1260	TC3 PLC / NC PTP 10 / NC I	cpu license
TC1270	TC3 PLC / NC PTP 10 / NC I / CNC	cpu license
TC1300	TC3 C++	cpu license
TC1320	TC3 C++ / MatSim	cpu license
TE1300	TC3 Scope View Professional	✓ cpu license
TE1400	TC3 Target For Matlab Simulink	cpu license

5. **Optional**: If you would like to add a license for a remote device, you first need to connect to the remote device via TwinCAT XAE toolbar



6. Switch to the tab **Order Information** and click the button **Activate 7 Days Trial License...** to activate a test version



TwinCAT Project4

Order Information
Manage Licenses
Project Licenses
Online Licenses
Activate 7 Days Trial License...
Generate License Request File...
Activate License Response File...

System Id:	3897F769-B69C-788A-6450-9EE7DAD97C1B
HW Platform:	other (90)
Beckhoff License Id:	
Customer Id:	
Customer Comment:	

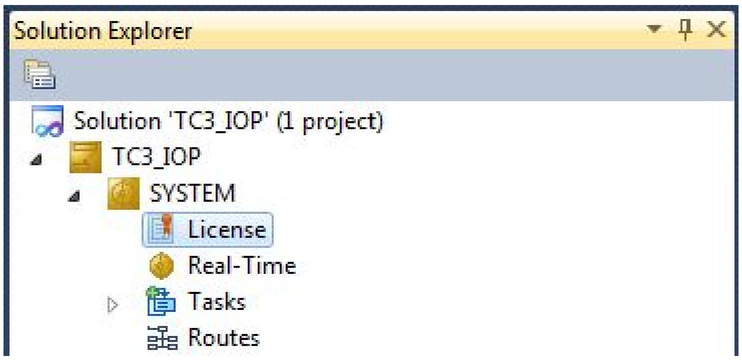
Order No	License	Instances	Current Status
TC1200	TC3 PLC	cpu license	expires on Mar 29, 2012 (trial l...
TF6420	TC3 Database-Server	cpu license	expires on Mar 29, 2012 (trial l...

7. Please restart TwinCAT 3 afterwards.

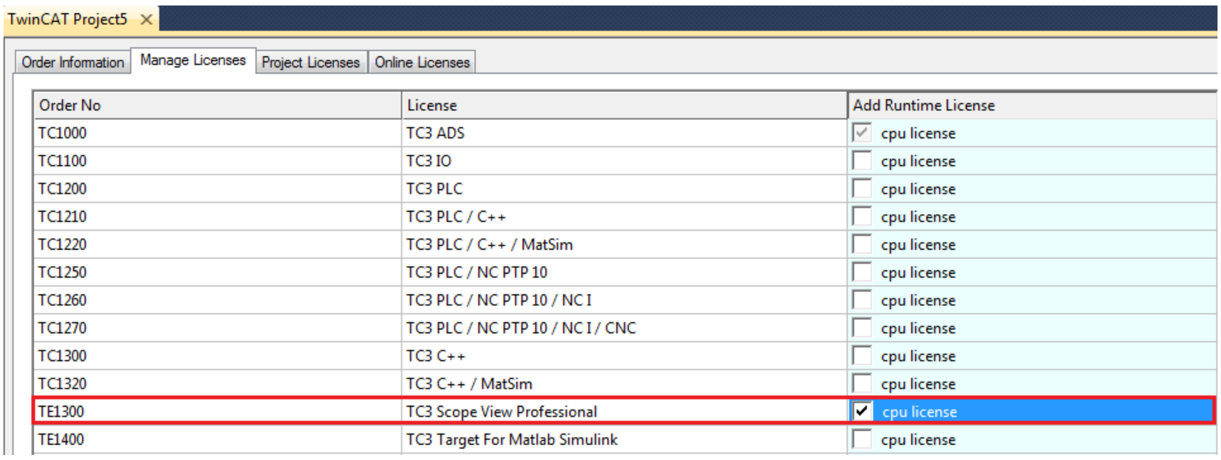
Licensing a full version

8. Start TwinCAT XAE
9. Open an existing TwinCAT 3 project or create a new project

10. In **Solution Explorer**, please navigate to the entry **SYSTEM\License**

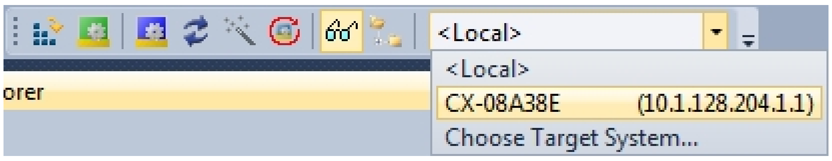


11. Open the tab **Manage Licenses** and add a **Runtime License** for your product (in this screenshot **TE1300: TC3 Scope View Professional**).

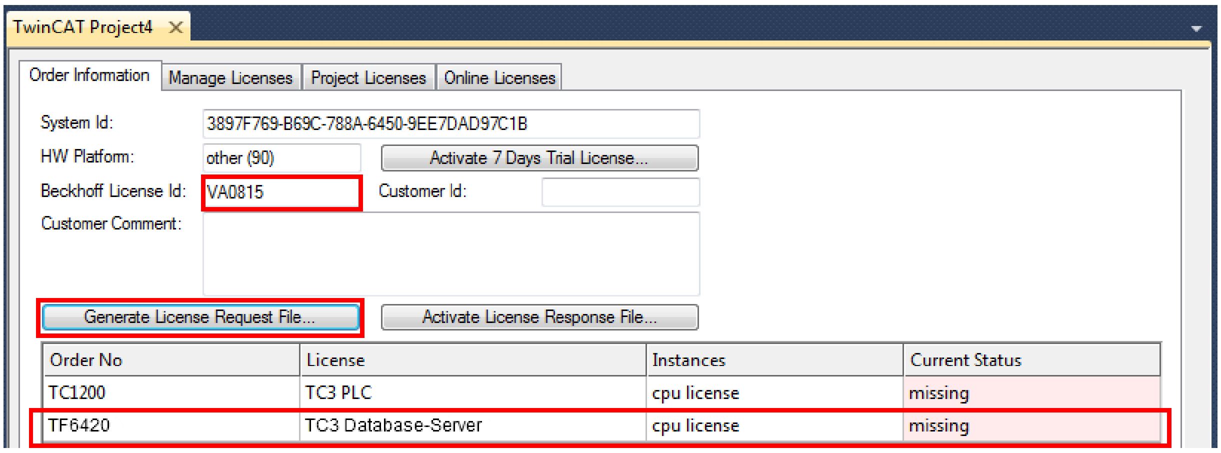


Order No	License	Add Runtime License
TC1000	TC3 ADS	✓ cpu license
TC1100	TC3 IO	cpu license
TC1200	TC3 PLC	cpu license
TC1210	TC3 PLC / C++	cpu license
TC1220	TC3 PLC / C++ / MatSim	cpu license
TC1250	TC3 PLC / NC PTP 10	cpu license
TC1260	TC3 PLC / NC PTP 10 / NC I	cpu license
TC1270	TC3 PLC / NC PTP 10 / NC I / CNC	cpu license
TC1300	TC3 C++	cpu license
TC1320	TC3 C++ / MatSim	cpu license
TE1300	TC3 Scope View Professional	✓ cpu license
TE1400	TC3 Target For Matlab Simulink	cpu license

12. **Optional:**If you would like to add a license for a remote device, you first need to connect to the remote device via TwinCAT XAE toolbar



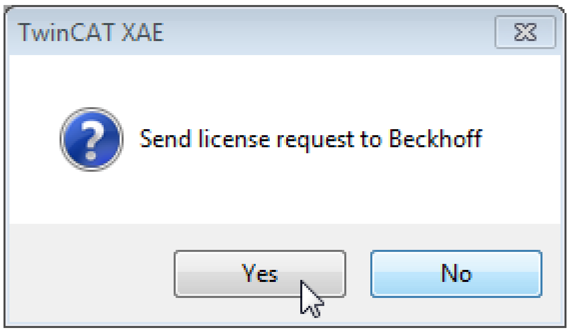
13. Navigate to the **Order Information** tab
The fields **System-ID** and **HW Platform** cannot be changed and just describe the platform for the licensing process in general a TwinCAT 3 license is always bound to these two identifiers:
the **System-ID** uniquely identifies your system.
The **HW Platform** is an indicator for the performance of the device.
14. Optionally, you may also enter an own order number and description for your convenience


TwinCAT Project4
Order Information
Manage Licenses
Project Licenses
Online Licenses
Activate 7 Days Trial License...
Generate License Request File...
Activate License Response File...

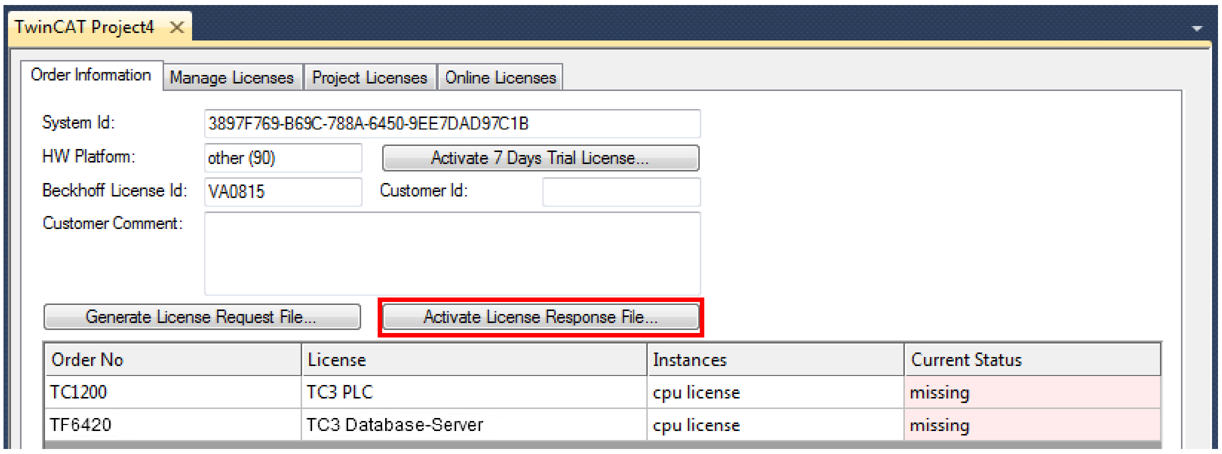
System Id: 3897F769-B69C-788A-6450-9EE7DAD97C1B
HW Platform: other (90)
Beckhoff License Id: VA0815
Customer Id:
Customer Comment:

Order No	License	Instances	Current Status
TC1200	TC3 PLC	cpu license	missing
TF6420	TC3 Database-Server	cpu license	missing

15. enter the **Beckhoff License ID** and click on **Generate License Request File....** If you are not aware of your **Beckhoff License ID** please contact your local sales representative.
16. After the license request file has been saved, the system asks whether to send this file via E-Mail to the Beckhoff Activation Server



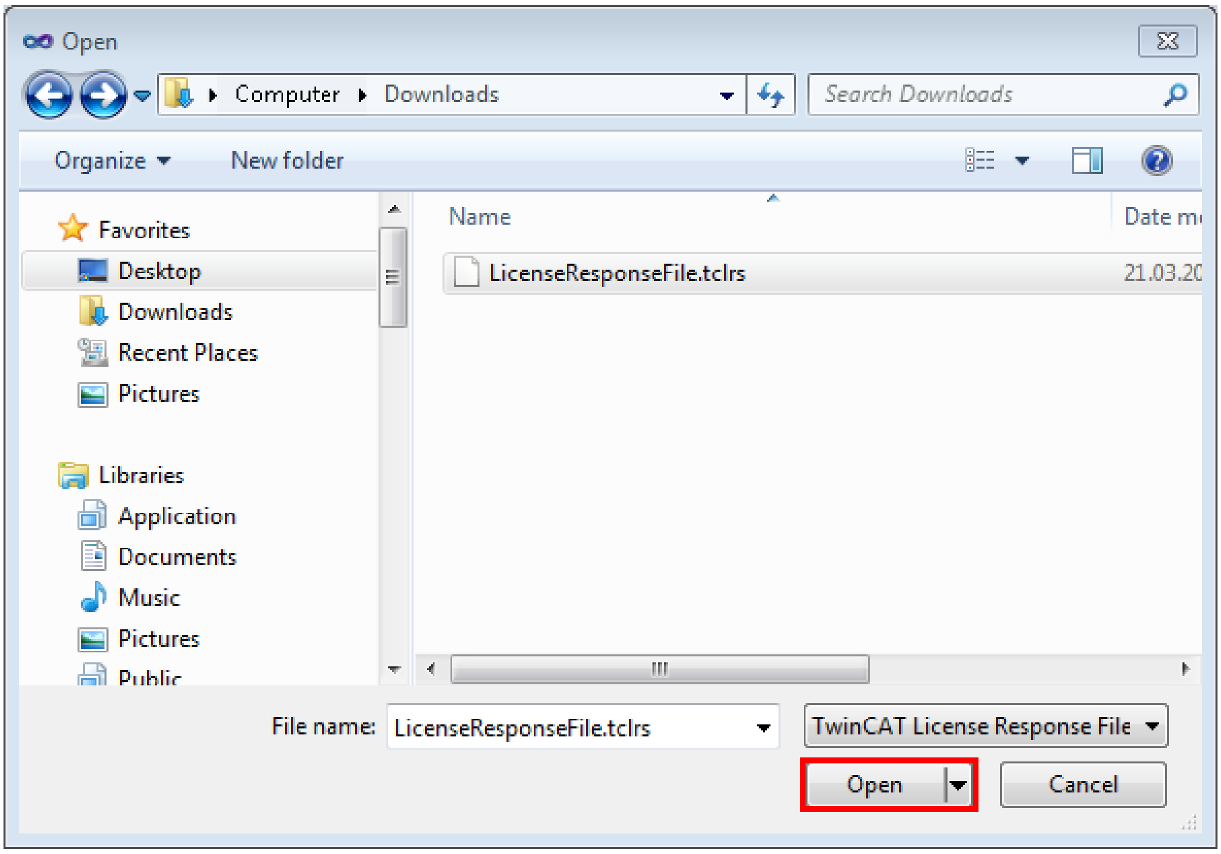
17. After clicking **Yes**, the standard E-Mail client opens and creates a new E-Mail message to "tclicense@beckhoff.com" which contains the "License Request File"
18. Send this Activation Request to Beckhoff
    NOTE! The License Response File will be sent to the same E-Mail address used for sending out the License Request File
19. After receiving the activation file, please click on the button **Activate License Response File...** in the TwinCAT XAE license Interface.


TwinCAT Project4
Order Information
Manage Licenses
Project Licenses
Online Licenses
Activate 7 Days Trial License...
Generate License Request File...
Activate License Response File...

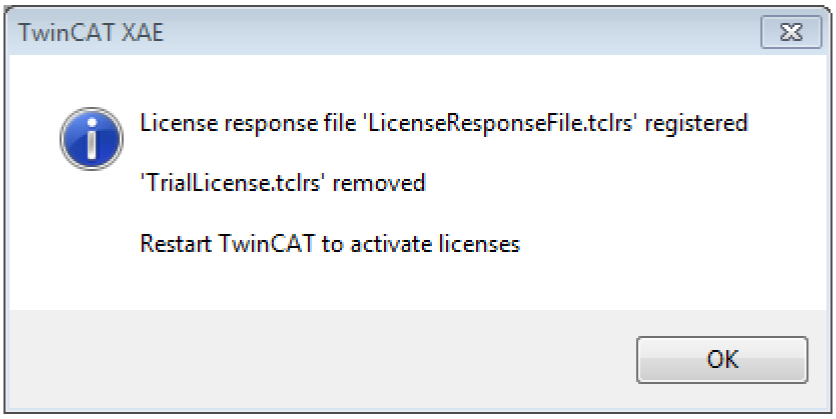
System Id: 3897F769-B69C-788A-6450-9EE7DAD97C1B
HW Platform: other (90)
Beckhoff License Id: VA0815
Customer Id:
Customer Comment:

Order No	License	Instances	Current Status
TC1200	TC3 PLC	cpu license	missing
TF6420	TC3 Database-Server	cpu license	missing

20. Select the received License response file and click on **Open**



21. The License Response File will be imported and all included licenses will be activated. If there have been any trial licenses, these will be removed accordingly.
22. Please restart TwinCAT to activate licenses.



License response file 'LicenseResponseFile.tclrs' registered

'TrialLicense.tclrs' removed

Restart TwinCAT to activate licenses

NOTE! The license file will be automatically copied to...\TwinCAT\3.1\Target\License on the local device.

3.4 Installation Windows CE

This part of the documentation describes, how you can install the TwinCAT 3 Function TF6310 TCP/IP on a Beckhoff Embedded PC Controller based on Windows CE.

The setup process consists of four steps:

1. Downloading the setup file
2. Installation on a host computer
3. Transfering the executable to the Windows CE device
4. Software installation

The last paragraph describes the Software upgrade

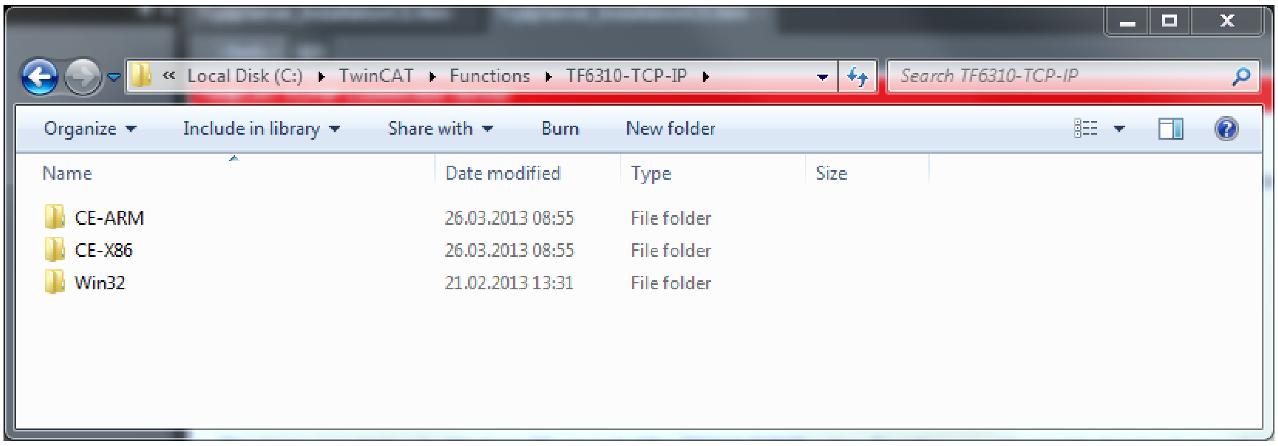
Downloading the setup file

The CAB installation file for Windows CE is part of the TFxxxx setup. Therefore you only need to download one setup file from www.beckhoff.com which contains binaries for Windows XP, Windows 7 and Windows CE (x86 and ARM).

Installation on a host computer

After installation, the install folder contains three directories - each one for a different hardware platform:

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



Name	Date modified	Type	Size
CE-ARM	26.03.2013 08:55	File folder	
CE-X86	26.03.2013 08:55	File folder	
Win32	21.02.2013 13:31	File folder	

The CE-ARM and CE-X86 folders contain the TFxxx (here TF6310) CAB-File for Windows CE - corresponding to the hardware platform of your Windows CE device. This file needs to be transfered to the Windows CE device.

Transfering the executable to the Windows CE device

Transfer the corresponding executable to you Windows CE device. This can be done via one of the following ways:

- via a Shared Folder
- via the integrated FTP-Server
- via ActiveSync
- via a CF/SD card

For more information, please consult the Windows CE section in our Infosys documentation system.

Software installation

After the CAB-File has been transfered via one of the above methods, you need to execute the file and acknowledge the following dialog with **Ok**. Restart your Windows CE device after the installation has finished.

After the restart has been completed, the *TFxxxx* executable files will be automatically started in background and is now available to use.

The software will be installed in the following directory on the CE device: *\Hard Disk\TwinCAT\Functions \TFxxxx*

Upgrade instructions

If you have already a version of TF6310 installed on your Windows CE device, you need to perform the following things on the Windows CE device to upgrade to a newer version:

1. Open the CE Explorer by clicking on **Start** > **Run** and entering **Explorer**
2. Navigate to *\Hard Disk\TwinCAT\Functions\TFxxx\xxxx*
3. Rename the file *Tc*.exe* to *Tc*.old*
4. Restart the Windows CE device
5. Transfer the new CAB-File to the CE device
6. Execute the CAB-File and install the new version
7. Delete *Tc*.old*.
8. Restart the Windows CE device

⇨ After the restart is complete, the new version is active.

4 PLC libraries

4.1 Overview

The PLC library **Tc2_TcXmlDataSrv** is supplied with the TC3 XML Server and copied into folder *...C:\TwinCAT\3.1\Components\Plc\Managed Libraries\Beckhoff Automation GmbH* during installation.

There are two function blocks for reading variables from the XML file:

- FB_XmlSrvRead
- FB_XmlSrvReadByName

and two function blocks for writing PLC variables to the XML file:

- FB_XmlSrvWrite
- FB_XmlSrvWriteByName

The first version (FB_XMLSrvRead, FB_XMLSrvWrite) uses the address and the size of the PLC variable for specifying the variable. The second version (FB_XMLSrvReadByName, FB_XMLSrvWriteByName) uses the symbol name for specifying the variable. The first version offers higher performance. In addition, the path of the XML file and the location of the variables within the XML document has to be transferred as input parameter to the function blocks.

Primitive data types

The following primitive data types are supported:

Data type	PLC example	XML example

UDINT, DINT, UINT, INT, USINT, SINT, DWORD, WORD, BYTE	value1 : DINT := -1;
	value2 : UDINT := 65535;
		<dataentry> <MAIN.value1>-1</MAIN.value1> <MAIN.value2>65535</MAIN.value2> </dataentry>

LREAL, REAL	value1 : LREAL = 1.2;	<dataentry> <MAIN.value1>1.2</MAIN.value1> </dataentry>
STRING	str1 : STRING = 'hallo';	<dataentry> <MAIN.str1>hallo</MAIN.str1> </dataentry>
TIME, DATE, TOD,DT	date1:DATE :=D#2005-05-04; (* Time types are stored in the XML file as DWORD*)	<dataentry> <MAIN.date1>1115164800</MAIN.date1> </dataentry>

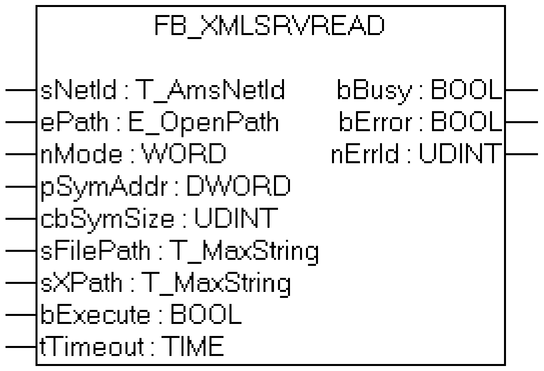
BOOL	bool1:BOOL := TRUE;
	bool2:BOOL := FALSE;
		<dataentry> <MAIN.bool1>true</MAIN.bool1> <MAIN.bool2>false</MAIN.bool2> </dataentry>

Requirements

Development environment	Target system type	PLC libraries to be linked
TwinCAT v3.1 Build 4011	PC or CX (x86, ARM)	Tc2_XmlDataSrv

4.2 Function blocks

4.2.1 FB_XmlSrvRead



The function block FB_XmlSrvRead can be used to initialise a PLC variable with data from an XML file. The input variable sXPath has to point to a valid node in the XML file specified via sFilePath. The symbol to be initialised is identified unambiguously by the symbol address and size.

VAR_INPUT

```
VAR_INPUT
    sNetId      : T_AmsNetId;
    ePath       : E_OpenPath := PATH_GENERIC;
    nMode       : WORD := XMLSRV_SKIPMISSING;
    pSymAddr    : DWORD;
    cbSymSize   : UDINT;
    sFilePath   : T_MaxString;
    sXPath      : T_MaxString;
    bExecute    : BOOL;
    tTimeout    : TIME := T#60s;
END_VAR
```

sNetId: String containing the network address of the TwinCAT 3 XML server. For the local computer (default) an empty string may be specified.

ePath: This input can be used to select a TwinCAT system path on the target device for opening the file.

nMode: This input can be used to control how the XML file is evaluated. Only the mode XMLSRV_SKIPMISSING is initially supported for the command XmlSrvRead.

pSymAddr: Address of the PLC variable to which the data from the XML file are to be written.

cbSymSize: Size of the PLC variable to which the data from the XML file are to be written.

sFilePath: Contains the path and file name for the file to be opened. The path can only point to the local computer's file system! This means that network paths cannot be used here!

sXPath: Contains the address of the tag in the XML document from which the data are to be written. The address must be a valid XPath instruction. The name of the tag must be different from the name of the symbol.

bExecute: The block is activated by a positive edge at this input.

tTimeout: Maximum time allowed for the execution of the function block.

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy   : BOOL;
    bError  : BOOL;
    nErrId  : UDINT;
END_VAR
```

bBusy: This output is set when the function block is activated. It remains set until feedback is received.

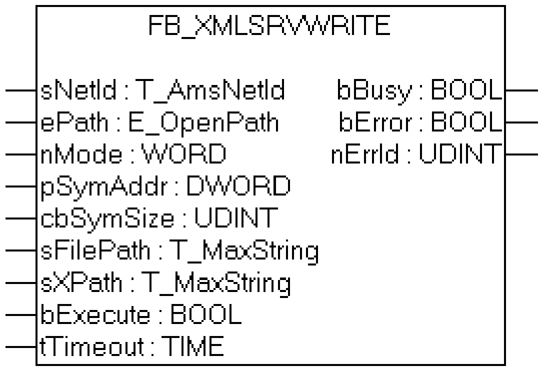
bError: This output is set up after the bBusy output has been reset if there has been an error in transmission of the command.

nErrId: Returns the TC3 XML server error number [▸ 34], if a bError output is set.

Requirements

Development environment	Target system type	PLC libraries to be linked
TwinCAT v3.1 Build 4011	PC or CX (x86, ARM)	Tc2_XmlDataSrv

4.2.2 FB_XmlSrvWrite



The function block FB_XmlSrvWrite can be used for writing the value of a PLC variable into an XML file. The input variable sXPath has to point to a valid node in the XML file specified via sFilePath. The symbol to be written is identified unambiguously by the symbol address and size.

VAR_INPUT

```
VAR_INPUT
    sNetId      : T_AmsNetId;
    ePath       : E_OpenPath := PATH_GENERIC;
    nMode       : WORD := XMLSRV_SKIPMISSING;
    pSymAddr    : DWORD;
    cbSymSize   : UDINT;
    sFilePath   : T_MaxString;
    sXPath      : T_MaxString;
    bExecute    : BOOL;
    tTimeout    : TIME := T#60s;
END_VAR
```

sNetId: String containing the network address of the TwinCAT 3 XML server. For the local computer (default) an empty string may be specified.

ePath: This input can be used to select a TwinCAT system path on the target device for opening the file.

nMode: This input can be used to control which data are written into the XML file. XMLSRV_SKIPMISSING and XMLSRV_ADDMISSING mode are available for the XmlSrvWrite command. In XMLSRV_SKIPMISSING mode, only those subelements of a PLC symbol that already exist in the XML file are written to the XML file. In XMLSRV_ADDMISSING mode, missing subelements are added to the XML file.

pSymAddr: Address of the PLC variable to be written to the XML file.

cbSymSize: Size of the PLC variable to be written to the XML file.

sFilePath: Contains the path and file name for the file to be opened. The path can only point to the local computer’s file system! This means that network paths cannot be used here!

sXPath: Contains the address of the tag in the XML document from which the data are to be written. The address must be a valid XPath instruction. The name of the tag must be different from the name of the symbol.

bExecute: The block is activated by a positive edge at this input.

tTimeout: Maximum time allowed for the execution of the function block.

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy   : BOOL;
    bError  : BOOL;
    nErrId  : UDINT;
END_VAR
```

bBusy: This output is set when the function block is activated. It remains set until feedback is received.

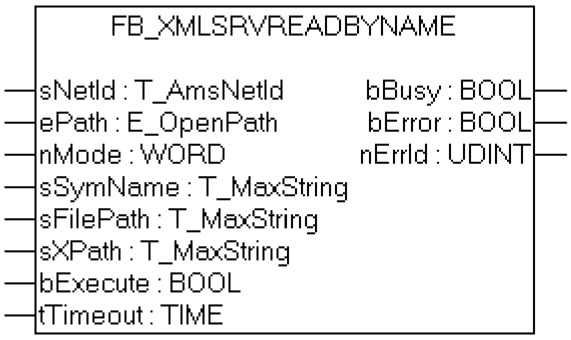
bError: This output is set up after the bBusy output has been reset if there has been an error in transmission of the command.

nErrId: Returns the TC3 XML server error number [▸ 34], if a bError output is set.

Requirements

Development environment	Target system type	PLC libraries to be linked
TwinCAT v3.1 Build 4011	PC or CX (x86, ARM)	Tc2_XmlDataSrv

4.2.3 FB_XmlSrvReadByName



The function block FB_XmlSrvReadByName can be used to initialize a PLC variable with data from an XML file. The input variable sXPath has to point to a valid node in the XML file specified via sFilePath. The symbol to be initialized is identified unambiguously by the symbol name.

VAR_INPUT

```
VAR_INPUT
    sNetId      : T_AmsNetId;
    ePath       : E_OpenPath := PATH_GENERIC;
    nMode       : WORD := XMLSRV_SKIPMISSING;
    sSymName    : T_MaxString;
    sFilePath   : T_MaxString;
    sXPath      : T_MaxString;
    bExecute    : BOOL;
    tTimeout    : TIME := T#60s;
END_VAR
```

sNetId: String containing the network address of the TwinCAT 3 XML server. For the local computer (default) an empty string may be specified.

ePath: This input can be used to select a TwinCAT system path on the target device for opening the file.

nMode: This input can be used to control how the XML file is evaluated. Only the mode XMLSRV_SKIPMISSING is currently supported for the command XmlSrvReadByName.

sSymName: Name of the PLC symbol to which the data from the XML file are to be written.

sFilePath: Contains the path and file name for the file to be opened. The path can only point to the local computer’s file system! This means that network paths cannot be used here!

sXPath: Contains the address of the tag in the XML document from which the data are to be written. The address must be a valid XPath instruction. The name of the tag must be different from the name of the symbol.

bExecute: The block is activated by a positive edge at this input.

tTimeout: Maximum time allowed for the execution of the function block.

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy   : BOOL;
    bError  : BOOL;
    nErrId  : UDINT;
END_VAR
```

bBusy: This output is set when the function block is activated. It remains set until feedback is received.

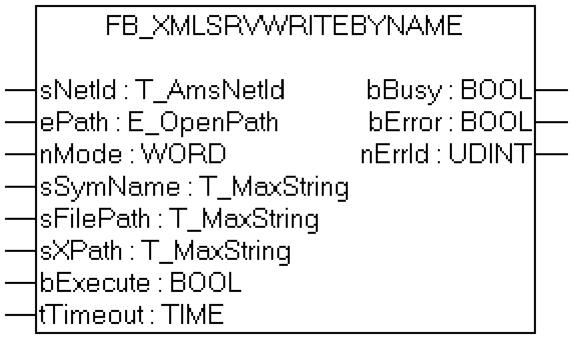
bError: This output is set up after the bBusy output has been reset if there has been an error in transmission of the command.

nErrId: Returns the TC3 XML server error number [▶ 34], if a bError output is set.

Requirements

Development environment	Target system type	PLC libraries to be linked
TwinCAT v3.1 Build 4011	PC or CX (x86, ARM)	Tc2_XmlDataSrv

4.2.4 FB_XmlSrvWriteByName



The function block FB_XmlSrvWriteByName can be used for writing the value of a PLC variable into an XML file. The input variable sXPath has to point to a valid node in the XML file specified via sFilePath. The symbol to be written is identified unambiguously by the symbol name.

VAR_INPUT

```
VAR_INPUT
    sNetId      : T_AmsNetId;
    ePath       : E_OpenPath := PATH_GENERIC;
    nMode       : WORD := XMLSRV_SKIPMISSING;
    sSymName    : T_MaxString;
    sFilePath   : T_MaxString;
    sXPath      : T_MaxString;
    bExecute    : BOOL;
    tTimeout    : TIME := T#60s;
END_VAR
```

sNetId: String containing the network address of the TwinCAT 3 XML server. For the local computer (default) an empty string may be specified.

ePath: This input can be used to select a TwinCAT system path on the target device for opening the file.

nMode: This input can be used to control which data are written into the XML file. XMLSRV_SKIPMISSING and XMLSRV_ADDMISSING mode are available for the XmlSrvWriteByte command. In XMLSRV_SKIPMISSING mode, only those subelements of a PLC symbol that already exist in the XML file are written to the XML file. In XMLSRV_ADDMISSING mode, missing subelements are added to the XML file.

sSymName: Name of the PLC symbol to be written to the XML file.

sFilePath: Contains the path and file name for the file to be opened. The path can only point to the local computer’s file system! This means that network paths cannot be used here!

sXPath: Contains the address of the tag in the XML document from which the data are to be written. The address must be a valid XPath instruction. The name of the tag must be different from the name of the symbol.

bExecute: The block is activated by a positive edge at this input.

tTimeout: Maximum time allowed for the execution of the function block.

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy   : BOOL;
    bError  : BOOL;
    nErrId  : UDINT;
END_VAR
```

bBusy: This output is set when the function block is activated. It remains set until feedback is received.

bError: This output is set up after the bBusy output has been reset if there has been an error in transmission of the command.

nErrId: Returns the TC3 XML server error number [▶ 34], if a bError output is set.

Requirements

Development environment	Target system type	PLC libraries to be linked
TwinCAT v3.1 Build 4011	PC or CX (x86, ARM)	Tc2_XmlDataSrv

4.3 Global constants

4.3.1 Global Variables

VAR_GLOBAL CONSTANT

```
VAR_GLOBAL CONSTANT

XMLSRV_AMSPORT :UINT :=10600;

XMLSRV_IGR_CLOSE :UDINT := 121;
XMLSRV_IGR_READ :UDINT := 122;
XMLSRV_IGR_WRITE :UDINT := 123;
XMLSRV_IGR_OPENREAD :UDINT := 124;
XMLSRV_IGR_OPENWRITE :UDINT := 125;

XMLSRV_SKIPMISSING :WORD := 0;
XMLSRV_ADDMISSING :WORD := 1; (*for write commands*)

XMLSRV_MAX_FRAGSIZE :UDINT := 16#40000;

XMLSRVERROR_INTERNAL :UDINT:= 16#8000;
XMLSRVERROR_NOTFOUND :UDINT:= 16#8001;
XMLSRVERROR_PARSERERROR :UDINT:= 16#8002;
XMLSRVERROR_INCOMPATIBLE :UDINT:= 16#8003;
XMLSRVERROR_NOMEMORY :UDINT:= 16#8004;
XMLSRVERROR_ADDNODE :UDINT:= 16#8005;
XMLSRVERROR_INVALIDXPATH :UDINT:= 16#8006;
END_VAR
```

Requirements

Development environment	Target system type	PLC libraries to be linked
TwinCAT v3.1 Build 4011	PC or CX (x86, ARM)	Tc2_XmlDataSrv

4.3.2 Library version

All libraries have a specific version. This version is shown in the PLC library repository too.
A global constant contains the library version information:

Global_Version

```
VAR_GLOBAL CONSTANT
    stLibVersion_Tc2_TcXmlDataSrv : ST_LibVersion;
END_VAR
```

Tc2_TcXmlDataSrv: Version number of the Tc2_TcXmlDataSrv library (type: ST_LibVersion)

To compare the existing version to a required version the function F_CmpLibVersion (defined in Tc2_System library) is offered.

NOTE! All other options for comparing library versions, which you may know from TwinCAT 2, are outdated!

5 Examples

The following pages contain examples (referred to as samples) that illustrate the operation of the TC3 XML server. The examples are numbered consecutively and can be downloaded as a PLC project from here: https://infosys.beckhoff.com/content/1033/TF6421_Tc3_XML_Server/Resources/zip/1635013643.zip

- Getting started [▶ 25]
  This section describes the general operation of the TwinCAT XML Data Server, the function of structures and arrays, and the access to individual nodes.
- Function blocks [▶ 26]
  The application and configuration of the function blocks of the PLC library *TcXmlDataSrv.Lib* is explained based on four examples. (Examples 1-4)
- Further examples [▶ 28]
  This document contains further examples, which describe the one-time initialization during PLC startup and cyclic writing, for instance. (Examples 5-6)
- Production example [▶ 31]
  The production example illustrates the processing of a small production order. It uses FB_XmlSrvReadByName and FB_XmlSrvWriteByName. (Example 7)

5.1 Getting Started

The following section describes the basic operating principle of the TC3 XML server, based on short examples.

The global variable .Var1 of type DINT is defined in the PLC. It should be saved in an XML file as follows:

```
<dataentry>
   <Var1>10</Var1>
</dataentry>
```

To this end the input variables of the function block FB_XmlSrvWrite have to be set as follows:

```
fbRead.pSymAddr     := ADR(value1);
fbRead.cbSymSize    := SIZEOF(value1);
fbRead.sFilePath    := Pfad zur XML-Datei z.B. 'C:\Test.xml';
fbRead.sXPath       := '/dataentry/Var1';
```

The root element name of the variables in the XML file is freely selectable. Simply adapt the path in the input variable sXPath. An XML file may contain definitions for several variables:

```
<dataentry>
   <Var1>10</Var1>
   <Var2>100</Var2>
   <Var3>
      <a>100</a>
      <b>10</b>
   </Var3>
</dataentry>
```

To access the symbol .Var3.a, for example, sXpath must be set to '/dataentry/Var3.a'.

Structures

XML files have the same hierarchical structure as the PLC. Individual subelements of the XML file may be skipped: The subelements of the structure must have the same names as in the PLC, otherwise they are skipped. If subelements of the XML file cannot be converted to the correct data type, they are also skipped.

Example:

The global variable .Var2 of type ST_MYSTRUCT is defined in the PLC:

```
TYPE ST_MYSTRUCT:
STRUCT
    a: UINT;
    b: DINT;
    c: LREAL;
    d: STRING;
END_STRUCT
END_TYPE
```

The XML file could then look as follows:

```
<variables>
   <Var1>10</Var1>
   <Var2>     <!-- sXPath := '/variables/Var2' -->
      <a>100</a>
      <b>-10</b>
      <c>1.2</c>
      <d>Hallo</d>
   </Var2>
</variables>
```

In this case all subelements are defined fully and correctly, so that the variable is fully initialized. In the following example, on the other hand, only subelement c is serialized:

```
<variables>
   <Var1>10</Var1>
   <Variable2> <!-- sXPath := '/variables/Variable2' -->
      <Info>dies ist ein Test</Info>
      <a>-100</a>
      <c>1.2</c>
   </Variable2>
</variables>
```

Subelement a cannot be converted because it is negative and UINT is required. Subelement b is missing completely. The tag <Info> is skipped, because it is not defined in the PLC file.

Arrays

In order to specify an array index, the attribute "index" has to be used for the individual array elements. Individual array elements can be omitted, in which case they will be skipped.

Example:

A variable .array1 of type ARRAY[1..4] OF DINT is defined in the PLC. The XML file will then look as follows:

```
<dataentry>
   <array1 index="1">10</array1>
   <array1 index="2">10</array1>
   <array1 index="3">10</array1>
   <array1 index="4">10</array1>
</dataentry>
```

Requirements

Development environment	Target system type	PLC libraries to be linked
TwinCAT v3.1 Build 4011	PC or CX (x86, ARM)	Tc2_XmlDataSrv

5.2 Function blocks

The following samples show the handling of the functionblocks of the Tc2_XmlDataSrv-Library. The PLC projekt, which contains the samples, can be downloaded here: https://infosys.beckhoff.com/content/1033/TF6421_Tc3_XML_Server/Resources/zip/1635013643.zip

All samples work with the structure ST_MYSTRUCT, which in turn includes ST_INNTERSTRUCT. Both Structures are shown in the following:

Structures

```
TYPE ST_MYSTRUCT:
STRUCT
   fReal      : REAL;
   bBool      : ARRAY [0..2] OF BOOL;
   stInner    : ST_INNTERSTRUCT;
END_STRUCT
END_TYPE

TYPE ST_INNTERSTRUCT:
```

```
STRUCT
    nInteger  : INT;
    sString   : STRING;
END_STRUCT
END_TYPE
```

Sample 1: Write operation with FB_XmlSrvWrite

In the first step the structure ST_MyStruct is to be written into an XML file. The mode is set to XMLSRV_ADMISSING, so that the XML file is generated automatically and the structure is created within it. Folders are not created automatically! For larger structures this approach is also recommend if the file is only to be read later. In this way the XML file does not have to be created manually, and errors are avoided.

```
(* Sample1 creates an XML-file under the path C:\Test.xml and writes value1 to it.
   FUNCTIONBLOCK: FB_XmlSrvWrite *)

PROGRAM Sample1
VAR
   value1          : ST_MyStruct;
   fbXmlSrvWrite   : FB_XmlSrvWrite;
   bExecute        : BOOL;
   sFilePath       : T_MaxString := 'C:\Test.xml'; (* CE: '\Hard Disk\Test.xml' *)
   sXPath          : T_MaxString := '/dataentry/MAIN.value1';
END_VAR

fbXmlSrvWrite(
 nMode        := XMLSRV_ADDMISSING,
 pSymAddr     := ADR(value1),
 cbSymSize    := SIZEOF(value1),
 sFilePath    := sFilePath,
 sXPath       := sXPath,
 bExecute     := bExecute
);
bExecute := TRUE;
```

Sample 2: Write operation with FB_XmlSrvWriteByName

Sample 2 leads to the same result as sample 1, but the block FB_XmlSrvWriteByName is used. Sample 1 offers higher performance.

```
(* Sample2 creates an XML-file under the path C:\Test.xml and writes value1 to it.
   FUNCTIONBLOCK: FB_XmlSrvWriteByName *)

PROGRAM Sample2
VAR
    value1          : ST_MyStruct;
    fbXmlSrvWrite   : FB_XmlSrvWriteByName;
    bExecute        : BOOL;
    sSymName        : T_MaxString := 'Sample2.value1';
    sFilePath       : T_MaxString := 'C:\Test.xml'; (* CE: '\Hard Disk\Test.xml' *)
   sXPath          : T_MaxString := '/dataentry/MAIN.value1';
END_VAR

fbXmlSrvWrite(
 nMode       := XMLSRV_ADDMISSING,
 sSymName    := sSymName,
 sFilePath   := sFilePath,
 sXPath      := sXPath,
 bExecute    := bExecute
);
bExecute:= TRUE;
```

XML file

In both examples the XML file *'Test.XML'* is created under *C:\*. In order to ensure that the XML file has the same content in both variants, the same sXPath is used, i.e. *'/dataentry/MAIN.value'*, although value1 is not stored in Main, but directly in the respective programs. sSymName (Sample 2) specifies the location of the variables in TwinCAT: *'Sample2.value1'!*

```
<dataentry>
  <MAIN.value1>
    <fReal>0</fReal>
    <bBool index="0">false</bBool>
    <bBool index="1">false</bBool>
    <bBool index="2">false</bBool>
    <stInner>
```

```
      <nInteger>0</nInteger>
      <sString></sString>
    </stInner>
  </MAIN.value1>
</dataentry>
```

Sample 3: Read operation with FB_XmlSrvRead

The following section describes the process of reading the structure of the XML file created in sample 1 and/or sample 2. Sample 3 uses the block FB_XmlSrvRead for this purpose.

```
(* Sample3 reads an XML-file (C:\Test.xml)
   FUNCTIONBLOCK: FB_XmlSrvRead *)

PROGRAM Sample3

VAR
    value1         : ST_MyStruct;
    fbXmlSrvRead   : FB_XmlSrvRead;
    bExecute       : BOOL;
    sFilePath      : T_MaxString := 'C:\Test.xml'; (* CE: '\Hard Disk\Test.xml' *)
    sXPath         : T_MaxString := '/dataentry/MAIN.value1';
END_VAR

fbXmlSrvRead(
 pSymAddr   := ADR(value1),
 cbSymSize  := SIZEOF(value1),
 sFilePath  := sFilePath,
 sXPath     := sXPath,
 bExecute   := bExecute
);
bExecute:= TRUE;
```

Sample 4: Read operation with FB_XmlSrvReadByName

Sample 4 shows the read operation using the block FB_XmlSrvReadByName.

```
(* Sample4 reads an XML-file (C:\Test.xml)
   FUNCTIONBLOCK: FB_XmlSrvReadByName *)

PROGRAM Sample4
VAR
   value1          : ST_MyStruct;
   fbXmlSrvRead    : FB_XmlSrvReadByName;
   bExecute        : BOOL;
   sSymName        : T_MaxString := 'Sample4.value1';
   sFilePath       : T_MaxString := 'C:\Test.xml'; (* CE: '\Hard Disk\Test.xml' *)
   sXPath          : T_MaxString := '/dataentry/MAIN.value1';
END_VAR

fbXmlSrvRead(
 sSymName    := sSymName,
 sFilePath   := sFilePath,
 sXPath      := sXPath,
 bExecute    := bExecute
);
bExecute:= TRUE;
```

Like for sample 2, note that sSymName and sXPath differ: sXPath indicates the path within the XML file. This was specified in sample 1 and/or sample 2. sSymName indicates the symbol name of the TwinCAT variable and is therefore *'Sample4.value1*.

Requirements

Development environment	Target system type	PLC libraries to be linked
TwinCAT v3.1 Build 4011	PC or CX (x86, ARM)	Tc2_XmlDataSrv

5.3 Further Samples

The following samples show different types of application of the TC3 XML Server. The PLC-projekt which contains the samples can be downloaded here: https://infosys.beckhoff.com/content/1033/TF6421_Tc3_XML_Server/Resources/zip/1635013643.zip

Sample 5 shows an initialization once at program startup, Sample 6 shows cyclic and event-driven printing procedures. Both samples again use the structure ST_MYSTRUCT, which in turn includes ST_INNTERSTRUCT. Both Structures are shown in the following:

Structures

```
TYPE ST_MYSTRUCT:
STRUCT
    fReal    : REAL;
    bBool    : ARRAY [0..2] OF BOOL;
    stInner  : ST_INNTERSTRUCT;
END_STRUCT
END_TYPE

TYPE ST_INNTERSTRUCT:
STRUCT
    nInteger : INT;
    sString  : STRING;
END_STRUCT
END_TYPE
```

Sample 5: One-time initialization at program startup

Sample 5 shows the one-time initialization of value1 on program startup. The function block FB_XmlSrvRead is used.

```
(* Sample5 reads and initializes value1 when the PLC is started FUNCTIONBLOCK: FB_XmlSrvRead *)

PROGRAM Sample5
VAR
   value1        : ST_MyStruct;
   fbXmlSrvRead  : FB_XmlSrvRead;
   bExecute      : BOOL;
   sFilePath     : T_MaxString := 'C:\Test.xml'; (* CE: '\Hard Disk\Test.xml' *)
   sXPath        : T_MaxString := '/dataentry/MAIN.value1';
   nState        : INT := 0;
END_VAR

CASE nState OF
0: (* initialize *)
   fbXmlSrvRead(
      pSymAddr   := ADR(value1),
      cbSymSize  := SIZEOF(value1),
      sFilePath  := sFilePath,
      sXPath     := sXPath,
      bExecute   := bExecute
   );
   fbXmlSrvRead(bExecute:= TRUE);
   nState:= 1;

1: (* wait for read operation *)
   fbXmlSrvRead(bExecute:= FALSE);
   IF NOT fbXmlSrvRead.bBusy AND NOT fbXmlSrvRead.bError THEN
      nState:= 2;
   ELSIF fbXmlSrvRead.bError THEN
      nState:= 100;
   END_IF

2: (* operations *)
 ;

100:(* errorState *)
 ;

END_CASE
```

Sample 6: Cyclic and event-driven writing

The following example generates a new XML file every 20 seconds and writes the familiar structure into it. As usual, the file name is generated based on the current Windows date and time and a string. The write process can also be started by pressing switch (or by setting the switch variable bButton). If the write process is triggered several times within one second, the current file is overwritten.

```
(* Sample6: Every 20s value1 will be written into a new XML-File named after the current date and
 time. Furthermore you can activate the printing procedure by pressing a button (or setting the
 corresponding variable *)
```

```
PROGRAM Sample6
VAR
   value1           : ST_MyStruct;
   fbXmlSrvWrite    : FB_XmlSrvWrite;

   sFileFolder      : T_MaxString :='C:\'; (* CE: '\Hard Disk\' *)
   sFileName        : T_MaxString:= '_test.xml';
   sFilePathWrite   : T_MaxString
   (*sFilePathWrite = sFileFolder + time + sFileName*)

   sXPathWrite      : T_MaxString :='/dataentry/MAIN.value1';
   ntGetTime        : NT_GetTime;
   stMyTimestruct   : TIMESTRUCT;
   iState           : INT := 1;
   bTwentySec       : BOOL:= FALSE;
   bButton          : BOOL:= FALSE;
   bTwentySecOver   : BOOL;
   triggerWrite     : R_TRIG;
   triggerButton    : R_TRIG;
END_VAR


triggerButton(CLK:= bButton);

CASE iState OF
0: (* idle state *)
 ;

1: (* initialize *)
   fbXmlSrvWrite(nMode:=XMLSRV_ADDMISSING, pSymAddr:= ADR(value1),
   cbSymSize:= SIZEOF(value1));
   ntGetTime(START:= TRUE, TIMESTR=>stMyTimestruct); (* get Windows time *)
   IF NOT ntGetTime.BUSY AND NOT ntGetTime.ERR THEN
      iState:= 2;
   ELSIF ntGetTime.ERR THEN
      iState:= 100;
 END_IF

2: (* working state *)
   (* change some values - replace with production-process *)
   value1.stInner.nInteger:= value1.stInner.nInteger + 1;
   IF value1.stInner.nInteger = 32767 THEN
      value1.stInner.nInteger:= 0;
   END_IF(* get Windows time *)
   ntGetTime(START:= FALSE);
   IF NOT ntGetTime.BUSY AND NOT ntGetTime.ERR THEN
      ntGetTime(START:= TRUE, TIMESTR=>stMyTimestruct);
   ELSIF ntGetTime.ERR THEN
      iState:= 100;
   END_IF

   (* check if 20s have passed*)
   IF stMyTimestruct.wSecond = 0 OR stMyTimestruct.wSecond = 20
                    OR stMyTimeStruct.wSecond = 40 THEN
      bTwentySecOver:= TRUE;
      ELSE
      bTwentySecOver:= FALSE;
   END_IF

   (* if 20s have passed => trigger writing-process *)
   triggerWrite(CLK:=bTwentySecOver);
   IF (triggerWrite.Q OR triggerButton.Q) AND NOT fbXmlSrvWrite.bBusy AND NOT fbXml-
SrvWrite.bError THEN
      (* create filename *)
      sFilePathWrite:= CONCAT(sFileFolder, SYSTEMTIME_TO_STRING(stMy-
Timestruct)); (* set folder + time *)
      sFilePathWrite:= DELETE(STR:= sFilePathWrite, LEN:= 4 , POS:= LEN(STR:=sFilePath-
Write)-3); (* delete milliseconds *)
      sFilePathWrite:= REPLACE(STR1:= sFilePathWrite , STR2:= '.' , L:= 1,
      P:= LEN(STR:=sFilePathWrite)-2); (* replace colon with point *)
      sFilePathWrite:= REPLACE(STR1:= sFilePathWrite , STR2:= '.' , L:= 1,
      P:= LEN(STR:=sFilePathWrite)-5); (* replace colon with point *)
      sFilePathWrite:= CONCAT(sFilePathWrite, sFileName); (* add filename (default: test) *)

      (*change value 1*)
      value1.stInner.sString := sFilePathWrite;
      value1.stInner.nInteger := stMyTimestruct.wSecond;
      (* write *)
      fbXmlSrvWrite(sFilePath:=sFilePathWrite, sXPath:=sXPathWrite, bExecute:= TRUE);

   ELSIF fbXmlSrvWrite.bError THEN
      iState:= 100;
   END_IF

   (* reset fbXmlSrvWrite *)
   IF fbXmlSrvWrite.bBusy AND NOT tGetTime.ERR THEN
      fbXmlSrvWrite(bExecute:= FALSE);
   ELSIF ntGetTime.ERR THEN
      iState:= 100;
   END_IF

100: (* error state*)
 ;

END_CASE
```

Requirements

Development environment	Target system type	PLC libraries to be linked
TwinCAT v3.1 Build 4011	PC or CX (x86, ARM)	Tc2_XmlDataSrv

5.4 Production example

Sample 7 (production example)

This section describes an example for a production order. Production data such as component length, component width, etc. are read in a structure for initialization (XML read), the production takes place with a corresponding quantity, and the production order is completed with an entry in the XML file (write). To start the program, the XML file first has to be saved in the location that matches the file path, and in the PLC program the variable bStart has to be set to TRUE.

The PLC project containing the examples can be downloaded from here: https://infosys.beckhoff.com/content/1033/TF6421_Tc3_XML_Server/Resources/zip/1635013643.zip

Variable declaration

```
PROGRAM Sample7
VAR
   fbXmlSrvReadByName   : FB_XmlSrvReadByName;
   fbXmlSrvWriteByName  : FB_XmlSrvWriteByName;
   value                : ST_MyProductionStruct;
   state                : INT := 0;
   R_Edge               : R_TRIG;
   bStart               : BOOL;
   bError               : BOOL;
   nErrId               : UDINT;
END_VAR
```

Structure ST_MyProductionStruct

```
TYPE ST_MyProductionStruct :
STRUCT
   rLength     : REAL;
   rWidth      : REAL;
   rHeight     : REAL;
   iQuantity   : INT;
   iCounter    : INT;
   bReady      : BOOL;
   stInfo      : STRING;
END_STRUCT
END_TYPE
```

PLC program

```
(* The production data is read, the production is carried out and, according to the
 quantity and the production order, completed with an entry in the XML file.
 To start the program the XML file needs to be stored in the correspond-
ing folder (sFilePath) and the variable bStart needs to be set TRUE in the PLC program. *)
```

```
R_Edge (CLK := bStart);
IF R_Edge.Q THEN
   state := 1;
END_IF


CASE state OF
0: (* idle state *)
   ;

1: (* init state *)
fbXmlSrvReadByName( sNetId := '',
            sSymName := 'Sample7.value',
            sFilePath := 'C:\Production1.xml',
            sXPath := '/dataentry/MAIN.value',
            bExecute := TRUE,
            tTimeout := t#10s,
            bError => bError,
            nErrId => nErrId);
state := 2;

2:
fbXmlSrvReadByName(bExecute := FALSE);
IF NOT fbXmlSrvReadByName.bBusy AND NOT fbXmlSrvReadByName.bError THEN
   state := 3;
ELSIF fbXmlSrvReadByName.bError THEN
   state := 100;
END_IF

3: (* working state *)
IF value.bReady = TRUE THEN
   value.stInfo := 'The order was already processed!';
   (* replace your production XML file! *)
   state := 4;
   RETURN;
END_IF

(* call production program with
 new length, width and height here *)
 value.iCounter := value.iCounter + 1;
 IF value.iCounter = value.iQuantity THEN
    value.bReady := TRUE;
    state := 4;
END_IF

4: (* documentation state *)
fbXmlSrvWriteByName( sNetId := '',
             nMode := XMLSRV_SKIPMISSING,
             sSymName := 'Sample7.value',
             sFilePath := 'C:\Production1.xml',
             sXPath := '/dataentry/MAIN.value',
             bExecute := TRUE,
             tTimeout := t#10s,
             bError => bError,
             nErrId => nErrId);
 state := 5;

5:
fbXmlSrvWriteByName(bExecute := FALSE);
IF NOT fbXmlSrvWriteByName.bBusy AND NOT fbXmlSrvWriteByName.bError THEN
   state := 0;
ELSIF fbXmlSrvWriteByName.bError THEN
   state := 100;
END_IF

100:(* error state *)
   ;
```

XML file

```
<dataentry>
  <MAIN.value>
    <rLength>65.85</rLength>
    <rWidth>30</rWidth>
    <rHeight>2.5</rHeight>
    <iQuantity>500</iQuantity>
    <iCounter>0</iCounter>
    <bReady>false</bReady>
    <stInfo></stInfo>
  </MAIN.value>
</dataentry>
```

Requirements

Development environment	Target system type	PLC libraries to be linked
TwinCAT v3.1 Build 4011	PC or CX (x86, ARM)	Tc2_XmlDataSrv

6 Appendix

6.1 Overview of TC3 XML Server error codes

Offset + error code	Range	Description
0x00000000 + TwinCAT system error codes	0x00000000-0x00007800	TwinCAT system error (including ADS error codes)
0x00008000 + internal TC3 XML Server error [▸ 34]	0x00008000-0x000080FF	Internal TC3 XML Server error codes

6.2 Internal error codes of the TwinCAT 3 XML Server

Code	Description	Symbolic name
0x00008000	Internal error	XMLSRVERROR_INTERNAL
0x00008001	Handle not found.	XMLSRVERROR_NOTFOUND
0x00008002	Error during parsing of the XML file	XMLSRVERROR_PARSERERROR
0x00008003	Incompatible data type.	XMLSRVERROR_INCOMPATIBLE
0x00008004	Error during memory allocation	XMLSRVERROR_NOMEMORY
0x00008005	Error during adding of an XML node	XMLSRVERROR_ADDNODE
0x00008006	Invalid sXPath	XMLSRVERROR_INVALIDXPATH
0x00008007	Invalid string in TC	XMLSRVERROR_INVALIDSTRING
0x00008800	Invalid handle of the client	XMLSRVERROR_INVALIDCLIENTHANDLE
0x0008900	Use of a wrong TcXmlDataSrv.exe (for TC2 than TC3)	XMLSRVERROR_INVALIDTWINCATVERSION

6.3 FAQ - Frequently asked questions and their answers

In this area we answer frequently asked questions to help you to work with TF6421 XML Server.

If you have any further questions please contact our support +49(0)5246/963-157).

What kind of information is saved to the XML file? [▸ 34]

Is it possible to write several variables into an XML file simultaneously? [▸ 34]

Is there a way to access an XML file in the network? [▸ 35]

Does the XML-Server have a feature to automatically create files? [▸ 35]

What happens if the XML Server is not able to convert a variable, because the types do not match (e.g. an INT has the value “hello” in an XML file)? [▸ 35]

? What kind of information is saved to the XML file?
! The only information saved to the XML-file is the name of the variable and the value. There is no information about the data type saved. You can find more information on that in the section “Getting Started [▸ 25]”.

? Is it possible to write several variables into an XML file simultaneously?

! Yes, this is possible. Therefore you have to wrap up your variables in a structure. You then write a variable of this type. This is shown in the samples [▸ 25].

? Is there a way to access an XML file in the network?
! No, this is not possible. sPath may only point at the local filesystem.

? Does the XML-Server have a feature to automatically create files?
! Yes, it does. To use this feature you have to set nMode:=1 (XMLSRV_ADDMISSING). The XML Server will create files and parts in the file automatically. Folders will not be created!

? What happens if the XML Server is not able to convert a variable, because the types do not match (e.g. an INT has the value “hallo” in an XML file)?
! In this case this variable will just be skipped. It does not lead to an error.