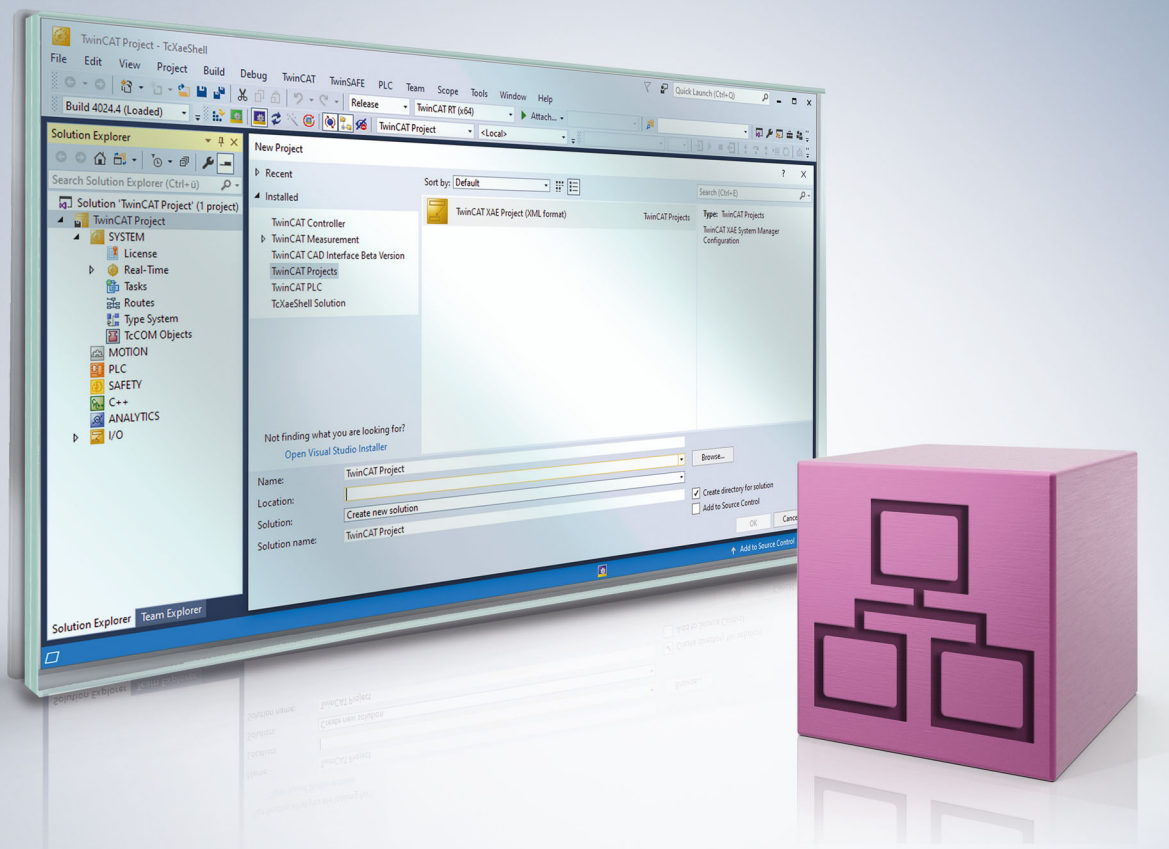
BECKHOFF New Automation Technology

Manual | EN

TF6421

TwinCAT 3 | XML Server



2022-11-28 | Version: 1.4.1

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 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®, TwinCAT/BSD®, TC/BSD®, EtherCAT®, EtherCAT G®, EtherCAT G10®, EtherCAT P®, Safety over EtherCAT®, TwinSAFE®, XFC®, XTS® and XPlanar® 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, EP1456722, EP2137893, DE102015105702
with corresponding applications or registrations in various other countries.

EtherCAT® is a 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.

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.

1.3 Notes on information security

The products of Beckhoff Automation GmbH & Co. KG (Beckhoff), insofar as they can be accessed online, are equipped with security functions that support the secure operation of plants, systems, machines and networks. Despite the security functions, the creation, implementation and constant updating of a holistic security concept for the operation are necessary to protect the respective plant, system, machine and networks against cyber threats. The products sold by Beckhoff are only part of the overall security concept. The customer is responsible for preventing unauthorized access by third parties to its equipment, systems, machines and networks. The latter should be connected to the corporate network or the Internet only if appropriate protective measures have been set up.

In addition, the recommendations from Beckhoff regarding appropriate protective measures should be observed. Further information regarding information security and industrial security can be found in our https://www.beckhoff.com/secguide.

Beckhoff products and solutions undergo continuous further development. This also applies to security functions. In light of this continuous further development, Beckhoff expressly recommends that the products are kept up to date at all times and that updates are installed for the products once they have been made available. Using outdated or unsupported product versions can increase the risk of cyber threats.

To stay informed about information security for Beckhoff products, subscribe to the RSS feed at https://www.beckhoff.com/secinfo.

2 Overview

The TwinCAT XML Server offers a PLC library with which write and read access to XML files can be implemented. The XML Server is characterized by its ease of use. It is particularly suitable, for example, for loading initialization data, as is often required when starting up a machine. It is also possible to save PLC variables formatted in an XML file including PLC comments. The structure of a variable in the XML document matches the structure of the variables in the PLC. This enables individual subelements of a variable to be accessed directly. Only those subelements (elements of a structure or an array) are transferred that are also defined in the XML file. When the PLC variables are written, missing elements can optionally be added.

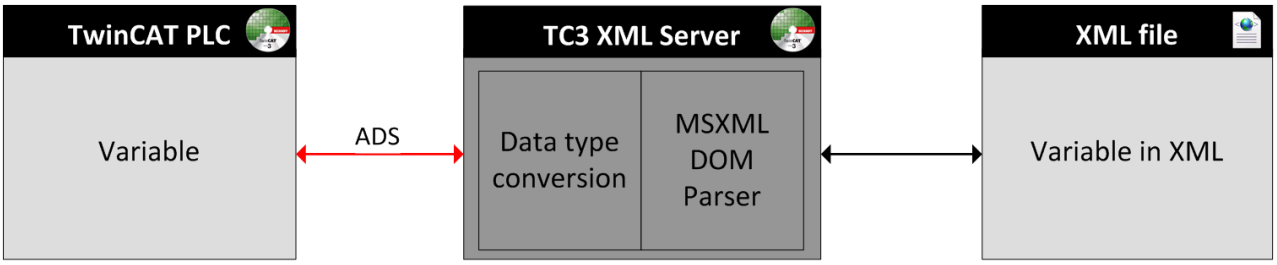
Components

- TwinCAT XML Server: is a service that is started and stopped together with TwinCAT. It is the link between TwinCAT and the XML file.
- PLC library [▶ 17]: the PLC library provides four different function blocks with which data from the PLC library can be written to and read from the XML file. Thus, variables can be stored formatted in an XML file and variables can be initialized in TwinCAT from an XML file.

Principle of operation

The XML Server communicates with the TwinCAT PLC via ADS. During a write operation, all variables to be written are converted to text and written to the XML file via the MSXML DOM parser. The XML file does not contain any information about the data types, but only the name of the variable as tag name and the value: <variable name> value </variable name>.

During a read operation, the data types for the variables to be read are determined via ADS and the texts from the XML file are converted accordingly.



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

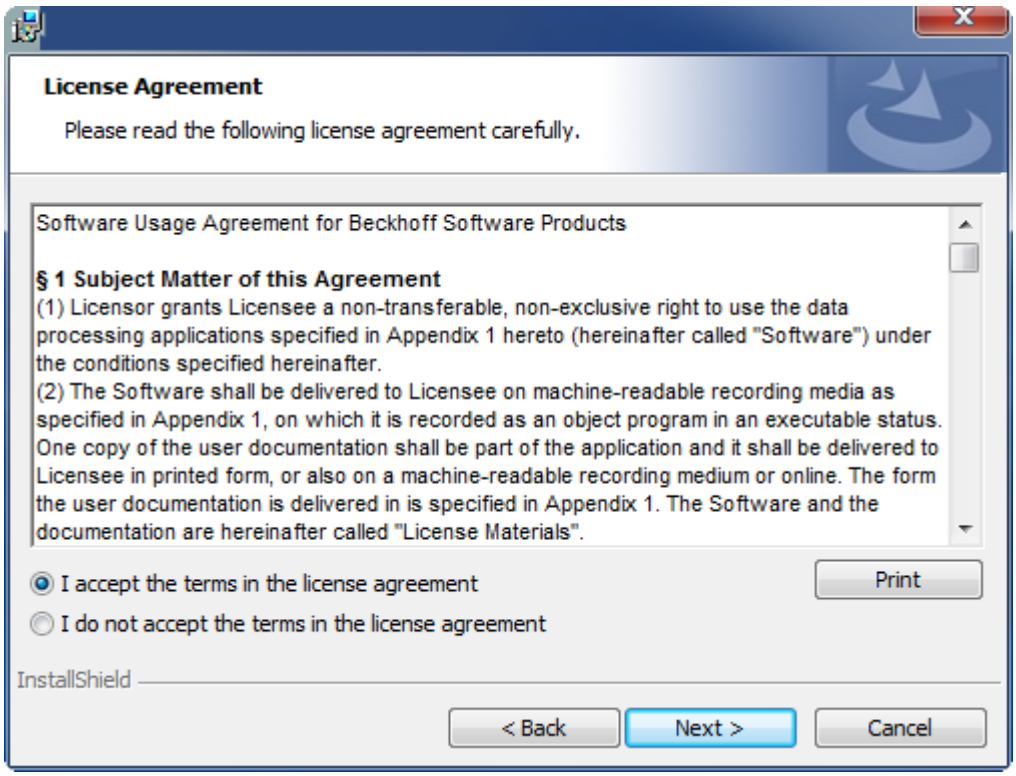
The following section describes how to install the TwinCAT 3 Function for Windows-based operating systems.

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

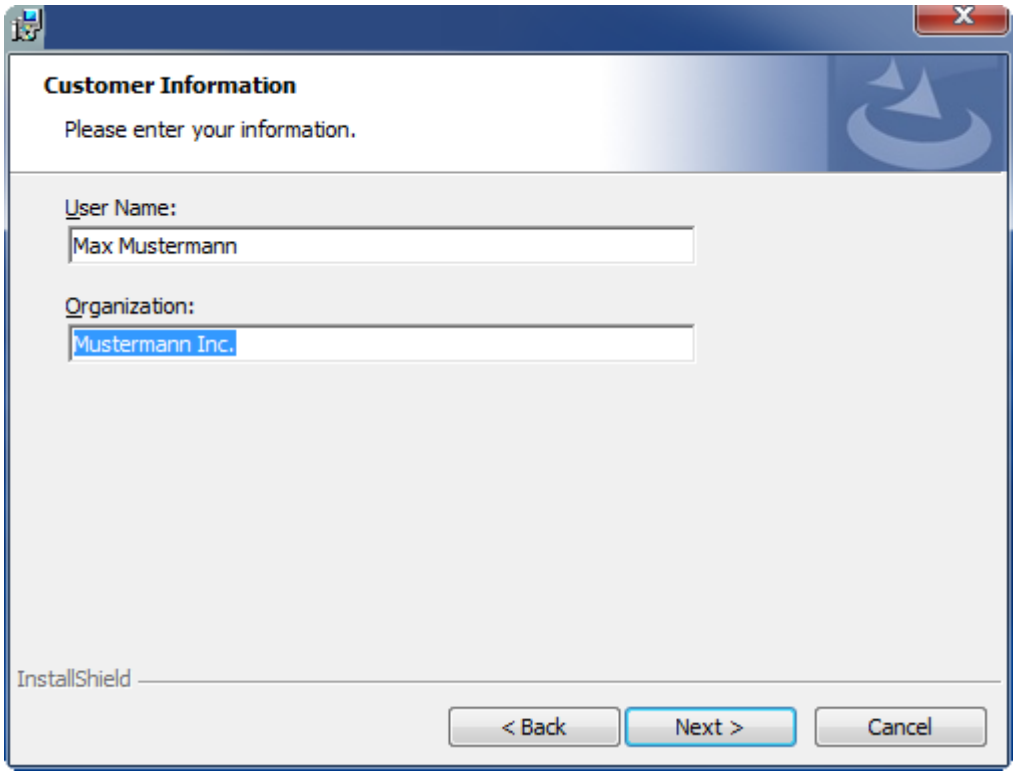
1. Run the setup file as administrator. To do this, select the command **Run as administrator** in the context menu of the file.

   ⇨ The installation dialog opens.

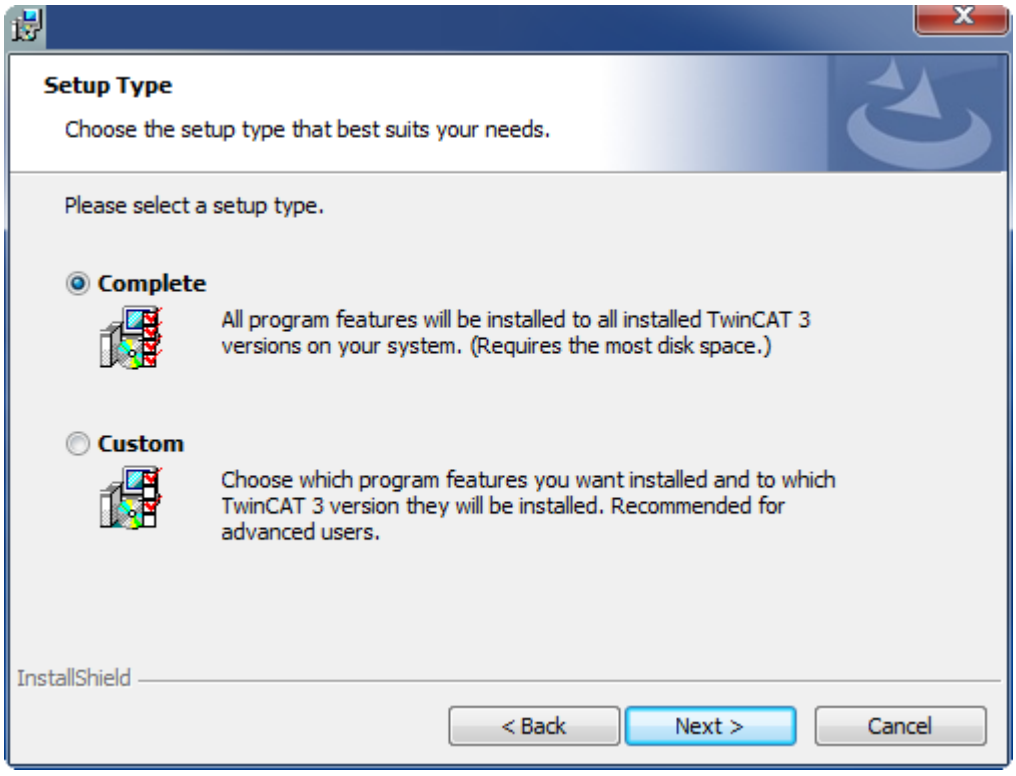
2. Accept the end user licensing agreement and click **Next**.



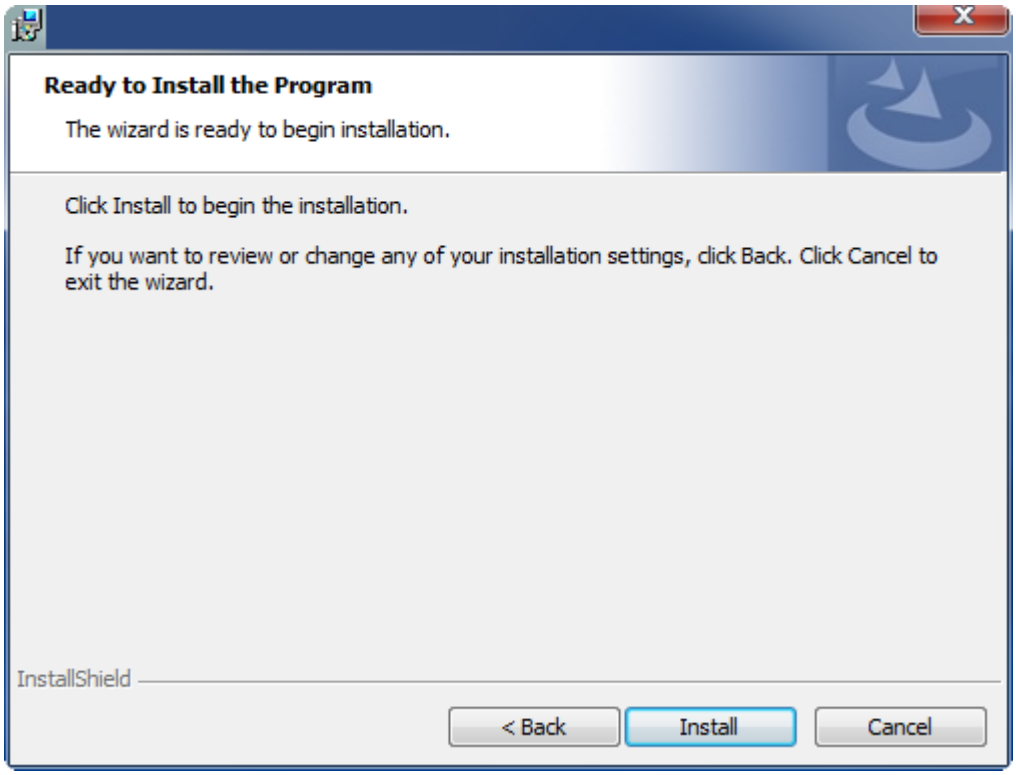
3. Enter your user data.



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**.

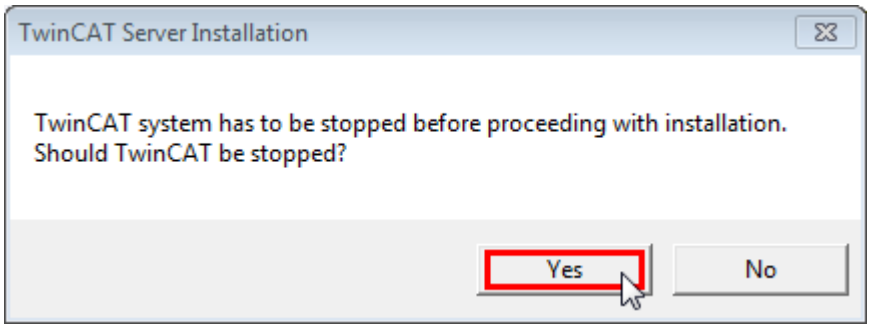


5. Select **Next**, then **Install** to start the installation.

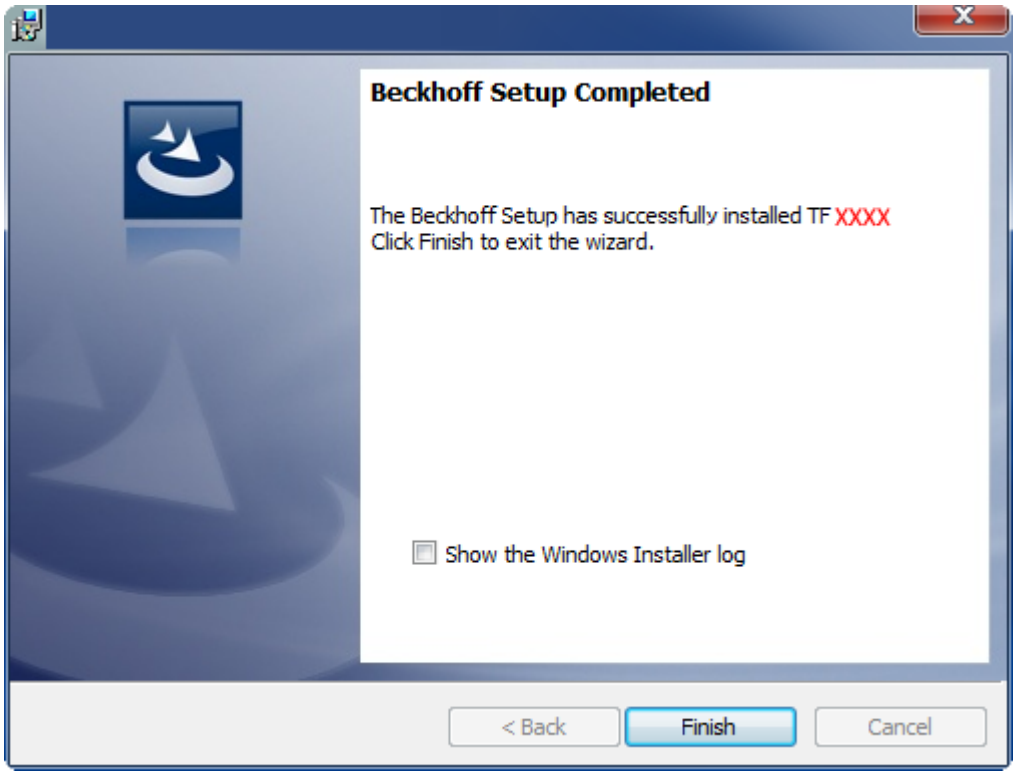


⇨ A dialog box informs you that the TwinCAT system must be stopped to proceed with the installation.

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



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



⇨ The TwinCAT 3 Function has been successfully installed and can be licensed (see Licensing [▶ 12]).

3.3 Licensing

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).

Licensing the full version of a TwinCAT 3 Function

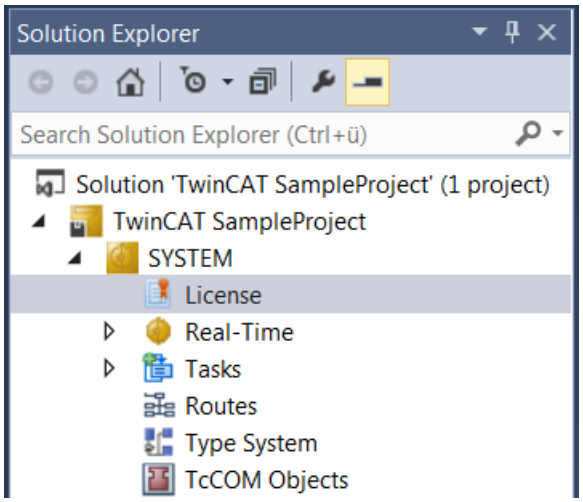
A description of the procedure to license a full version can be found in the Beckhoff Information System in the documentation "TwinCAT 3 Licensing".

Licensing the 7-day test version of a TwinCAT 3 Function

A 7-day test version cannot be enabled for a TwinCAT 3 license dongle.

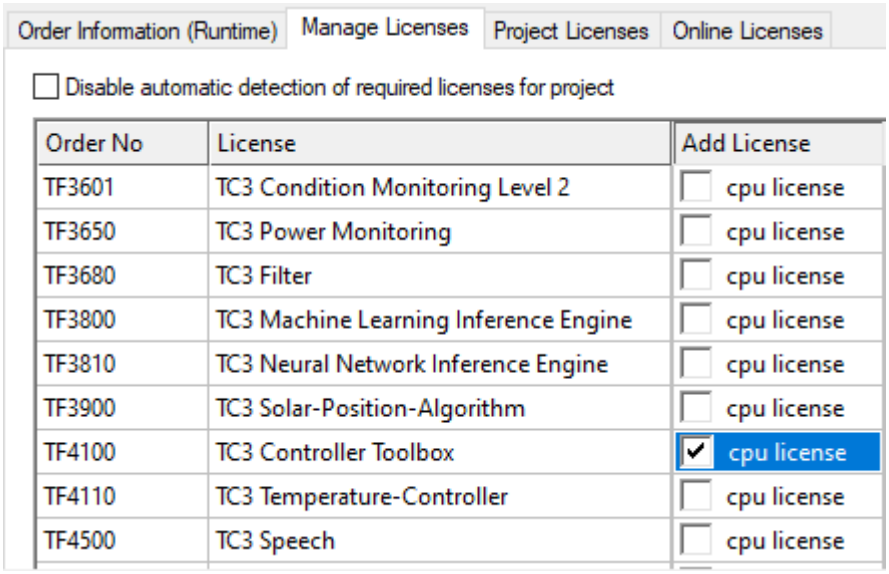
1. Start the TwinCAT 3 development environment (XAE).
2. Open an existing TwinCAT 3 project or create a new project.
3. If you want to activate the license for a remote device, set the desired target system. To do this, select the target system from the **Choose Target System** drop-down list in the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

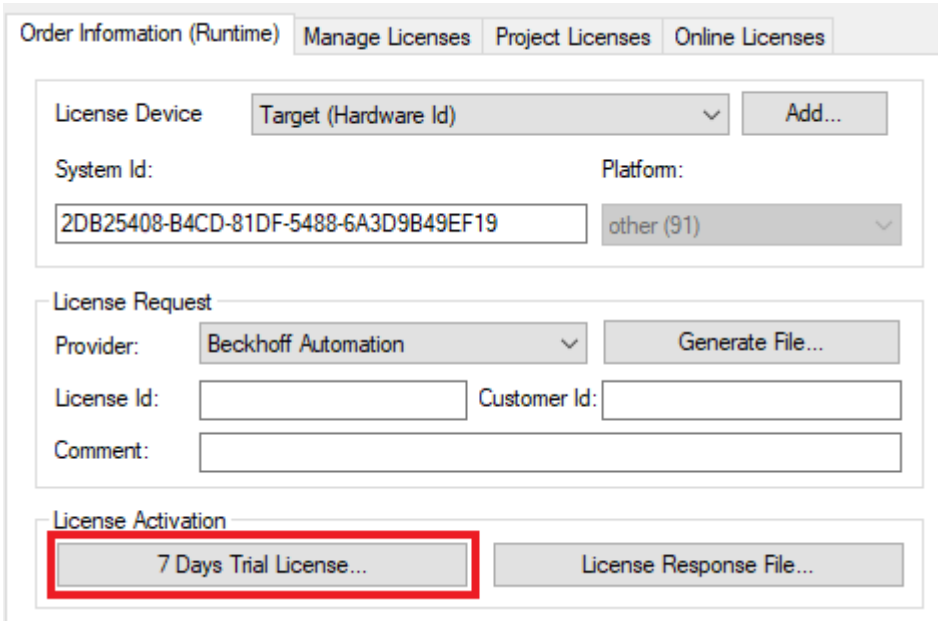
5. Open the **Manage Licenses** tab. In the **Add License** column, check the check box for the license you want to add to your project (e.g. "TF4100 TC3 Controller Toolbox").



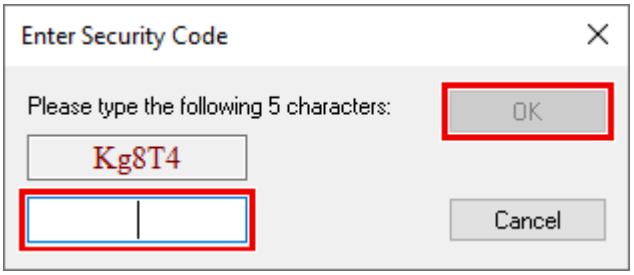
6. Open the **Order Information (Runtime)** tab.

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

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



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



8. Enter the code exactly as it is displayed and confirm the entry.
9. Confirm the subsequent dialog, which indicates the successful activation.
   ⇨ In the tabular overview of licenses, the license status now indicates the expiry date of the license.
10. Restart the TwinCAT system.

⇨ The 7-day trial version is enabled.

3.4 Installation Windows CE

The following section describes how to install a TwinCAT 3 function (TFxxx) on a Beckhoff Embedded PC with Windows CE.

1. Download and install the setup file [▶ 14]
2. Transfer the CAB file to the Windows CE device [▶ 15]
3. Run the CAB file on the Windows CE device [▶ 15]

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

- Software upgrade [▶ 15]

Download and install the setup file

The CAB installation file for Windows CE is part of the TFxxx 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).

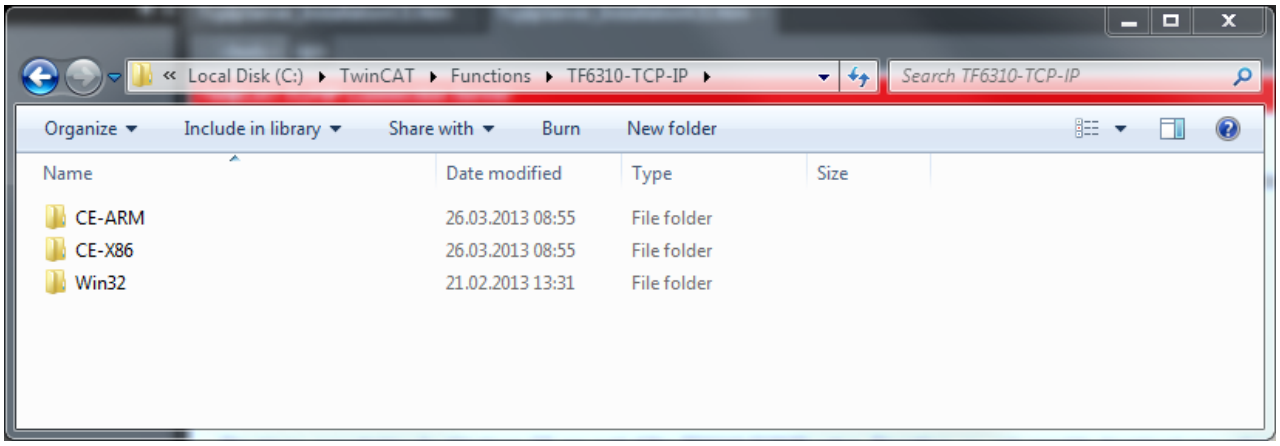
Download the TFxxx setup file and install the TwinCAT 3 function as described in the Installation [▶ 9] section.

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.

Example: "TF6310" installation folder



Transfer the CAB file to the Windows CE device

Transfer the corresponding CAB 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 of the TwinCAT 3 function (TFxxxx) are automatically loaded in the background and are then available.

The software is installed in the following directory on the Windows CE device:
\Hard Disk\TwinCAT\Functions\TFxxxx

Software upgrade

If an older version of the TwinCAT 3 function 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\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 Windows CE device.
6. Run the CAB file on the Windows CE device and install the new version.
7. Delete the file *Tc*.old*.
8. Restart the Windows CE device.

⇨ The new version is active after the restart.

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:

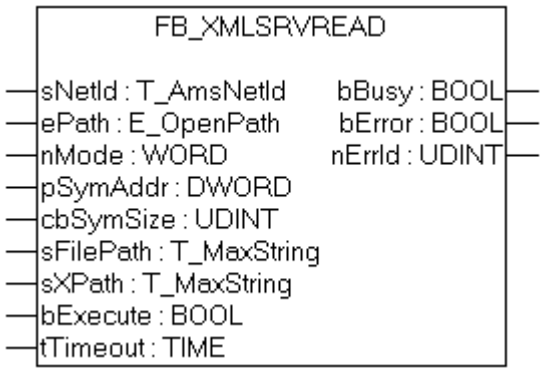
Data type	PLC example	XML example
UDINT, DINT, UINT, INT, USINT, SINT, DWORD, WORD, BYTE	value1 : DINT := -1; value2 : UDINT := 65535;	<dataentry> <MAIN.value1>-1</MAIN.value1> <MAIN.value2>65535</MAIN.value2> </dataentry>
LREAL, REAL	value1 : LREAL = 1.2;	<dataentry> <MAIN.value1>1.2</MAIN.value1> </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, x64, 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.

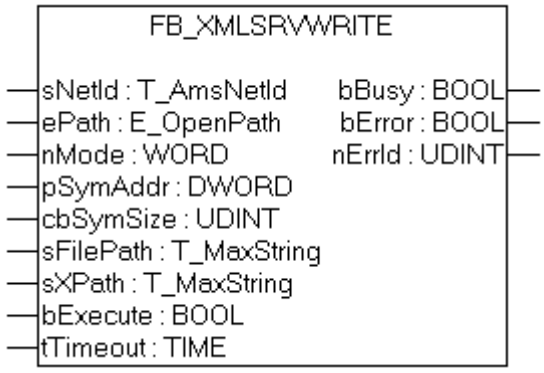
bError: This output is set up after the bBusy output has been reset if there has been an error in transmission of the command.

nErrId: Returns the TC3 XML server error number [▶ 33], 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, x64, 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 modes 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. From product version 3.2.31.0 it is possible to write the comments from the declaration section into the XML file with the parameters XMLSRV_SERIALIZESYMCOMMENT and XMLSRV_SERIALIZETYPECOMMENT.

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 of 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 function block is enabled via a positive edge at this input.

tTimeout: maximum time that must not be exceeded when the function block is executed.

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.

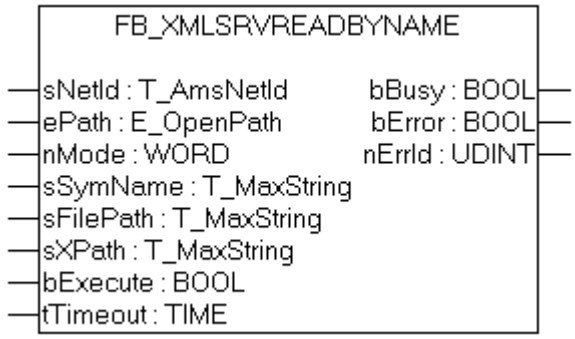
bError: This output is set up after the bBusy output has been reset if there has been an error in transmission of the command.

nErrId: Returns the TC3 XML server error number [▶ 33], 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, x64, 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.

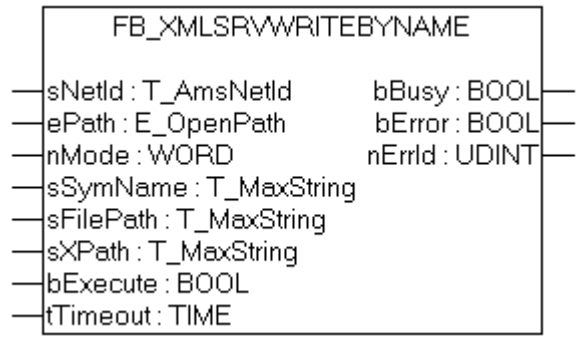
bError: This output is set up after the bBusy output has been reset if there has been an error in transmission of the command.

nErrId: Returns the TC3 XML server error number [▶ 33], 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, x64, 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. From product version 3.2.31.0 it is possible to write the comments from the declaration section into the XML file with the parameters XMLSRV_SERIALIZESYMCOMMENT and XMLSRV_SERIALIZETYPECOMMENT.

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

sFilePath: contains the path and file name of 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 function block is enabled via a positive edge at this input.

tTimeout: maximum time that must not be exceeded when the function block is executed.

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 [▶ 33], 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, x64, 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, x64, 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.

Comparing versions

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/1635013643/.zip

- Getting started [▶ 24]
  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 [▶ 25]
  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 [▶ 30]
  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   :='C:\Test.xml'; (*Pfad zur XML-Datei*)
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, x64, 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/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, x64, 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/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 fbXmlSrvWrite.bError T
HEN
      (* create filename *)
      sFilePathWrite:= CONCAT(sFileFolder, SYSTEMTIME_TO_STRING(stMyTimestruct)); (* set folder + ti
me *)
      sFilePathWrite:= DELETE(STR:= sFilePathWrite, LEN:= 4 , POS:= LEN(STR:=sFilePathWrite)-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, x64, 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/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 corresponding folder (sFilePath) and th
e 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, x64, ARM)	Tc2_XmlDataSrv

6 Appendix

6.1 Overview of TC3 XML Server error codes

Offset + error code	Range	Description
0x00000000 + TwinCAT system error codes [▶ 33]	0x00000000-0x00007800	TwinCAT system error (including ADS error codes)
0x00008000 + internal TC3 XML Server error [▶ 37]	0x00008000-0x000080FF	Internal TC3 XML Server error codes

6.2 ADS Return Codes

Grouping of error codes:
Global error codes: ADS Return Codes [▶ 33]... (0x9811_0000 ...)
Router error codes: ADS Return Codes [▶ 33]... (0x9811_0500 ...)
General ADS errors: ADS Return Codes [▶ 34]... (0x9811_0700 ...)
RTime error codes: ADS Return Codes [▶ 36]... (0x9811_1000 ...)

Global error codes

Hex	Dec	HRESULT	Name	Description
0x0	0	0x98110000	ERR_NOERROR	No error.
0x1	1	0x98110001	ERR_INTERNAL	Internal error.
0x2	2	0x98110002	ERR_NORTIME	No real time.
0x3	3	0x98110003	ERR_ALLOCLOCKEDMEM	Allocation locked – memory error.
0x4	4	0x98110004	ERR_INSERTMAILBOX	Mailbox full – the ADS message could not be sent. Reducing the number of ADS messages per cycle will help.
0x5	5	0x98110005	ERR_WRONGRECEIVEHMSG	Wrong HMSG.
0x6	6	0x98110006	ERR_TARGETPORTNOTFOUND	Target port not found – ADS server is not started or is not reachable.
0x7	7	0x98110007	ERR_TARGETMACHINENOTFOUND	Target computer not found – AMS route was not found.
0x8	8	0x98110008	ERR_UNKNOWNCMDID	Unknown command ID.
0x9	9	0x98110009	ERR_BADTASKID	Invalid task ID.
0xA	10	0x9811000A	ERR_NOIO	No IO.
0xB	11	0x9811000B	ERR_UNKNOWNAMSCMD	Unknown AMS command.
0xC	12	0x9811000C	ERR_WIN32ERROR	Win32 error.
0xD	13	0x9811000D	ERR_PORTNOTCONNECTED	Port not connected.
0xE	14	0x9811000E	ERR_INVALIDAMSLENGTH	Invalid AMS length.
0xF	15	0x9811000F	ERR_INVALIDAMSNETID	Invalid AMS Net ID.
0x10	16	0x98110010	ERR_LOWINSTLEVEL	Installation level is too low –TwinCAT 2 license error.
0x11	17	0x98110011	ERR_NODEBUGINTAVAILABLE	No debugging available.
0x12	18	0x98110012	ERR_PORTDISABLED	Port disabled – TwinCAT system service not started.
0x13	19	0x98110013	ERR_PORTALREADYCONNECTED	Port already connected.
0x14	20	0x98110014	ERR_AMSSYNC_W32ERROR	AMS Sync Win32 error.
0x15	21	0x98110015	ERR_AMSSYNC_TIMEOUT	AMS Sync Timeout.
0x16	22	0x98110016	ERR_AMSSYNC_AMSERROR	AMS Sync error.
0x17	23	0x98110017	ERR_AMSSYNC_NOINDEXINMAP	No index map for AMS Sync available.
0x18	24	0x98110018	ERR_INVALIDAMSPORT	Invalid AMS port.
0x19	25	0x98110019	ERR_NOMEMORY	No memory.
0x1A	26	0x9811001A	ERR_TCPSEND	TCP send error.
0x1B	27	0x9811001B	ERR_HOSTUNREACHABLE	Host unreachable.
0x1C	28	0x9811001C	ERR_INVALIDAMSFRAGMENT	Invalid AMS fragment.
0x1D	29	0x9811001D	ERR_TLSSEND	TLS send error – secure ADS connection failed.
0x1E	30	0x9811001E	ERR_ACCESSDENIED	Access denied – secure ADS access denied.

Router error codes

Hex	Dec	HRESULT	Name	Description
0x500	1280	0x98110500	ROUTERERR_NOLOCKEDMEMORY	Locked memory cannot be allocated.
0x501	1281	0x98110501	ROUTERERR_RESIZEMEMORY	The router memory size could not be changed.
0x502	1282	0x98110502	ROUTERERR_MAILBOXFULL	The mailbox has reached the maximum number of possible messages.
0x503	1283	0x98110503	ROUTERERR_DEBUGBOXFULL	The Debug mailbox has reached the maximum number of possible messages.
0x504	1284	0x98110504	ROUTERERR_UNKNOWNPORTTYPE	The port type is unknown.
0x505	1285	0x98110505	ROUTERERR_NOTINITIALIZED	The router is not initialized.
0x506	1286	0x98110506	ROUTERERR_PORTALREADYINUSE	The port number is already assigned.
0x507	1287	0x98110507	ROUTERERR_NOTREGISTERED	The port is not registered.
0x508	1288	0x98110508	ROUTERERR_NOMOREQUEUES	The maximum number of ports has been reached.
0x509	1289	0x98110509	ROUTERERR_INVALIDPORT	The port is invalid.
0x50A	1290	0x9811050A	ROUTERERR_NOTACTIVATED	The router is not active.
0x50B	1291	0x9811050B	ROUTERERR_FRAGMENTBOXFULL	The mailbox has reached the maximum number for fragmented messages.
0x50C	1292	0x9811050C	ROUTERERR_FRAGMENTTIMEOUT	A fragment timeout has occurred.
0x50D	1293	0x9811050D	ROUTERERR_TOBEREMOVED	The port is removed.

General ADS error codes

Hex	Dec	HRESULT	Name	Description
0x700	1792	0x98110700	ADSERR_DEVICE_ERROR	General device error.
0x701	1793	0x98110701	ADSERR_DEVICE_SRVNOTSUPP	Service is not supported by the server.
0x702	1794	0x98110702	ADSERR_DEVICE_INVALIDGRP	Invalid index group.
0x703	1795	0x98110703	ADSERR_DEVICE_INVALIDOFFSET	Invalid index offset.
0x704	1796	0x98110704	ADSERR_DEVICE_INVALIDACCESS	Reading or writing not permitted.
0x705	1797	0x98110705	ADSERR_DEVICE_INVALIDSIZE	Parameter size not correct.
0x706	1798	0x98110706	ADSERR_DEVICE_INVALIDDATA	Invalid data values.
0x707	1799	0x98110707	ADSERR_DEVICE_NOTREADY	Device is not ready to operate.
0x708	1800	0x98110708	ADSERR_DEVICE_BUSY	Device is busy.
0x709	1801	0x98110709	ADSERR_DEVICE_INVALIDCONTEXT	Invalid operating system context. This can result from use of ADS blocks in different tasks. It may be possible to resolve this through multitasking synchronization in the PLC.
0x70A	1802	0x9811070A	ADSERR_DEVICE_NOMEMORY	Insufficient memory.
0x70B	1803	0x9811070B	ADSERR_DEVICE_INVALIDPARM	Invalid parameter values.
0x70C	1804	0x9811070C	ADSERR_DEVICE_NOTFOUND	Not found (files, ...).
0x70D	1805	0x9811070D	ADSERR_DEVICE_SYNTAX	Syntax error in file or command.
0x70E	1806	0x9811070E	ADSERR_DEVICE_INCOMPATIBLE	Objects do not match.
0x70F	1807	0x9811070F	ADSERR_DEVICE_EXISTS	Object already exists.
0x710	1808	0x98110710	ADSERR_DEVICE_SYMBOLNOTFOUND	Symbol not found.
0x711	1809	0x98110711	ADSERR_DEVICE_SYMBOLVERSIONINVALID	Invalid symbol version. This can occur due to an online change. Create a new handle.
0x712	1810	0x98110712	ADSERR_DEVICE_INVALIDSTATE	Device (server) is in invalid state.
0x713	1811	0x98110713	ADSERR_DEVICE_TRANSMODENOTSUPP	AdsTransMode not supported.
0x714	1812	0x98110714	ADSERR_DEVICE_NOTIFYHNDINVALID	Notification handle is invalid.
0x715	1813	0x98110715	ADSERR_DEVICE_CLIENTUNKNOWN	Notification client not registered.
0x716	1814	0x98110716	ADSERR_DEVICE_NOMOREHDLS	No further handle available.
0x717	1815	0x98110717	ADSERR_DEVICE_INVALIDWATCHSIZE	Notification size too large.
0x718	1816	0x98110718	ADSERR_DEVICE_NOTINIT	Device not initialized.
0x719	1817	0x98110719	ADSERR_DEVICE_TIMEOUT	Device has a timeout.
0x71A	1818	0x9811071A	ADSERR_DEVICE_NOINTERFACE	Interface query failed.
0x71B	1819	0x9811071B	ADSERR_DEVICE_INVALIDINTERFACE	Wrong interface requested.
0x71C	1820	0x9811071C	ADSERR_DEVICE_INVALIDCLSID	Class ID is invalid.
0x71D	1821	0x9811071D	ADSERR_DEVICE_INVALIDOBJID	Object ID is invalid.
0x71E	1822	0x9811071E	ADSERR_DEVICE_PENDING	Request pending.
0x71F	1823	0x9811071F	ADSERR_DEVICE_ABORTED	Request is aborted.
0x720	1824	0x98110720	ADSERR_DEVICE_WARNING	Signal warning.
0x721	1825	0x98110721	ADSERR_DEVICE_INVALIDARRAYIDX	Invalid array index.
0x722	1826	0x98110722	ADSERR_DEVICE_SYMBOLNOTACTIVE	Symbol not active.
0x723	1827	0x98110723	ADSERR_DEVICE_ACCESSDENIED	Access denied.
0x724	1828	0x98110724	ADSERR_DEVICE_LICENSENOTFOUND	Missing license.
0x725	1829	0x98110725	ADSERR_DEVICE_LICENSEEXPIRED	License expired.
0x726	1830	0x98110726	ADSERR_DEVICE_LICENSEEXCEEDED	License exceeded.
0x727	1831	0x98110727	ADSERR_DEVICE_LICENSEINVALID	Invalid license.
0x728	1832	0x98110728	ADSERR_DEVICE_LICENSESYSTEMID	License problem: System ID is invalid.
0x729	1833	0x98110729	ADSERR_DEVICE_LICENSENOTIMELIMIT	License not limited in time.
0x72A	1834	0x9811072A	ADSERR_DEVICE_LICENSEFUTUREISSUE	Licensing problem: time in the future.
0x72B	1835	0x9811072B	ADSERR_DEVICE_LICENSETIMETOLONG	License period too long.
0x72C	1836	0x9811072C	ADSERR_DEVICE_EXCEPTION	Exception at system startup.
0x72D	1837	0x9811072D	ADSERR_DEVICE_LICENSEDUPLICATED	License file read twice.
0x72E	1838	0x9811072E	ADSERR_DEVICE_SIGNATUREINVALID	Invalid signature.
0x72F	1839	0x9811072F	ADSERR_DEVICE_CERTIFICATEINVALID	Invalid certificate.
0x730	1840	0x98110730	ADSERR_DEVICE_LICENSEOEMNOTFOUND	Public key not known from OEM.
0x731	1841	0x98110731	ADSERR_DEVICE_LICENSERESTRICTED	License not valid for this system ID.
0x732	1842	0x98110732	ADSERR_DEVICE_LICENSEDEMODENIED	Demo license prohibited.
0x733	1843	0x98110733	ADSERR_DEVICE_INVALIDFNCID	Invalid function ID.
0x734	1844	0x98110734	ADSERR_DEVICE_OUTOFRANGE	Outside the valid range.
0x735	1845	0x98110735	ADSERR_DEVICE_INVALIDALIGNMENT	Invalid alignment.
0x736	1846	0x98110736	ADSERR_DEVICE_LICENSEPLATFORM	Invalid platform level.

Hex	Dec	HRESULT	Name	Description
0x737	1847	0x98110737	ADSERR_DEVICE_FORWARD_PL	Context – forward to passive level.
0x738	1848	0x98110738	ADSERR_DEVICE_FORWARD_DL	Context – forward to dispatch level.
0x739	1849	0x98110739	ADSERR_DEVICE_FORWARD_RT	Context – forward to real time.
0x740	1856	0x98110740	ADSERR_CLIENT_ERROR	Client error.
0x741	1857	0x98110741	ADSERR_CLIENT_INVALIDPARM	Service contains an invalid parameter.
0x742	1858	0x98110742	ADSERR_CLIENT_LISTEMPTY	Polling list is empty.
0x743	1859	0x98110743	ADSERR_CLIENT_VARUSED	Var connection already in use.
0x744	1860	0x98110744	ADSERR_CLIENT_DUPLINVOKEID	The called ID is already in use.
0x745	1861	0x98110745	ADSERR_CLIENT_SYNCTIMEOUT	Timeout has occurred – the remote terminal is not responding in the specified ADS timeout. The route setting of the remote terminal may be configured incorrectly.
0x746	1862	0x98110746	ADSERR_CLIENT_W32ERROR	Error in Win32 subsystem.
0x747	1863	0x98110747	ADSERR_CLIENT_TIMEOUTINVALID	Invalid client timeout value.
0x748	1864	0x98110748	ADSERR_CLIENT_PORTNOTOPEN	Port not open.
0x749	1865	0x98110749	ADSERR_CLIENT_NOAMSADDR	No AMS address.
0x750	1872	0x98110750	ADSERR_CLIENT_SYNCINTERNAL	Internal error in Ads sync.
0x751	1873	0x98110751	ADSERR_CLIENT_ADDHASH	Hash table overflow.
0x752	1874	0x98110752	ADSERR_CLIENT_REMOVEHASH	Key not found in the table.
0x753	1875	0x98110753	ADSERR_CLIENT_NOMORESYM	No symbols in the cache.
0x754	1876	0x98110754	ADSERR_CLIENT_SYNCRESINVALID	Invalid response received.
0x755	1877	0x98110755	ADSERR_CLIENT_SYNCPORTLOCKED	Sync Port is locked.
0x756	1878	0x98110756	ADSERR_CLIENT_REQUESTCANCELLED	The request was cancelled.

RTime error codes

Hex	Dec	HRESULT	Name	Description
0x1000	4096	0x98111000	RTERR_INTERNAL	Internal error in the real-time system.
0x1001	4097	0x98111001	RTERR_BADTIMERPERIODS	Timer value is not valid.
0x1002	4098	0x98111002	RTERR_INVALIDTASKPTR	Task pointer has the invalid value 0 (zero).
0x1003	4099	0x98111003	RTERR_INVALIDSTACKPTR	Stack pointer has the invalid value 0 (zero).
0x1004	4100	0x98111004	RTERR_PRIOEXISTS	The request task priority is already assigned.
0x1005	4101	0x98111005	RTERR_NOMORETCB	No free TCB (Task Control Block) available. The maximum number of TCBs is 64.
0x1006	4102	0x98111006	RTERR_NOMORESEMAS	No free semaphores available. The maximum number of semaphores is 64.
0x1007	4103	0x98111007	RTERR_NOMOREQUEUES	No free space available in the queue. The maximum number of positions in the queue is 64.
0x100D	4109	0x9811100D	RTERR_EXTIRQALREADYDEF	An external synchronization interrupt is already applied.
0x100E	4110	0x9811100E	RTERR_EXTIRQNOTDEF	No external sync interrupt applied.
0x100F	4111	0x9811100F	RTERR_EXTIRQINSTALLFAILED	Application of the external synchronization interrupt has failed.
0x1010	4112	0x98111010	RTERR_IRQLNOTLESSOREQUAL	Call of a service function in the wrong context
0x1017	4119	0x98111017	RTERR_VMXNOTSUPPORTED	Intel VT-x extension is not supported.
0x1018	4120	0x98111018	RTERR_VMXDISABLED	Intel VT-x extension is not enabled in the BIOS.
0x1019	4121	0x98111019	RTERR_VMXCONTROLSMISSING	Missing function in Intel VT-x extension.
0x101A	4122	0x9811101A	RTERR_VMXENABLEFAILS	Activation of Intel VT-x fails.

Specific positive HRESULT Return Codes:

HRESULT	Name	Description
0x0000_0000	S_OK	No error.
0x0000_0001	S_FALSE	No error. Example: successful processing, but with a negative or incomplete result.
0x0000_0203	S_PENDING	No error. Example: successful processing, but no result is available yet.
0x0000_0256	S_WATCHDOG_TIMEOUT	No error. Example: successful processing, but a timeout occurred.

TCP Winsock error codes

Hex	Dec	Name	Description
0x274C	10060	WSAETIMEDOUT	A connection timeout has occurred - error while establishing the connection, because the remote terminal did not respond properly after a certain period of time, or the established connection could not be maintained because the connected host did not respond.
0x274D	10061	WSAECONNREFUSED	Connection refused - no connection could be established because the target computer has explicitly rejected it. This error usually results from an attempt to connect to a service that is inactive on the external host, that is, a service for which no server application is running.
0x2751	10065	WSAEHOSTUNREACH	No route to host - a socket operation referred to an unavailable host.
More Winsock error codes: Win32 error codes			

6.3 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

6.4 FAQ - Frequently asked questions and their answers

In this section frequently asked questions are answered, in order to facilitate your work with the XML Server. If you have any further questions, please contact our support +49(0)5246/963-157)

What information is stored in the XML file? [▶ 37]

Can multiple variables be written to an XML file at the same time? [▶ 37]

Is it possible to access an XML file on the network? [▶ 37]

Can the XML Server automatically generate XML files? [▶ 38]

What happens if the XML server cannot convert a variable (e.g. if an integer variable in the XML file has the value "hello")? [▶ 38]

Are alias data types supported? [▶ 38]

? What information is stored in the XML file?
! Only the name of the variable and its value are stored in the XML file - not its data type. In addition, as of product version 3.2.31.0, a PLC comment can also be written to the XML file. For more information see: "Getting Started [▶ 24]".

? 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 [▶ 24].

? 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.

? Are alias data types supported?

! Yes, alias data types are supported starting from product version 3.2.31.0.

More Information:
www.beckhoff.com/tf6421

Beckhoff Automation GmbH & Co. KG
Hülshorstweg 20
33415 Verl
Germany
Phone: +49 5246 9630
info@beckhoff.com
www.beckhoff.com