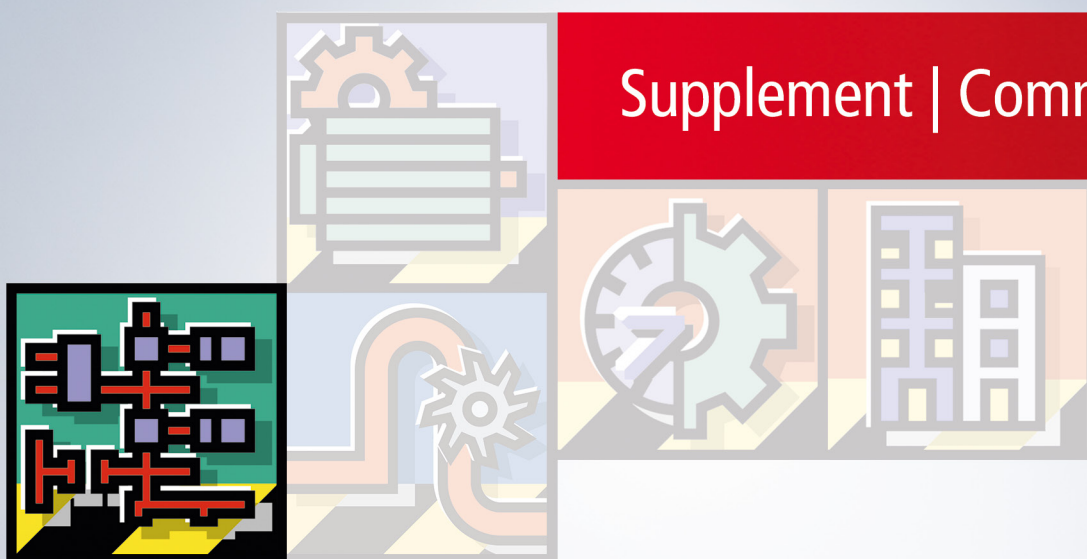


Manual | EN

# TS6421

TwinCAT 2 | XML Data Server





# Table of contents

<b>1 Foreword</b> .....	<b>5</b>
1.1 Notes on the documentation .....	5
1.2 Safety instructions .....	6
1.3 Notes on information security.....	7
<b>2 Overview</b> .....	<b>8</b>
<b>3 System Requirements</b> .....	<b>9</b>
<b>4 Installation</b> .....	<b>10</b>
<b>5 PLC API</b> .....	<b>13</b>
5.1 Function blocks .....	13
5.1.1 FB_XmlSrvRead .....	13
5.1.2 FB_XmlSrvWrite.....	14
5.1.3 FB_XmlSrvReadByName.....	16
5.1.4 FB_XmlSrvWriteByName.....	17
5.2 Functions.....	18
5.2.1 F_GetVersionTcXmlDataSrv .....	18
5.3 Global constants .....	19
5.3.1 Global Variables.....	19
<b>6 Samples</b> .....	<b>20</b>
6.1 Getting Started .....	20
6.2 Function blocks .....	22
6.3 Further Samples.....	24
6.4 Sample 7 (Production Sample) .....	26
<b>7 Appendix</b> .....	<b>29</b>
7.1 TwinCAT XML Data Server error codes.....	29
7.2 Internal error codes of the TwinCAT XML Data Server.....	29



# 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

© Beckhoff Automation GmbH & Co. KG, Germany.

The reproduction, distribution and utilization of this document as well as the communication of its contents to others without express authorization are prohibited.

Offenders will be held liable for the payment of damages. All rights reserved in the event of the grant of a patent, utility model or design.

## 1.2 Safety instructions

### Safety regulations

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

### Exclusion of liability

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

### Personnel qualification

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

### Description of symbols

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

#### **DANGER**

##### **Serious risk of injury!**

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

#### **WARNING**

##### **Risk of injury!**

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

#### **CAUTION**

##### **Personal injuries!**

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

#### **NOTE**

##### **Damage to the environment or devices**

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



##### **Tip or pointer**

This symbol indicates information that contributes to better understanding.

## 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.

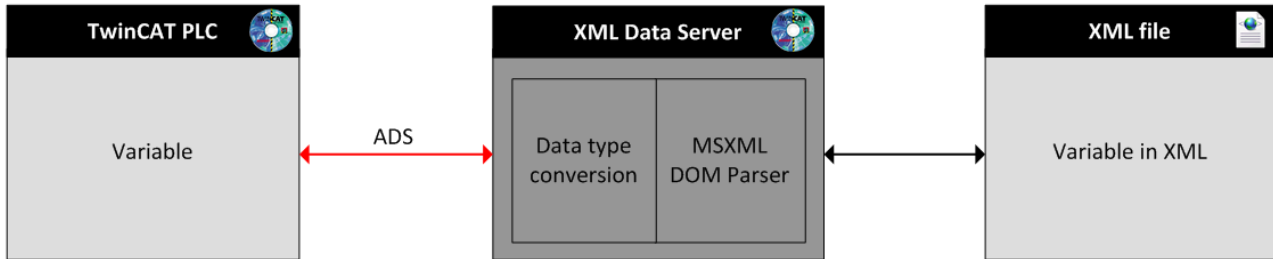
In addition, the recommendations from Beckhoff regarding appropriate protective measures should be observed. Further information regarding information security and industrial security can be found in our <https://www.beckhoff.com/secguide>.

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 Data Server enables initialising of TwinCAT PLC variables with data stored in an XML file, or formatted saving of PLC variables in an XML file. 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.





### 3 System Requirements

Technical Data	TS6421 XML Data Server
Target System	Windows NT/2000/XP/Vista/7 PC (x86-kompatibel)
Min. TwinCAT Version	2.10.0
Min. TwinCAT Level	TwinCAT PLC

## 4 Installation

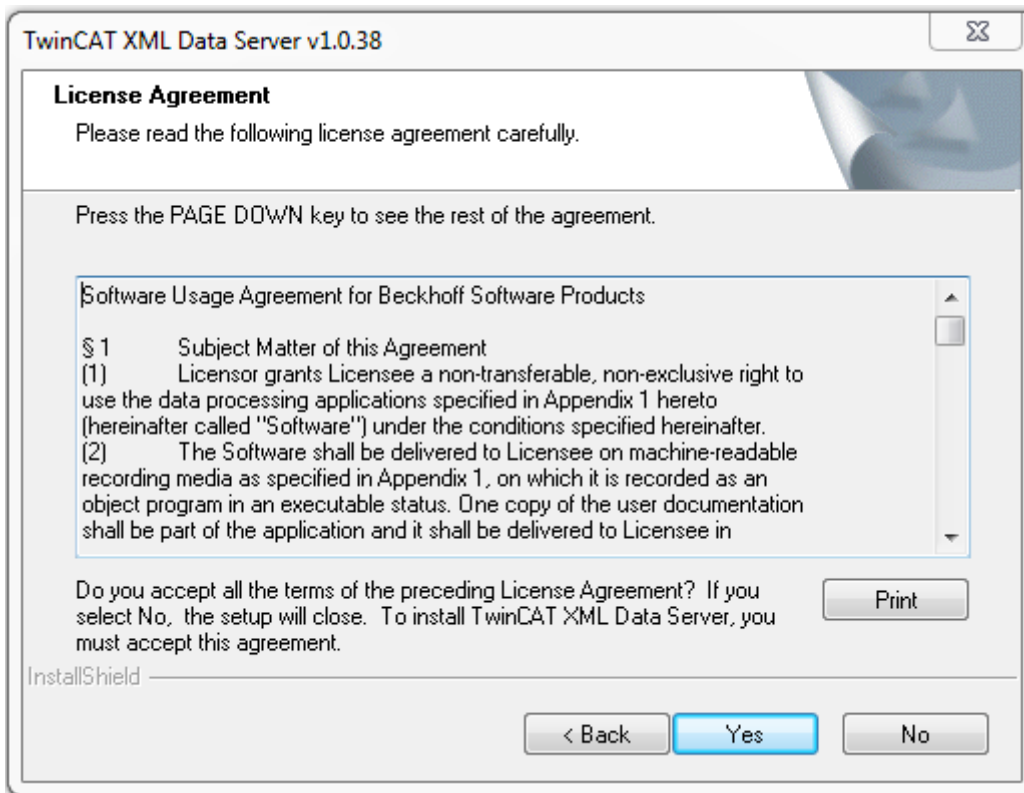
This part of the documentation gives a step-by-step explanation of the TwinCAT 3 XML Data Server setup process for Windows based operating systems. The following topics are part of this document:

- Starting the installation
- After the installation

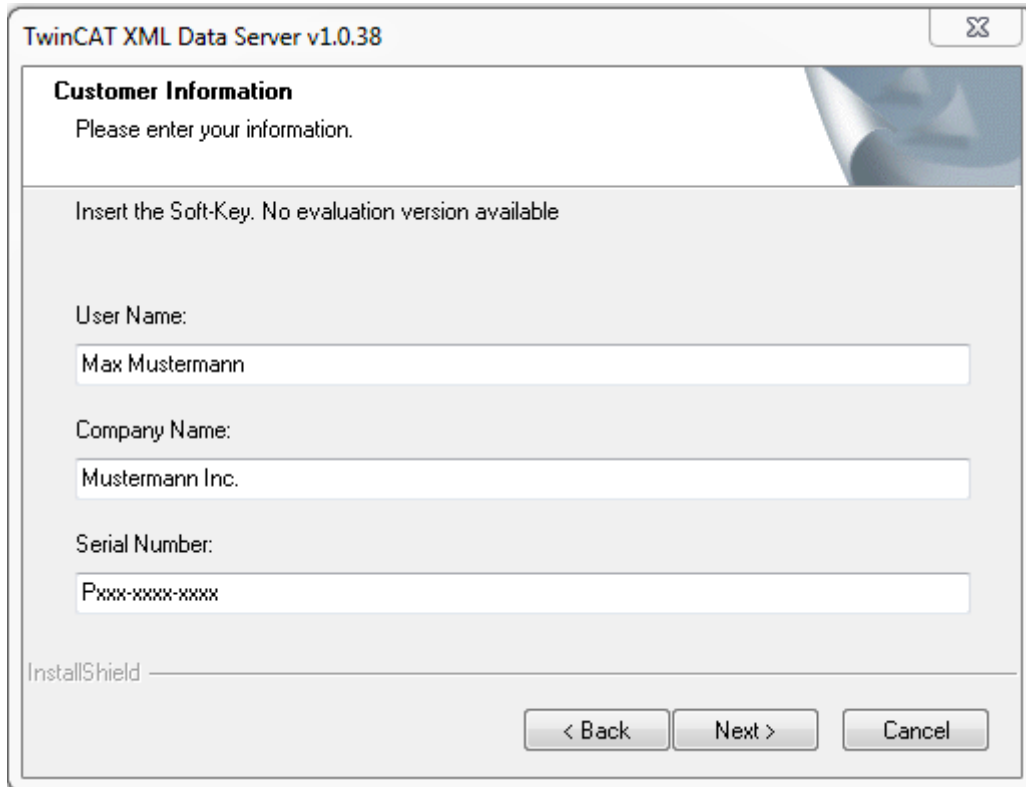
### Starting the installation

To install the Supplement, please perform the following steps:

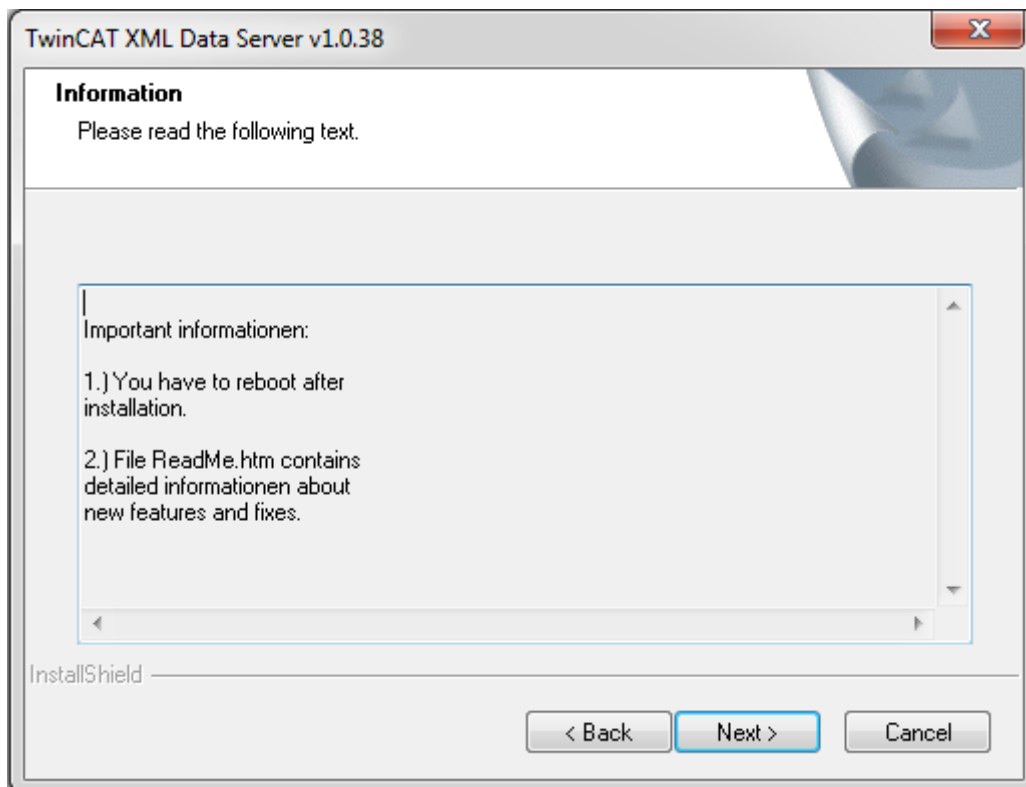
- Double-click the downloaded setup file "**TcXmlDataSrv.exe**".  
**Under Windows 32-bit/64-bit, please start the installation with "Run as Administrator" by right-clicking the setup file and selecting the corresponding option in the context menu.**
- Click on "**Next**" and accept the license agreement.



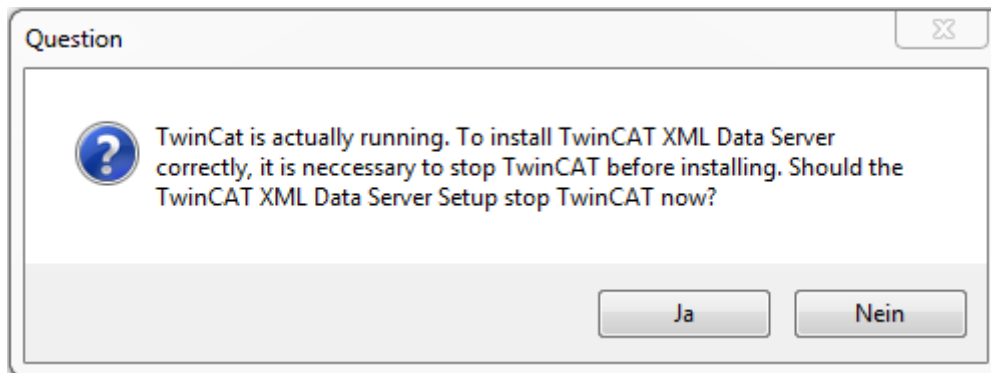
- Enter your user information:



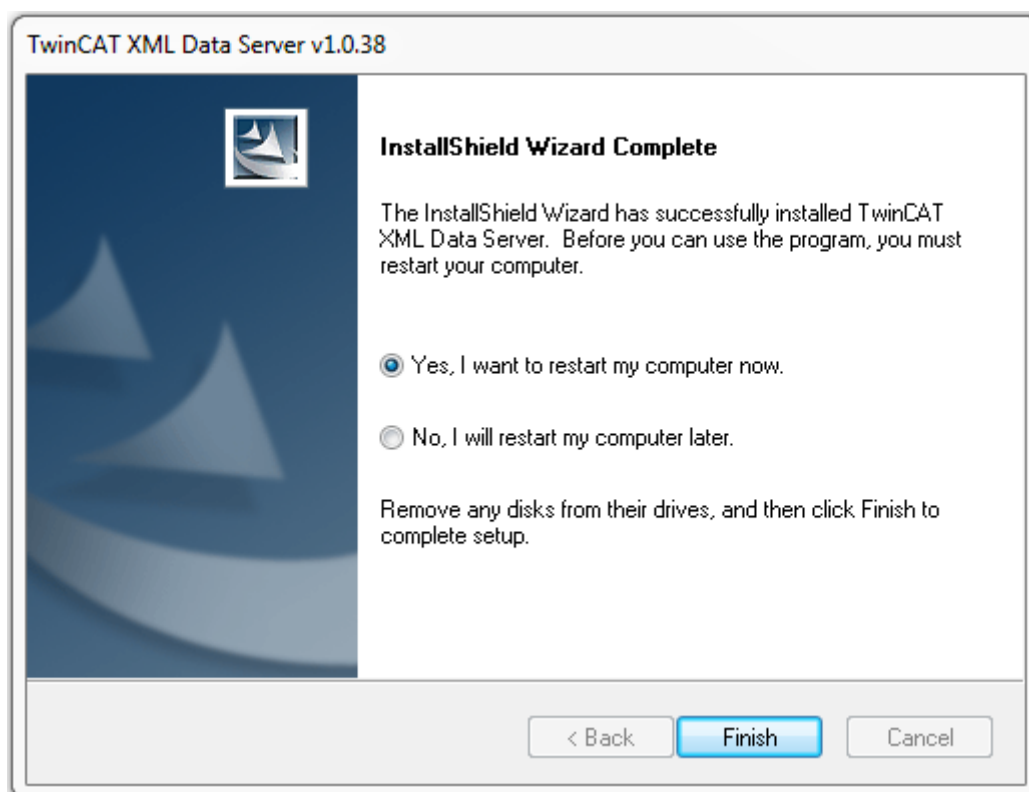
- Click on "Next" to start the installation



- TwinCAT system must be stopped before proceeding with installation. Select **"Yes"** to stop TwinCAT system service



- Select **"Finish"** to end the setup process



### After the installation

After restarting your computer, the supplement "TwinCAT XML Data Server" is ready for use.

## 5 PLC API

The PLC library TcXmlDataSrv.Lib is supplied with the TwinCAT XML Data Server and copied into folder ... \TwinCAT\PLC\Lib during installation.

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

- FB\_XmlSrvRead
- FB\_XmlSrvReadByName

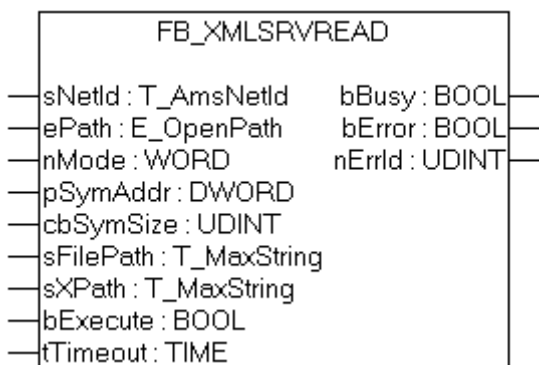
and two function blocks for writing PLC variables to the XML file:

- 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, although it is only supported from Build 1235 of TwinCAT version 2.10. For older TwinCAT versions only the function blocks FB\_XMLSrvReadByName and FB\_XMLSrvWriteByName can be used. In addition, the path of the XML file and the location of the variables within the XML document must be transferred as input parameter to the function blocks in XPath format.

### 5.1 Function blocks

#### 5.1.1 FB\_XmlSrvRead



The function block FB\_XmlSrvRead can be used to initialize a PLC variable with data from an XML file. The input variable sXPath must point to a valid node in the XML file specified via sFilePath. The symbol to be initialized 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 XML Data 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 influence how the XML file is analyzed. For the XmlSrvRead command, only XMLSRV\_SKIPMISSING mode is currently supported.

**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 rising 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.

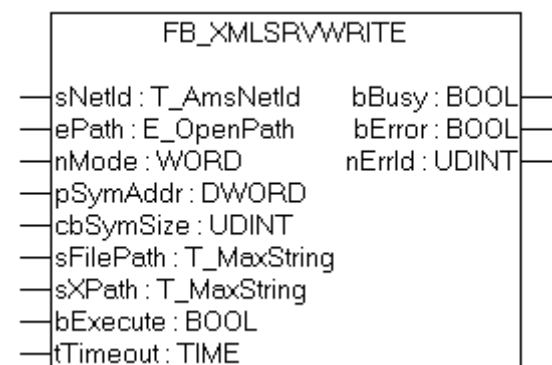
**bError:** This output is set up after the bBusy output has been reset if there has been an error in transmission of the command.

**nErrId :** If the bError output is set, this parameter returns the [TwinCAT XML Data Server error number](#) [▶ 29].

**Requirements**

Development environment	Target system type	PLC libraries to be linked
TwinCAT v2.10.0 from Build 1235	PC or CX (x86)	TcXmlDataSrv.Lib ( Standard.Lib; TcBase.Lib; TcSystem.Lib are included automatically )

**5.1.2 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 XML Data 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 specify which data are written to the XML file. XMLSRV\_SKIPMISSING and XMLSRV\_ADDMISSING mode are available for the XmlSrvWrite command. In XMLSRV\_SKIPMISSING mode, only those sub elements of a PLC symbol that already exist in the XML file are written to the XML file. In XMLSRV\_ADDMISSING mode, missing sub elements 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 rising 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.

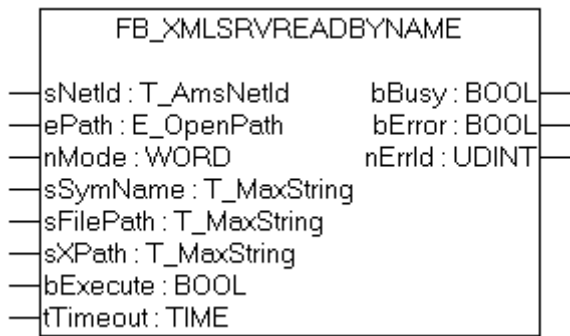
**bError:** This output is set up after the bBusy output has been reset if there has been an error in transmission of the command.

**nErrId :** If the bError output is set, this parameter returns the TwinCAT XML Data Server error number [▶ 29].

**Requirements**

Development environment	Target system type	PLC libraries to be linked
TwinCAT v2.10.0 from Build 1235	PC or CX (x86)	TcXmlDataSrv.Lib ( Standard.Lib; TcBase.Lib; TcSystem.Lib are included automatically )

### 5.1.3 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 XML Data 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 influence how the XML file is analyzed. For the XmlSrvReadByName command, only XMLSRV\_SKIPMISSING mode is currently supported.

**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 rising 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.

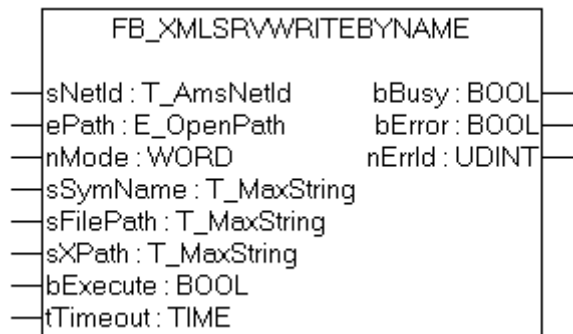
**nErrId :** If the bError output is set, this parameter returns the TwinCAT XML Data Server error number [► 29].



Requirements

Development environment	Target system type	PLC libraries to be linked
TwinCAT v2.10.0	PC or CX (x86)	TcXmlDataSrv.Lib ( Standard.Lib; TcBase.Lib; TcSystem.Lib are included automatically )

5.1.4 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 must 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 XML Data 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 specify which data are written to the XML file. XMLSRV\_SKIPMISSING and XMLSRV\_ADDMISSING mode are available for the XmlSrvWriteByte command. In XMLSRV\_SKIPMISSING mode, only those sub elements of a PLC symbol that already exist in the XML file are written to the XML file. In XMLSRV\_ADDMISSING mode, missing sub elements 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 rising 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.

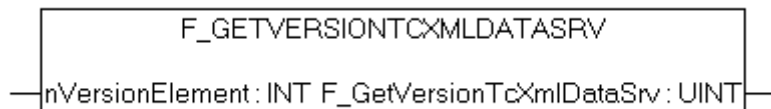
**nErrId** : If the bError output is set, this parameter returns the [TwinCAT XML Data Server error number](#) [▶ 29].

**Requirements**

Development environment	Target system type	PLC libraries to be linked
TwinCAT v2.10.0	PC or CX (x86)	TcXmlDataSrv.Lib ( Standard.Lib; TcBase.Lib; TcSystem.Lib are included automatically )

## 5.2 Functions

### 5.2.1 F\_GetVersionTcXmlDataSrv



This function can be used to read PLC library version information.

**FUNCTION F\_GetVersionTcXmlDataSrv : UINT**

```
VAR_INPUT
  nVersionElement : INT;
END_VAR
```

**nVersionElement** : Version element to be read. Possible parameters:

- 1 : major number;
- 2 : minor number;
- 3 : revision number;

**Requirements**

Development environment	Target system type	PLC libraries to be linked
TwinCAT 2.10.0	PC or CX (x86)	TcXmlDataSrv.Lib ( Standard.Lib; TcBase.Lib; TcSystem.Lib are included automatically )

## 5.3 Global constants

### 5.3.1 Global Variables

```

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	PLC Libraries to include
TwinCAT v2.10.0	PC or CX (x86)	TcXmlDataSrv.Lib ( Standard.Lib; TcBase.Lib; TcSystem.Lib are included automatically )

## 6 Samples

In the following you will find Samples that show the handling of the TwinCAT XML Data Server. The samples are consecutively numbered and can be [https://infosys.beckhoff.com/content/1033/tcxmldata\\_srv/Resources/11416819467/.zip](https://infosys.beckhoff.com/content/1033/tcxmldata_srv/Resources/11416819467/.zip).

[Getting Started \[► 20\]](#)

The basic handling of the TwinCAT XML Data Server is shown. Besides you get to know how to work with structures and arrays.

[Function Blocks \[► 22\]](#)

Four samples show the usage and configuration of the function blocks included in TcXmlDataSrv.Lib. (Sample 1-4)

[Further Samples \[► 24\]](#)

This document includes further samples which show an initialization once at program startup and cyclic as well as event-driven writing. (Sample 5-6)

[Production Sample \[► 26\]](#)

This sample shows the processing of a small production order. It makes use of FB\_XmlSrvReadByName and FB\_XmlSrvWriteByName. (Sample 7)

### 6.1 Getting Started

In the following you will get to know the basics in working with the TwinCAT XML Data Server.

#### Primitive data types

The table below shows the supported primitive data types

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-Typen werden in der XML-Datei als DWORD gespeichert*)	<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>

Example: The plc global variable Var1 ist typed as DINT. This ist the XML structure

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

The input parameters of the function block FB\_XmlSrvRead must be set as follows:

```
fbRead.pSymAddr      := ADR(value1);
fbRead.cbSymSize     := SIZEOF(value1);
fbRead.sFilePath     := 'C:\Test.xml';
fbRead.sXPath        := '/dataentry/Var1';
```

The root element in the XML file and the name of the variable are freely selectable in the XML file. These values must be specified in the input parameters sXPath, sFilePath. Also, multiple variable definitions can be stored in a single xml file:

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

To access the definition vor Var3.a set sXPath:='/dataentry/Var3.a'

## Structures

Structures in the XML file have the same hierarchical structure as in the PLC. However, there is the possibility to skip individual sub elements in the XML file. The individual sub elements of the structure must have the same names as in the PLC, otherwise they are skipped. If sub elements in the XML file cannot be converted to the correct data type, they will also be skipped.

Example: The global variable Var2 is typed as ST\_MYSTRUCT:

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

A possible xml file representation:

```
<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 sub elements are completely and correctly defined, so the variable is fully initialized. In the following example, however, only the sub element c is serialized:

```
<variables>
  <Var1>10</Var1>
  <Variable2>                            <!-- XPath := '\variables\Variable2' --!>
  <Info>Test entry</Info>
  <a>-100</a>
  <c>1.2</c>
</Variable2>
</variables>
```

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

## Arrays

To specify the index of arrays, the "Index" attribute must be used for the individual array elements. Individual array elements can also be omitted. These are then simply skipped.

Example: the plc variable `array1` is defined as: `ARRAY[1..4] OF DINT`.

A possible xml file representation:

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

## Requirements

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0	PC or CX (x86)	TcXmlDataSrv.Lib

## 6.2 Function blocks

The following samples show the handling of the functionblocks of the tcXmlDataSrv-Library. The PLC project, which contains the samples, can be [https://infosys.beckhoff.com/content/1033/tcxmldata\\_srv/Resources/11416819467/.zip](https://infosys.beckhoff.com/content/1033/tcxmldata_srv/Resources/11416819467/.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: Basic printing procedure with `FB_XmlSrvWrite`

In the first step the structure `ST_MyStruct` shall be written to an XML file. The mode is set to `XMLSRV_ADMISSING`, so that the XML file will be created automatically (if it does not already exist) as well as the structure in it. Folders will not be created automatically! This method is recommended even then, if you do not intend to write but just to read the file. So, the XML file does not have to be created manually, which saves time and avoids mistakes.

```
(* 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: Basic printing procedure with FB\_XmlSrvWriteByName

Sample 2 has the same result as sample 1 but makes use of FB\_XmlSrvWriteByNamen. However, Sample 1 is more performant.

```
(* 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

No matter which function block is used, the XML file 'Test.xml' will be created under C:\. For that you get the same XML-content both times, sXPath must be the same ('/dataentry/MAIN.value1'). It does not matter that value1 is not directly created in the main, but in the specific program. This path in TwinCAT is set by sSymName (Sample 2): '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: Basic reading procedure with FB\_XmlSrvRead

In the following the structure created in Sample 1 or Sample 2 shall be read.

```
(* 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: Basic reading procedure with FB\_XmlSrvReadByName

Sample 4 shows the reading procedure under usage of 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;
```

Here again (as in sample 2) you must differentiate between sSymName and sXPath: sXPath sets the path within the XML file - determined in sample 1 and 2. sSymName on the other hand sets the symbol name of the variable in TwinCAT, which is 'Sample4.value1'.

## Requirements

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0	PC or CX (x86)	TcXmlDataSrv.Lib

## 6.3 Further Samples

The following samples show different types of application of the TwinCAT XML Data Servers. The PLC-project, which contains the samples, can be [https://infosys.beckhoff.com/content/1033/tcxmldata\\_srv/Resources/11416819467/.zip](https://infosys.beckhoff.com/content/1033/tcxmldata_srv/Resources/11416819467/.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
TEND_TYPE

```

```

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

```

### Sample 5: Initialization once at program startup

Sample 5 shows the initialization of value1 once at program startup. Therefore 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 printing

The following sample creates a new XML file every 20 seconds and writes the above-mentioned structure to it. The filename is made up of the current windows-date, -time and the string sFileName. Furthermore, the printing procedure can be started by pressing a button (or setting the respective variable bButton). If the printing procedure is triggered twice a second, the latest file will be 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 AN
D 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 THEN (* create filename *)
  sFilePathWrite:= CONCAT(sFileFolder, SYSTEMTIME_TO_STRING(stMyTimeStruct)); (* set folder + time *)
  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) *)
(* write *)
  fbXmlSrvWrite(sFilePath:=sFilePathWrite, sXPath:=sXPathWrite, bExecute:= TRUE);
ELSIF fbXmlSrvWrite.bError THEN
  iState:= 100;
END_IF(* reset fbXmlSrvWrite *)IF fbXmlSrvWrite.bBusy AND NOT ntGetTime.ERR THEN
  fbXmlSrvWrite(bExecute:= FALSE);
ELSIF ntGetTime.ERR THEN
  iState:= 100;
END_IF
100: (* error state*)
;
END_CASE

```

## Requirements

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0	PC or CX (x86)	TcXmlDataSrv.Lib

## 6.4 Sample 7 (Production Sample)

The production data are 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 at the corresponding place of the data path and the variable bStart needs to be set TRUE in the PLC program.

The **PLC-projekt**, which includes this sample, can be <https://infosys.beckhoff.com/content/1033/tcxmldatasrv/Resources/11416819467/.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 are 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 at the corresponding place
   of the data path 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_IFCASE 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 *)
    ;
END_CASE

```

**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**

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0	PC or CX (x86)	TcXmlDataSrv.Lib

## 7 Appendix

### 7.1 TwinCAT XML Data Server error codes

#### Requirements

Offset + error code	Range	Description
0x00000000 + TwinCAT system error	0x00000000-0x00007800	TwinCAT system error (including ADS error codes)
0x00008000 + <a href="#">internal TwinCAT XML Data Server error</a> [► 29]	0x00008000-0x000080FF	Internal TwinCAT XML Data Server error

### 7.2 Internal error codes of the TwinCAT XML Data Server

#### Requirements

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 Client Handle	XMLSRVERROR_INVALIDCLIENTHANDLE
0x00008900	Wrong Version of TcXmlDataSrv.exe in use (TC2 instead of TC3)	XMLSRVERROR_INVALIDTWINCATVERSION



More Information:  
[www.beckhoff.com/ts6421](http://www.beckhoff.com/ts6421)

Beckhoff Automation GmbH & Co. KG  
Hülshorstweg 20  
33415 Verl  
Germany  
Phone: +49 5246 9630  
[info@beckhoff.com](mailto:info@beckhoff.com)  
[www.beckhoff.com](http://www.beckhoff.com)

