

Manual | EN

TcEventLogger



TwinCAT 2 | System



Table of contents

1 Foreword	7
1.1 Notes on the documentation	7
1.2 Safety instructions	8
1.3 Notes on information security.....	9
2 Introduction	10
3 Setup	13
4 Alarm & Events for Windows CE	15
5 Connectivity	16
5.1 DCOM	16
5.1.1 DCOM Samples	16
5.2 ADS	17
5.2.1 Compatibility.....	17
5.2.2 Interfaces	19
6 Event Configuration	20
6.1 Beckhoff TcEventConfigurator	20
6.2 Getting started.....	20
6.3 TcEventConfigurator Reference.....	21
6.4 TcEventConfigurator - File Formats	23
6.5 TcEventConfigEditor ActiveX	24
6.6 EventConfiguration on Windows CE	25
6.7 COM.....	25
6.7.1 IEvtConfiguration.....	26
6.7.2 IEvtCfgSource	32
6.7.3 IEvtCfgEvent	41
6.7.4 IEvtCfgDocLink	50
6.7.5 IEvtCfgEnumParser	55
6.7.6 IEvtCfgParser.....	55
6.7.7 EvtCfgTpsParser.....	60
6.7.8 EvtCfgXmlParser.....	61
6.7.9 IEvtCfgEditor	61
7 Event HMI	67
7.1 Classes	68
7.1.1 TcEventView	68
7.2 Interfaces	68
7.2.1 ITcEventView	68
7.2.2 ITcEventViewLight	112
7.3 Enums.....	113
7.3.1 TCEVENTVIEW_CONTEXTMENUFLAGS.....	113
7.3.2 TCEVENTVIEW_DISPLAYMODE	114
7.3.3 TC_SORTMODE.....	114
8 Event Formatting	116
8.1 TcEventFormatter	116

8.1.1	ITcLogFormatterC	116
8.2	TcXmlFormatter.....	122
8.2.1	TcEventSourceLocation	122
8.2.2	XmlEventConfiguration	123
9	API	127
9.1	TcEventLogger Return Codes.....	127
9.2	Classes	128
9.2.1	TcEventLog	128
9.2.2	TcEvent.....	128
9.3	Interfaces	129
9.3.1	ITcEventLog	129
9.3.2	ITcEventLogC	137
9.3.3	ITcEventLogC2	139
9.3.4	ITcEventLogC3	141
9.3.5	ITcEventLogEvents	144
9.3.6	ITcEnumEvents.....	150
9.3.7	ITcEnumEventEx	155
9.3.8	ITcEvent.....	157
9.3.9	ITcEventC	175
9.3.10	ITcEnumEventDocLink.....	176
9.4	Enums.....	179
9.4.1	TcEventClass	179
9.4.2	TcEventConCodes	179
9.4.3	TcEventFlags	180
9.4.4	TcEventPriority.....	180
9.4.5	TcEventStates.....	181
9.5	Structures.....	181
9.5.1	TcEventDocLink.....	181
9.5.2	TcEventHeader	181
10	Samples	183
10.1	Configuration of Messages	184
10.2	PLC interface	186
10.2.1	Issue a simple message.....	186
10.2.2	Issue a complex message.....	187
10.2.3	Issue a message with format parameters	189
10.3	Visual Studio C++	190
10.3.1	Integration of TcEventViewer ActiveX control	190
10.3.2	Console Application - Read logged events via DCOM.....	191
10.3.3	Connection Points	192
10.3.4	Add the TcEvtCfgEditor ActiveX into an Dialog	192
10.4	Visual Studio C#.....	193
10.4.1	Displaying Logged Events in C#	193
10.4.2	Displaying Active Events in C#	195
10.5	Visual Basic.....	196
10.5.1	Integration of TcEventViewer ActiveX control	196

10.5.2	View active alarms in user defined ListView	197
10.6	C++Builder 2009	198
10.6.1	Integration in CodeGear C++Builder 2009.....	198
10.6.2	Console Application - Read logged alarms via DCOM	206
10.6.3	Console Application - Read logged alarms via ADS proxy	208
10.6.4	View active alarms in user defined ListView	211
10.6.5	Einbinden von TcEventViewer-ActiveX-Control	214

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 Introduction

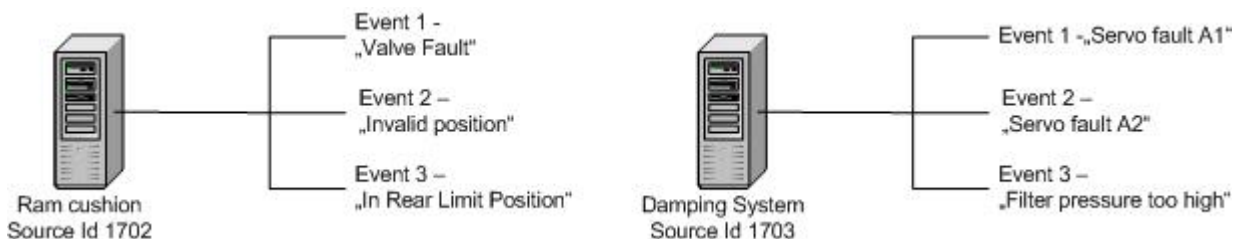
The TcEventLogger is Beckhoffs standard alarm and event concept. It helps to trigger and visualize warnings, errors, information or status messages from the PLC. By using the TcEventlogger for your Alarm & Event implementation you are able to realize the following requirements:

1. Issue Alarms and Events from the PLC
2. Manage events through the TcEventLogger
3. COM APIs for HMI development
4. Fully featured ActiveX control for display of events
5. Support for implementation of distributed systems: The HMI can easily display events of a remote PC
6. Load language specific message texts from a database

The TwinCAT Message-Concept:

Messages (Events)

Each message is identified by a Source- and an Event ID. The Source ID is assigned to a machine and the Event ID is assigned to a certain event of that machine.

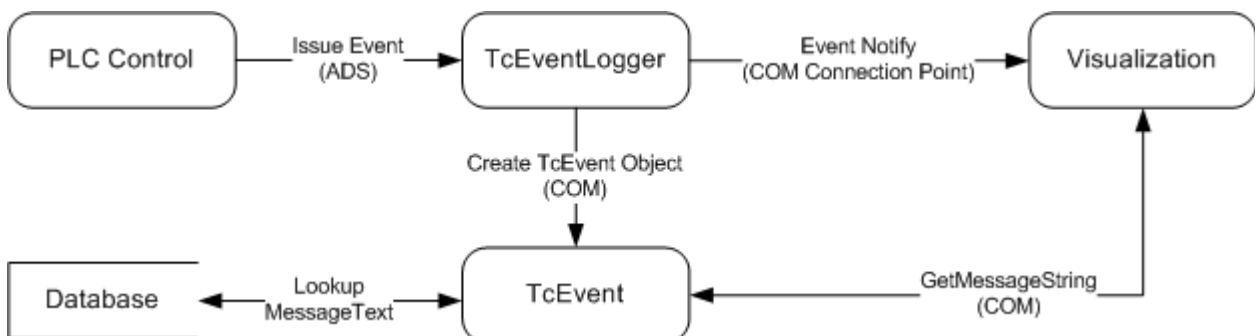


Source names and event messages are stored in a certain database file. When an event is issued, the TcEventLogger looks up the appropriate message in its database and provides it to its clients. Clients typically are applications like HMIs.

For each event additional "document links" may be stored in the database. Each DocLink is identified by its name and contains a link to a file. Processing of these links is up to the clients - the TcEventLogger only provides functionality to lookup the link itself (see '[message formatting \[► 10\]](#)' for details).

Message Triggering

You can use a simple [Function Block](#) to signal an event from the PLC. As soon as an event is issued (or its state changes) the TcEventLogger notifies its clients about this event. The client is called and a [TcEvent \[► 157\]](#) COM object supplied. Using this event object, the client can access the event and e.g. query a completely formatted message string in the appropriate language from the database.



Message formatting

The TcEventLogger offers the [TcEventFormatter \[116\]](#) concept to load message texts from a database. Beckhoff supplies default formatters e.g. for loading message texts from a xml file. Custom formatters can be implemented and injected into the system.

Default formatters supplied by Beckhoff:

Formatter	Description
TcEventFormatter	Loads data from the TwinCAT Project Storage (.tps).
TcXMLFormatter	Loads data from XML files.

The formatter to be used for an event is chosen in the PLC, when issuing the event.

Message Configuration

To load message texts from a database the events need to be configured. The [TcEventConfigurator \[20\]](#) offers a GUI to do that. It supports both standard formatters and additional Plain Text or Excel format for translation purposes. Use the TcEventConfigurator used to do the entire event configuration:

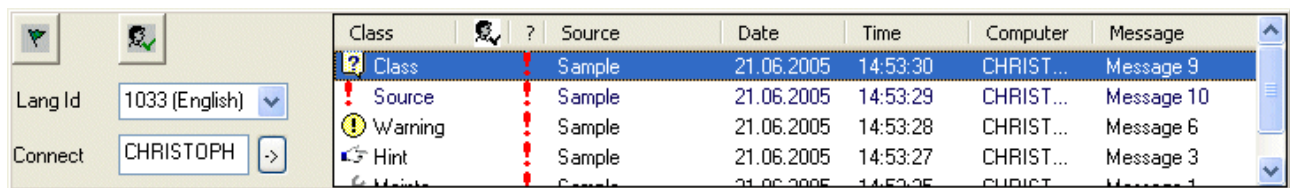
- Configure messages
- Save data in different formats
- Generate PLC Sample Code

Visualizations

To reduce the effort of creating custom visualizations, a standard implementation is supplied with the TcEventLogger : The [TcEventViewer \[67\]](#).

The TcEventViewer is an extensive configurable ActiveX control for usage in customers HMIs.

The TcEventBar is an example implementation for displaying events. It utilizes the TcEventViewer ActiveX.



The TcEventBar can be found in the 'TwinCAT/EventLogger' directory.

Setting up the TcEventBar

The TcEventBar uses a registry key (`HKEY_CURRENT_USER\Software\TwinCAT\TcEventBar`) for the properties (since v.2.9.0.11) . In the "ContextMenuFlags" value you can save the context menu settings of the TcEventBar.

The following flags can be set to disable the menu items.

TcEventView item:	Description
0x00000001	"Clear active events"
0x00000002	"Clear logged events"
0x00000004	"Set filter"
0x00000008	"Confirm"
0x00000010	"Reset"

TcEventView item:	Description
0x00000020	"Details"
0x0000FFFF	Hide the complete context menu

Main window context menu:	Description
0x00010000	"Always on top"
0x00020000	"Autohide"
0x00040000	"Exit"
0xFFFF0000	Hide the complete context menu

Connectivity

The TcEventLogger provides two ways for implementing distributed systems:

DCOM

DCOM is only supported on NT / XP Platforms. There are well known issues using DCOM connections in the scope of configuration and network timeouts.

ADS Stream Interface

The TcEventLogger supports a binary ADS Interface to implement distributed systems.

To keep programming efforts at a minimum, Beckoff provides a [proxy \[► 19\]](#) implementation which wraps binary encoding and exchange of data. The [TcEventLogAdsProxy \[► 19\]](#) class provides the whole set of the [TcEventLogger COM API \[► 127\]](#) for client programming.

3 Setup

Log TwinCAT System errors

It is possible since TwinCAT 2.8 to log NC errors and I/O errors with the TcEventLogger.

This can be controlled with the following registry key:

HKEY_LOCAL_MACHINE\SOFTWARE\Beckhoff\TwinCAT\System[LogMessageType]

Possible values for LogMessageType are:

Value	Description
0	TwinCAT System message logged into the operating system.
1	TwinCAT System message logged into the TcEventLogger [▶ 127] . They available for Custom programs and the HMI in the list of logged events. System messages use the TcXmlFormatter [▶ 116] .
2	Default mode. TwinCAT system message will be logged into the operating system and the TcEventLogger [▶ 127] .

Message sources, database organisation, and the prompt on a TwinCAT System:

source	message text as xml file	Messages on a TwinCAT (PC) System			prompt	
		database	event viewer of the OS	TcEventViewer-Client e.g. TcEvent-Bar.exe	event viewer of the OS	TwinCAT System Manager Logger
OS	-	-	X	-	X	-
AD-SLOGSTR, AD-SLOGDINT, AD-SLOGLREAL functions	-	-	X	-	X	X
ADSLOGEVENT module	X*	X	-	X	-	-
TwinCAT System NC and IO-messages	X TwinCAT 2.8 or newer	X** only at LogMessageType 1 or 2	X only at LogMessageType 0 or 2	X** only at LogMessageType 1 or 2	X only at LogMessageType 0 or 2	X

- : not possible

X : possible

* only with the TcXmlFormatter.

** Not all messages will be logged to reduce the system load.

Disable log

The TcEventLogger has a small log with 128 messages max. The TcEventlogger saves this log together with its configuration in the project storage. To disable this and switch to temporary log files you can set a value in the registry:

```
HKLM\SYSTEM\CurrentControlSet\Services\TcEventLogger\LogToTcStg (REG_BINARY)
```

- not there or value = 0x1 log in the storage
- there and value = 0x0 disable log in the storage

DCOM Security

From TwinCAT 2.11. TcEventLogger implicitly sets default access rights. To set access rights explicitly via dcomcnfg or registry (e.g. for remote access) set the following value in the registry:

```
HKLM\SYSTEM\CurrentControlSet\Services\TcEventLogger\UseAppIdSecurity  
(REG_DWORD)
```

- Not existing OR value = 0 : implicit configuration of access rights
- Value >= 1 : configuration via Registry/ Dcomcnfg

4 Alarm & Events for Windows CE

The TcEventLoggerCE is available on Beckhoffs Windows CE platforms since Image version

- 2.20a
- 3.06a

On Windows CE only the [ADS communication channel \[▶ 17\]](#) is supported. Use the [TcEventLogAdsProxy \[▶ 19\]](#) library to implement event logging solutions.

5 Connectivity

Certain production environments may require the possibility for displaying events of certain machines on a host computer. The TcEventLogger does support two ways for implementing such systems:

- **DCOM**
This is a proprietary protocol developed by Microsoft. Method calls are serialized over the network so that one can call methods inside the remote TcEventLogger. This protocol is only available on Windows NT/ XP based systems.
- **ADS**
Beckhoff has defined a binary protocol for exchange of event states and state callbacks. This protocol is supported on NT/ XP as well as on Windows CE.

Exchanging the communication layer during the development process or even at runtime is possible with minimal effort.

5.1 DCOM

General

The TcEventLogger does support DCOM connections to remote systems. DCOM allows to create a virtual local instance of an object which is physically running on a remote PC. By utilizing DCOM you can create an instance of a remote eventlogger and programatically query its active or logged events.

EventViewer/ HMI

The TcEventViewer ActiveX supports DCOM remote connections. Use the AddConnection method to attach to a remote eventlogger. The TcEventViewer supports connections to a virtually unlimited number of remote machines at a time.

Notes

DCOM needs to be configured to allow access from another computer. Check the MSDN for further information.

DCOM is not supported on Beckhoffs Windos CE based platforms.

5.1.1 DCOM Samples

C++

```
void ConnectComputer(BSTR computer)
{
    ITcEventLogPtr spTcEventLog
    COSERVERINFO comServerInfo={0};

    comServerInfo.pwszName = computer;
    MULTI_QI mQI={&IID_ITcEventLog, NULL,0};

    hr::CoCreateInstanceEx( CLSID_TcEventLog,
                            NULL,
                            CLSCTX_SERVER,
                            &comServerInfo,
                            1,
                            &mQI);

    if ( SUCCEEDED(hr) )
    {
        spTcEventLog =mQI.pItf;
        mQI.pItf->Release();
    }

    //...
}
```


5.2 ADS

General

Use the TcEventLogAdsProxy for connecting your application to an instance of the TcEventLogger. The EventLogger may either be running locally or on a remote computer.

This class implements the ITcEventLogAdsProxy interface for creating ADS connections and inherits from the ITcEventlog interfaces. The ITcEventLog provides an object oriented and easy to use [programming interface](#) [► 19]

Requirements

Connections to the AdsProxy are supported

- on Windows NT/ XP/ Vista since TwinCAT 2.10.1327
- on Windows CE since image version 2.20a or 3.06a

Setup

1. Create an ADS route between local and remote system (Not necessary for local connections)
2. Call the [ITcEventLogAdsProxy::Connect\(\)](#) [► 19] method

5.2.1 Compatibility

The WindowsCE Eventlogger offers a high degree of compatibility to the standard Win32 based logger. Minimal effort is required to develop an application that supports both versions. This document provides some implementation hints on developing a compatible application.

See samples for

- [C#](#) [► 17]
- [C++](#) [► 17]
- [TcEventViewer](#) [► 18]
- [TcEventBar](#) [► 19]

Developing a C# application that supports both versions of the TcEventLogger

1. Add references to the "Beckhoff TcEventLogger" and the "Beckhoff TcEventLogAdsProxy" COM type libraries
2. Add one global instance of the TcEventLogger
3. Use these two methods to connect either to a local PC Based TcEventLogger or an remote WindowsCE system. If your application is a Windows CE and shall connect the local logger pass *Null* for the NetId

```
void ConnectRemoteLogger( String strNetId )
{
    TcEventLogAdsProxyLib.TcEventLogger adsLogClient = new
    TcEventLogAdsProxyLib.TcEventLogger ( );
```

```

        adsLogClient.Connect( strNetId );
        m_logger = (TcEventLog)adsLogClient;
    }

void ConnectLocalPC()
{
    m_logger = new TCEVENTLOGGERLib.TcEventLog();
}

```

4. Write currently active events to the console window

```

foreach( TcEvent evt in m_logger.EnumActiveEventsEx() )
    Console.WriteLine( evt.GetMsgString(1033) );

```

Developing a C++ application that supports both versions of the TcEventLogger

1. Import type libraries of the "Beckhoff TcEventLogger" and the "Beckhoff TcEventLogCEProxy"

```

#pragma warning(disable: 4192)
#import "C:\TwinCAT\TcEventLogger\TcEventlogger.exe" no_namespace,
        named_guids
#import "C:\TwinCAT\TcEventLogger\TcEventLogAdsProxy.dll" no_namespace,
        named_guids
#pragma warning(default: 4192)

```

2. Add a member variable for the TcEventLogger object

```
ITcEventLogPtr m_spTcEventLog;
```

3. Use these two methods to connect either to a local PC Based TcEventLogger or an remote WindowsCE system. If your application is a Windows CE and shall connect the local logger pass *Null* for the NetId

```

HRESULT ConnectRemoteLogger( BSTR strNetId )
{
    HRESULT hr = E_FAIL;
    if ( m_spTcEventLog == NULL )
    {
        hr = m_spTcEventLog.CreateInstance(CLSID_TcEventLogAdsProxy);
        if ( SUCCEEDED(hr) )
        {
            ITcEventLogAdsProxyPtr spLogCE = m_spTcEventLog;
            hr = spLogCE->Connect( strNetId ); // 5.1.47.197.1.1
        }
    }
}

```

```

void ConnectPC( BSTR strComputer )
{
    HRESULT hr=E_FAIL;

    COSERVERINFO comServerInfo={0};

    comServerInfo.pwszName = strComputer ;
    MULTI_QI mQI={&IID_ITcEventLog, NULL, 0};

    if ( m_spTcEventLog == NULL )
        hr::CoCreateInstanceEx(CLSID_TcEventLog,
                                NULL,
                                CLSCTX_SERVER,
                                &comServerInfo,
                                1,

```

```

        &mQI);
    if ( SUCCEEDED(hr) )
    {
        m_spTcEventLog = mQI.pItf;
        mQI.pItf->Release();
    }
}

```

Connect the TcEventViewer with the TcEventLoggerCE

1. TcEventViewer version 2.9.0.73 is required (Shipped with TwinCAT build 2.10.1317 or higher)
2. For connecting a WindowsCE device pass "ADS://<AmsNetId>" to the [ITcEventView::AddConnection\(\)](#) [▶ 77] method

Using the TcEventBar with the TcEventLoggerCE

1. TcEventViwer version 2.9.0.73 is required (Shipped with TwinCAT build 2.10.1317 or higher)
2. In the *Connect* field type "ADS://<AmsNetId>"

5.2.2 Interfaces

Supported Interfaces :

Interface	Description
ITcEventLogAdsProxy [▶ 19]	This is the main Interface of the TcEventLogAdsProxy. Use this interface to connect an instance of the TcEventLogger. After the proxy has successfully connected continue using the standard TcEventLogger interfaces
ITcEventLog [▶ 129]	This it the main Interface of the TcEventLogger for COM Clients like HMI. It offers functions to control active and logged alarms and to add and remove alarms.
ITcEventLogC [▶ 137]	This is the basic interface of ITcEventLogC3 and ITcEventLogC2.
ITcEventLogC2 [▶ 139]	This interface derives from ITcEventLogC and is the basic interface for ITcEventLogC3.
ITcEventLogC3 [▶ 141]	This is a application-specific interface of the TcEventLoggers for COM Clients that use application-specific interfaces like HMI. is interface derives from ITcEventLogC2.
ITcEventLogEvents [▶ 144]	Callback functions for callbacks from the TcEventlogger to COM-Clients. Clients can use this to get informations when an event occurs or is cancelled.

5.2.2.1 ITcEventLogAdsProxy

The ITcEventLogAdsProxy interface provides the basic functionality for connecting to a Eventlogger.

ITcEventLog Meth-ods and Proper-ties	Description
Connect	Connect to a remote TcEventLoggerCE by passing the target AmsNetId or Computer Name as a string. Pass NULL or 'localhost' to connect to a local instance of the TcEventLogger. NOTE: precondition for a successful connection to a remote instance is that an ADS route has been configured.
Disconnect	Disconnect from the TcEvenLoggerCE. This method is called internally upon destruction of the object.
Connected	Gets whether the proxy is currently connected to an Instance of the TcEventloggerCE.

6 Event Configuration

6.1 Beckhoff TcEventConfigurator

The TcEventConfigurator is a powerful tool, provided to simplify the setup of your eventing system. Using this tool will render it unnecessary to understand the TcEventlogger in its complete complexity.

This Documentation does cover the following topics:

1. [Getting started \[▶ 20\]](#) with the TcEventConfigurator (Using the Wizards to create a configuration).
2. Short usage [Reference \[▶ 21\]](#) for the configurators UI.
3. COM [Interface \[▶ 25\]](#) documentation (for developers who want to integrate the configurator in their own Application).
4. [Formats \[▶ 23\]](#) of the configurator.

The Beckhoff EventConfigurator is available since TwinCAT version 2.10.0 (Build 1253). Windows 7 support has been added in EventConfigurator version 2.0.0 (Build 68).

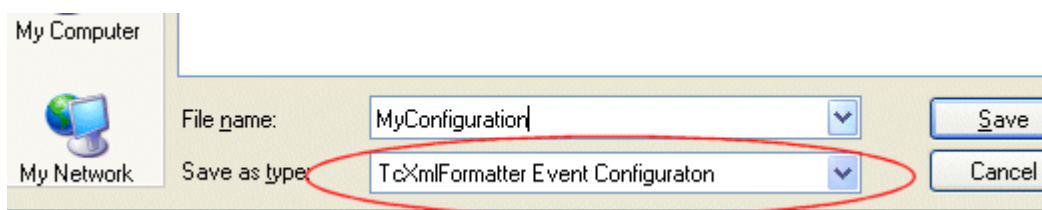
Requirements:

.NET Framework 2.0

6.2 Getting started

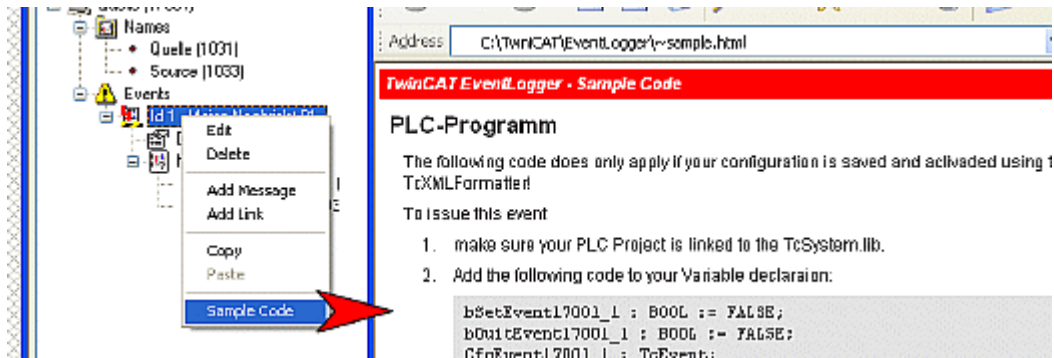
This tutorial gives you step by step instructions on how to create and issue your first event. You will start with using the TcEventConfigurators Wizard to create an event source and the event itself. Then you will use the Configurator to generate some code fragments which you can paste to a PLC project to issue the event.

1. Start the TcEventConfigurator.
2. If you start the Configurator for the first time, a Wizard will pop up. If not so, you may start the wizard through the toolbar. Follow the Wizards instructions to create a Source and an Event.
3. If you want to create more events repeat step 2.
4. Save your configuration using the TcXmlEventFormatter.

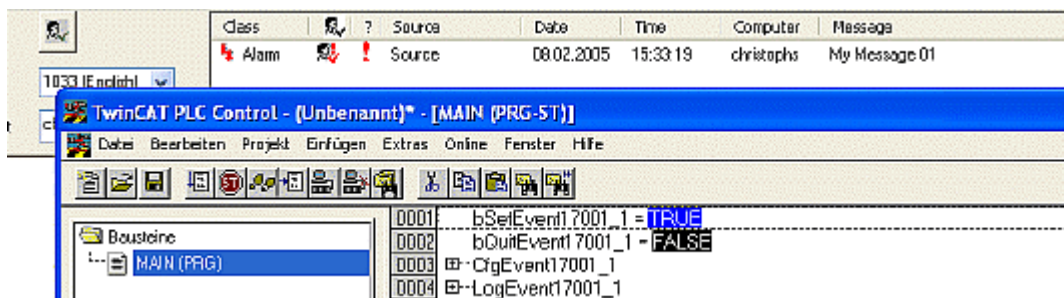


5. A PLC program is required to issue events. You may either use an existing program or create a new project. The TcEventConfigurator will create sample code and instruct you on how to use it. Expand the tree to the Event you want to issue.

Right click on the event and select 'Sample Code' from the popup menu. A HTML page containing code fragments and further instructions will pop up.

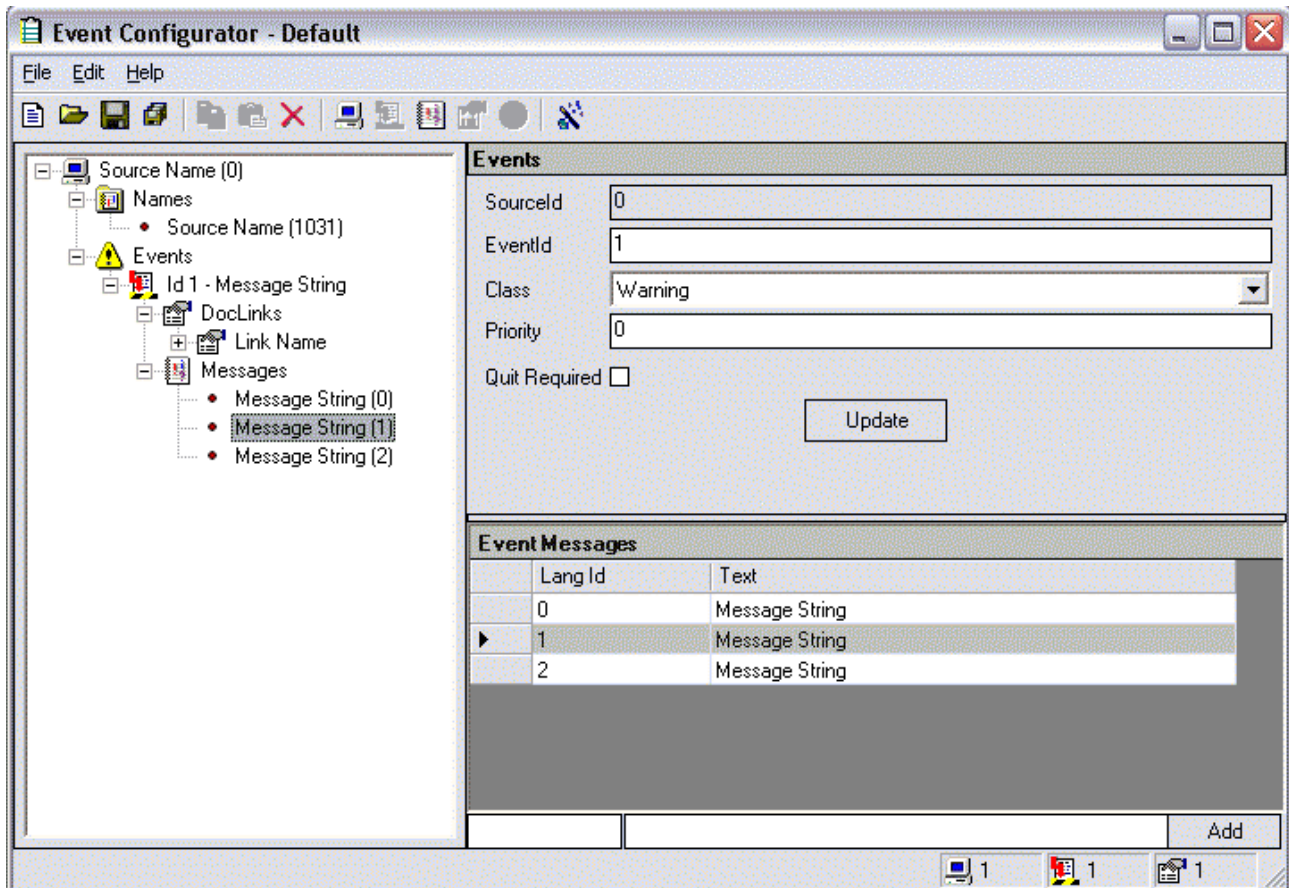


6. Now paste the code samples to your PLC program, start it and run the TcEventBar (Located in %TwinCAT%\EventLogger). Use the switches in your PLC program to issue events.








6.3 TcEventConfigurator Reference

The TcEventConfigurator consists of a tree view to the left and a context dependent property page to the right



The TreeView does display the hierarchical structure of the Event Configuration. Via the right click context menus you are able to add remove and edit each item in the configuration.

	Source. Each source is displayed through a source item
	Folder containing all names associated with each source
	The Event tag. Expand to display all events wich are child to the source
	Message Tag. Expand to view the message string(s) associated with the EventID.
	Document links for this Event

The right pane is used to show detailed lists of the items selected in the tree. You may also edit data in this pane.

Adding/Editing Source Names, Messages or URLs

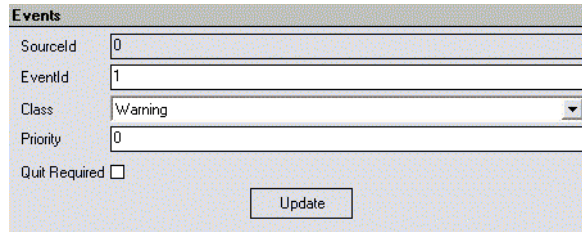
Select what you want to Add, if you select a Source you can add a Sourcename, if you select an Event you can add Messages and if you select a DocLink you can add URLs. You will get a editable list of the entries for the selected Item with fields for adding entries below it. Enter a numeric LangID in the first field and the text into the second as shown in the picture and click 'Add'.

Event Messages	
Lang Id	Text
0	Message String
1	Message String 1
2	Message String

1033	...	Add
------	-----	-----

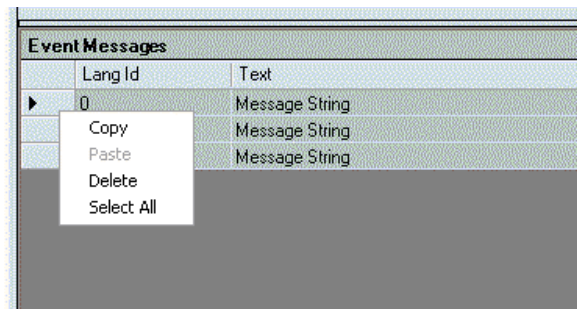
Editing Events

If you want to change the properties of an event you have to select the event in the Tree. You will get a property page on the right side and the messages for this event below. You can change the properties there, dont forget to klick 'Update' after you changed them.



Deleting, Copying and entries

You can use the message table to copy or delete messages. Just select one or more rows in the table and click the right mouse button.



6.4 TcEventConfigurator - File Formats

The TcEventConfigurator offers The following formats:

Format	File Extension	Description
Event Configuration Project Ex	ecpx	Native TcEventConfigurator format. This is a improved version of to ecp format
Event Configuration Project	ecp	Native TcEventConfigurator format. Was replaced by the ecpx format
TcXMLFormatter	xml	Load/Save data from XML files.
TcEventFormatter	tps	Load/Save data from the Project Storage.
Excel	xls	Load/Save Data to Excel xls Files. Used to exchange data with translation bureaus.
Plain Text Format	txt	Load/Save data from txt files.

Format features

Not all formats support all features of the TcEventConfigurator. See a table of supported features below.

If you do not have special needs that require usage of another than the ecp format you should always decide for that format as it is the configurators native format and will always support every feature of the Configurator.

Format	Activation *	Save event properties **	Save multiple Source IDs ***
ecpx	X	X	X
ecp	X	X	X
xml	X	X	X
tps	X	X	
xls		X	X

Format	Activation *	Save event properties **	Save multiple Source IDs ***
txt			

* If a configuration is activated, the TcEventLogger will search this database for message texts.

** Event properties are the event type (Warning/Message/...), Priority etc.

*** The Multiple Source ID feature is that one actual Source is associated with multiple SourceIDs.

Plain Text Format Description

The Plain Text Format can be used for transmission to translation bureaus. It stores data in human readable form and can be modified by hand. See the sample below on how to extend a text file with another language:

This sample configuration shows a basic Plain Text Document, containing the languages German and English.

```
Size[1,1,1]
Sourcename [0,1031] ='Ursprung'
Sourcename [0,1033] ='Source'
Message [0,1,1031] ='Meldungstext'
Message [0,1,1033] ='Message String'
DocLink [0,1,'New Doklink 1',1031] ='www.anyurl.de'
DocLink [0,1,'New Doklink 1',1033] ='www.anyurl.com'
```

To add a third language to the Configuration you just have to copy a line and change the Language ID (blue) and the text for it (green).

Check the following example for details:

```
Size[1,1,1]

Sourcename [0,1031] ='Ursprung'
Sourcename [0,1033] ='Source'
Sourcename [0,1034] ='Fuente'

Message [0,1,1031] ='Meldungstext'
Message [0,1,1033] ='Message String'
Message [0,1,1034] ='Texto'

DocLink [0,1,'New Doklink 1',1031] ='www.anyurl.de'
DocLink [0,1,'New Doklink 1',1033] ='www.anyurl.com'
DocLink [0,1,'New Doklink 1',1034] ='www.anyurl.es'
```

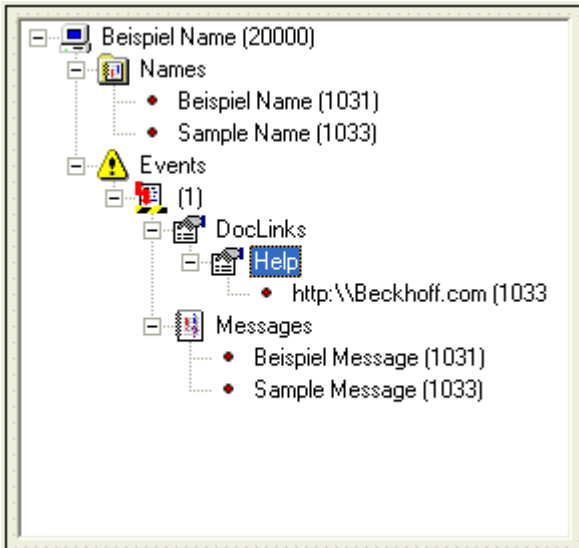
The following table gives a list of some common LCIDs*.

LCID	Description
1031	German
1033	US English
1034	Spanish
1036	French

* See the MSDN for detailed information.

6.5 TcEventConfigEditor ActiveX

The TcEvtCfgEditor is a simple TreeView Control to display and edit event configuration. It provides basic functionality to set up a complete event configuration for the TcEventLogger.



In theory the TcEvtCfgEditor does provide anything necessary to edit configurations, but in practice you will need to extend its features using the [TcEvtConfiguration COM interfaces](#) [[▶ 25](#)], just like the EventConfigApp does.

To customize the editors behavior, use the [IEvtCfgEditor Interface](#) [[▶ 61](#)]

6.6 EventConfiguration on Windows CE

The TcEventConfigurator is not available for Windows CE. You can create event configurations on your Desktop System and manually activate these files on the embedded device.

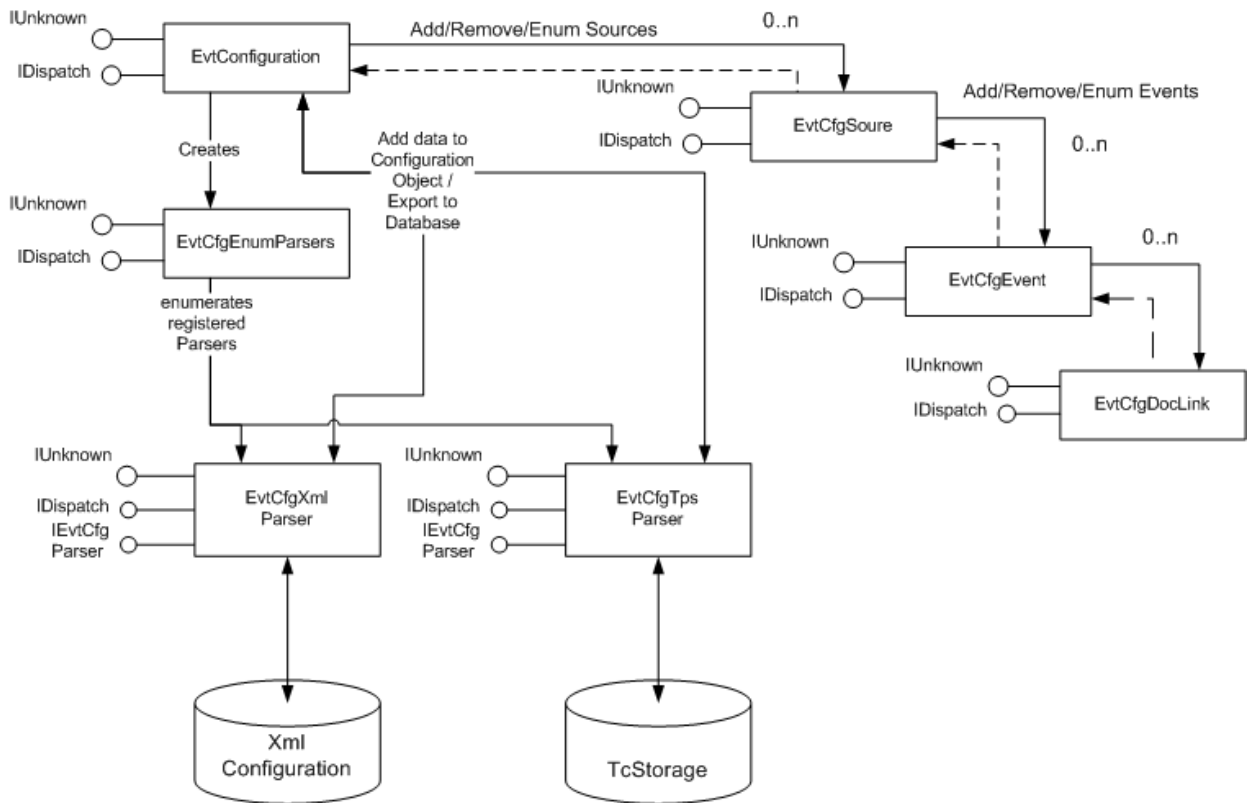
To manually activate a configuration:

1. Save it in Xml or EcpX format.
2. Activate the file(s) as described in the [TcXmlFormatter section](#) [[▶ 122](#)].

6.7 COM

The TcEvtConfiguration object is the core component of the Configurator. It exposes all its functionality via COM through the ITcEvtConfig Interface. Through its exposed methods you may programmatically open, create edit and save Configurations.

With the TcEventConfig Library a set of common import/export filters is supplied. If you want to use a custom database to store your event information, you have the possibility to implement an own parser. For detailed information on that see the ITcEvtCfgParser interface.



Use these Interfaces to manipulate a configuration

Interface	Description
IEvtConfiguration [▶ 26]	Core component of the configurator. From here files are loaded, saved and manipulated
IEvtCfgSource [▶ 32]	Component representing one event Source. From here the source name may be set and events belonging to that source may be accessed
IEvtCfgEvent [▶ 41]	Component representing one event. Through this component the e.g., event messages may be edited. Also, DocLinks associated with that event can be accessed
IEvtCfgDocLink [▶ 50]	Component representing one DocLink. Use to set up one events DocLinks
IEvtCfgEnumParsers [▶ 55]	Enumerates through all registered import/export parsers and returns the parsers Interfaces
IEvtCfgParser [▶ 55]	Import and export configurations to various storage formats

6.7.1 IEvtConfiguration

The IEvtConfiguration Interface represents the configurators core component. Through its methods configurations may be saved, loaded and modified. It also implements the _IEvtCfgEvents interface to notify an Application about changes in the configuration.

Table 1: IEvtConfiguration Methods and Properties in Vtable Order

IEvtConfiguration Methods	Description
AddSource [▶ 27]	Add a new Source to the configuration
GetSource [▶ 30]	Retrieve one Source from the configuration
RemoveSource [▶ 31]	Remove a Source from the configuration
GetFirstSource [▶ 29]	Start enumeration of sources and retrieve the first source
GetNextSource [▶ 30]	Continue enumeration of sources

IEvtConfiguration Methods	Description
EnumParsers [▶ 28]	Get an enumeration object to retrieve registered parsers
SourceCount [▶ 31]	Get the number of sources in the configuration
ClearConfig [▶ 28]	Reset the whole configuration
GetConfigSize [▶ 29]	Get the complete count of sources, events and doc links in the configuration
CopyTo [▶ 28]	Copy the current configuration to another or create a new Configuration containing the same data

Table 2: *_IEvtCfgEvents Methods and Properties in Vtable Order*

_IEvtCfgEvents Methods	Description
OnNewSource	A new source has been added
OnEditSource	A source has been edited
OnDeleteSource	A source was removed from the configuration
OnNewEvent	An event has been added
OnEditEvent	An event has been edited
OnDeleteEvent	An event was removed
OnNewLink	A link has been added
OnEditLink	A link has been edited
OnDeleteLink	A link was removed

6.7.1.1 AddSource

Add a new Source to the configuration.
Sources are identified by their IDs- Duplicate IDs are not allowed.

```
HRESULT AddSource([in]
IEvtCfgSource* pSource);
```

Parameters

pSource

[in] pointer to the new source object to be added.

Return Values

S_OK

The new source was successfully added to the configuration. The OnNewSource Event is fired.

E_FAIL

pSource could not be added to the configuration. This can happen if either the configuration does already contain a source with the same Id or the Source is already child of another configuration

E_POINTER

pSource was no valid pointer

E_ACCESSDENIED

The object is locked due to an asynchronous operation

6.7.1.2 ClearConfig

Reset to an empty configuration.

```
HRESULT ClearConfig();
```

Parameters

Return Values

S_OK

The configuration was cleared.

E_ACCESSDENIED

The object is locked due to an asynchronous operation

6.7.1.3 CopyTo

Create a copy of the current configuration.

```
HRESULT CopyTo([out, retval]  
IEvtConfiguration** ppConfiguration);
```

Parameters

ppConfiguration

[out, retval] pointer to a pointer to the target Configuration. If *ppConfiguration is NULL, a new Configuration will be created.

Return Values

S_OK

ppConfiguration contains a exact copy of the configuration.

E_POINTER

ppConfiguration was no valid pointer

E_ACCESSDENIED

The object is locked due to an asynchronous operation

Any CoCreateInstance() return value

6.7.1.4 EnumParsers

Query for an IEvtCfgEnumParser Object to enumerate through registered parsers.

```
HRESULT EnumParsers([out, retval]  
IDispatch** ppEnum);
```

Parameters

ppEnum

[out, retval] pointer to a pointer to store the returned enumeration object.

Return Values

S_OK

ppEnum contains a valid pointer to an IDispatch pointer, on which you may query for the IEvtCfgEnumParser interface.

E_ACCESSDENIED

The object is locked due to an asynchronous operation

6.7.1.5 GetConfigSize

Retrieve the size of the whole configuration (all Sources, Events, DocLinks)

```
HRESULT GetConfigSize([out] long*
nSources, [out] long* nEvents, [out] long* nDocLinks);
```

Parameters

nSources

[out] Pointer to a variable to store the number of sources. If null, the source count is not returned

nEvents

[out] Pointer to a variable to store the number of events. If null, the event count is not returned

nDocLinks

[out] Pointer to a variable to store the number of links. If null, the link count is not returned

Return Values

S_OK

The function completed successfully.

E_ACCESSDENIED

The object is locked due to an asynchronous operation

6.7.1.6 GetFirstSource

Get the first source in the configuration.

This method resets the internal iterator and calls [GetNextSource \[► 30\]](#)

To continue enumeration call [GetNextSource \[► 30\]](#)

```
HRESULT GetFirstSource([out,
retval] IEvtCfgSource** ppSource);
```

Parameters

[out, retval] pointer to a pointer to store the returned Source.

Return Values

S_OK

ppSource contains a valid pointer to an IEvtCfgSource object.

S_FALSE

The configuration does not contain any sources. *ppSource is set to NULL.

E_POINTER

ppSource was no valid pointer

E_ACCESSDENIED

The object is locked due to an asynchronous operation

6.7.1.7 GetNextSource

Get the next source in the configuration according to the internal iterator.

Before starting enumeration of sources call [GetFirstSource \[► 29\]](#)

After adding or removing Sources you must restart enumeration by calling [GetFirstSource \[► 29\]](#)

```
HRESULT GetNextSource([out,
retval] IEvtCfgSource** ppSource);
```

Parameters

[out, retval] pointer to a pointer to store the returned Source.

Return Values

S_OK

ppSource contains a valid pointer to an IEvtCfgSource object.

S_FALSE

The configuration does not contain more sources. *ppSource is set to NULL.

E_POINTER

ppSource was no valid pointer

E_ACCESSDENIED

The object is locked due to an asynchronous operation

6.7.1.8 GetSource

Get a source from the configuration.

The required source is identified by its Source Id. If the Source Id is not known, you can enumerate all sources in the configuration by using the [GetFistSource \[► 29\]](#) and [GetNextSource \[► 30\]](#) methods.

```
HRESULT GetSource([in] long nId,
[out,retval] IEvtCfgSource** ppSource);
```

Parameters

nId

[in] Id of the Source to be retrieved.

ppSource

[out, retval] pointer to a pointer to store the returned Source.

Return Values

S_OK

ppSource contains a valid pointer to an IEvtCfgSource object.

E_INVALIDARG

No source with the specified Id exists in the configuration

E_ACCESSDENIED

The object is locked due to an asynchronous operation

6.7.1.9 RemoveSource

Remove a source from the configuration.

The source to be removed is identified by its Id.

```
HRESULT RemoveSource([in] long  
nId);
```

Parameters

nId

[in] Id of the Source to be removed.

Return Values

S_OK

The specified source was removed from the configuration.

E_INVALIDARG

No source with the specified Id does not exist in the configuration

E_ACCESSDENIED

The object is locked due to an asynchronous operation

6.7.1.10 SourceCount

Get the number of sources in the configuration.

```
HRESULT SourceCount([out, retval]  
long *pVal);
```

Parameters

pVal

[out, retval] pointer to a variable to store the returned Value.

Return Values

S_OK

ppSource contains a valid pointer to an IEvtCfgSource object.

E_POINTER

pVal was no valid Pointer

E_ACCESSDENIED

The object is locked due to an asynchronous operation

6.7.2 IEvtCfgSource

The ITcEvtCfgSource Interfaces provides access to objects representing one source in the event configuration. Each source is identified by its unique Id and may contain a Name, describing the source, in multiple languages. If part of an configuration, a source object has exactly one parent object of type [IEvtConfiguration](#) [▶ 26]. One Source object can not be child of multiple configurations. A Source may -and mostly will- contain multiple children of type [ITcEvtCfgEvent](#) [▶ 41].

Table 3: Methods and Properties in Vtable Order

ITcEvtConfig Methods	Description
Name [▶ 39]	Get/Set the source name in a specified language
AddEvent [▶ 32]	Add an Event to this source
GetEvent [▶ 35]	Get an Event with a certain Id
RemoveEvent [▶ 40]	Remove an event
Id [▶ 37]	Get/Set the Source Id
GetFirstName [▶ 36]	Start enumerating through source names
GetNextName [▶ 37]	Continue enumerating source names
GetFirstEvent [▶ 35]	Start enumerating children
GetNextEvent [▶ 36]	Continue enumerating children
GetDefaultName [▶ 34]	Get a default name according to the local machine language
RemoveName [▶ 41]	Remove a name with a certain language
Parent [▶ 40]	Get/Set the parent of this source
IsParentOnlyReference [▶ 38]	Check whether the object is only more referenced by its parent
EventCount [▶ 33]	Get the number of events which are children of this source
CopyTo [▶ 33]	Copy the source to another EvtCfgSource object

6.7.2.1 AddEvent

Add a new event to this source.

Events are identified by their Ids- Duplicate IDs are not allowed. Also events may only have one parent. If the event is already child of onother source, the function will fail.

```
HRESULT AddEvent([in]
IEvtCfgEvent* pEvent);
```

Parameters

pEvent

[in] pointer to the new event object to be added.

Return Values

S_OK

The new event was successfully added. The OnNewEvent event is fired.

E_FAIL

pEvent could not be added to the configuration. This can happen either if the source does already contain an event with the same Id or the event is already child of another source

E_POINTER

pEvent was no valid pointer

E_ACCESSDENIED

The object is locked due to an asynchronous operation

Also see about this

 IEvtCfgSource [[▶ 32](#)]

6.7.2.2 CopyTo

Create a copy of the source object.

```
HRESULT CopyTo([out, retval]  
ITcEvtCfgSource** ppSource);
```

Parameters

ppSource

[out, retval] pointer to a pointer to the target source. If *ppSource is NULL, a new source will be created.

Return Values

S_OK

ppSource contains an exact copy of this source.

E_POINTER

ppSource was no valid pointer

E_ACCESSDENIED

The object is locked due to an asynchronous operation

Any CoCreateInstance() return value

Also see about this

 IEvtCfgSource [[▶ 32](#)]

6.7.2.3 EventCount

Get the number of events which are children of this source.

Property get

```
HRESULT EventCount([out, retval]  
long *pVal);
```

Parameters

pVal

[out, retval] pointer to a long variable that receives the number of children

Return Values

S_OK

Function was successfully called

E_POINTER

pVal was no valid pointer

E_ACCESSDENIED

The object is locked due to an asynchronous operation

Also see about this

 IEvtCfgSource [[▶ 32](#)]

6.7.2.4 GetDefaultName

Get an default name for this source.

It will check for a names in the following order and return the first 'hit'

1. Local machines language id
2. English (1031 / 9)
3. German (1033 / 7)
4. Lowest existing lcid

Property get

```
HRESULT Name([out, retval] BSTR  
*name);
```

Parameters

name

[out, retval] pointer to a BSTR variable to receive the source name

Return Values

S_OK

Function was successfully called

S_FALSE

No name is specified for this source. name is set to NULL

E_POINTER

name was no valid pointer

E_ACCESSDENIED

The object is locked due to an asynchronous operation

Also see about this

 IEvtCfgSource [[▶ 32](#)]

6.7.2.5 GetEvent

Get an event which is child of this source.

The event is identified by its Event Id. If the Event Id is not known, you can enumerate all children of this source through the `GetFistEvent` and `GetNextEvent` methods.

```
HRESULT GetEvent([in] long nId,  
[out,retval] IEvtCfgEvent** ppEvent);
```

Parameters

nId

[in] Id of the event to be retrieved.

ppSource

[out, retval] pointer to a pointer to store the returned event object.

Return Values

S_OK

ppSource contains a valid pointer to an `IEvtCfgSource` object.

E_INVALIDARG

No source with the specified Id exists in the configuration

E_POINTER

ppEvent was no valid pointer.

E_ACCESSDENIED

The object is locked due to an asynchronous operation

6.7.2.6 GetFirstEvent

Get the first name of the source.

This method resets the internal iterator and calls [GetNextEvent \[▶ 36\]](#)

To continue enumeration call [GetNextEvent \[▶ 36\]](#).

If the source is locked by an asynchronous operation, this method will not return `E_FAIL`, but wait until the lock is released.

```
HRESULT GetFirstEvent([out,retval]  
IEvtCfgEvent** ppEvent);
```

Parameters

ppEvent

[out, retval] pointer to a `IEvtCfgEvent` interface pointer to store the first child of this source.

Return Values

S_OK

name contains the first name of this source.

S_FALSE

The source does not have any childs. *ppEvent is set to NULL.

E_POINTER

ppEvent was no valid pointer

E_ACCESSDENIED

The object is locked due to an asynchronous operation

6.7.2.7 GetFirstName

Get the first name of the source.

This method resets the internal iterator and calls [GetNextName \[▶ 37\]](#)

To continue enumeration call [GetNextName \[▶ 37\]](#)

```
HRESULT GetFirstName([out] BSTR*
pName, [out,retval] long* pLangId);
```

Parameters

name

[out] pointer to a BSTR variable to store the first name associated with this source.

pLangId

[out, retval] pointer to a long variable to store the lang Id of the returned name.

Return Values**S_OK**

name contains the first name of this source.

S_FALSE

The source does not have any name. *name is set to NULL.

E_POINTER

name was no valid pointer

E_ACCESSDENIED

The object is locked due to an asynchronous operation

6.7.2.8 GetNextEvent

Get the next child of this source according to the internal iterator.

Before starting enumeration of events call [GetFirstEvent \[▶ 35\]](#)

After adding or removing events you must restart enumeration by calling [GetFirstEvent \[▶ 35\]](#).

If the source is locked by an asynchronous operation, this method will not return E_FAIL, but wait until the lock is released.

```
HRESULT GetNextEvent([out, retval]
IEvtCfgSEvent** ppEvent);
```

Parameters

ppEvent

[out, retval] pointer to a pointer to store the returned event.

Return Values

S_OK

ppEvent contains a valid pointer to an IEvtCfgSource object.

S_FALSE

The configuration does not contain more events. *ppEvent is set to NULL.

E_POINTER

ppEvent was no valid pointer

E_ACCESSDENIED

The object is locked due to an asynchronous operation

6.7.2.9 GetNextName

Get the next name of this source according to the internal iterator.

Before starting enumeration of source names call [GetFirstName \[► 36\]](#)

After adding or removing sources names you must restart enumeration by calling [GetFirstName \[► 36\]](#)

```
HRESULT GetNextName([out, retval]  
BSTR* name);
```

Parameters

name

[out] pointer to a BSTR variable to store the next name associated with this source.

pLangId

[out, retval] pointer to a long variable to store the lang Id of the returned name.

Return Values

S_OK

name contains the next name of this source.

S_FALSE

The source does not have further names. *name is set to NULL.

E_POINTER

name was no valid pointer

E_ACCESSDENIED

The object is locked due to an asynchronous operation

6.7.2.10 Id

Get or set the source Id.

If the source is already child of a configuration, it will be removed and added with the new Id. The SourceRemoved and NewSource events will be issued.

Property get

```
HRESULT Id([out, retval] long *pVal);
```

Parameters

pVal

[out, retval] pointer to a long variable that receives the Source Id

Return Values

S_OK

Function was successfully called

E_POINTER

pVal was no valid pointer

Property set

```
HRESULT Id([in] long newVal);
```

Parameters

newVal

[in] long variable holding the new source Id

Return Values

S_OK

The Source Id was successfully set to newVal

E_FAIL

The source is child of an EvtConfiguration object wich does already contain another source with the specified Id.

The Id remains unchanged.

E_ACCESSDENIED

The object is locked due to an asynchronous operation

6.7.2.11 IsParentOnlyReference

Check if the source is only more referenced by its parent.

```
HRESULT IsParentOnlyReference([out, retval] BOOL* pVal);
```

Parameters

pVal

[out, retval] pointer to a BOOL variable to store the return value.

Return Values

S_OK

Success.

S_FALSE

The source does not have have a parent. *pVal is False.

E_POINTER

pVal was no valid pointer

6.7.2.12 Name

Get or set the source name in one language.

Property get

```
HRESULT Name([in] long langId,  
[out, retval] BSTR *pVal);
```

Parameters

langId

[in] long value specifying the language in which to retrieve the source name

pVal

[out, retval] pointer to a BSTR variable that receives the source name in the specified language

Return Values

S_OK

Function was successfully called

E_FAIL

No name is specified in the questioned language

E_POINTER

pVal was no valid pointer

E_ACCESSDENIED

The object is locked due to an asynchronous operation

Property set

```
HRESULT Name([in] long langId,  
[in] BSTR *pVal);
```

Parameters

langId

[in] long value specifying the language in which to set the source name

pVal

[in] BSTR variable holding the source name for the specified language

Return Values

S_OK

Function was successfully called

E_FAIL

A name for the language Id does already exist

E_ACCESSDENIED

The object is locked due to an asynchronous operation

6.7.2.13 Parent

Get or set the parent of this source.

This property may be used to retrieve the parent of this source (if one). Setting the parent is for internal use only. Setting this value to inappropriate values will result in undefined states and side effects. To set the parent for a source use the [IEvtConfiguration::AddSource \[► 27\]](#) or [IEvtConfiguration::RemoveSource \[► 31\]](#) methods.

Property get

```
HRESULT Name([out, retval]  
IDispatch** ppVal);
```

Parameters

ppVal

[out, retval] pointer to a IDispatch* variable that receives the the parent of this source

Return Values

S_OK

Function was successfully called. If the source does have a parent, *ppVal is valid, otherwise it is set to NULL

E_POINTER

ppVal was no valid pointer

E_ACCESSDENIED

The object is locked due to an asynchronous operation

6.7.2.14 RemoveEvent

Remove an event from this source.

The event to be removed is identified by its Event Id.

```
HRESULT RemoveEvent([in] long nId);
```

Parameters

nId

[in] Id of the event to be removed.

Return Values

S_OK

The specified event was removed from the configuration.

E_INVALIDARG

No event with the specified Id does not exist in the configuration

E_ACCESSDENIED

The object is locked due to an asynchronous operation

Also see about this

 [IEvtCfgSource](#) [[▶](#) 32]

6.7.2.15 RemoveName

Delete a name for a certain language.

```
HRESULT RemoveName([in] long
nId);
```

Parameters

nId

[in] Lang Id of the name to be removed.

Return Values

S_OK

The specified name was removed.

E_INVALIDARG

No name with the specified Lang Id does not exist.

E_ACCESSDENIED

The object is locked due to an asynchronous operation

Also see about this

 [IEvtCfgSource](#) [[▶](#) 32]

6.7.3 IEvtCfgEvent

The IEvtCfgEvent Interface represents one event in the configuration. Through its methods the events may be created and modified.

Table 4: Methods and Properties in Vtable Order

IEvtCfgEvent Methods	Description
Id [▶ 46]	Get or set the Event Id
Class [▶ 42]	Get or Set the Event Class
Priority [▶ 49]	Get or set the Event Priority
MustConfirm [▶ 48]	Get or set weather the event must be confirmed
MsgString [▶ 48]	Get or set the Message String for a particular language
AddDocLink [▶ 42]	Add a Doc Link to the Event
RemoveDocLink [▶ 50]	Remove a Doc Link from the event
GetDocLink [▶ 44]	Get a Doc Link

IEvtCfgEvent Methods	Description
GetFirstMessage [▶ 44]	Start enumerating message strings
GetNextMessage [▶ 45]	Continue enumerating message strings
GetFirstFormatString [▶ 44]	Start enumerating format strings
GetNextFormatString [▶ 45]	Continue enumerating format strings
GetFirstDocLink [▶ 44]	Start enumerating DocLinks
GetNextDocLink [▶ 45]	Continue enumerating DocLinks
Parent [▶ 48]	Get or set the parent of the event
IsParentOnlyReference [▶ 46]	Check whether the event is only more referenced by its parent
DocLinkCount [▶ 43]	Number of Doc Links associated with this event
CopyTo [▶ 43]	Create a copy of the event object

6.7.3.1 AddDocLink

Add a new DocLink to this event.

DocLinks are identified by their Names. DocLinks may only have one parent. If the DocLink is already child of another event, the function will fail.

```
HRESULT AddDocLink([in] IEvtCfgDocLinkt* pLink);
```

Parameters

pLink

[in] pointer to the new DocLink object to be added.

Return Values

S_OK

The new DocLink was successfully added.

E_FAIL

pLink could not be added to the event. This can happen if the Link is already child of another event

E_POINTER

pLink was no valid pointer

E_ACCESSDENIED

The object is locked due to an asynchronous operation

6.7.3.2 Class

Get or set the Event Class.

Property get

```
HRESULT Class([out, retval] long *pVal);
```

Parameters

pVal

[out, retval] pointer to a long variable that receives the Event Class

Return Values

S_OK

Function was successfully called

E_POINTER

pVal was no valid pointer

Property set

```
HRESULT Class([in] long  
newVal);
```

Parameters

newVal

[in] long variable holding the new Event Class

Return Values

S_OK

The Event Class was successfully set to newVal

6.7.3.3 CopyTo

Create a copy of the event object.

```
HRESULT CopyTo([out, retval]  
ITcEvtCfgSource** ppEvent);
```

Parameters

ppEvent

[out, retval] pointer to a pointer to the target event. If *ppEvent is NULL, a new event will be created.

Return Values

S_OK

ppEvent contains a exact copy of this event.

E_POINTER

ppEvent was no valid pointer

E_ACCESSDENIED

The object is locked due to an asynchronous operation

Any CoCreateInstance() return value

6.7.3.4 DocLinkCount

Get the number of links which are children of this event.

Property get

```
HRESULT DocLinkCount([out, retval]  
long *pVal);
```

Parameters

pVal

[out, retval] pointer to a long variable that receives the numer of children

Return Values

S_OK

Function was successfully called

E_POINTER

pVal was no valid pointer

6.7.3.5 GetDocLink

Get the DocLink with the specified name.

```
HRESULT GetDocLink([in] BSTR name,  
[out,retval] IEvtCfgDocLink** ppLink);
```

Parameters

name

[in] name of the link to be retrieved.

ppLink

[out, retval] pointer to a pointer to store the returned DocLink object.

Return Values

S_OK

ppLink contains a valid pointer to an IEvtCfgSource object.

E_INVALIDARG

No link with the specified name is child of the event

E_POINTER

ppLink was no valid pointer.

6.7.3.6 GetFirstMessage

Get the first message string of the event.

This method resets the internal iterator and calls [GetNextMessage \[▶ 45\]](#)

To continue enumeration call [GetNextMessagee \[▶ 45\]](#)

```
HRESULT GetFirstMessage([out]  
BSTR* message, [out,retval] long* pLangId);
```

Parameters

message

[out] pointer to a BSTR variable to store the first message associated with this source.

pLangId

[out, retval] pointer to a long variable to store the lang Id of the returned message.

Return Values

S_OK

message contains the first message of this source.

S_FALSE

The source does not have any message. *message is set to NULL.

E_POINTER

message was no valid pointer

E_ACCESSDENIED

The object is locked due to an asynchronous operation

6.7.3.7 GetNextMessage

Get the next message of this event according to the internal iterator.

Before starting enumeration of message strings call [GetFirstMessage \[▶ 44\]](#)

After adding or removing message strings you must restart enumeration by calling [GetFirstMessage](#)

```
HRESULT GetNextMessage([out] BSTR*  
message, [out, retval] long* langId);
```

Parameters

message

[out] pointer to a BSTR variable to store the next message associated with this event.

langId

[out, retval] pointer to a long variable to store the lang Id of the returned message.

Return Values

S_OK

message contains the next name of this event.

S_FALSE

The event does not have further messages. *message is set to NULL.

E_POINTER

message was no valid pointer

E_ACCESSDENIED

The object is locked due to an asynchronous operation

6.7.3.8 Id

Get or set the event Id.

If the event is already child of a Source, it will be removed and added with the new Id. The EventRemoved and NewEvent events will be issued.

Property get

```
HRESULT Id([out, retval] long  
*pVal);
```

Parameters

pVal

[out, retval] pointer to a long variable that receives the Event Id

Return Values

S_OK

Function was successfully called

E_POINTER

pVal was no valid pointer

Property set

```
HRESULT Id([in] long newVal);
```

Parameters

newVal

[in] long variable holding the new Event Id

Return Values

S_OK

The Event Id was successfully set to newVal

E_FAIL

The event is child of an Source object which does already contain another event with the specified Id. The Id remains unchanged.

E_ACCESSDENIED

The object is locked due to an asynchronous operation

6.7.3.9 IsParentOnlyReference

Check if the event is only more referenced by its parent.

```
HRESULT IsParentOnlyReference([out,  
retval] BOOL* pVal);
```

Parameters

pVal

[out, retval] pointer to a BOOL variable to store the return value.

Return Values

S_OK

Success.

S_FALSE

The event does not have a parent. *pVal is False.

E_POINTER

pVal was no valid pointer

6.7.3.10 MsgString

Get or set the message string in a particular language.

Property get

```
HRESULT MsgString([in] long  
langId, [out, retval] BSTR *pVal);
```

Parameters

langId

[in] the language id for which the message string is to be retrieved

pVal

[out, retval] pointer to a BSTR variable that receives the string

Return Values

S_OK

Function was successfully called

E_POINTER

pVal was no valid pointer

E_FAIL

no message exists in the specified language

Property set

```
HRESULT MsgString([in] long  
langId, [in] BSTR newVal);
```

Parameters

langId

[in] the language id for which the message string is to be set

newVal

[in] BSTR variable holding the new value

Return Values

S_OK

The function completed successfully

E_FAIL

A message string in the specified language already exists

6.7.3.11 MustConfirm

Get or set the Events Priority.

Property get

```
HRESULT MustConfirm([out, retval]  
BOOL *pVal);
```

Parameters

pVal

[out, retval] pointer to a BOOL variable that receives the current state

Return Values

S_OK

Function was successfully called

E_POINTER

pVal was no valid pointer

Property set

```
HRESULT MustConfirm([in] BOOL  
newVal);
```

Parameters

newVal

[in] BOOL variable holding the new value

Return Values

S_OK

The function completed successfully

6.7.3.12 Parent

Get or set the parent of this source.

This property may be used to retrieve the parent of this event (if one). Setting the parent is for internal use only. Setting this value to inappropriate values will result in undefined states and side effects. To set the parent for a event use the [IEvtCfgSource::AddEvent \[▶ 32\]](#) or [IEvtCfgSource::RemoveEvent \[▶ 40\]](#) methods.

Property get

```
HRESULT Name([out, retval]  
IDispatch** ppVal);
```

Parameters

ppVal

[out, retval] pointer to a IDispatch* variable that receives the the parent of this event

Return Values

S_OK

Function was successfully called. If the event does have a parent, *ppVal is valid, otherwise it is set to NULL

E_POINTER

ppVal was no valid pointer

E_ACCESSDENIED

The object is locked due to an asynchronous operation

6.7.3.13 Priority

Get or set the Events Priority.

Property get

```
HRESULT Priority([out, retval]  
long *pVal);
```

Parameters

pVal

[out, retval] pointer to a long variable that receives the Event Priority

Return Values

S_OK

Function was successfully called

E_POINTER

pVal was no valid pointer

Property set

```
HRESULT Priority([in] long  
newVal);
```

Parameters

newVal

[in] long variable holding the new Event Priority

Return Values

S_OK

The Event Priority was successfully set to newVal

6.7.3.14 RemoveDocLink

ITcEvtCfgEvent

Remove a DocLink from this event.

```
HRESULT RemoveDocLink([in]
IEvtCfgDocLink* pLink);
```

Parameters

pLink

[in] the DocLink to be removed.

Return Values

S_OK

The specified link was removed from the configuration.

E_INVALIDARG

the link is not child of this event

E_POINTER

pLink was no valid pointer

E_ACCESSDENIED

The object is locked due to an asynchronous operation

6.7.4 IEvtCfgDocLink

The IEvtCfgDocLink Interface provides access to the DocLinks properties.

DocLinks may be used to store additional Information concerning an event. A DocLink consists of a name, uniquely identifying it, and multiple links each for a certain language.

Table 5: Methods and Properties in Vtable Order

IEvtCfgDocLink Methods	Description
Name [▶ 53]	Get or set the DocLinks name
Link [▶ 52]	Get or set a Link in a particular language
GetFirstLink [▶ 51]	Start enumerating links
GetNextLink [▶ 51]	Continue enumerating links
RemoveLink [▶ 54]	Remove a link in a certain Language
Parent [▶ 54]	Get or set the parent of the DocLink
IsParentOnlyReference [▶ 52]	Check if the DocLink is only more referenced by its parent
CopyTo [▶ 50]	Create a copy of this DocLink object

6.7.4.1 CopyTo

Create a copy of the DocLink object.

```
HRESULT CopyTo([out, retval]
ITcEvtCfgDocLink** ppLink);
```

Parameters

ppLink

[out, retval] pointer to a pointer to the target link. If *ppLink is NULL, a new DocLink will be created.

Return Values

S_OK

ppLink contains a exact copy of this link.

E_POINTER

ppLink was no valid pointer

6.7.4.2 GetFirstLink

Get the first link from this object.

This method resets the internal iterator and calls [GetNextLink \[► 51\]](#)

To continue enumeration call [GetNextLink](#)

```
HRESULT GetFirstLink([out] BSTR*  
name, [out,retval] long* pLangId);
```

Parameters

name

[out] pointer to a BSTR variable to store the first URL associated with this doclink.

pLangId

[out, retval] pointer to a long variable to store the lang Id of the returned URL.

Return Values

S_OK

name contains the first URL of this doclink.

S_FALSE

The doclink does not have any URL. *name is set to NULL.

E_POINTER

name was no valid pointer

6.7.4.3 GetNextLink

Get the next link according to the internal iterator.

Before starting enumeration of links call [GetFirstLink \[► 51\]](#)

After adding or removing links you must restart enumeration by calling [GetFirstLink](#)

```
HRESULT GetNextLink([out] BSTR*  
link, [out, retval] long* langId);
```

Parameters

link

[out] pointer to a BSTR variable to store the next link.

langId

[out, retval] pointer to a long variable to store the lang Id of the returned link.

Return Values

S_OK

link contains the next link.

S_FALSE

The DocLink does not have further links. *link is set to NULL.

E_POINTER

link was no valid pointer

6.7.4.4 IsParentOnlyReference

Check if the DocLink is only more referenced by its parent.

```
HRESULT IsParentOnlyReference([out,  
retval] BOOL* pVal);
```

Parameters

pVal

[out, retval] pointer to a BOOL variable to store the return value.

Return Values

S_OK

Success.

S_FALSE

The DocLink does not have have a parent. *pVal is False.

E_POINTER

pVal was no valid pointer

6.7.4.5 Link

Get or set the link for a particular language.

Property get

```
HRESULT Link([in] long langId,  
[out, retval] BSTR *pVal);
```

Parameters

langId

[in] the language id for which the link is to be retrieved

pVal

[out, retval] pointer to a BSTR variable that receives the link

Return Values

S_OK

Function was successfully called

E_POINTER

pVal was no valid pointer

E_FAIL

no link exists in the specified language

Property set

```
HRESULT Link([in] long langId,  
[in] BSTR newVal);
```

Parameters

langId

[in] the language id for which the link is to be set

newVal

[in] BSTR variable holding the new value

Return Values

S_OK

The function completed successfully

E_FAIL

A link in the specified language already exists

6.7.4.6 Name

Get or set the name of the DocLink.

Property get

```
HRESULT Name([out, retval] BSTR  
*pVal);
```

Parameters

pVal

[out, retval] pointer to a BSTR variable that receives the name

Return Values

S_OK

Function was successfully called

E_POINTER

pVal was no valid pointer

Property set

```
HRESULT Name([in] BSTR *pVal);
```

Parameters

pVal

[in] BSTR variable holding the new name

Return Values

S_OK

Function was successfully called

6.7.4.7 Parent

Get or set the parent of this source.

This property may be used to retrieve the parent of this DocLink (if one). Setting the parent is for internal use only. Setting this value to inappropriate values will result in undefined states and side effects. To set the parent for a event use the [IEvtCfgDocLink::AddDocLink \[► 42\]](#) or [IEvtCfgIEvtCfgDocLink::RemoveDocLink \[► 50\]](#) methods.

Property get

```
HRESULT Name([out, retval]  
IDispatch** ppVal);
```

Parameters

ppVal

[out, retval] pointer to a IDispatch* variable that receives the the parent of this DocLink. The parent will always be of the type [IEvtCfgEvent \[► 41\]](#)

Return Values

S_OK

Function was successfully called. If the event does have a parent, *ppVal is valid, otherwise it is set to NULL

E_POINTER

ppVal was no valid pointer

6.7.4.8 RemoveLink

Remove a link from this DocLink object.

```
HRESULT RemoveLink([in] long  
langId);
```

Parameters

langId

[in] the langId of the link to be removed.

Return Values

S_OK

The specified link was removed.

E_INVALIDARG

No link with the specified lang Id does exist.

6.7.5 IEvtCfgEnumParser

The IEvtCfgEnumParser Interface is used to enumerate all registered parsers. EvtCfgParser objects are created through the [IEvtConfiguration::EnumParsers \[▶ 28\]](#) method

Table 6: Methods and Properties in Vtable Order

IEvtCfgEnumParser Methods	Description
Next	Enum next X parser objects
Reset	Reset the enumerator
Skip	Skip the next X parsers
VbNext	VB and .NET compatible Next() method

6.7.6 IEvtCfgParser

With the TcEventConfig library a set of standard parsers are shipped. Amon these are e.g. the Xml and Tps parsers.

Each of these parsers implement the IEvtCfgParser Interface.

Custom parsers can be added to the system at will. Required is only the implementation of this interface and a simple registration.

To build your own parser see the c++ sample.

The following interface description shows, how the EvtCfgParser Interface is defined and how each implementation has to react on function calls.

See a list of supplied parsers below, to check for their specific implementation details.

Table 7: IEvtCfgParser Methods and Properties in Vtable Order

IEvtCfgParser Methods	Description
OpenConfiguration [▶ 60]	Open an existing configuration file
Import [▶ 59]	Import from an opened file
Export [▶ 58]	Export to an opened file
CloseConfiguration [▶ 57]	Close a configuration file
DefFileExt [▶ 57]	Get the default file extension for the implemented import format
DefFileType [▶ 58]	Get a short description of the filetype for the implemented format
CreateConfiguration [▶ 57]	Create and open a new configuration file
ActivateConfiguration [▶ 56]	Set the opened configuration as current active configuration
GetConfigSize [▶ 59]	Get the Size of an opened configuration file
AsyncImport [▶ 56]	Asynchronously imports a configuration
AsyncExport [▶ 56]	Asynchronously exports a configuration

Table 8: _IEvtCfgParserEvents Methods and Properties in Vtable Order

_IEvtCfgParserEvents Methods	Description
SourceLoaded	A source has been loaded
SourceSaved	A source has been saved

_IEvtCfgParserEvents Methods	Description
EventLoaded	An event has been loaded
EventSaved	An event has been saved
LinkLoaded	A link has been loaded
LinkSaved	A link has been saved
LoadComplete	Loading has completed
SaveComplete	Saving has completed

Table 9: Parser implementations

Parser	Description
EvtCfgTpsParser [► 60]	Import/Export to TwinCAT storage.
EvtCfgXMLParser [► 61]	Import/Export to XML file format.

6.7.6.1 ActivateConfiguration

Activate the configuration for use with a EventFormatter.
This behavior (if any) is absolutely parser specific.

```
HRESULT ActivateConfiguration();
```

Parameters

Return Values

S_OK

The configuration has been activated.

E_ACCESSDENIED

The parser is locked by an asynchronous operation.

E_FAIL

The configuration could not be activated.

6.7.6.2 AsyncExport

Asynchronous alternative to [Export \[► 58\]\(\)](#).

The function will return immediately and you have to rely on [_IEvtCfgParser \[► 55\]](#) events to check progress.

6.7.6.3 AsyncImport

ITcEvtCfgParser

Asynchronous alternative to [Import \[► 59\]\(\)](#).

The function will return immediately and you have to rely on [_IEvtCfgParser](#) events to check progress.

Also see about this

 [IEvtCfgParser \[► 55\]](#)

6.7.6.4 CloseConfiguration

Close a previous opened configuration file.

```
HRESULT CloseConfiguration();
```

Parameters

Return Values

S_OK

The configuration is closed.

E_ACCESSDENIED

The parser is locked by an asynchronous operation.

6.7.6.5 CreateConfiguration

Create a new configuration file and open it.

The specified file may not already exist.

```
HRESULT CreateConfiguration([in]  
BSTR filename);
```

Parameters

filename

[in] Path and filename.

Return Values

S_OK

The configuration has been created and opened.

E_ACCESSDENIED

The parser is locked by an asynchronous operation.

E_FAIL

The configuration could not be created.

6.7.6.6 DefFileExt

Get the default file extension for use with one parser implementation.

Property get

```
HRESULT DefFileExt([out, retval]  
BSTR *pVal);
```

Parameters

pVal

[out, retval] pointer to a BSTR variable that receives the file extension

Return Values

S_OK

Function was successfully called

E_POINTER

pVal was no valid pointer

6.7.6.7 DefFileType

Get a short textual description on the file type used by the parser implementation.

Property get

```
HRESULT DefFileType([out, retval]  
BSTR *pVal);
```

Parameters

pVal

[out, retval] pointer to a BSTR variable that receives the file description

Return Values

S_OK

Function was successfully called

E_POINTER

pVal was no valid pointer

6.7.6.8 Export

Export from the supplied [IEvtConfiguration](#) [► 26] object into the opened configuration file..

```
HRESULT Export([in]  
IEvtConfiguration* pConfig);
```

Parameters

pConfig

[in] Source configuration.

Return Values

S_OK

The configuration has been exported from pConfig.

E_ACCESSDENIED

The parser is locked by an asynchronous operation.

E_FAIL

The configuration could not be exported.

6.7.6.9 GetConfigSize

Get the number of Sources, Events and DocLinks in a configuration file.

This does only work for opened files. To get the size of a configuration before exporting it, call [IEvtConfiguration::GetConfigSize\(\)](#) [► 29].

```
HRESULT GetConfigSize([out] long* nSources, [out] long* nEvents, [out] long* nLinks);
```

Parameters

nSources

[out] Number of Sources in the configuration

nEvents

[out] Number of Events in the configuration

nLinks

[out] Number of DocLinks in the configuration

Return Values

S_OK

Success.

E_ACCESSDENIED

The parser is locked by an asynchronous operation.

E_FAIL

Error.

Remarks

Returned values may not be exact, as parsers may guess the size for performance reasons or rely on meta data previously added to the file.

For example the XML and Tps parsers will return 0, when opening a 'HandMade' file the first time. The method will only work for files, saved with the Parser.

6.7.6.10 Import

Import from the currently opened configuration file into the supplied [IEvtConfiguration](#) [► 26] object..

```
HRESULT Import([in] IEvtConfiguration* pConfig);
```

Parameters

pConfig

[in] Target configuration.

Return Values

S_OK

The configuration has been imported into pConfig.

E_ACCESSDENIED

The parser is locked by an asynchronous operation.

E_FAIL

The configuration could not be imported.

6.7.6.11 OpenConfiguration

Open a configuration file. All subsequent calls will from now on refer to this file.

When processing has finished, call [CloseConfiguration](#) [► 57].

```
HRESULT OpenConfiguration([in]BSTR name);
```

Parameters

name

[in] Path and file to be opened.

Return Values

S_OK

The configuration is opened.

E_ACCESSDENIED

The parser is locked by an asynchronous operation.

E_FAIL

The file could not be opened.

6.7.7 EvtCfgTpsParser

The EvtCfgTpsParser does work on the TwinCAT storage file format.

The default storage file is 'TwinCAT\default.tps'.

To have your event messages fetched from the TwinCAT storage, you need to choose the TcEventFormatter when firing events from the PLC.

Do so by specifying 'TcEventLogger.TcEventFormatter' as progId in the TcEvent structure.

When calling ActivateConfiguration, the parser will set the opened configuration file as TwinCATs current project.

If you create a new configuration and register it this will have two side effects:

- The EventLoggers list of logged events is lost.
- Additional TwinCAT servers will no more be registered.

When you activate the previous configuration, these side effects will resolve again.

Changing the active configuration will first take effect after restarting TwinCAT.

6.7.8 EvtCfgXmlParser

The EvtCfgTpsParser does work on the XML file format.
 Each Source and its events will be saved in an own file.
 All available source Ids are stored in your project file which links to each source description file.

To have your event messages fetched from an XML description, you need to choose the TcXMLFormatter when firing events from the PLC.
 Do so by specifying 'TcEventFormatter.TcXmlFormatter' as progId in the TcEvent structure.

When calling ActivateConfiguration, the parser will add your project info to the TcEventSourceLocation.xml file (Located in TwinCAT\Resource). Any information previously stored in this file will remain unchanged- new entries made by the parser will be marked so. Only these marked entries will be removed when you activate another configuration.

6.7.9 IEvtCfgEditor

The IEvtCfgEditor Interface provides access to the EventEditors properties.

Table 10: Methods and Properties in Vtable Order

IEvtCfgEditor Methods	Description
GetConfiguration [▶ 63]	Get the EventConfiguration [▶ 26]
SelectItem [▶ 64]	Select one item in the TreeView
GetSelectedItem [▶ 63]	Get the object associated with the selected TreeView item
ExpandSelection [▶ 63]	Expand the selected item
CollapseSelection [▶ 62]	Collapse the selected item
Copy [▶ 62]	Copy the selected item
Paste [▶ 64]	Paste
CanCopy [▶ 61]	Get or set whether copy is possible
CanPaste [▶ 62]	Get or set whether paste is possible
_IEvtCfgEditorEvents Methods	Description
SelChanged	The selected item changed.

Table 11: Enums

Name	Description
EVT_CFG_EDITOR_ITEMTYPES [▶ 65]	Determine which item is selected in the tree.

6.7.9.1 CanCopy

Check whether the current selection can be copied.

```
HRESULT CanCopy([out, retval] BOOL
*pVal);
```

Parameters

pVal

[out, retval] TRUE if a copy operation is possible, otherwise FALSE.

Return Values

S_OK

Success.

6.7.9.2 CanPaste

Check whether pasting is possible.

```
HRESULT CanPaste([out, retval]  
BOOL *pVal);
```

Parameters

pVal

[out, retval] TRUE if pasting is possible, otherwise FALSE.

Return Values

S_OK

Success.

6.7.9.3 CollapseSelection

Collapse the tree views currently selected item.

```
HRESULT CollapseSelection();
```

Parameters**Return Values**

S_OK

The item is collapsed.

6.7.9.4 Copy

Copy the selected item.

The data of the copy operation is not stored System wide- It is lost after unloading the editor and cannot be shared among multiple instances.

```
HRESULT Copy();
```

Parameters**Return Values**

E_FAIL

The current selection cannot be copied. Call [CanCopy \[▶ 61\]](#) to check whether the selection can be copied

S_OK

The item is copied.

6.7.9.5 ExpandSelection

Expand the treeviews currently selected item.

```
HRESULT ExpandSelection();
```

Parameters

Return Values

S_OK

The item is expanded.

6.7.9.6 GetConfiguration

Get the underlying [configuration \[▶ 26\]](#) object.

```
HRESULT GetConfiguration([out,  
retval] IEvtConfiguration** ppConfig);
```

Parameters

ppConfig

[out, retval] pointer to a pointer to the underlying configuration.

Return Values

S_OK

*ppConfig contains a valid pointer.

E_FAIL

The editor was not initialized properly and does not hold a configuration object.

E_POINTER

ppConfig was no valid pointer

6.7.9.7 GetSelectedItem

Get the current selection in the tree View.

The item is identified by the type of COM object returned (can be [IEvtCfgSource \[▶ 32\]](#), [IEvtCfgEvent \[▶ 41\]](#) or [IEvtCfgDocLink \[▶ 50\]](#)).

Subitems are identified by Enum [EVTFCGEDITOR_ITEMTYPES \[▶ 65\]](#) and a language Id.

```
HRESULT GetSelectedItem([out]  
IDispatch** ppItem, [out] EVTFCGEDITOR_ITEMTYPES* subItem, [out]  
LONG* langId);
```

Parameters

ppItem

[out] Pointer to IDispatch pointer. It contains a pointer to one of the interfaces mentioned above

subItem

[out] Selected subitem. If TYPE_NULL, the item itself is selected. See the [EVTCFGEDITOR_ITEMTYPES \[► 65\]](#) for more details.

If subItem is TYPE_SRCNAME, TYPE_EVTMSG or TYPE_URL, langId contains valid data.

langId

[out] Language identifier of the selected item.

Return Values

S_OK

Function completed successfully.

E_POINTER

pItem was no valid pointer.

6.7.9.8 Paste

Paste data as child of the selected item. If the item copied cannot be pasted as child of the currently selected item, the function will fail.

```
HRESULT Paste();
```

Parameters

Return Values

E_FAIL

The item cannot be pasted as child of the current selection. Call [CanPaste \[► 62\]](#) to check whether pasting is possible

S_OK

The item is pasted.

6.7.9.9 SelectItem

Select an item in the tree View.

The item is identified by the type of COM object supplied (can be [IEvtCfgSource \[► 32\]](#), [IEvtCfgEvent \[► 41\]](#) or [IEvtCfgDocLink \[► 50\]](#)).

If necessary, subitems are identified by enum [EVTCFGEDITOR_ITEMTYPES \[► 65\]](#) and a language Id.

```
HRESULT SelectItem([in] IDispatch* pItem,
                  [in, defaultValue(TYPE_NULL)] EVTCFGEDITOR_ITEMTYPES subItem,
                  [in, defaultValue(0)] LONG langId
);
```

Parameters

pItem

[in] IDispatch pointer which contains a pointer to one of the interfaces mentioned above

subItem

[in] Subitem to be selected. If TYPE_NULL, the item itself is selected. See the [EVTCFGEDITOR_ITEMTYPES \[▶ 65\]](#) for more details.

If subItem is TYPE_SRCNAME, TYPE_EVTMSG or TYPE_URL, a language Id for the item to be selected must be specified.

langId

[in] Language identifier of the item to be selected.

Return Values

S_OK

The item was selected as requested.

E_FAIL

The requested Item could not be selected, e.g. because the subitem does not match the supplied item type, or an item could not be found in the specified language.

E_POINTER

pltem was no valid pointer








Also see about this




 IEvtCfgEditor [▶ 61]

6.7.9.10 EVTCFGEDITOR_ITEMTYPES

This enum is used to determine which item is selected in the tree.

```
typedef enum {
    TYPE_NULL = 0,
    TYPE_SRC,
    TYPE_SRCNAMETAG,
    TYPE_SRCNAME,
    TYPE_EVTTAG,
    TYPE_EVT,
    TYPE_EVTMSG,
    TYPE_EVTMSGTAG,
    TYPE_LNKTAG,
    TYPE_LNK,
    TYPE_URL
}EVTCFGEDITOR_ITEMTYPES;
```

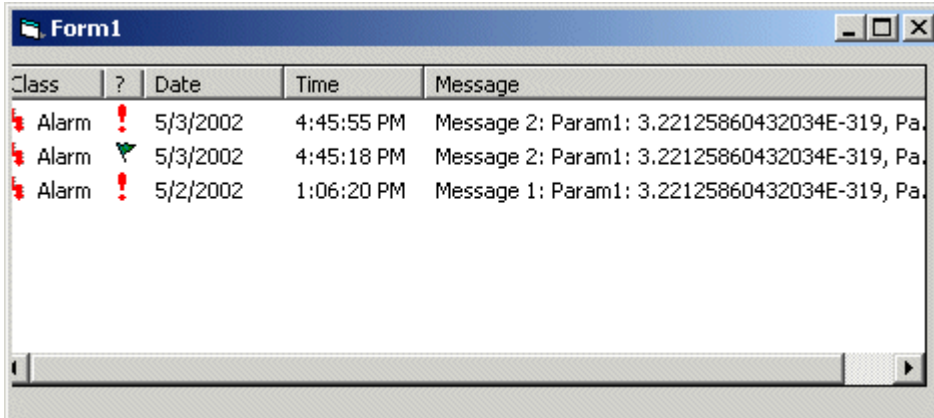
Value	Description	Editor Icon
TYPE_NULL	Undefined type, most likely indicating an error	
TYPE_SRC	A Source item	
TYPE_SRCNAMETAG	The Names tag beneath TYPE_SRC	
TYPE_SRCNAME	A sourcename - child of TYPE_SRCNAMETAG	
TYPE_EVTTAG	The event tag beneath TYPE_SRC	
TYPE_EVT	One Event	
TYPE_EVTMSGTAG	The messages tag beneath TYPE_EVT	
TYPE_EVTMSG	An event message string - child of TYPE_EVT	

Value	Description	Editor Icon
TYPE_LNKTAG	The DocLink tag beneath TYPE_EVT	
TYPE_LNK	A Doc Link Item	
TYPE_URL	An url - child of TYPE_LNK	

7 Event HMI

The TcEventViewer.dll provides the Beckhoff TcEventViewer. The EventViewer is an ActiveX control that is the HMI building Block of the TcEvent Logger. By Just placing it on the form and configure some properties it can be used to implement in less than 5 minutes the visible part of the TcEvent Logger.

The next screenshot shows a Visual Basic program using the TcEventLogger to display logged alarms.



The TcEventViewer can display active and logged alarms. All columns of the TcEventViewer can be disabled and changed in the size from the code.

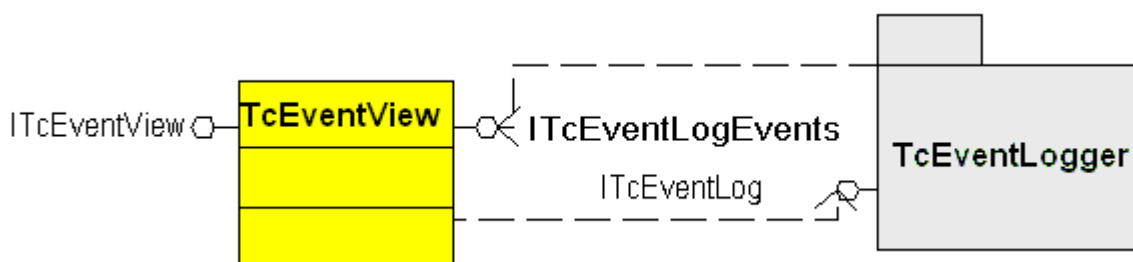
All strings of the events are displayed in the selected language since the language switching is done by the Formatter.

Since version 2.9.0.56, the TcEventviewer is divided into the TcEventviewer and TcEventviewerLight. The Light version of the TcEventviewer is minimized in code and maximized in performance

Interfaces

The TcEventViewer communicates to other components through it's interfaces.

The next figure shows an overview of the main interface of the TcEventViewer:



Interface	Description
ITcEventView [▶ 68]	Main interface of the TcEventViewer to COM clients like the HMI. Main features are to control the appearances of the ActiveX control.
ITcEventViewLight [▶ 112]	Interface of the EventViewers light version
ITcEventLogEvents [▶ 144]	The Event interface is provided by the TcEventLogger. When the TcEventViewer is started it gets a list of logged or active events by calling the methods ITcEventLog [▶ 129]::EnumActiveEvents [▶ 132] or ITcEventLog [▶ 129]::EnumLoggedEvents [▶ 133] . After that the TcEventLogger will notify the TcEventViewer using the ITcEventLogEvents interface. This notifications are used to keep the list of displayed events in sync with the server.

Classes

The TcEventViewer provides just one class to the outside: **TcEventView**

Client Server

The TcEventLogger provides a client server architecture. The TcEventLogger is the server that contains all informations about active an logged events. The TcEventViewer is a client of the TcEventLogger. When the EventViewer is started it first gets the list of active or logged events, displays them and synchronizes the list by the events fired to the [ITcEventLogEvents \[▶ 144\]](#) interface.

The TcEventViewer can be used in 3 different Client Server scenarios.

1. One TcEventViewer displays the events of one TcEventLogger. In this scenario the HMI with the TcEventViewer and the TcEventLogger are located on the same machine.
2. One TcEventViewer displays the alarms of several TcEventLoggers located on several machines. The remote connection is done by using *DCOM.
3. Several TcEventViewers display the events of one TcEventLogger. The remote connection is established using *DCOM.

*DCOM DCOM has to be configured correctly on the client and server machine by using dcomcnfg.exe. For more information about DCOM configuration see MSDN library. Due to DCOM has very long timeouts DCOM should only be used on a very stable network connection. Another problem is that the EventLogger is not multithreaded; the TcEventViewer may block the TcEventLogger.

Also see about this

 [TcEventLog \[▶ 128\]](#)

7.1 Classes

7.1.1 TcEventView

The main class of the TcEventViewer. Provides interfaces to the COM Client like the HMI. This class displays the alarm in a list view.

The class can be controlled through the following interfaces.

Interface	Description
ITcEventView [▶ 68]	Main interface of the TcEventViewer to COM clients like the HMI. Main features are to control the appearances of the ActiveX control.
ITcEventLogEvents [▶ 144]	The Event interface provided to the TcEventLogger. When the TcEventViewer is started it gets a list of logged or active events by calling the methods ITcEventLog [▶ 129]::EnumActiveEvents [▶ 132] or ITcEventLog [▶ 129]::EnumLoggedEvents [▶ 133] . After that the TcEventLogger will notify the TcEventViewer through ITcEventLogEvents interface. This notification is used to keep the list of displayed events in sync with the server.

7.2 Interfaces

7.2.1 ITcEventView

The interface ITcEventView is used to control the apperance of the TcEventVierwer ActiveX control. The interface derives from **IDispatch** so that it can be used by all languages that support COM automation interfaces. IDispatch itself derives from IUnknown.

Table 12: Methods and Properties in Vtable Order

IUnknown Methods	Description
QueryInterface	Returns a pointer to the queried interface
AddRef	Increments the reference counter
Release	Decrements the reference counter

IDispatch Methods	Description
GetIDsOfNames	Maps a single member name and an optional set of parameter names to a corresponding set of integer dispatch identifiers (DISPIDs), which can be used on subsequent calls to IDispatch::Invoke .
GetTypeInfo	Retrieves the type information of an object.
GetTypeInfoCount	Retrieves the number of type information interfaces that an object provides, either 0 or 1.
Invoke	Provides access to properties and methods exposed by an object.

ITcEventView Methods and Properties	Description
ShowLoggedEvents [▶ 70]	displays the logged events
ShowActiveEvents [▶ 71]	displays the active events
ConfirmEvent [▶ 71]	confirms one event
ResetEvent [▶ 72]	resets one event
LangId [▶ 72]	sets and returns the language that's used for displaying event message and the source name
DisableMenuItems [▶ 73]	returns or sets the disable state of context menu
SelectEvent [▶ 76]	marks one event as selected
AddConnection [▶ 77]	adds a remote connection to another EventLogger.
DeleteConnection [▶ 78]	deletes a remote connection to another EventLogger.
DisplayComputerName [▶ 78]	enables or disables the column that displays the computer name or returns the state.
DisplayEventId [▶ 79]	enables or disables the column that displays the event id or returns the state.
AlarmTextColor [▶ 82]	sets or returns the color that is used to display alarms
WarningTextColor [▶ 84]	sets or returns the color that is used to display warnings
InstructionTextColor [▶ 85]	sets or returns the color that is used to display introductions
ParamErrorTextColor [▶ 86]	sets or returns the color that is used to display parameter errors
StateInfoTextColor [▶ 87]	sets or returns the color that is used to display state information's
HintTextColor [▶ 88]	sets or returns the color that is used to display hints
MessageTextColor [▶ 89]	sets or returns the color that is used to display messages
MaintenanceTextColor [▶ 89]	sets or returns the color that is used to display maintenance information's
UndefTextColor [▶ 90]	sets or returns the color that is used to display undefined events
ColWidthClass [▶ 91]	sets or returns the width of the columns event class
ColWidthMustCon [▶ 92]	sets or returns the width of the column confirmation state
ColWidthState [▶ 93]	sets or returns the width of the columns event state

ITcEventView Methods and Properties	Description
ColWidthSource [▶ 94]	sets or returns the width of the columns event source
ColWidthDate [▶ 95]	sets or returns the width of the columns event date
ColWidthTime [▶ 96]	sets or returns the width of the columns event time
ColWidthEventId [▶ 96]	sets or returns the width of the columns event id
ColWidthComputer [▶ 97]	sets or returns the width of the column computer name
ColWidthMsg [▶ 98]	sets or returns the width of the columns event message
DisplayClass [▶ 99]	enables or disables the column that displays the event class or returns the state.
DisplayMustCon [▶ 100]	enables or disables the column that displays the confirmation state or returns the state.
DisplayState [▶ 101]	enables or disables the column that displays the event state or returns the state.
DisplaySource [▶ 102]	enables or disables the column that displays the source name or returns the state.
DisplayDate [▶ 103]	enables or disables the column that displays the event date or returns the state.
DisplayTime [▶ 104]	enables or disables the column that displays the event time or returns the state.
DisplayMsg [▶ 104]	enables or disables the column that displays the event message or returns the state.
GetSelectedEvent [▶ 105]	returns index of the event that is selected by the user
Mode [▶ 107]	sets or returns the display mode of the TcEventViewer
MaxEvents [▶ 108]	sets or returns the maximum number of displayed events.
DisplayTitlebar [▶ 80]	Enables or disables the column header.
DisplayUserComment [▶ 81]	Enables or disables the column that displays the userComment or returns the state.
DisableContextMenu [▶ 74]	Enables or disables the ContextMenu.
DisableScrollbars [▶ 75]	Enables or disables the Scrollbars.
SortMode [▶ 110]	Gets or sets the sort order and mode.
putFont [▶ 109]	Sets the font for the control.
getExtendedInfo [▶ 106]	Gets extended information about an event.
UseExternalLanguageResource [▶ 111]	Gets or sets whether an external language resource is used.
BackgroundColor [▶ 83]	Gets or sets the background color for the control.
ModifyLVStyle [▶ 84]	Delegate calls to the CWindow::ModifyStyle method.
SetFilter [▶ 109]	Method to set up a display filter

7.2.1.1 ShowLoggedEvents

[ITcEventView \[▶ 68\]](#)

This method displays all logged events of all EventLogger's the TcEventView is connected to. Before it displays the event it deletes all old events.

```
HRESULT ShowLoggedEvents();
```

Parameters

Return Values

S_OK

Function was successfully called

Visual Basic sample code

```
Option Explicit

' add the Beckhoff TcEvent View Library to the components
' place a TcEventView on the form, and assign the name TcEventView1

Private Sub Form_Load()
    TcEventView1.ShowLoggedEvents
End Sub
```

7.2.1.2 ShowActiveEvents

[ITcEventView |▶ 68|](#)

This method displays all active events of all EventLogger's the TcEventView is connected to. Before it displays the event it deletes all old events.

```
HRESULT ShowActivatedEvents();
```

Parameters

Return Values

S_OK

Function was successfully called

Visual Basic sample code

```
Option Explicit

' add the Beckhoff TcEvent View Library to the components
' place a TcEventView on the form, and assign the name TcEventView1

Private Sub Form_Load()
    TcEventView1.ShowActiveEvents
End Sub
```

7.2.1.3 ConfirmEvent

[ITcEventView |▶ 68|](#)

This method confirms one displayed event by the given index in the list view and confirmation code.

```
HRESULT ConfirmEvent(long nIndex, long nCode);
```

Parameters

nIndex

[in] the index of the event that should be confirmed. The index in the list view starts at the top with the index 0.

nCode

[in] The confirmation code that is described through the Enum TcEventConCodes

Return Values

S_OK

Function was successfully called

Visual Basic sample code

```
Option Explicit

' add the Beckhoff TcEvent View Library to the components
' place a TcEventView on the form, and assign the name TcEventView1

Private Sub Form_Load()
    Call TcEventView1.ConfirmEvent(0, 0)
End Sub
```

7.2.1.4 ResetEvent

[ITcEventView](#) | [▶ 68](#)

This method resets one displayed event by the given index in the list view.

```
HRESULT ResetEvent(long
nIndex);
```

Parameters

nIndex

[in] the index of the event that should be reset. The index in the list view starts at the top with the index 0.

Return Values

S_OK

Function was successfully called

Visual Basic sample code

```
Option Explicit

' add the Beckhoff TcEvent View Library to the components
' place a TcEventView on the form, and assign the name TcEventView1

Private Sub Form_Load()
    Call TcEventView1.ResetEvent(0)
End Sub
```

7.2.1.5 LangId

[ITcEventView](#) | [▶ 68](#)

This property sets and returns the language that is used for displaying the viewer, event messages and source names.

You may either supply only the primary language id (e.g. ENG = 9; GER = 7) or a complete LCID (e.g. ENG = 1033; GER = 1031). The TcEventViewer will then decide what the parameter is. Setting the language id will have two effects: on the one hand it will switch its internal string table to the specified language, on the other hand it will delegate the supplied value to the TcEventFormatter. It is important to understand that the supplied values will only be delegated to the formatter. If you specify Lang id as 7 (GER) the default formatters will lookup the string '7'. If in your resource file german is defined as '1031', the resource will not be found - and vice versa.

Property get

```
HRESULT get_LangId([out, retval] long *pVal);
```

Parameters

pVal

[out, retval] pointer to a long variable that receives the actual selected language id

Return Values

S_OK

Function was successfully called

E_POINTER

pVal was no valid pointer

Property let

```
HRESULT put_LangId([in] long newVal);
```

Parameters

newVal

[in] The language Id of the requested language. The requested language should be tagged by *LCIDs. The next table shows a sample for some language ids. In the configuration of the event formatter the language are tagged by the same language id

LCID	Description
1031	German
1033	US English
1034	Spanish - Spain
1036	French

Return Values

S_OK

Function was successfully called

Visual Basic sample code

```
Option Explicit
' add the Beckhoff TcEvent View Library to the components
' place a TcEventView on the form, and assign the name TcEventView1

Private Sub Form_Load()
' prints the old language id
Debug.Print TcEventView1.LangId
' sets the language id to English
TcEventView1.LangId = 1033
End Sub
```

Remarks

If a new language is assigned all displayed events are retrieved again in the new selected languages

7.2.1.6 DisableMenuItems

[TcEventView |▶ 68]

This property returns or sets the disable state of the context menu.

Property get

```
HRESULT get_DisableMenuItems([out, retval] long *pVal);
```

Parameters

pVal

[out, retval] pointer to a long variable that receives the actual disable state of the context menu. The disable state is represented by or junction of the enum type [TCEVENTVIEW_CONTEXTMENUFLAGS](#) [► 113].

Return Values

S_OK

Function was successfully called

E_POINTER

pVal was no valid pointer

Property set

```
HRESULT put_DisableMenuItems([in] long newVal);
```

Parameters

newVal

[in] a long variable set holds the new disable state of the context. The disable state is represented by or junction of the enum type [TCEVENTVIEW_CONTEXTMENUFLAGS](#).

Return Values

S_OK

Function was successfully called

Visual Basic sample code

```
Option Explicit

' add the Beckhoff TcEvent View Library to the components
' place a TcEventView on the form, and assign the name TcEventView1

Private Sub Form_Load()
    Dim state As Long
    ' get the actual context menu state
    state = TcEventView1.DisableMenuItems
    ' disable the confirm menu item of the context menu
    state = state Or TCEVENTVIEW_DISABLECONFIRM
    TcEventView1.DisableMenuItems = state
End Sub
```

7.2.1.7 DisableContextMenu

[TcEventView [► 68]

This property enables or disables the context menu popup or returns the state.

Property get

```
HRESULT get_DisableContextMenu([out, retval] VARIANT_BOOL*pVal);
```

Parameters

pVal

[out, retval] pointer to VARIANT_BOOL variable that receives the enable/disable state of the context menu. It's visible if the variable is VARINAT_TRUE (TRUE) and invisible if the variable is VARIANT_FALSE (FALSE).

Return Values

S_OK

Function was successfully called

E_POINTER

pVal was no valid pointer

Property set

```
HRESULT put_DisableContextMenu([in] VARIANT_BOOL newVal);
```

Parameters

newVal

[in] A variable that sets the enable/disable state of the context menu. It's visible if the variable is VARINAT_TRUE (TRUE) and invisible if the variable is VARIANT_FALSE (FALSE).

Return Values

S_OK

Function was successfully called

Visual Basic sample code

```
Option Explicit

' add the Beckhoff TcEvent View Library to the components
' place a TcEventView on the form, and assign the name TcEventView1

Private Sub Form_Load()
    TcEventView1.DisableContextMenu = True
End Sub
```

7.2.1.8 DisableScrollbars

[ITcEventView](#) [▶ 68](#)

This property sets or returns whether the list view uses Scrollbars.

Property get

```
HRESULT get_DisableScrollbars([out, retval] long* pVal);
```

Parameters

pVal

[out, retval] pointer to variable that receives the enable/disable state.

Return Values

S_OK

Function was successfully called

E_POINTER

pVal was no valid pointer

Property set

```
HRESULT put_DisableScrollbars([in] long newVal);
```

Parameters

newVal

[in] A variable that sets the the enable/disable state

Return Values

S_OK

Function was successfully called

Visual Basic sample code

```
Option Explicit

' add the Beckhoff TcEvent View Library to the components
' place a TcEventView on the form, and assign the name TcEventView1

Private Sub Form_Load()
    TcEventView1.DisableScrollbars = true
End Sub
```

7.2.1.9 SelectEvent

[ITcEventView](#) | [▶ 68](#)

This method selects an event displayed in the list view by the given index. The selected event will be displayed highlighted.

```
HRESULT SelectEvent(long nIndex);
```

Parameters

nIndex

[in] the index of the event that should be selected. The index in the list view starts at the top with the index 0.

Return Values

S_OK

Function was successfully called

Visual Basic sample code

```
Option Explicit

' add the Beckhoff TcEvent View Library to the components
' place a TcEventView on the form, and assign the name TcEventView1

Private Sub Form_Load()
    Call TcEventView1.SelectEvent(1)
End Sub
```

Remarks

This method has the same behavior as if the alarm is selected by a left mouse button click on the event item in the list view.

7.2.1.10 AddConnection

[ITcEventView](#) [► 68]

This method adds a connection to a remote pc on the network.

By default, the connection is established using *DCOM. By using the prefix *ADS://* a connection is established by using **ADS

```
HRESULT AddConnection(BSTR strComputerName);
```

Parameters

strComputerName

[in] BSTR string that holds the name of the remote PC the EventView want to connect to.

Return Values

S_OK

Function was successfully called

```
TCEVTVIEWERR_ADVISESINK = 0xF1000000
```

A connection to the remote computer has been established but the EventSink could not be advised. In this case events can only be polled by calling ShowEvents.

The high order byte does describe the error case, the low order bytes are the actual Win32 error code returned by the DispEventAdvise call.

```
TCEVENTERR_ADS = 0x98200000
```

When using ADS for remote connections, ADS errors may be passed through. The original ADS error code is stored in the low order word.

Visual Basic sample code

```
Option Explicit

' add the Beckhoff TcEvent View Library to the components
' place a TcEventView on the form, and assign the name TcEventView1

Private Sub Form_Load()
    Call TcEventView1.AddConnection("BeltPC1")
End Sub
```

Remarks

*DCOM: DCOM should just be used on very stable networks. DCOM must be set up correctly on the PC that uses the TcEventView and on the PC that provides the TcEventLogger by using dcomcnfg.exe. For more information see MSDN library.

**ADS: An ADS connection needs to be configured before connection. Both systems need to meet certain minimum requirements. Further information can be found [here](#) [► 17]

7.2.1.11 DeleteConnection

[ITcEventView](#) [► 68]

This method removes a connection a remote TcEventLogger on another PC that was established before by [AddConnection](#) [► 77].

```
HRESULT DeleteConnection(BSTR
strComputerName);
```

Parameters

strComputerName

[in] BSTR string that holds the name of the remote PC the EventView wants to disconnect from.

Return Values

S_OK

Function was successfully called

Visual Basic sample code

```
Option Explicit

' add the Beckhoff TcEvent View Library to the components
' place a TcEventView on the form, and assign the name
TcEventView1

Private Sub Form_Load()
    Call TcEventView1.DeleteConnection("BeltPC1")
End Sub
```

Remarks

7.2.1.12 DisplayComputerName

[ITcEventView](#) [► 68]

This property enables or disables the column that displays the computer name or returns the state.

Property get

```
HRESULT get_DisplayComputerName([out,
retval] VARIANT_BOOL*pVal);
```

Parameters

pVal

[out, retval] pointer to VARIANT_BOOL variable that receives the visible state of the list view column Computer Name. It's visible if the variable is VARINAT_TRUE (TRUE) and invisible if the variable is VARIANT_FALSE (FALSE).

Return Values

S_OK

Function was successfully called

E_POINTER

pVal was no valid pointer

Property let

```
HRESULT put_DisplayComputerName([in]  
VARIANT_BOOL newVal);
```

Parameters

newVal

[in] A variable that sets the visible state of the list view column Computer Name. It's visible if the variable is VARINAT_TRUE (TRUE) and invisible if the variable is VARIANT_FALSE (FALSE).

Return Values

S_OK

Function was successfully called

Visual Basic sample code

```
Option Explicit  
  
' add the Beckhoff TcEvent View Library to the components  
' place a TcEventView on the form, and assign the name TcEventView1  
  
Private Sub Form_Load()  
    TcEventView1.DisplayComputerName = True  
End Sub
```

7.2.1.13 DisplayEventId

[ITcEventView | 68](#)

This property enables or disables the column that displays the Event Id or returns the state.

Property get

```
HRESULT get_DisplayEventId([out, retval] VARIANT_BOOL*pVal);
```

Parameters

pVal

[out, retval] pointer to VARIANT_BOOL variable that receives the visible state of the list view column Event Id. It's visible if the variable is VARINAT_TRUE (TRUE) and invisible if the variable is VARIANT_FALSE (FALSE).

Return Values

S_OK

Function was successfully called

E_POINTER

pVal was no valid pointer

Property let

```
HRESULT put_DisplayEventId([in] VARIANT_BOOL newVal);
```

Parameters

newVal

[in] A variable that sets the visible state of the list view column Event Id. It's visible if the variable is VARINAT_TRUE (TRUE) and invisible if the variable is VARIANT_FALSE (FALSE).

Return Values

S_OK

Function was successfully called

Visual Basic sample code

```
Option Explicit

' add the Beckhoff TcEvent View Library to the components
' place a TcEventView on the form, and assign the name TcEventView1

Private Sub Form_Load()
    TcEventView1.DisplayEventId = True
End Sub
```

7.2.1.14 DisplayTitlebar

[ITcEventView](#) [▶ 68](#)

This property enables or disables the Viewers title bar (column header)

Property get

```
HRESULT get_DisplayTitlebar([out, retval] VARIANT_BOOL*pVal);
```

Parameters

pVal

[out, retval] pointer to VARIANT_BOOL variable that receives the visible state of the ListViews Titlebar. It's visible if the variable is VARINAT_TRUE (TRUE) and invisible if the variable is VARIANT_FALSE (FALSE).

Return Values

S_OK

Function was successfully called

E_POINTER

pVal was no valid pointer

Property set

```
HRESULT put_DisplayTitlebar([in]
VARIANT_BOOL newVal);
```

Parameters

newVal

[in] A variable that sets the visible state of the ListViews Titlebar. It's visible if the variable is VARINAT_TRUE (TRUE) and invisible if the variable is VARIANT_FALSE (FALSE).

Return Values

S_OK

Function was successfully called

Visual Basic sample code

```
Option Explicit

' add the Beckhoff TcEvent View Library to the components
' place a TcEventView on the form, and assign the name
TcEventView1

Private Sub Form_Load()
    TcEventView1.DisplayTitlebar = True
End Sub
```

7.2.1.15 DisplayUserComment

[ITcEventView](#) [► 68]

This property enables or disables the column that displays the UserComment or returns the state.

Property get

```
HRESULT get_DisplayUserComment([out, retval] VARIANT_BOOL*pVal);
```

Parameters

pVal

[out, retval] pointer to VARIANT_BOOL variable that receives the visible state of the list view column 'user comment'. It's visible if the variable is VARINAT_TRUE (TRUE) and invisible if the variable is VARIANT_FALSE (FALSE).

Return Values

S_OK

Function was successfully called

E_POINTER

pVal was no valid pointer

Property set

```
HRESULT put_DisplayUserComment([in] VARIANT_BOOL newVal);
```

Parameters

newVal

[in] A variable that sets the visible state of the list view column 'user comment'. It's visible if the variable is VARINAT_TRUE (TRUE) and invisible if the variable is VARIANT_FALSE (FALSE).

Return Values

S_OK

Function was successfully called

Visual Basic sample code

```
Option Explicit

' add the Beckhoff TcEvent View Library to the components
' place a TcEventView on the form, and assign the name TcEventView1

Private Sub Form_Load()
    TcEventView1.DisplayUserComment = True
End Sub
```

7.2.1.16 AlarmTextColor

[ITcEventView](#) [▶ 68]

This property sets or returns the color that is used to display alarm. The alarm had been issued from the PLC function block ADSLOGEVENT by setting the flag from FALSE to TRUE (or other ADS device) or from a call to a Report Event Function like [IT](#) [▶ 137][cEventLogC](#) [▶ 137]::[ReportEvent](#) [▶ 138],[ITcEventC3](#) [▶ 141]::[ReportEventEx](#) [▶ 143] or [ITcEventLog](#) [▶ 129]::[ReportEvent](#) [▶ 129] with the [TcEventClass](#) [▶ 179] set to TCEVENTCLASS_ALARM .

Property get

```
HRESULT get_AlarmTextColor([out, retval] OLE_COLOR* pVal);
```

Parameters

pVal

[out, retval] pointer to OLE_COLOR variable that receives the text color that is used to display alarms.

Return Values

S_OK

Function was successfully called

E_POINTER

pVal was no valid pointer

Property let

```
HRESULT put_AlarmTextColor([in] OLE_COLOR newVal);
```

Parameters

newVal [in] A variable that sets the text color that is used to display alarms.

Return Values

S_OK

Function was successfully called

Visual Basic sample code

```
Option Explicit

' add the Beckhoff TcEvent View Library to the components
' place a TcEventView on the form, and assign the name TcEventView1

Private Sub Form_Load()
    ' display alarms with red text color
    TcEventView1.AlarmTextColor = RGB(255, 0, 0)
End Sub
```

7.2.1.17 BackgroundColor

[ITcEventView](#) | [▶ 68](#)

This property sets or returns the background color of the control.

Property get

```
HRESULT get_BackgroundColor([out, retval] OLE_COLOR* pVal);
```

Parameters

pVal

[out, retval] pointer to OLE_COLOR variable that receives the background color.

Return Values

S_OK

Function was successfully called

E_POINTER

pVal was no valid pointer

Property set

```
HRESULT put_BackgroundColor([in] OLE_COLOR newVal);
```

Parameters

newVal [in] A variable that sets the background color .

Return Values

S_OK

Function was successfully called

Visual Basic sample code

```
Option Explicit

' add the Beckhoff TcEvent View Library to the components
' place a TcEventView on the form, and assign the name TcEventView1

Private Sub Form_Load()
    ' display alarms with red text color
    TcEventView1.BackgroundColor = RGB(255, 0, 0)
End Sub
```

7.2.1.18 ModifyLVStyle

[ITcEventView](#) |▶ 68|

This method delegates its parameters to the Cwindow::ModifyStyle method of the listview control. For further information refer to the MSDN.

```
HRESULT ModifyLVStyle([in]long dwRemove, [in]long dwAdd);
```

Parameters

dwRemove

[in] Window flags to be removed.

dwAdd

[in] Window flags to be added.

Return Values

S_OK

Function was successfully called

7.2.1.19 WarningTextColor

[ITcEventView](#) |▶ 68|

This property sets or returns the color that is used to display warnings. The warning had been issued from the PLC function block ADSLOGEVENT by setting the flag from FALSE to TRUE (or other ADS device) or the from a call to a Report Event Function like [ITcEventLogC::ReportEvent](#) |▶ 138|, [ITcEventC3::ReportEventEx](#) |▶ 143| or [ITcEventLog::ReportEvent](#) |▶ 129| with the [TcEventClass](#) |▶ 179| set to TCEVENTCLASS_WARNING.

Property get

```
HRESULT get_WarningTextColor([out, retval] OLE_COLOR* pVal);
```

Parameters

pVal

[out, retval] pointer to OLE_COLOR variable that receives the text color that is used to display warnings.

Return Values

S_OK

Function was successfully called

E_POINTER

pVal was no valid pointer

Property let

```
HRESULT put_WarningTextColor([in] OLE_COLOR newVal);
```

Parameters

newVal

[in] A variable that sets the text color that is used to display warnings.

Return Values

S_OK

Function was successfully called

Visual Basic sample code

```
Option Explicit

' add the Beckhoff TcEvent View Library to the components
' place a TcEventView on the form, and assign the name TcEventView1

Private Sub Form_Load()
    ' display warning with red text color
    TcEventView1.WarningTextColor = RGB(255, 0, 0)
End Sub
```

7.2.1.20 InstructionTextColor

[ITcEventView](#) [▶ 68](#)

This property sets or returns the color that is used to display instructions. The instruction had been issued from the PLC function block ADSLOGEVENT by setting the flag from FALSE to TRUE (or other ADS device) or from a call to a Report Event Function like [ITcEventLogC::ReportEvent](#) [▶ 138](#), [ITcEventC3::ReportEventEx](#) [▶ 143](#) or [ITcEventLogC::ReportEvent](#) [▶ 129](#) with the [TcEventClass](#) [▶ 179](#) set to TCEVENTCLASS_INSTRUCTION.

Property get

```
HRESULT get_InstructionTextColor([out, retval] OLE_COLOR* pVal);
```

Parameters

pVal

[out, retval] pointer to OLE_COLOR variable that receives the text color that is used to display instruction.

Return Values

S_OK

Function was successfully called

E_POINTER

pVal was no valid pointer

Property let

```
HRESULT put_InstructionTextColor([in] OLE_COLOR newVal);
```

Parameters

newVal

[in] A variable that sets the text color that is used to display instruction.

Return Values

S_OK

Function was successfully called

Visual Basic sample code

```
Option Explicit

' add the Beckhoff TcEvent View Library to the components
' place a TcEventView on the form, and assign the name TcEventView1

Private Sub Form_Load()
    ' display instruction with red text color
    TcEventView1.InstructionTextColor = RGB(255, 0, 0)
End Sub
```

7.2.1.21 ParamErrorTextColor

[ITcEventView](#) | [▶ 68](#)

This property sets or returns the color that is used to display parameter errors. The parameter error had been issued from the PLC function block ADSLOGEVENT by setting the flag from FALSE to TRUE (or other ADS device) or from a call to a Report Event Function like [ITcEventLogC::ReportEvent](#) | [▶ 138](#), [ITcEventC3::ReportEventEx](#) | [▶ 143](#) or [ITcEventLog::ReportEvent](#) | [▶ 129](#) with the [TcEventClass](#) | [▶ 179](#) set to TCEVENTCLASS_PARAMERROR.

Property get

```
HRESULT get_ParamErrorTextColor([out, retval] OLE_COLOR* pVal);
```

Parameters

pVal

[out, retval] pointer to OLE_COLOR variable that receives the text color that is used to display parameter errors.

Return Values

S_OK

Function was successfully called

E_POINTER

pVal was no valid pointer

Property let

```
HRESULT put_ParamErrorTextColor([in] OLE_COLOR newVal);
```

Parameters

newVal

[in] A variable that sets the text color that is used to display parameter errors.

Return Values

S_OK

Function was successfully called

Visual Basic sample code

```
Option Explicit

' add the Beckhoff TcEvent View Library to the components
```

```
' place a TcEventView on the form, and assign the name TcEventView1

Private Sub Form_Load()
    ' display parameter error with red text color
    TcEventView1.ParamErrorTextColor = RGB(255, 0, 0)
End Sub
```

7.2.1.22 StateInfoTextColor

[TcEventView ▶ 68]

This property sets or returns the color that is used to display state information. The state information had been issued from the PLC function block ADSLOGEVENT by setting the flag from FALSE to TRUE (or other ADS device) or from a call to a Report Event Function like ITcEventLogC::ReportEvent [▶ 138], ITcEventC3::ReportEventEx [▶ 143] or ITcEventLog::ReportEvent [▶ 129] with the TcEventClass [▶ 179] set to TCEVENTCLASS_STATEINFO.

Property get

```
HRESULT get_StateInfoTextColor([out, retval] OLE_COLOR* pVal);
```

Parameters

pVal

[out, retval] pointer to OLE_COLOR variable that receives the text color that is used to display state informations.

Return Values

S_OK

Function was successfully called

E_POINTER

pVal was no valid pointer

Property let

```
HRESULT put_StateInfoTextColor([in] OLE_COLOR newVal);
```

Parameters

newVal

[in] A variable that sets the text color that is used to display state informations.

Return Values

S_OK

Function was successfully called

Visual Basic sample code

```
Option Explicit

' add the Beckhoff TcEvent View Library to the components
' place a TcEventView on the form, and assign the name TcEventView1

Private Sub Form_Load()
```

```
' display state information with red text color
TcEventView1.StateInfoTextColor = RGB(255, 0,0)
End Sub
```

7.2.1.23 HintTextColor

[ITcEventView](#) [► 68]

This property sets or returns the color that is used to display hints. The hint had been issued from the PLC function block ADSLOGEVENT by setting the flag from FALSE to TRUE (or other ADS device) or the from a call to a Report Event Function like [ITcEventLogC::ReportEvent](#) [► 138], [ITcEventC3::ReportEventEx](#) [► 143] or [ITcEventLog::ReportEvent](#) [► 129] with the [TcEventClass](#) [► 179] set to TCEVENTCLASS_HINT.

Property get

```
HRESULT get_HintTextColor([out, retval] OLE_COLOR* pVal);
```

Parameters

pVal

[out, retval] pointer to OLE_COLOR variable that receives the text color that is used to display hints.#

Return Values

S_OK

Function was successfully called

E_POINTER

pVal was no valid pointer

Property let

```
HRESULT put_HintTextColor([in] OLE_COLOR newVal);
```

Parameters

newVal

[in] A variable that sets the text color that is used to display hints.

Return Values

S_OK

Function was successfully called

Visual Basic sample code

```
Option Explicit

' add the Beckhoff TcEvent View Library to the components
' place a TcEventView on the form, and assign the name
TcEventView1

Private Sub Form_Load()
    ' display hints with red text color
    TcEventView1.HintTextColor = RGB(255, 0, 0)
End Sub
```


7.2.1.24 MessageTextColor

[ITcEventView](#) | [▶ 68](#)

This property sets or returns the color that is used to display messages. The message had been issued from the PLC function block ADSLOGEVENT by setting the flag from FALSE to TRUE (or other ADS device) or from a call to a Report Event Function like [ITcEventLogC::ReportEvent](#) | [▶ 138](#), [ITcEventC3::ReportEventEx](#) | [▶ 143](#) or [ITcEventLog::ReportEvent](#) | [▶ 129](#) with the [TcEventClass](#) | [▶ 179](#) set to TCEVENTCLASS_MESSAGE.

Property get

```
HRESULT get_MessageTextColor([out, retval] OLE_COLOR* pVal);
```

Parameters

pVal

[out, retval] pointer to OLE_COLOR variable that receives the text color that is used to display messages.

Return Values

S_OK

Function was successfully called

E_POINTER

pVal was no valid pointer

Property let

```
HRESULT put_MessageTextColor([in] OLE_COLOR newVal);
```

Parameters

newVal

[in] A variable that sets the text color that is used to display messages.

Return Values

S_OK

Function was successfully called

Visual Basic sample code

```
Option Explicit

' add the Beckhoff TcEvent View Library to the components
' place a TcEventView on the form, and assign the name TcEventView1

Private Sub Form_Load()
    ' display messages with red text color
    TcEventView1.MessageTextColor = RGB(255, 0, 0)
End Sub
```

7.2.1.25 MaintenanceTextColor

[ITcEventView](#) | [▶ 68](#)

This property sets or returns the color that is used to display maintenance messages. The maintenance message had been issued from the PLC function block ADSLOGEVENT by setting the flag from FALSE to TRUE (or other ADS device) or from a call to a Report Event Function like [ITcEventLogC::ReportEvent \[▶ 138\]](#), [ITcEventC3::ReportEventEx \[▶ 143\]](#) or [ITcEventLog::ReportEvent \[▶ 129\]](#) with the [TcEventClass \[▶ 179\]](#) set to TCEVENTCLASS_MAINTENANCE.

Property get

```
HRESULT get_MaintenanceTextColor([out, retval] OLE_COLOR* pVal);
```

Parameters

pVal

[out, retval] pointer to OLE_COLOR variable that receives the text color that is used to display maintenance messages.

Return Values

S_OK

Function was successfully called

E_POINTER

pVal was no valid pointer

Property let

```
HRESULT put_MaintenanceTextColor([in] OLE_COLOR newVal);
```

Parameters

newVal

[in] A variable that sets the text color that is used to display maintenance messages.

Return Values

S_OK

Function was successfully called

Visual Basic sample code

```
Option Explicit

' add the Beckhoff TcEvent View Library to the components
' place a TcEventView on the form, and assign the name TcEventView1

Private Sub Form_Load()
    ' display maintenance messages with red text color
    TcEventView1.MaintenanceTextColor = RGB(255, 0, 0)
End Sub
```

7.2.1.26 UndefTextColor

[ITcEventView \[▶ 68\]](#)

This property sets or returns the color that is used to displayed undefined messages. The undefined message had been issued from the PLC function block ADSLOGEVENT by setting the flag from FALSE to TRUE (or other ADS device) or the from a call to a Report Event Function like [ITcEventLogC::ReportEvent \[► 138\]](#), [ITcEventC3::ReportEventEx \[► 143\]](#) or [ITcEventLog::ReportEvent \[► 129\]](#) with the [TcEventClass \[► 179\]](#) set to a value that was not defined.

Property get

```
HRESULT get_UndefTextColor([out, retval] OLE_COLOR* pVal);
```

Parameters

pVal

[out, retval] pointer to OLE_COLOR variable that receives the text color that is used to display undefined messages.

Return Values

S_OK

Function was successfully called

E_POINTER

pVal was no valid pointer

Property let

```
HRESULT put_UndefTextColor([in] OLE_COLOR newVal);
```

Parameters

newVal

[in] A variable that sets the text color that is used to display undefined messages.

Return Values

S_OK

Function was successfully called

Visual Basic sample code

```
Option Explicit

' add the Beckhoff TcEvent View Library to the components
' place a TcEventView on the form, and assign the name TcEventView1

Private Sub Form_Load()
    ' display undefined messages with red text color
    TcEventView1.UndefTextColor = RGB(255, 0, 0)
End Sub
```

7.2.1.27 ColWidthClass

[ITcEventView \[► 68\]](#)

This property sets or returns the width of the list view columns Event Class.

Property get

```
HRESULT get_ColWidthClass([out, retval] long* pVal);
```

Parameters

pVal

[out, retval] pointer to variable that receives width of the list view columns Event Class

Return Values

S_OK

Function was successfully called

E_POINTER

pVal was no valid pointer

Property set

```
HRESULT put_ColWidthClass([in] long newVal);
```

Parameters

newVal

[in] A variable that sets the width of the list view column Event Class

Return Values

S_OK

Function was successfully called

Visual Basic sample code

```
Option Explicit

' add the Beckhoff TcEvent View Library to the components
' place a TcEventView on the form, and assign the name
TcEventView1

Private Sub Form_Load()
    TcEventView1.ColWidthClass = 200
End Sub
```

7.2.1.28 ColWidthMustCon[ITcEventView](#) [▶ 68]

This property sets or returns the width of the list view columns confirmation state.

Property get

```
HRESULT get_ColWidthMustCon([out, retval] long* pVal);
```

Parameters

pVal

[out, retval] pointer to variable that receives width of the list view columns confirmation state

Return Values

S_OK

Function was successfully called

E_POINTER

pVal was no valid pointer

Property let

```
HRESULT set_ColWidthMustCon([in] long newVal);
```

Parameters

newVal

[in] A variable that sets the width of the list view column confirmation state

Return Values

S_OK

Function was successfully called

Visual Basic sample code

```
Option Explicit

' add the Beckhoff TcEvent View Library to the components
' place a TcEventView on the form, and assign the name TcEventView1

Private Sub Form_Load()
    TcEventView1.ColWidthMustCon = 200
End Sub
```

7.2.1.29 ColWidthState

[ITcEventView](#) [▶ 68](#)

This property sets or returns the width of the list view columns Event State.

Property get

```
HRESULT get_ColWidthState([out, retval] long* pVal);
```

Parameters

pVal

[out, retval] pointer to variable that receives width of the list view columns Event State

Return Values

S_OK

Function was successfully called

E_POINTER

pVal was no valid pointer

Property let

```
HRESULT put_ColWidthState([in] long newVal);
```

Parameters

newVal

[in] A variable that sets the width of the list view column Event State

Return Values

S_OK

Function was successfully called

Visual Basic sample code

```
Option Explicit

' add the Beckhoff TcEvent View Library to the components
' place a TcEventView on the form, and assign the name TcEventView1

Private Sub Form_Load()
    TcEventView1.ColWidthState = 200
End Sub
```

7.2.1.30 ColWidthSource

[ITcEventView](#) [▶ 68](#)

This property sets or returns the width of the list view columns Event Source.

Property get

```
HRESULT get_ColWidthSource([out, retval] long* pVal);
```

Parameters

pVal

[out, retval] pointer to variable that receives width of the list view columns Event Source

Return Values

S_OK

Function was successfully called

E_POINTER

pVal was no valid pointer

Property let

```
HRESULT put_ColWidthSource([in] long newVal);
```

Parameters

newVal

[in] A variable that sets the width of the list view column Event Source

Return Values

S_OK

Function was successfully called

Visual Basic sample code

```
Option Explicit

' add the Beckhoff TcEvent View Library to the components
' place a TcEventView on the form, and assign the name TcEventView1

Private Sub Form_Load()
    TcEventView1.ColWidthSource = 200
End Sub
```

7.2.1.31 ColWidthDate

[TcEventView ▶ 68]

This property sets or returns the width of the list view columns Event Date.

Property get

```
HRESULT get_ColWidthDate([out, retval] long* pVal);
```

Parameters

pVal

[out, retval] pointer to variable that receive width of the list view columns Event Date

Return Values

S_OK

Function was successfully called

E_POINTER

pVal was no valid pointer

Property let

```
HRESULT put_ColWidthDate([in] long newVal);
```

Parameters

newVal

[in] A variable that sets the width of the list view column Event Date

Return Values

S_OK

Function was successfully called

Visual Basic sample code

```
Option Explicit

' add the Beckhoff TcEvent View Library to the components
' place a TcEventView on the form, and assign the name TcEventView1

Private Sub Form_Load()
    TcEventView1.ColWidthDate = 200
End Sub
```

7.2.1.32 ColWidthTime

[ITcEventView](#) | [▶ 68](#)

This property sets or returns the width of the list view columns Event Time.

Property get

```
HRESULT get_ColWidthTime([out, retval] long* pVal);
```

Parameters

pVal

[out, retval] pointer to variable that receives width of the list view columns Event Time

Return Values

S_OK

Function was successfully called

E_POINTER

pVal was no valid pointer

Property let

```
HRESULT put_ColWidthTime([in] long newVal);
```

Parameters

newVal

[in] A variable that sets the width of the list view column Event Time

Return Values

S_OK

Function was successfully called

Visual Basic sample code

```
Option Explicit

' add the Beckhoff TcEvent View Library to the components
' place a TcEventView on the form, and assign the name TcEventView1

Private Sub Form_Load()
    TcEventView1.ColWidthTime = 200
End Sub
```

7.2.1.33 ColWidthEventId

[ITcEventView](#) | [▶ 68](#)

This property sets or returns the width of the list view columns Event Id.

Property get

```
HRESULT get_ColWidthEventId([out, retval] long* pVal);
```


Parameters

pVal

[out, retval] pointer to variable that receives width of the list view columns Event Id

Return Values

S_OK

Function was successfully called

E_POINTER

pVal was no valid pointer

Property set

```
HRESULT put_ColWidthEventId([in] long newVal);
```

Parameters

newVal

[in] A variable that sets the width of the list view column Event Id

Return Values

S_OK

Function was successfully called

Visual Basic sample code

```
Option Explicit

' add the Beckhoff TcEvent View Library to the components
' place a TcEventView on the form, and assign the name TcEventView1

Private Sub Form_Load()
    TcEventView1.ColWidthEventId = 200
End Sub
```

7.2.1.34 ColWidthComputer

[TcEventView | ▶ 68]

This property sets or returns the width of the list view columns Computer Name

Property get

```
HRESULT get_ColWidthComputer([out, retval] long* pVal);
```

Parameters

pVal

[out, retval] pointer to variable that receives width of the list view columns Computer Name

Return Values

S_OK

Function was successfully called

E_POINTER

pVal was no valid pointer

Property let

```
HRESULT put_ColWidthComputer[in] long newVal);
```

Parameters

newVal

[in] A variable that sets the width of the list view column Computer Name

Return Values

S_OK

Function was successfully called

Visual Basic sample code

```
Option Explicit

' add the Beckhoff TcEvent View Library to the components
' place a TcEventView on the form, and assign the name TcEventView1

Private Sub Form_Load()
    TcEventView1.ColWidthComputer= 200
End Sub
```

7.2.1.35 ColWidthMsg

[\[TcEventView |> 68\]](#)

This property sets or returns the width of the list view columns Event Message

Property get

```
HRESULT get_ColWidthMsg([out, retval] long* pVal);
```

Parameters

pVal

[out, retval] pointer to variable that receives width of the list view columns Event Message

Return Values

S_OK

Function was successfully called

E_POINTER

pVal was no valid pointer

Property let

```
HRESULT put_ColWidthMsg[in] long newVal);
```

Parameters

newVal

[in] A variable that sets the width of the list view column Event Message

Return Values

S_OK

Function was successfully called

Visual Basic sample code

```
Option Explicit

' add the Beckhoff TcEvent View Library to the components
' place a TcEventView on the form, and assign the name TcEventView1

Private Sub Form_Load()
    TcEventView1.ColWidthMsg = 200
End Sub
```

7.2.1.36 DisplayClass

[ITcEventView](#) | [▶ 68](#)

This property enables or disables the column that displays the Event Class or returns the state.

Property get

```
HRESULT get_DisplayClass([out, retval] VARIANT_BOOL*pVal);
```

Parameters

pVal

[out, retval] pointer to VARIANT_BOOL variable that receives the visible state of the list view column Class. It's visible if the variable is VARINAT_TRUE (TRUE) and invisible if the variable is VARIANT_FALSE (FALSE).

Return Values

S_OK

Function was successfully called

E_POINTER

pVal was no valid pointer

Property set

```
HRESULT put_DisplayClass([in] VARIANT_BOOL newVal);
```

Parameters

newVal

[in] A variable that sets the visible state of the list view column Class. It's visible if the variable is VARINAT_TRUE (TRUE) and invisible if the variable is VARIANT_FALSE (FALSE).

Return Values

S_OK

Function was successfully called

Visual Basic sample code

```
Option Explicit

' add the Beckhoff TcEvent View Library to the components
' place a TcEventView on the form, and assign the name TcEventView1

Private Sub Form_Load()
    TcEventView1.DisplayClass = True
End Sub
```

7.2.1.37 DisplayMustCon[ITcEventView](#) [▶ 68]

This property enables or disables the column that displays the confirmation state or returns the state.

Property get

```
HRESULT get_DisplayMustCon([out, retval] VARIANT_BOOL*pVal);
```

Parameters

pVal

[out, retval] pointer to VARIANT_BOOL variable that receives the visible state of the list view column confirmation state. It's visible if the variable is VARINAT_TRUE (TRUE) and invisible if the variable is VARIANT_FALSE (FALSE).

Return Values

S_OK

Function was successfully called

E_POINTER

pVal was no valid pointer

Property let

```
HRESULT put_DisplayMustCon([in] VARIANT_BOOL newVal);
```

Parameters

newVal

[in] A variable that set the visible state of the list view column confirmation state. It's visible if the variable is VARINAT_TRUE (TRUE) and invisible if the variable is VARIANT_FALSE (FALSE).

Return Values

S_OK

Function was successfully called

Visual Basic sample code

```
Option Explicit

' add the Beckhoff TcEvent View Library to the components
' place a TcEventView on the form, and assign the name
TcEventView1

Private Sub Form_Load()
    TcEventView1.DisplayMustCon = True
End Sub
```

7.2.1.38 DisplayState

[ITcEventView](#) [▶ 68]

This property enables or disables the column that displays the Event State or returns the state.

Property get

```
HRESULT get_DisplayState([out, retval] VARIANT_BOOL*pVal);
```

Parameters

pVal

[out, retval] pointer to VARIANT_BOOL variable that receives the visible state of the list view Event State. It's visible if the variable is VARINAT_TRUE (TRUE) and invisible if the variable is VARIANT_FALSE (FALSE).

Return Values

S_OK

Function was successfully called

E_POINTER

pVal was no valid pointer

Property let

```
HRESULT put_DisplayState([in] VARIANT_BOOL newVal);
```

Parameters

newVal

[in] A variable that sets the visible state of the list view column Event State. It's visible if the variable is VARINAT_TRUE (TRUE) and invisible if the variable is VARIANT_FALSE (FALSE).

Return Values

S_OK

Function was successfully called

Visual Basic sample code

```
Option Explicit

' add the Beckhoff TcEvent View Library to the components
' place a TcEventView on the form, and assign the name TcEventView1

Private Sub Form_Load()
    TcEventView1.DisplayState = True
End Sub
```

7.2.1.39 DisplaySource

[TcEventView ▶ 68]

This property enables or disables the column that displays the Event Source Name or returns the state.

Property get

```
HRESULT get_DisplaySource([out, retval] VARIANT_BOOL*pVal);
```

Parameters

pVal

[out, retval] pointer to VARIANT_BOOL variable that receives the visible state of the list view Event Source. It's visible if the variable is VARINAT_TRUE (TRUE) and invisible if the variable is VARIANT_FALSE (FALSE).

Return Values

S_OK

Function was successfully called

E_POINTER

pVal was no valid pointer

Property let

```
HRESULT put_DisplaySource([in] VARIANT_BOOL newVal);
```

Parameters

newVal

[in] A variable that sets the visible state of the list view column Event Source. It's visible if the variable is VARINAT_TRUE (TRUE) and invisible if the variable is VARIANT_FALSE (FALSE).

Return Values

S_OK

Function was successfully called

Visual Basic sample code

```
Option Explicit

' add the Beckhoff TcEvent View Library to the components
' place a TcEventView on the form, and assign the name TcEventView1
```

```
Private Sub Form_Load()  
    TcEventView1.DisplaySource = True  
End Sub
```

7.2.1.40 DisplayDate

[ITcEventView](#) | [68](#)

This property enables or disables the column that displays the Event Date or returns the state.

Property get

```
HRESULT get_DisplayDate([out, retval] VARIANT_BOOL*pVal);
```

Parameters

pVal

[out, retval] pointer to VARIANT_BOOL variable that receives the visible state of the list view Event Date. It's visible if the variable is VARINAT_TRUE (TRUE) and invisible if the variable is VARIANT_FALSE (FALSE).

Return Values

S_OK

Function was successfully called

E_POINTER

pVal was no valid pointer

Property set

```
HRESULT put_DisplayDate([in] VARIANT_BOOL newVal);
```

Parameters

newVal

[in] A variable that sets the visible state of the list view column Event Date. It's visible if the variable is VARINAT_TRUE (TRUE) and invisible if the variable is VARIANT_FALSE (FALSE).

Return Values

S_OK

Function was successfully called

Visual Basic sample code

```
Option Explicit  
  
' add the Beckhoff TcEvent View Library to the components  
' place a TcEventView on the form, and assign the name  
TcEventView1  
  
Private Sub Form_Load()  
    TcEventView1.DisplayDate = True  
End Sub
```

7.2.1.41 DisplayTime

[ITcEventView](#) [► 68]

This property enables or disables the column that displays the Event Time or returns the state.

Property get

```
HRESULT get_DisplayTime([out, retval] VARIANT_BOOL*pVal);
```

Parameters

pVal

[out, retval] pointer to VARIANT_BOOL variable that receives the visible state of the list view Event Time. It's visible if the variable is VARINAT_TRUE (TRUE) and invisible if the variable is VARIANT_FALSE (FALSE).

Return Values

S_OK

Function was successfully called

E_POINTER

pVal was no valid pointer

Property let

```
HRESULT put_DisplayTime([in] VARIANT_BOOL newVal);
```

Parameters

newVal

[in] A variable that set the visible state of the list view column Event Time. It's visible if the variable is VARINAT_TRUE (TRUE) and invisible if the variable is VARIANT_FALSE (FALSE).

Return Values

S_OK

Function was successfully called

Visual Basic sample code

```
Option Explicit

' add the Beckhoff TcEvent View Library to the components
' place a TcEventView on the form, and assign the name
TcEventView1

Private Sub Form_Load()
    TcEventView1.DisplayTime = True
End Sub
```

7.2.1.42 DisplayMsg

[ITcEventView](#) [► 68]

This property enables or disables the column that displays the Event Message Text or returns the state.

Property get

```
HRESULT get_DisplayMsg([out, retval] VARIANT_BOOL*pVal);
```

Parameters

pVal

[out, retval] pointer to VARIANT_BOOL variable that receives the visible state of the list view Event Message. It's visible if the variable is VARINAT_TRUE (TRUE) and invisible if the variable is VARIANT_FALSE (FALSE).

Return Values

S_OK

Function was successfully called

E_POINTER

pVal was no valid pointer

Property let

```
HRESULT put_DisplayMsg([in] VARIANT_BOOL newVal);
```

Parameters

newVal

[in] A variable that sets the visible state of the list view column Event Message. It's visible if the variable is VARINAT_TRUE (TRUE) and invisible if the variable is VARIANT_FALSE (FALSE).

Return Values

S_OK

Function was successfully called

Visual Basic sample code

```
Option Explicit

' add the Beckhoff TcEvent View Library to the components
' place a TcEventView on the form, and assign the name
TcEventView1

Private Sub Form_Load()
    TcEventView1.DisplayMsg = True
End Sub
```

7.2.1.43 GetSelectedEvent

[ITcEventView](#) [► 68]

This method returns the index of the selected Event. The selection is done by by a left mouse button click on the event item in the list view or by the method [SelectEvent](#) [► 76].

```
HRESULT GetSelectedEvent([out,retval] long* pSelEvent);
```

Parameters

pSelEvent

[out, retval] A pointer to variable that receives the index of the selected event. The index in the list view starts at the top with the index 0.

Return Values

S_OK

Function was successfully called

E_POINTER

pSelEvent was no valid pointer

Visual Basic sample code

```
Option Explicit

' add the Beckhoff TcEvent View Library to the components
' place a TcEventView on the form, and assign the name TcEventView1
' place a command button on the form, and assign the name Command1

Private Sub Command1_Click()
    ' get the selected event
    Dim index As Long
    index = TcEventView1.GetSelectedEvent()

    ' reset the event
    Call TcEventView1.ResetEvent(index)
End Sub
```

Remarks**7.2.1.44 getExtendedInfo**

[TcEventView | ▶ 68]

This method returns the Extended Information string (if specified) associated with an event.

```
HRESULT getExtendedInfo([in] long item, [out,retval] BSTR* message);
```

Parameters

item

[in] index of the event item in the list view control.

message

[out, retval] A pointer to variable that receives the extended information associated with the event.

Return Values

S_OK

Function was successfully called

S_FALSE

Function was successfully called but no extended information was found for the specified event

E_POINTER

pSelEvent was no valid pointer

Visual Basic sample code

Option Explicit

```
' add the Beckhoff TcEvent View Library to the components
' place a TcEventView on the form, and assign the name TcEventView1
String extInfo = TcEventView1.getExtendedInfo(0)
```

7.2.1.45 Mode

[ITcEventView](#) [► 68]

This property sets or returns the display mode of the TcEventViewer. The displayed modes are defined by the enum [TCEVENTVIEW_DISPLAYMODE](#) [► 114].

Property get

```
HRESULT get_Mode([out, retval] long* pVal);
```

Parameters

pVal

[out, retval] pointer to variable that receives the display mode of the type TCEVENTVIEW_DISPLAYMODE

Return Values

S_OK

Function was successfully called

E_POINTER

pVal was no valid pointer

Property let

```
HRESULT put_Mode([in] long newVal);
```

Parameters

newVal

[in] A variable that sets the display mode of the type TCEVENTVIEW_DISPLAYMODE

Return Values

S_OK

Function was successfully called

Visual Basic sample code

Option Explicit

```
' add the Beckhoff TcEvent View Library to the components
' place a TcEventView on the form, and assign the name TcEventView1
```

```
Private Sub Form_Load()
    TcEventView1.Mode = TCEVENTVIEW_MODE_TRACE
End Sub
```

7.2.1.46 MaxEvents

[ITcEventView](#) [► 68]

This property sets or returns the number of maximum displayed events in the TcEventViewer.

Property get

```
HRESULT get_MaxEvents([out, retval] long* pVal);
```

Parameters

pVal

[out, retval] pointer to variable that receives the max number of events

Return Values

S_OK

Function was successfully called

E_POINTER

pVal was no valid pointer

Property let

```
HRESULT put_MaxEvents([in] long newVal);
```

Parameters

newVal

[in] A variable that sets the the max number of events

Return Values

S_OK

Function was successfully called

Visual Basic sample code

```
Option Explicit

' add the Beckhoff TcEvent View Library to the components
' place a TcEventView on the form, and assign the name
TcEventView1

Private Sub Form_Load()
    TcEventView1.MaxEvents = 20
End Sub
```

Remarks

If **MaxEvent** is set to 0 all events are displayed. The reducing of the max number of events can speed up the performance of the TcEventView.

7.2.1.47 putFont

[ITcEventView](#) [▶ 68]

Method to modify the list views font.

```
HRESULT putFont(BSTR faceName, long size, unsigned short flags);
```

Parameters

faceName

[in] Name of the font to use.

size

[in] Size of the font.

flags

[in] Or junction of one or more of the following values:

```
BOLD 0x1
ITALIC 0x2
UNDERLINE 0x4
STRIKEOUT 0x8
```

Return Values

S_OK

Function was successfully called

Remarks

The putFont method does not conform to the standard COM interfaces. From languages like Visual Basic it is not usable.

7.2.1.48 SetFilter

[ITcEventView](#) [▶ 68]

This method is used to set up a display filter. Then, the TcEventViewer will display events of the requested type only - e.g. only alarms or events of a specified source.

```
HRESULT SetFilter([in]TCEVENTVIEW_FILTER Filter, [in]VARIANT Value);
```

Parameters

Filter

[in] Kind of filter to select. This value is defined through the **TCEVENTVIEW_FILTER** enum

Value

[in] Value to filter for. The following table shows which vartype is required for which filter

Filter	Vartype
TCEVENTVIEW_FILTER_NONE	Ignored
TCEVENTVIEW_FILTER_CLASS TCEVENTVIEW_FILTER_SOURCEID TCEVENTVIEW_FILTER_ID	Value convertable to Long (VT_I4)

Filter	Vartype
TCEVENTVIEW_FILTER_SOURCENAME	String (VT_BSTR)

Return Values

S_OK

Function was successfully called

E_INVALIDARG

Either the selected filter type is invalid or the supplied Vartype is not convertible to the required type.

7.2.1.49 SortMode

[ITcEventView](#) [► 68]

This property returns or sets the current selected sort order. This is what also happens when clicken the column header. If you disable the column header, or want to initialize the Viewer with a specific sort mode use this property.

Property get

```
HRESULT get_SortMode([out, retval] long *pVal);
```

Parameters

pVal

[out, retval] pointer to a long variable that receives the actual sort mode. Sort modes are defined as enum [TC_SORTMODE](#) [► 114].

Return Values

S_OK

Function was successfully called

E_POINTER

pVal was no valid pointer

Property set

```
HRESULT put_SortMode([in] long newVal);
```

Parameters

newVal

[in] a long variable set holds the new sort mode. Sort modes are defined as enum [TC_SORTMODE](#) [► 114].

Return Values

S_OK

Function was successfully called

Visual Basic sample code

```
Option Explicit
' add the Beckhoff TcEvent View Library to the components
' place a TcEventView on the form, and assign the name
```

```
TcEventView1  
  
Private Sub Form_Load()  
    TcEventView1.SortMode = SORT_BY_EVT_ID_DESCENDING  
End Sub
```

Also see about this

 [TC_SORTMODE](#) [▶ 114]

7.2.1.50 UseExternalLanguageResource

[ITcEventView](#) [▶ 68]

This property sets or returns if the Event View uses an external or the standard internal String Tables. The event view will refer to its threads LCID when deciding in what language e.g. the column header it will be displayed. The TcEventview currently supports German, English (U.S.), French, Italian and Spanish. If your preferred language is none of those, you may set this property to true. The TcEventView will then search the folder '%TwinCat\Resource' for the file 'TcEventViewLangRes.xml' in this file you may specify additional languages. This sample configuration shows you how to set up your XML resource.

Property get

```
HRESULT get_UseExternalLanguageResource([out, retval] long* pVal);
```

Parameters

pVal

[out, retval] pointer to variable that receives the current state.

Return Values

S_OK

Function was successfully called

E_POINTER

pVal was no valid pointer

Property set

```
HRESULT put_UseExternalLanguageResource([in] long newVal);
```

Parameters

newVal

[in] A variable that sets the state.

Return Values

S_OK

Function was successfully called

Visual Basic sample code

```
Option Explicit  
  
' add the Beckhoff TcEvent View Library to the components  
' place a TcEventView on the form, and assign the name TcEventView1
```

```
Private Sub Form_Load()
    TcEventView1.UseExternalLanguageResource = true
End Sub
```

7.2.2 ITcEventViewLight

The interface ITcEventViewLight is used to control the appearance of the light version of the TcEventViewer ActiveX control. The interface derives from **IDispatch** so that it can be used by all languages that support COM automation interfaces. IDispatch itself derives from IUnknown.

Table 13: Methods and Properties in Vtable Order

IUnknown Methods	Description
QueryInterface	Returns a pointer to the queried interface
AddRef	Increments the reference counter
Release	Decrements the reference counter

IDispatch Methods	Description
GetIDsOfNames	Maps a single member name and an optional set of parameter names to a corresponding set of integer dispatch identifiers (DISPIDs), which can be used on subsequent calls to IDispatch::Invoke .
GetTypeInfo	Retrieves the type information of an object.
GetTypeInfoCount	Retrieves the number of type information interfaces that an object provides, either 0 or 1.
Invoke	Provides access to properties and methods exposed by an object.

ITcEventView Methods and Properties	Description
ShowLoggedEvents [▶ 70]	displays the logged events
ShowActiveEvents [▶ 71]	displays the active events
ConfirmEvent [▶ 71]	confirms one event
ResetEvent [▶ 72]	resets one event
LangId [▶ 72]	sets and returns the language that's used for displaying event message and the source name
DisableMenuItems [▶ 73]	returns or sets the disable state of context menu
SelectEvent [▶ 76]	marks one event as selected
AddConnection [▶ 77]	adds a remote connection to another EventLogger.
DeleteConnection [▶ 78]	deletes a remote connection to another EventLogger.
DisplayComputerName [▶ 78]	enables or disables the column that displays the computer name or returns the state.
DisplayEventId [▶ 79]	enables or disables the column that displays the event id or returns the state.
ColWidthClass [▶ 91]	sets or returns the width of the columns event class
ColWidthMustCon [▶ 92]	sets or returns the width of the column confirmation state
ColWidthState [▶ 93]	sets or returns the width of the column event state
ColWidthSource [▶ 94]	sets or returns the width of the column event source
ColWidthDate [▶ 95]	sets or returns the width of the column event date
ColWidthTime [▶ 96]	sets or returns the width of the column event time

ITcEventView Methods and Properties	Description
ColWidthEventId [▶ 96]	sets or returns the width of the columns event id
ColWidthComputer [▶ 97]	sets or returns the width of the column computer name
ColWidthMsg [▶ 98]	sets or returns the width of the column event message
DisplayClass [▶ 99]	enables or disables the column that displays the event class or returns the state.
DisplayMustCon [▶ 100]	enables or disables the column that displays the confirmation state or returns the state.
DisplayState [▶ 101]	enables or disables the column that displays the event state or returns the state.
DisplaySource [▶ 102]	enables or disables the column that displays the source name or returns the state.
DisplayDate [▶ 103]	enables or disables the column that displays the event date or returns the state.
DisplayTime [▶ 104]	enables or disables the column that displays the event time or returns the state.
DisplayMsg [▶ 104]	enables or disables the column that displays the event message or returns the state.
GetSelectedEvent [▶ 105]	returns index of the event that is selected by the user
Mode [▶ 107]	sets or returns the display mode of the TcEventViewer
MaxEvents [▶ 108]	sets or returns the maximum number of displayed events.
DisplayTitlebar [▶ 80]	Enables or disables the column header.
DisableContextMenu [▶ 74]	Enables or disables the ContextMenu.
DisableScrollbars [▶ 75]	Enables or disables the Scrollbars.

7.3 Enums

7.3.1 TCEVENTVIEW_CONTEXTMENUFLAGS

This enum is used to disable context menu functions

```
enum TCEVENTVIEW_CONTEXTMENUFLAGS
{
    TCEVENTVIEW_NOTHINGDISABLED =0x0000,
    TCEVENTVIEW_DISABLECONFIRM =0x0001,
    TCEVENTVIEW_DISABLERESET =0x0002,
    TCEVENTVIEW_DISABLEDETAILS =0x0004,
    TCEVENTVIEW_DISABLEDOCLINKS =0x0008,
    TCEVENTVIEW_DISABLECLEARACTIVEALL =0x0010,
    TCEVENTVIEW_DISABLECLEARLOGGEDALL =0x0020,
    TCEVENTVIEW_DISABLECOMMENT =0x0040,
    TCEVENTVIEW_DISABLEALL =0xFFFF
}
```

Parameters

Item	Description
TCEVENTVIEW_NOTHINGDISABLED	all functions of the context menu are enabled
TCEVENTVIEW_DISABLECONFIRM	removes confirmation functions from the context menu
TCEVENTVIEW_DISABLERESET	removes reset functions from the context menu
TCEVENTVIEW_DISABLEDETAILS	removes detail functions from the context menu
TCEVENTVIEW_DISABLEDOCLINKS	remove document links functions from the context menu

Item	Description
TCEVENTVIEW_DISABLECLEARACTIVEALL	removes clear active functions from the context menu
TCEVENTVIEW_DISABLECLEARLOGGEDALL	remove clear logged functions from the context menu
TCEVENTVIEW_DISABLEALL	disable all functions of the context menu

7.3.2 TCEVENTVIEW_DISPLAYMODE

This enum is used to set the display mode of the TcEventViewer

```
enum TCEVENTVIEW_DISPLAYMODE
{
    TCEVENTVIEW_MODE_ACTIVE = 0x00000000,
    TCEVENTVIEW_MODE_LOGGED = 0x00000001,
    TCEVENTVIEW_MODE_SNAPSHOT = 0x00000002,
    TCEVENTVIEW_MODE_TRACE = 0x00000003
}
```

Parameters

Item	Description
TCEVENTVIEW_MODE_ACTIVE	Display active events. The TcEventView is kept in sync with the list of active events of the connected TcEventLog servers.
TCEVENTVIEW_MODE_LOGGED	Display logged events. The TcEventView is kept in sync with the list of logged events of the connected TcEventLog servers.
TCEVENTVIEW_MODE_SNAPSHOT	Returns a snapshot of the logged or active events. The TcEventView is not kept in sync with the list of logged events of the connected TcEventLog servers. A new snapshot of active events is displayed by calling ITcEventView::ShowActiveEvents [▶ 71] . A new snapshot of logged events is displayed by calling ITcEventView::ShowLoggedEvents [▶ 70] .
TCEVENTVIEW_MODE_TRACE	New occurring events are added to the TcEventView; no event is deleted

7.3.3 TC_SORTMODE

This enum is used to set the order in which events are displayed

```
enum TC_SORTMODE
{
    SORT_BY_CLASS_ASCENDING = 0x00000000,
    SORT_BY_CLASS_DESCENDING = 0x00000001,
    SORT_BY_SOURCE_ASCENDING = 0x00000002,
    SORT_BY_SOURCE_DESCENDING = 0x00000003,
    SORT_BY_DATE_ASCENDING = 0x00000004,
    SORT_BY_DATE_DESCENDING = 0x00000005,
    SORT_BY_TIME_ASCENDING = 0x00000006,
    SORT_BY_TIME_DESCENDING = 0x00000007,
    SORT_BY_EVT_ID_ASCENDING = 0x00000008,
    SORT_BY_EVT_ID_DESCENDING = 0x00000009,
    SORT_BY_COMPUTER_ASCENDING = 0x0000000A,
    SORT_BY_COMPUTER_DESCENDING = 0x0000000B
}
```

Parameters

Item	Description
SORT_BY_CLASS_ASCENDING	Sort events ascending by the event classes
SORT_BY_CLASS_DESCENDING	Sort events descending by the event classes
SORT_BY_SOURCE_ASCENDING	Sort events ascending by event sources

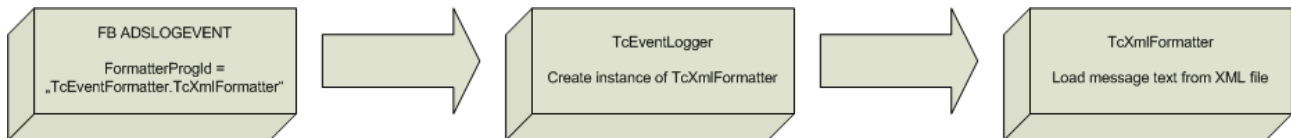
Item	Description
SORT_BY_SOURCE_DESCENDING	Sort events descending by event sources
SORT_BY_DATE_ASCENDING	Sort events ascending by date
SORT_BY_DATE_DESCENDING	Sort events descending by date
SORT_BY_TIME_ASCENDING	Sort events ascending by time
SORT_BY_TIME_DESCENDING	Sort events descending by time
SORT_BY_EVT_ID_ASCENDING	Sort events ascending by event id
SORT_BY_EVT_ID_DESCENDING	Sort events descending by event id
SORT_BY_COMPUTER_ASCENDING	Sort events ascending by computer name
SORT_BY_COMPUTER_DESCENDING	Sort events descending by computer name

8 Event Formatting

An event formatter is used to load event specific messages in a certain language from a database.

Beckhoff provides a set of standard EventFormatter implementations. Anyhow, the EventFormatter programming interface is documented so that customers can implement specific Formatters to support proprietary file or database formats.

When an event is issued in the PLC, a formatter ProgId must be given. The TcEventlogger will then try to create an instance of the COM Object with that ProgId. If such an object exists and implements the [ITcLogFormatterC \[▶ 116\]](#) interface the eventlogger will query it for the certain message text.



TwinCAT standard formatters:

Formatter	Description	Supported Platforms
TcEventFormatter	Load data from the Project Storage.	NT/2000/XP/Vista.
TcXMLFormatter [▶ 122]	Load data from XML files.	NT/2000/XP/Vista/CE.

8.1 TcEventFormatter

8.1.1 ITcLogFormatterC

Table 14: Methods and Properties in Vtable Order

Unknown Methods	Description
QueryInterface	returns a pointer to the interface you query for
AddRef	Increment the reference counter
Release	Decrements the reference counter

ITcLogFormatterC Methods	Description
GetFormatString [▶ 116]	Returns the format string
GetCompleteString [▶ 117]	Returns a formatted message string in the requested language
SetFormatString [▶ 118]	Sets the format string for a certain event
GetClassPrio [▶ 119]	Returns the event class and priority
EnumDocLinks [▶ 120]	Returns an enumeration object
GetSourceName [▶ 121]	Returns the Sourcename in a certain language

8.1.1.1 GetFormatString

This method returns the Formatstring assigned from **TcEvent.DataFormatStrAddress** when the event was shown by the PLC method **ADSLOGEVENT**.

```
HRESULT GetFormatString([in]long nEventId,
                        [in]long nSrcId,
                        [in] langId,
                        [out, retval] BSTR * szFormat)
```

Parameter

nEventId

[in] Variable with the Event Id

nSrcId

[in] Variable with the Source Id

langId

[in] The language id of the needed language. The needed language should be marked with *LCIDs. The following table gives a list of some language ids.

LCID	Description
1031	German
1033	US English
1034	Spanish
1036	French

szFormat

[out, retval] Pointer to a BSTR string that returns the format.

Return Values

S_OK

Function successful.

E_POINTER

szFormat was an invalid pointer.

E_NOTIMPL

This method is not implemented.

Notes

The TcXmlFormatter does not support this method.

*LCID: more information in the MSDN Library.

8.1.1.2 GetCompleteString

The method GetCompleteString returns a formatted message string in the requested language. This method is used of the [TcEvent \[► 128\]](#) object method [ITcEvent \[► 157\]::GetMsgString \[► 158\]](#).

```
HRESULT GetCompleteString([in]long nEventId,
                          [in]long nSrcId,
                          [in] langId,
                          [in] TcEventHeader* pEventHead,
                          [in] SAFEARRAY(VARIANT)* eventData,
                          [out, retval] BSTR* msg)
```

Parameter

nEventId

[in] Variable with the Event Id

nSrcId

[in] Variable with the Source Id

langId

[in] The language id of the needed language. The needed language should be marked with *LCIDs. The following table gives a list of some language ids.

LCID	Description
1031	German
1033	US English
1034	Spanish
1036	French

pEventHead

[in] Pointer to a [TcEventHeader](#) [► 181] object. This object is the event configuration.

eventData

[in] Pointer to a Safearray, that holds the event arguments. Its documented in the configuration of the Formatter how the event arguments have to be palced in the shown event message.

The followd data types are supported by the default Formatter:

int (16bit), long (32bit), float(32bit), double(64bit), string(BSTR). The eventdata is assigned when the event is shown.

msg

[out, retval] PPointer to a BSTR string that returns the formatted string.

Return Values

S_OK

Function successful.

E_POINTER

szFormat was an invalid pointer.

E_NOTIMPL

This method is not implemented.

Notes

Most standard Formatters, like the XML based Formatter ([TcEventFormatter.TcXmlFormatter](#)) try to return a string in the default language when the requested language cannot be found in the configuration of the formatter.

*LCID: more informations in the MSDN Library.

8.1.1.3 SetFormatString

[[TcLogFormatterC](#)] [► 116]

This method returns the Format string assigned from **TcEvent.DataFormatStrAddress** when the object returned from the PLC module **ADSLOGEVENT** was shown.

```
HRESULT SetFormatString([in]long nEventId,
                        [in]long nSrcId,
                        [in] langId,
                        [in] BSTR * szFormatString)
```

Parameter

nEventId

[in] Variable with the Event Id

nSrcId

[in] Variable with the Source Id

langId

[in] The language id of the needed language. The needed language should be marked with *LCIDs. The following table gives a list of some language ids.

LCID	Description
1031	German
1033	US English
1034	Spanish
1036	French

msg

[in] a BSTR string with the Format string

Return Values

S_OK

Function successful.

E_NOTIMPL

This method is not implemented.

Notes

The TcXmlFormatter does not support this method.

*LCID: more information in the MSDN Library.

8.1.1.4 GetClassPrio

This method returns the Class and Priority. This is used for alerts that are triggered by the PLC method ADSLOGEVENT, by setting the flag to TRUE, or by callig a function like [ITcEventLogC \[▶ 137\]::ReportEvent \[▶ 138\]](#), [ITcEventC3 \[▶ 141\]::ReportEventEx \[▶ 143\]](#) or [ITcEventLog \[▶ 129\]::ReportEvent \[▶ 129\]](#) with the [TcEventFlags \[▶ 180\]](#) set to TCEVENTFLAG_PRIOCLASS.

```
HRESULTGetClassPrio([in]long nEventId,
                    [in]long nSrcId,
                    [out] TcEventClass * pClass,
                    [out] long * pPriority)
```

pClass

[in] Pointer to a [TcEventClass](#) [[179](#)] value, tht hold the EventClass

pPriority

[out, retval] Pointer to a long variable, that gets the Event Priority. The Priority is defined by [TcEventPriority](#) [[180](#)].

Return Values

S_OK

Function successful.

E_POINTER

pClass oder pPriority are no valid pointers.

8.1.1.5 EnumDocLinks

This method returns an enum object that is used to repeat listed events. DocLinks are path and filename of the documents, like HTML pages, that are used to describe the Event in the given language.

This method is called from the [TcEvent](#) [[128](#)] object method [ITcEvent](#) [[157](#)]::EnumDocLinks. [[174](#)]

```
HRESULTEnumDocLinks ([in]long nEventId,
                    [in]long nSrcId,
                    [in] langId,
                    [out, retval] ITcEnumEventDocLink ** ppEnum
                    )
```

Parameter

nEventId

[in] Variable with the Event Id.

nSrcId

[in] Variable with the Source Id.

langId

[in] The language id of the needed language. The needed language should be marked with *LCIDs. The following table gives a list of some language ids.

LCID	Description
1031	German
1033	US English
1034	Spanish
1036	French

ppEnum

[out, retval] Pointer to a [ITcEnumEventDocLink](#) [[176](#)] pointer that gets the enum object

Return Values

S_OK

Function successful.

E_POINTER

ppEnum was an invalid pointer.

Notes

*LCID: more information in the MSDN Library.

Also see about this

 [ITcEnumEventDocLink](#) [[▶ 176](#)]

8.1.1.6 GetSourceName

The method `GetSourceName` returns a formatted Sourcename for the requested language. The [TcEvent](#) [[▶ 128](#)] object method [ITcEvent](#) [[▶ 157](#)]::SourceName [[▶ 163](#)] calls this method.

```
HRESULTGetSourceName( [in]long nSrcId,
                     [in]langId,
                     [out,retval] BSTR* szName)
```

Parameter

nSrcId

[in] Variable with the Source Id

langId

[in] The language id of the needed language. The needed language should be marked with *LCIDs. The following table gives a list of some language ids.

LCID	Description
1031	German
1033	US English
1034	Spanish
1036	French

szName

[out, retval]

Pointer to a BSTR string that returns the Sourcename

Return Values

S_OK

Function successful.

E_POINTER

szName was an invalid pointer.

Notes

Most standard Formatters, like the XML based Formatter (`TcEventFormatter.TcXmlFormatter`) try to return a string in the default language when the requested language cannot be found in the configuration of the formatter.

*LCID: more information in the MSDN Library.

8.2 TcXmlFormatter

The TcXmlFormatter offers functions to format text for TcEventLogger Events. It can handle several languages and can add floats, integers, or strings into the printed string. The printed strings are saved in a XML file. The TcEventFormatter is in the TcEventFormatter.dll.

Database

The TcXmlFormatter uses XML files as Database. Its configured with 2 different XML files:

1. In the file [TcEventSourceLocation.xml](#) [▶ 122] are the Sources defined. The Id and name of the Event file are saved here for every Source.
2. The messages are described by [XmlEventConfiguration](#). [▶ 123]

When the XML file is changed, the TcEventLogger must be restarted (or the ITcServer Reset() method must be called).

To restart the EventLogger

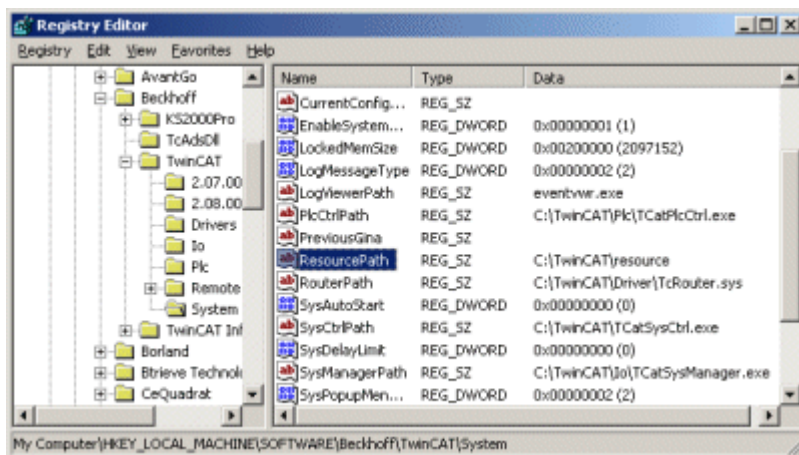
- 1) Stop all Clients that use the TcEventLogger, like HMI, TcEventbar.
- 2) Stop TwinCAT.
- 3) Start the Task Manager and ensure, that the program TcEventLogger.exe is not running. If the TcEventLogger.exe is still running you might need to restart the operating system.
- 4) Start TwinCAT again. The new Event configuration is now valid.

8.2.1 TcEventSourceLocation

The file **TcEventSourceLocation.xml** defines the location of the event configuration for every Source of the TcXmlFormatter [▶ 122]. This file is located in the TwinCAT/Resource directory.

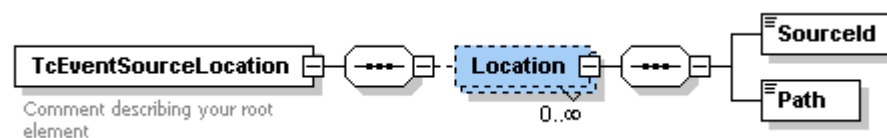
The current path of the Resource directory can be read from the Registry under the key value **HKEY_LOCAL_MACHINE\SOFTWARE\Beckhoff\TwinCAT\System**[ResourcePath].

The following screenshot shows the Registry value:



Scheme

This picture shows the XML scheme, that describes the **TcEventSourceLocation** XML file.



This table describes the nodes of the XML file:

Interface	Description
TcEventSourceLocation	The only root element. It contains a list of the location elements from 0 to infinite.
Location	Location is the location of the XML files with the Source Ids that belongs to the XmlEventConfiguration [▶ 123] . The nodes has 2 subnodes: SourceId and Path.
SourceId	Integer with the Source Id.
Path	String with the relative path of filename to the XmlEventConfiguration [▶ 123] for a Source Id.

Example

The following text shows an example **TcEventSourceLocation.xml** file.

```
<?xml version="1.0" encoding="UTF-8"?>
<TcEventSourceLocation>
  <Location>
    <SourceId>300</SourceId>
    <Path>TcIoMessages.xml</Path>
  </Location>
  <Location>
    <SourceId>500</SourceId>
    <Path>TcNcMessages.xml</Path>
  </Location>
  <Location>
    <SourceId>1</SourceId>
    <Path>user.xml</Path>
  </Location>
</TcEventSourceLocation>
```

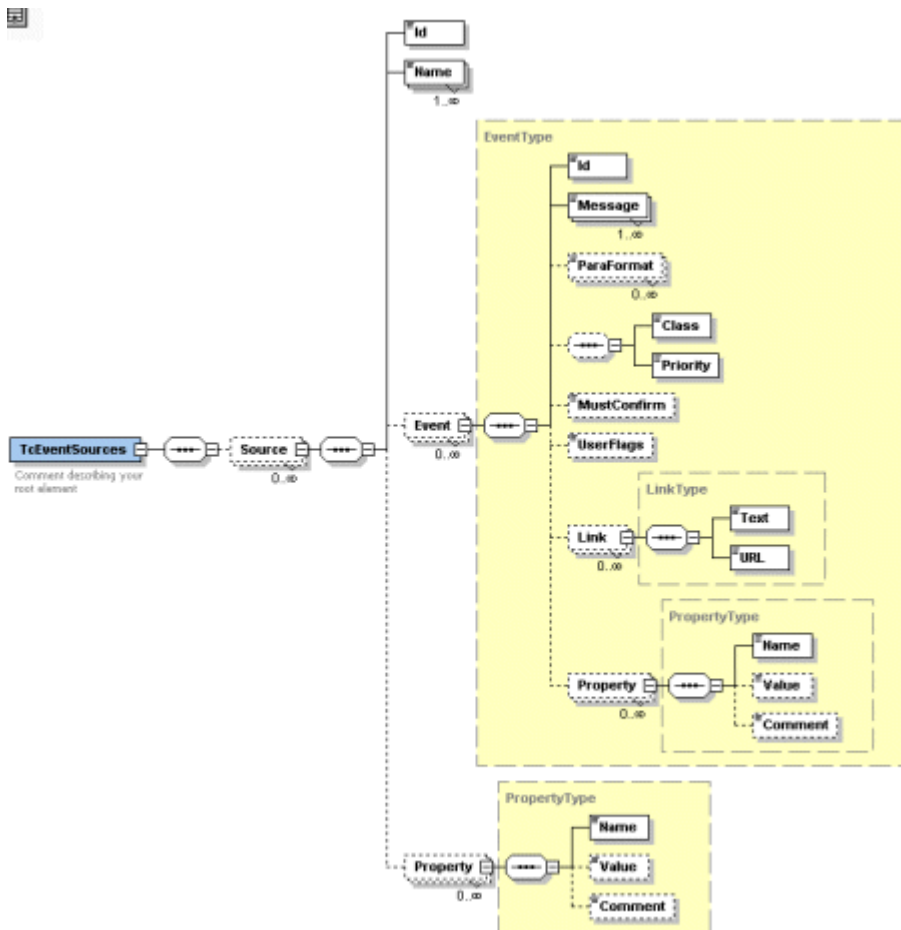
This example shows the location of the I/O messages, NC messages and user defined messages with the Source Id 1 in the user.xml file, that is located in the same directory.

8.2.2 XmlEventConfiguration

The file **XmlEventConfiguration** defines the event configuration of a Source Id in different languages. The **XmlEventConfiguration** Source Id and the location must be added to the [TcEventSourceLocation \[▶ 122\]](#) as a **Location** node.

Scheme

The following screenshot shows a XML scheme with the structure of the **XmlEventConfiguration xml file**.



The following table describes the nodes of the XML file

Interface	Description
TcEventSource	The only root element, it contains a list of Source nodes from 0 to infinite.
Source	Sub element of the TcEventSource node. This node is the configuration of a Source. The sub elements are Ids of the Source. 1 to infinite for the Source Name in different languages and 0 to infinite for the Events and Properties .
id	Sub element of the Source node. An Integer with the Source Id.
Name	Sub element of the Source node. The Source Name for the different languages. The language is defined by the attribute *Lcid . If no *Lcid attribute exists, the *Lcid 1033 (US-English) is used.
Event	Sub element of the Source node. This node is the configuration of a Event. The sub elements are Event id , a Message string for the different Languages from 0 to infinite, ParaFormat is the format string for the Event data, the MustConfirm flag, the UserFlags , DocLinks from 0 to infinite and Property from 0 to infinite.
Property	Sub element of the Source node. The property is displayed by name, value and comment.
id	Sub element of the Event node. A Integer with the Event Id.
Message	Sub element of the Event node. The Event message for the different Languages. The language is defined by the attribute *Lcid . If no *Lcid attribute exists, the *Lcid 1033 (US-English) is used. The eventData was assigned when the Event was started from the PLC module ADSLOGEVENT by setting the Flag from FALSE to TRUE or by calling the function ITcEventLogC::ReportEvent [▶ 138] , ITcEventC3 ::ReportEventEx [▶ 143] or ITcEventLog:: ReportEvent [▶ 129] . It can added to the message with %1, %2, %3 ... , %x is the number of the index of eventData .
ParaFormat	Sub element of the Event node. Defines a printf format description, that is used for eventData . The attribute ParaNo represents the Index of eventData for that the format string is used.

Interface	Description
UserFlags	Sub element of the Event node. An integer that defines the user-defined flags.
Link	Sub element of the Event node. A DocLink is a link to a document that offers more informations in different languages. You can access to the DocLink Property with the TcEvent [▸ 128] object ITcEvent [▸ 157] :: EnumDocLink [▸ 174] and in the context menu of the TcEventViewer [▸ 68] . The language is defined by the attribute *Lcid . The DocLink has a sub element Text that shows the URL for the requested *Lcid .
Property	Sub element of the Event node. The Property element has a Name, a Value, and a Comment.

Example

The following text shows a example of a **XmlEventConfiguration** file.

```
<?xml version="1.0" encoding="UTF-8"?>
<TcEventSources>
  <Source>
    <Id>1</Id>
    <Name LcId="1033">Axis Controller</Name>
    <Name LcId="1031">Achsen Controller</Name>

    <Event>
      <Id>1</Id>

      <Message LcId="1033">The Axis:%1 stop at the position %2
mm</Message>
      <Message LcId="1031">Die Achse:%1 hielt an der Position %2
mm</Message>

      <!-- format for the second parameter -->
      <ParaFormat ParaNo="2">%.3f</ParaFormat>

      <!--the event class and prio is defined here -->

      <!-- the event class is a WARNING-->
      <Class>6</Class>

      <!-- the event priority is implicit -->
      <Priority>0</Priority>

      <MustConfirm>true</MustConfirm>
      <Link LcId="1033">
        <Text>Help</Text>
        <URL>file:///C:/1033/AxisError.html</URL>
      </Link>
      <Link LcId="1031">
        <Text>Help</Text>
        <URL>file:///C:/1031/AxisError.html</URL>
      </Link>
      <Property>
        <Name>Log</Name>
        <Value>true</Value>
      </Property>
    </Event>
    <Event>
      <Id>2</Id>

      <Message LcId="1033">Emergency Stop!</Message>
      <Message LcId="1031">Not Stopp!</Message>
    </Event>
  </Source>
</TcEventSources>
```

This example shows the Event configuration for 2 Event with the languages German and English.

The following table gives a list of some language *LCIDs.

LCID	Description
1031	German
1033	US English

LCID	Description
1034	Spanish
1036	French

*LCID: more information in the MSDN Library.

9 API

The TcEventLogger is a mechanism to manage messages (eG from the PLC). It's realized as TcCOM-Server and controlled from the TwinCAT System. It offers multiple COM Interfaces to query and manipulate messages:

Interface	Description
ITcServer	Used by the TwinCAT System to manage this Server.
ITcAdsAsync	Used internal to implement AdS functions. The TcEventLogger get its AdS commands from port 110.
ITcEventLog [▶ 129]	This is the main Interface of the TcEventLogger for COM Clients like HMI. It offers functions to control active and logged alarms and to add and remove alarms.
ITcEventLogC [▶ 137]	This is the basic interface of ITcEventLogC3 and ITcEventLogC2.
ITcEventLogC2 [▶ 139]	This interface derives from ITcEventLogC and is the basic interface for ITcEventLogC3.
ITcEventLogC3 [▶ 141]	This is an application-specific interface of the TcEventLoggers for COM Clients that use application-specific interfaces like HMI. is interface deriving from ITcEventLogC2.
ITcEventLogEvents [▶ 144]	Callback functions for callbacks from the TcEventlogger to COM-Clients. Clients can use this to get informations when an event occurs or is cancelled.
ITcEvent [▶ 157]	Message object. This object can be used to get informations about a message (e.G. event- and source ID) or the message text. It can also be used to reset messages.
ITcEnumEvents [▶ 150]	This interface can be used to enumerate the active or logged messages.
ITcEnumEventDocLink [▶ 176]	This interface can be used to enumerate the Doc Links.

9.1 TcEventLogger Return Codes

Code (hex)	Description
0x00000000	No Errorr
0x98210001	Event object is in signaled state
0x98210002	Event object is confirmed
0x98210003	Event object is reset
0x98210004	Event object is confirmed but not reset
0x98210005	Event object does not need confirmation
0x98210006	Reset or confirm for unknown event
0x98210007	Event is loaded from persistent storage
0x98210008	Event configuration could not be retrieved from the storage
0x98210009	Could not access storage

9.2 Classes

9.2.1 TcEventLog

The Main class of the TcEvent Logger. Provides Interface to the COM Client and to the ADS world. This Class implements the state machine for handling the behavior of the alarms and the signal transfer between the PLC and the COM clients. Internally it controls the TcEventKeyToEvent Map that lists active alarms and an object of the Type IEnumSTATSTGPtr that holds a list of streams that present the Logged alarms. On request to a method call on the ITeEventLog interface it returns one enumeration object for active events and one for logged events.

Requirements

Interface	Description
ITcServer	Used internally by the TwinCAT system to control this server.
ITcAdsAsync	Used internally to implement ADS functionality into the TcEventLogger. The TcEvent Logger receives ADS commands on ADS port 110 through this interface.
ITcEventLog [► 129]	This is the main interface of the TcEventLogger to COM clients like the HMI. Main features to control active and logged alarms and to add new alarms or delete existing ones.
ITcEventLogC3 [► 141]	This is a custom interface of the TcEventLogger to COM clients like the HMI that supports custom interfaces. This interface derives from ITcEventLogC2.
ITcEventLogC2 [► 139]	Interface derives from ITcEventLogC2 and it the base interface of ITcEventLogC3
ITcEventLogC [► 137]	This interface is the base interface of ITcEventLogC3 and ITcEventLogC2
ITcEventLogBase C	Not used yet

Additionally this class provides the event interface [ITcEventLogEvents \[► 144\]](#) that could be implemented by the client.

9.2.2 TcEvent

This class represents one event. The class provides interfaces to the COM clients. Objects of this class are issued by the TcEventLog the main class of the TcEvent Logger. The class provides through it's interface all parameter of the alarms and uses internally the assigned formatter object to return the event message language independent.

Interface	Description
ITcEvent [► 157]	The main Interface provides all parameters of an event object.
ITcEventC [► 175]	This custom interface provides additional parameters of the alarm.

9.3 Interfaces

9.3.1 ITcEventLog

The ITcEventLog is a standard interface of the main class of the [TcEventLogger](#) [▶ 127]. The interface provides the main features to control active and logged alarms and also to add new alarms or delete existing ones. The interface derives from **IDispatch** so that it could be used by all languages that support COM automation interfaces. IDispatch itself derives from IUnknown.

Table 15: Methods and Properties in Vtable Order

IUnknown Methods	Description
QueryInterface	Returns a pointer to the interface you query for
AddRef	Increments the reference counter
Release	Decrements the reference counter

IDispatch Methods	Description
GetIDsOfNames	Maps a single member name and an optional set of parameter names to a corresponding set of integer dispatch identifiers (DISPIDs), which can then be used on subsequent calls to IDispatch::Invoke .
GetTypeInfo	Retrieves the type information for an object.
GetTypeInfoCount	Retrieves the number of type information interfaces that an object provides, either 0 or 1.
Invoke	Provides access to properties and methods exposed by an object.

ITcEventLog Methods and Properties	Description
ReportEvent [▶ 129]	This method is used to issue a new event from the client.
ClearActiveEvents [▶ 131]	This method resets all Active events.
GetLastEvent [▶ 131]	This method returns the most recent events
EnumActiveEvents [▶ 132]	Returns an enumeration object that is used to iterate through the list of active events.
EnumActiveEventsEx [▶ 133]	Returns a standard enumerator for iterating through the list of active events.
EnumLoggedEvents [▶ 133]	Returns an enumeration object that is used to iterate through the list of logged events.
EnumLoggedEventsEx [▶ 135]	Returns a standard enumerator for iterating through the list of logged events.
Export [▶ 135]	Copies the list of logged event to a structured storage and returns the storage.
ExportText	Not implemented
ActiveEvents [▶ 136]	Returns the number of active events.
LoggedEvents [▶ 136]	Returns the number of logged events.
ClearLoggedEvents [▶ 137]	This method resets all logged events.

9.3.1.1 ReportEvent

[ITcEventLog](#) [▶ 129]

This method is used to issue an event from a COM client like the HMI program to the TcEventLogger.

```
HRESULT ReportEvent([in] SAFEARRAY(VARIANT)* eventHead,
                   [in] SAFEARRAY(VARIANT)* eventData)
);
```

Parameters

eventHead

[in] Pointer to a safearray with the bounds 0 to 9 that represents the event header. The array represents the same information as TcEventHeader. The items of the array are explained in the following table

index	TcEventHeader [▶ 181] item	Data Type	Description
0	nClass	TcEventClass [▶ 179]	The class of the event, like alarm, warning, hint
1	nPriority	TcEventPriority [▶ 180] (long)	The event priority. Since now the priority is always: TcEventPriority.TCEVENTPRIO_IMPLICIT=0
2	dwFlags	TcEventFlags [▶ 180] (long)	The flags control the behavior of the alarm. That could be combined by an OR junction.
3	dwUserData	long	A spare variable that can be used by the user.
4	nId	long	The event id, that unique represents this type of event for this one Event Source.
5	nInvokeld	long	
6	fDate	VARIANT-DATE	The date and time when this event occur
7	nMs	long	The millisecond part of the event date
8	varSource	variant	The source id source name. The event source is used to distinguish different device. Like I/O Event, or different parts of the PLC program.
9	szFmtProgId	BSTR string	The PrgId of the formatter. If we want to use the standard XML formatter the string should be set to "TcEventFormatter.TcXmlFormatter"

eventData

[in] A pointer to a safe array that represents the event arguments. In the configuration of the event formatter, it's defined how to place the event arguments into the displayed event message. The following data types are supported by the standard formatter. int (16bit), long (32bit), float(32bit), double(64bit), string(BSTR)

Return Values

S_OK

Function was successfully called

TCEVENTERR_ISSIGNALLED

The event with this source id and event id was already signaled.

Visual Basic sample code

```
' get the one and only event logger
Dim evtLogger As TCEVENTLOGGERLib.TcEventLog
Set evtLogger = New TCEVENTLOGGERLib.TcEventLog

' create event header
Dim arrHeader(0 To 9) As Variant

arrHeader(0) = TcEventClass.TCEVENTCLASS_ALARM ' event class
arrHeader(1) = TcEventPriority.TCEVENTPRIO_IMPLICIT ' event priority
arrHeader(2) = TcEventFlags.TCEVENTFLAG_LOG ' event flags
arrHeader(3) = 0 'user data
arrHeader(4) = 1 ' event id
```

```

arrHeader(5) = 0 'invoke id
arrHeader(6) = Now ' event date and time
arrHeader(7) = 123 ' millisecond part of the event date and time
arrHeader(8) = 1 ' event source id
arrHeader(9) = "TcEventFormatter.TcXmlFormatter" ' XML formatter ProgId

' create event parameters
Dim arrParam(0 To 1) As Variant
arrParam(0) = 1234 ' first parameter as long
arrParam(1) = 1234.777 ' second parameter as double

' log the event
Call evtLogger.ReportEvent(arrHeader, arrParam)

```

Remarks

After this method call the Event Logger will raise the event [OnNewEvent \[► 145\]](#) on all client that implements the event interface [ITcEventLogEvents \[► 144\]](#) (VB: Dim WithEvents).

For C++ and Visual Basic programmers it's more convenient to use the method [ITcEventLogC \[► 137\]::ReportEvent \[► 138\]](#) or [ITcEventC3 \[► 141\]::ReportEventEx \[► 143\]](#).

9.3.1.2 ClearActiveEvents

[ITcEventLog \[► 129\]](#)

This method is used to reset all active events.

```
HRESULT ClearActiveEvents();
```

Parameters

Return Values

S_OK

Function was successfully called

Visual Basic sample code

```

' get the one and only event logger
Dim evtLogger As TCEVENTLOGGERLib.TcEventLog
Set evtLogger = New TCEVENTLOGGERLib.TcEventLog

' clear all active alarms
evtLogger.ClearActiveEvents

```

Remarks

After this method call the Event Logger will raise the event [OnActiveEventsCleared \[► 148\]](#) on all client that implements the event interface [ITcEventLogEvents \[► 144\]](#) (VB: Dim WithEvents).

9.3.1.3 GetLastEvent

[ITcEventLog \[► 129\]](#)

This method returns the most recent active event.

```
HRESULT GetLastEvent([out,retval] IDispatch** lastEvent);
```

Parameters

lastEvent

[out, retval] pointer to an object of the interface type IDispatch that returns the most recent active event.

Return Values

S_OK

Function was successfully called

E_POINTER

the pointer assign to lastEvent was not valid

Visual Basic sample code

```
' get the one and only event logger
Dim evtLogger As TCEVENTLOGGERLib.TcEventLog
Set evtLogger = New TCEVENTLOGGERLib.TcEventLog

' get the most recent logged event
Dim resentEvt As TcEvent
Set resentEvt = evtLogger.GetLastEvent
```

9.3.1.4 EnumActiveEvents

[ITcEventLog](#) [▶ 129]

Returns an enumeration object that is used to iterate through the list of active events.

This method is provided for backward compatibility and usage from c++ client. Languages like VB or the .NET family should use [EnumActiveEventsEx](#) [▶ 131].

```
HRESULT EnumActiveEvents ([out, retval] IUnknown** ppEnum);
```

Parameters

ppEnum

[out, retval] Pointer to the [ITcEnumEvents](#) [▶ 150] interface pointer that receives the enumeration object of the active events.

Return Values

S_OK

Function was successfully called

E_POINTER

ppEnum was no valid pointer.

Visual Basic sample code

```
' get the one and only event logger
Dim evtLogger As TCEVENTLOGGERLib.TcEventLog
Set evtLogger = New TCEVENTLOGGERLib.TcEventLog

' get the active event enumeration object
Dim enumEvt As ITcEnumEvents
Set enumEvt = evtLogger.EnumActiveEvents

' try to get max events an loop through the event list
Const coMax As Long = 20
Dim i As Long
Dim nFetched As Long
Dim evt As TcEvent
Dim arrEvt(1 To coMax) As Object
Do
    nFetched = enumEvt.Next(coMax, arrEvt(1))
    For i = 1 To nFetched
        Set evt = arrEvt(i)
```

```
' print the event message in english
Debug.Print evt.GetMsgString(1033)
' release the evt object
Set arrEvt(i) = Nothing
Next i
Loop While (nFetched >= coMax)
```

Remarks

The behaviour of the imported TcEventLogger type libraries is different under CodeGear C++Builder 2009 and Visual Studio C++. Both development environments generate different Code. In C++Builder the returned enumeration object is not automatically enabled if it leaves the visible range ("out of scope") e.g. after allocation:

```
ITcEnumEventsPtr enumEvt = evtLogger->EnumActiveEvents();
```

the no longer used object must be enabled in C++Builder.

```
enumEvt->Release();
```

There is no difference to the Microsoft Visual Studio C++ implementation. The release call is not necessary. For this please use the **EnumActiveEventsEx()** method in C++Builder. This method enables the returned object under C++Builder automatically.

9.3.1.5 EnumLoggedEventsEx

[ITcEventLog](#) [[▶ 129](#)]

Returns an enumeration object that is used to iterate through the list of active events. This Method is provided for compatibility with the .NET language family. If you are programming in C++ you might rather want to use the previous version : [EnumActiveEvents](#) [[▶ 132](#)].

```
HRESULTEnumLoggedEventsEx([out, retval] ITcEnumEventEx** ppEnum);
```

Parameters

ppEnum

[out, retval] Pointer to the [ITcEnumEventsEx](#) [[▶ 155](#)] interface pointer that receives the enumeration object of the active events.

Return Values

S_OK

Function was successfully called

E_POINTER

ppEnum was no valid pointer.

Visual Studio 2005 C# Sample Code:

```
foreach (TcEvent evt in new TcEventLog().EnumActiveEventsEx())
    evt.Reset();
```

Also see about this

 [ITcEnumEvents](#) [[▶ 150](#)]

9.3.1.6 EnumLoggedEvents

[ITcEventLog](#) [[▶ 129](#)]

Returns an enumeration object that is used to iterate through the list of logged events.

This method is provided for backward compatibility and usage from c++ client. Languages like VB or the .NET family should use [EnumLoggedEventsEx](#) [► 135].

```
HRESULT EnumLoggedEvents ([out, retval] IUnknown** ppEnum);
```

Parameters

ppEnum

[out, retval] Pointer to the [ITcEnumEvents](#) [► 150] interface pointer that receives the enumeration object of the logged events.

Return Values

S_OK

Function was successfully called

E_POINTER

ppEnum was no valid pointer.

Visual Basic sample code

```
' get the one and only event logger
Dim evtLogger As TCEVENTLOGGERLib.TcEventLog
Set evtLogger = New TCEVENTLOGGERLib.TcEventLog

' get the active event enumeration object
Dim enumEvt As ITcEnumEvents
Set enumEvt = evtLogger.EnumLoggedEvents

' try to get max events in a loop through the event list
Const coMax As Long = 20
Dim i As Long
Dim nFetched As Long
Dim evt As TcEvent
Dim arrEvt(1 To coMax) As Object
Do
    nFetched = enumEvt.Next(coMax, arrEvt(1))
    For i = 1 To nFetched
        Set evt = arrEvt(i)
        ' print the event message in english
        Debug.Print evt.GetMsgString(1033)
        ' release the evt object
        Set arrEvt(i) = Nothing
    Next i
Loop While (nFetched >= coMax)
```

Remarks

The behavior of the imported TcEventLogger type libraries is different under CodeGear C++Builder 2009 and Visual Studio C++. Both development environments generate different Code. In C++Builder the returned enumeration object is not automatically enabled if it leaves the visible range ("out of scope") e.g. after allocation:

```
ITcEnumEventsPtr enumEvt = evtLogger->EnumLoggedEvents();
```

the no longer used object has to be enabled in C++Builder:

```
enumEvt->Release();
```

There is a difference to the Microsoft Visual Studio C++ implementation. The the release call is not necessary. For this please use the [EnumLoggedEventsEx\(\)](#) method in C++Builder. This method enables the returned object under C++Builder automatically.

9.3.1.7 EnumLoggedEventsEx

[ITcEventLog](#) [[▶ 129](#)]

Returns an enumeration object that is used to iterate through the list of logged events. This Method is provided for compatibility with the .NET language family. If you are programming in C++ you might rather want to use the previous version : [EnumLoggedEvents](#) [[▶ 133](#)].

```
HRESULT EnumLoggedEventsEx ([out, retval] ITcEnumEventEx** ppEnum);
```

Parameters

ppEnum

[out, retval] Pointer to the [ITcEnumEventsEx](#) [[▶ 155](#)] interface pointer that receives the enumeration object of the logged events.

Return Values

S_OK

Function was successfully called

E_POINTER

ppEnum was no valid pointer.

Visual Studio 2005 C# Sample Code:

```
foreach (TcEvent evt in new TcEventLog().EnumLoggedEventsEx())  
    evt.Reset();
```

Also see about this

 [ITcEnumEvents](#) [[▶ 150](#)]

9.3.1.8 Export

[ITcEventLog](#) [[▶ 129](#)]

Copies the list of logged events to a structured storage and returns the storage.

```
HRESULT Export ([in] IUnknown *pExport);
```

Parameters

ppEnum

[in] IUnknown pointer to an IStorage interface that receives the copy of the structured storage that holds the list of logged events in streams.

Return Values

S_OK

Function was successfully called

E_POINTER

pExport was no valid pointer.

Remarks

Since the structured storage is the native database format and the structure design is not documented the export function should not be used by user programs.

9.3.1.9 ActiveEvents

[ITcEventLog](#) [[▶](#) [129](#)]

This property returns the number of active events.

```
HRESULT ActiveEvents([out, retval] long *pVal);
```

Parameters

pVal

[out, retval] A pointer to a long value that returns the current number of active events.

Return Values

S_OK

function was successfully called

E_POINTER

pVal was no valid pointer.

Visual Basic sample code

```
' get the one and only event logger
Dim evtLogger As TCEVENTLOGGERLib.TcEventLog
Set evtLogger = New TCEVENTLOGGERLib.TcEventLog

' get the number of active events
Dim countActive As Long
countActive = evtLogger.ActiveEvents
```

9.3.1.10 LoggedEvents

[ITcEventLog](#) [[▶](#) [129](#)]

This property returns the number of logged events.

```
HRESULT LoggedEvents([out, retval] long *pVal);
```

Parameters

pVal

[out, retval] A pointer to a long value that returns the current number of logged events.

Return Values

S_OK

Function was successfully called

E_POINTER

pVal was no valid pointer.

Visual Basic sample code

```
' get the one and only event logger
Dim evtLogger As TCEVENTLOGGERLib.TcEventLog
Set evtLogger = New TCEVENTLOGGERLib.TcEventLog

' get the number of logged events
Dim countLogged As Long
countLogged = evtLogger.LoggedEvents
```

9.3.1.11 ClearLoggedEvents

[ITcEventLog](#) [[▶ 129](#)]

This method is used to reset all logged events.

```
HRESULT ClearLoggedEvents();
```

Parameters

Return Values

S_OK

Function was successfully called

Visual Basic sample code

```
' get the one and only event logger
Dim evtLogger As TCEVENTLOGGERLib.TcEventLog
Set evtLogger = New TCEVENTLOGGERLib.TcEventLog

' clear all logged alarms
evtLogger.ClearLoggedEvents
```

Remarks

After this method call the Event Logger will raise the event [OnLoggedEventsCleared](#) [[▶ 148](#)] on all client that implements the event interface [ITcEventLogEvents](#) [[▶ 144](#)] (VB: Dim WithEvents).

9.3.2 ITcEventLogC

The ITcEventLogC is an interface of the main class of the [TcEventLogger](#) [[▶ 127](#)]. The interface provides the features add new alarms. The interface derives from **IUnknown** so that it could be used by all languages that support COM custom interfaces.

Table 16: Methods in Vtable Order

IUnknown Methods	Description
QueryInterface	returns a pointer to the interface you query for
AddRef	Increment the reference counter
Release	Decrements the reference counter

ITcEventLogC Methods	Description
ReportEvent [▶ 129]	This method is used to issue a new event from the client.
EnumEventSources	Deprecated. Was used on the deprecated structured storage based formatter to loop through the list of event source storages.
OpenEventSource	Deprecated. Was used on the deprecated structured storage-based formatter to return one particular event source storages.

9.3.2.1 ReportEvent

[ITcEventLogC](#) | [137](#)

This method is used to issue an alarm from a COM client like the HMI program to the TcEventLogger.

```
HRESULT ReportEvent([in] TcEventHeader* pEventHead,
                   [in] SAFEARRAY(VARIANT)* pEventData
);
```

Parameters

pEventHead

[in] Pointer to an object of the type TcEventHeader. The object represents the alarm configuration.

eventData

[in] A pointer to a safe array that represents the event arguments. In the configuration of the event formatter it's defined how to place the event arguments into the displayed event message.

The following data types are supported by the standard formatter.

int (16bit), long (32bit), float(32bit), double(64bit), string(BSTR)

Return Values

S_OK

Function was successfully called

TCEVENTERR_ISSIGNALLED

the event with this source id and event id was already signaled.

E_POINTER

pEventHead or pEventData were no valid pointers.

Visual Basic sample code

```
' get the one and only event logger
Dim evtLogger As TCEVENTLOGGERLib.TcEventLog
Set evtLogger = New TCEVENTLOGGERLib.TcEventLog

' cast to ITcEventLogC interface
Dim evtLoggerC As TCEVENTLOGGERLib.ITcEventLogC
Set evtLoggerC = evtLogger

' create event header
Dim header As TcEventHeader
header.nClass = TcEventClass.TCEVENTCLASS_ALARM ' event class
header.nPriority = TcEventPriority.TCEVENTPRIO_IMPLICIT ' event priority
header.dwFlags = TcEventFlags.TCEVENTFLAG_LOG ' event flags
header.dwUserData = 0 'user data
header.nId = 1 ' event id
header.nInvokeId = 0 'invoke id
header.fDate = Now ' event date and time
header.nMs = 123 ' milli second part of the event date and time
header.varSource = 1 ' event source id
header.szFmtProgId = "TcEventFormatter.TcXmlFormatter" ' event prog id

' create event params
Dim arrParam() As Variant
ReDim arrParam(0 To 1) As Variant
arrParam(0) = 1234 ' first parameter as long
arrParam(1) = 1234.777 ' second parameter as double

' log the event
Call evtLoggerC.ReportEvent(header, arrParam)
```

Remarks

After this method call the Event Logger will raise the event `OnNewEvent` [▶ 145] on all client that implement the event interface `ITcEventLogEvents` [▶ 144] (VB: Dim WithEvents).

The method can just be used to issue an alarm. A second call to reset the alarm is not possible with this method. To change the state of an already issued alarm we need the event object itself. For more convenience the method `ITcEventLogC3` [▶ 141>::`ReportEventEx` [▶ 143] directly returns the new created event object.

Also see about this

📄 `TcEventHeader` [▶ 181]

9.3.3 ITcEventLogC2

The `ITcEventLogC2` is an interface of the main class of the `TcEventLogger` [▶ 127]. The interface provides the deprecated features to control the deprecated structured storage based formatter. The interface derives from `ITcEventLogC` [▶ 137] and **IUnknown** so that it could be used by all languages that support COM custom interfaces.

Methods in Vtable Order

Table 17: Methods in Vtable Order

IUnknown Methods	Description
<code>QueryInterface</code>	returns a pointer to the interface you query for
<code>AddRef</code>	Increments the reference counter
<code>Release</code>	Decrements the reference counter

ITcEventLogC Methods	Description
<code>ReportEvent</code> [▶ 129]	This method is used to issue a new event from the client.
<code>EnumEventSources</code>	Deprecated. Was used on the deprecated structured storage based formatter to loop through the list of event source storages.
<code>OpenEventSource</code>	Deprecated. Was used on the deprecated structured storage based formatter to return one particular event source storages.

ITcEventLogC2 Methods	Description
<code>CreateEventSource</code>	Deprecated. Was used on the deprecated structured storage based formatter to create a new event source storage.
<code>RemoveEventSource</code>	Deprecated. Was used on the deprecated structured storage based formatter to remove an existing event source storage.
<code>RenameEventSource</code>	Deprecated. Was used on the deprecated structured storage based formatter to rename an existing event source storage.

9.3.3.1 ReportEvent

`ITcEventLogC`

This method is used to issue an alarm from a COM client like the HMI program to the `TcEventLogger`.

```
HRESULT ReportEvent([in] TcEventHeader* pEventHead,
                   [in] SAFEARRAY(VARIANT)* pEventData
);
```

Parameters

pEventHead

[in] Pointer to an object of the type TcEventHeader. The object represents the alarm configuration.

eventData

[in] A pointer to a safe array that represents the event arguments. In the configuration of the event formatter, it's defined how to place the event arguments into the displayed event message.

The following data types are supported by the standard formatter.

int (16bit), long (32bit), float(32bit), double(64bit), string(BSTR)

Return Values

S_OK

Function was successfully called

TCEVENTERR_ISSIGNALLED

the event with this source id and event id was already signaled.

E_POINTER

pEventHead or pEventData were no valid pointers.

Visual Basic sample code

```
' get the one and only event logger
Dim evtLogger As TCEVENTLOGGERLib.TcEventLog
Set evtLogger = New TCEVENTLOGGERLib.TcEventLog

' cast to ITcEventLogC interface
Dim evtLoggerC As TCEVENTLOGGERLib.ITcEventLogC
Set evtLoggerC = evtLogger

' create event header
Dim header As TcEventHeader
header.nClass = TcEventClass.TCEVENTCLASS_ALARM ' event class

header.nPriority = TcEventPriority.TCEVENTPRIO_IMPLICIT ' event priority
header.dwFlags = TcEventFlags.TCEVENTFLAG_LOG ' event flags
header.dwUserData = 0 'user data
header.nId = 1 ' event id
header.nInvokeId = 0 'invoke id
header.fDate = Now ' event date and time
header.nMs = 123 ' milli second part of the event date and time
header.varSource = 1 ' event source id
header.szFmtProgId = "TcEventFormatter.TcXmlFormatter" ' event prog id

' create event params
Dim arrParam() As Variant
ReDim arrParam(0 To 1) As Variant
arrParam(0) = 1234 ' first parameter as long
arrParam(1) = 1234.777 ' second parameter as double

' log the event
Call evtLoggerC.ReportEvent(header, arrParam)
```

Remarks

After this method call the Event Logger will raise the event [OnNewEvent \[► 145\]](#) on all client that implement the event interface [ITcEventLogEvents \[► 144\]](#) (VB: Dim WithEvents).

The method can just be used to issue an alarm. A second call to reset the alarm is not possible with this method. To change the state of an already issued alarm we need the event object itself. For more convenience the method [ITcEventLogC3 \[▶ 141\]::ReportEventEx \[▶ 143\]](#) directly returns the new created event object.

Also see about this

- ITcEventLogC [▶ 137]
- TcEventHeader [▶ 181]

9.3.4 ITcEventLogC3

The ITcEventLogC is an interface of the main class of the [TcEventLogger \[▶ 127\]](#). The interface provides the features add new alarms. The interface derives from [ITcEventLogC2 \[▶ 139\]](#), [ITcEventLogC \[▶ 137\]](#) and **IUnknown** so that it could be used by all languages that support COM custom interfaces.

Methods in Vtable Order

Table 18: Methods in Vtable Order

IUnknown Methods	Description
QueryInterface	returns a pointer to the interface you query for
AddRef	Increments the reference counter
Release	Decrements the reference counter

ITcEventLogC [▶ 137] Methods	Description
ReportEvent [▶ 129]	This method is used to issue a new event from the client.
EnumEventSources	Deprecated. Was used on the deprecated structured storage based formatter to loop through the list of event source storages.
OpenEventSource	Deprecated. Was used on the deprecated structured storage based formatter to return one particular event source storages.

ITcEventLogC2 [▶ 139] Methods	Description
CreateEventSource	Deprecated. Was used on the deprecated structured storage based formatter to create a new event source storage.
RemoveEventSource	Deprecated. Was used on the deprecated structured storage based formatter to remove an existing event source storage.
RenameEventSource	Deprecated. Was used on the deprecated structured storage based formatter to rename an existing event source storage.

ITcEventLogC3 Methods	Description
ReportEventEx [▶ 143]	This method is used to issue a new event from the client. The function returns the new issued alarm for further control

9.3.4.1 ReportEvent

[ITcEventLogC \[▶ 137\]](#)

This method is used to issue an alarm from a COM client like the HMI program to the TcEventLogger.

```
HRESULT ReportEvent([in] TcEventHeader* pEventHead,
                   [in] SAFEARRAY(VARIANT)* pEventData
);
```

Parameters

pEventHead

[in] Pointer to an object of the type TcEventHeader. The object represents the alarm configuration.

eventData

[in] A pointer to a safearray that represents the event arguments. In the configuration of the event formatter, it's defined how to place the event arguments into the displayed event message.

The following data types are supported by the standard formatter.

int (16bit), long (32bit), float(32bit), double(64bit), string(BSTR)

Return Values

S_OK

Function was successfully called

TCEVENTERR_ISSIGNALLED

the event with this source id and event id was already signaled.

E_POINTER

pEventHead or pEventData were no valid pointers.

Visual Basic sample code

```
' get the one and only event logger
Dim evtLogger As TCEVENTLOGGERLib.TcEventLog
Set evtLogger = New TCEVENTLOGGERLib.TcEventLog

' cast to ITcEventLogC interface
Dim evtLoggerC As TCEVENTLOGGERLib.ITcEventLogC
Set evtLoggerC = evtLogger

' create event header
Dim header As TcEventHeader
header.nClass = TcEventClass.TCEVENTCLASS_ALARM ' event class
header.nPriority = TcEventPriority.TCEVENTPRIO_IMPLICIT ' event priority
header.dwFlags = TcEventFlags.TCEVENTFLAG_LOG ' event flags
header.dwUserData = 0 'user data
header.nId = 1 ' event id
header.nInvokeId = 0 'invoke id
header.fDate = Now ' event date and time
header.nMs = 123 ' milli second part of the event date and time
header.varSource = 1 ' event source id
header.szFmtProgId = "TcEventFormatter.TcXmlFormatter" ' event prog id

' create event params
Dim arrParam() As Variant
ReDim arrParam(0 To 1) As Variant
arrParam(0) = 1234 ' first parameter as long
arrParam(1) = 1234.777 ' second parameter as double

' log the event
Call evtLoggerC.ReportEvent(header, arrParam)
```

Remarks

After this method call the Event Logger will raise the event [OnNewEvent \[► 145\]](#) on all client that implement the event interface [_ITcEventLogEvents \[► 144\]](#) (VB: Dim WithEvents).

The method can just be used to issue an alarm. A second call to reset the alarm is not possible with this method. To change the state of an already issued alarm we need the event object itself. For more convenience the method [ITcEventLogC3 \[▶ 141\]::ReportEventEx \[▶ 143\]](#) directly returns the new created event object.

Also see about this

[TcEventHeader \[▶ 181\]](#)

9.3.4.2 ReportEventEx

[ITcEventLogC3 \[▶ 141\]](#)

This method is used to issue an alarm from a COM client like the HMI program to the TcEventLogger.

```
HRESULT ReportEventEx([in] TcEventHeader* pEventHead,
                    [in] SAFEARRAY(VARIANT)* pEventData
                    [out, retval] [out, retval] ITcEvent** pEvent
);
```

Parameters

pEventHead

[in] Pointer to an object of the type TcEventHeader. The object represents the alarm configuration.

eventData

[in] A pointer to a safe array that represents the event arguments. In the configuration of the event formatter it's defined how to place the event arguments into the displayed event message. The following data types are supported by the standard formatter. int (16bit), long (32bit), float(32bit), double(64bit), string(BSTR)

pEvent

[out, retval] A pointer to an ITcEvent pointer that receives the new logged event object

Return Values

S_OK

Function was successfully called

TCEVENTERR_ISSIGNALLED

the event with this source id and event id was already signaled.

E_POINTER

pEventHead or pEventData or pEvent were no valid pointers.

Visual Basic sample code

```
' get the one and only event logger
Dim evtLogger As TCEVENTLOGGERLib.TcEventLog
Set evtLogger = New TCEVENTLOGGERLib.TcEventLog

' cast to ITcEventLogC3 interface
Dim evtLoggerC As TCEVENTLOGGERLib.ITcEventLogC3
Set evtLoggerC = evtLogger

' create event header
Dim header As TcEventHeader
header.nClass = TcEventClass.TCEVENTCLASS_ALARM ' event class
header.nPriority = TcEventPriority.TCEVENTPRIO_IMPLICIT ' event priority
header.dwFlags = TcEventFlags.TCEVENTFLAG_LOG ' event flags
header.dwUserData = 0 'user data
header.nId = 1 ' event id
header.nInvokeId = 0 'invoke id
header.fDate = Now ' event date and time
header.nMs = 123 ' milli second part of the event date and time
```

```

header.varSource = 1 ' event source id
header.szFmtProgId = "TcEventFormatter.TcXmlFormatter" ' event prog id

' create event params
Dim arrParam() As Variant
ReDim arrParam(0 To 1) As Variant
arrParam(0) = 1234 ' first parameter as long
arrParam(1) = 1234.777 ' second parameter as double

' log the event
Dim evt As TcEvent
Set evt = evtLoggerC.ReportEventEx(header, arrParam)

' reset the event
evt.Reset

```

Remarks

After this method call the Event Logger will raise the event [OnNewEvent \[▶ 145\]](#) on all client that implement the event interface [ITcEventLogEvents \[▶ 144\]](#) (VB: Dim WithEvents).

Also see about this

 [TcEventHeader \[▶ 181\]](#)

9.3.5 ITcEventLogEvents

The [_ITcEventLogEvents](#) is the standard event interface of the main class of the [TcEventLogger \[▶ 127\]](#). The interface provides the callback method to keep a client in sync with the actual state of the TcEvent Logger. The interface derives from **IDispatch** so that it could be used by all languages that support COM automation interfaces. IDispatch itself derives from IUnknown.

Table 19: Methods in Vtable Order

IUnknown Methods	Description
QueryInterface	returns a pointer to the interface you query for
AddRef	Increments the reference counter
Release	Decrements the reference counter

IDispatch Methods	Description
GetIDsOfNames	Maps a single member name and an optional set of parameter names to a corresponding set of integer dispatch identifiers (DISPIDs), which can then be used on subsequent calls to IDispatch::Invoke .
GetTypeInfo	Retrieves the type information for an object.
GetTypeInfoCount	Retrieves the number of type information interfaces that an object provides, either 0 or 1.
Invoke	Provides access to properties and methods exposed by an object.

ITcEventLog Methods	Description
OnNewEvent [▶ 145]	This event method is called when a new event was issued
OnConfirmEvent [▶ 171]	This event method is called when an event was confirmed
OnResetEvent [▶ 146]	This event method is called when an event was reset
OnSignalEvent [▶ 174]	This event method is called when an event was signaled
OnActiveEventsCleared [▶ 148]	This event method is called when all active events are cleared
OnLoggedEventsCleared [▶ 148]	This event method is called when all logged events are cleared
OnClearEvent [▶ 149]	This event method is called when one events is cleared

ITcEventLog Methods	Description
OnShutdown [▶ 149]	This event method is called when the operating system shuts down.
On [▶ 149] NewEventConfiguration [▶ 150]	This event method is called when a new configuration was loaded due to a call to <code>ITcEventLog::Reset</code> .

9.3.5.1 OnNewEvent

[_ITcEventLogEvents](#) [[▶ 144](#)]

This event method is called on all clients that implement and connect this event interface when a new event is issued. The event had been issued from the PLC function block ADSLOGEVENT by setting the flag from FALSE to TRUE (or other ADS device) or the from a call to a Report Event Function like [ITcEventLogC](#) [[▶ 137](#)]`::ReportEvent` [[▶ 138](#)], [ITcEventC3](#) [[▶ 141](#)]`::ReportEventEx` [[▶ 143](#)] or [ITcEventLog](#) [[▶ 129](#)]`::ReportEvent` [[▶ 129](#)].

```
HRESULT OnNewEvent([in] IDispatch*
  evtObj);
```

Parameters

evtObj

[in] **IDispatch** pointer to a new event object of the class [TvEvent](#) [[▶ 128](#)]. The event object provides the full access to the new issue event. Each client receives a reference to the event object and no copy.

Return Values

S_OK

Function was successfully called

E_POINTER

evtObj was no valid pointer.

Visual Basic sample code

```
Option Explicit

Dim WithEvents evtLogger As TCEVENTLOGGERLib.TcEventLog

' form load
Private Sub Form_Load()
    ' get the one and only event logger
    Set evtLogger = New TcEventLog
End Sub

' event method
Private Sub evtLogger_OnNewEvent(ByVal evtObj As Object)
    Dim evt As TcEvent
    Set evt = evtObj
    ' print the message string in English
    Debug.Print evt.GetMsgString(1033)
End Sub
```

9.3.5.2 OnConfirmEvent

[_ITcEventLogEvents](#) [[▶ 144](#)]

This event method is called on all clients that implement and connect this event interface when a event is confirmed. The event was confirmed from the PLC function block by setting **EventQuit=TRUE** (or other ADS device) or the from a call to [ITcEvent \[▶ 157\]::Confirm \[▶ 167\]](#)

```
HRESULT OnConfirmEvent([in] IDispatch*
evtObj);
```

Parameters

evtObj

[in] **IDispatch** pointer to a event object of the class [TcEvent \[▶ 128\]](#). The event object provides the full access to the confirmed event. Each client receives a reference to the event object and no copy.

Return Values

S_OK

Function was successfully called

E_POINTER

evtObj was no valid pointer.

Visual Basic sample code

```
Option Explicit

Dim WithEvents evtLogger As TCEVENTLOGGERLib.TcEventLog

' form load
Private Sub Form_Load()
    ' get the one and only event logger
    Set evtLogger = New TcEventLog
End Sub

' event method
Private Sub evtLogger_OnConfirmEvent(ByVal evtObj As Object)
    Dim evt As TcEvent
    Set evt = evtObj
    ' print the date and time of the confirmation
    Debug.Print evt.DateConfirmed
End Sub
```

Remarks

9.3.5.3 OnResetEvent

[ITcEventLogEvents \[▶ 144\]](#)

This event method is called on all clients that implement and connect this event interface when a event is reset. The event was reset from the PLC function block ADSLOGEVENT by setting **Event** from **TRUE** to **FALSE** (or other ADS device) or the from a call to [ITcEvent \[▶ 157\]::Reset \[▶ 168\]](#)

```
HRESULT OnResetEvent([in] IDispatch* evtObj);
```

Parameters

evtObj

[in] **IDispatch** pointer to a event object of the class [TvEvent \[▶ 128\]](#). The event object provides the full access to the reset event. Each client receives a reference to the event object and no copy.

Return Values

S_OK

Function was successfully called

E_POINTER

evtObj was no valid pointer.

Visual Basic sample code

```
Option Explicit

Dim WithEvents evtLogger As TCEVENTLOGGERLib.TcEventLog

' form load
Private Sub Form_Load()
    ' get the one and only event logger
    Set evtLogger = New TcEventLog
End Sub

' event method
Private Sub evtLogger_OnResetEvent(ByVal evtObj As Object)
    Dim evt As TcEvent
    Set evt = evtObj
    ' print the date and time of the reset
    Debug.Print evt.DateReset
End Sub
```

9.3.5.4 OnSignalEvent

[ITcEventLogEvents](#) [[▶ 144](#)]

This event method is called on all clients that implement and connect this event interface when a event is signaled. Only events that have to be confirmed could be signaled. A signaled event, is waiting for it's confirmation and was reset before. The event was signaled from the PLC function block ADSLOGEVENT by setting **Event** from **FALSE** to **TRUE** (or other ADS device) or the from a call to [ITcEvent](#) [[▶ 157](#)][::Signal](#) [[▶ 174](#)]

```
HRESULT OnSignalEvent([in] IDispatch* evtObj);
```

Parameters

evtObj

[in] **IDispatch** pointer to a event object of the class [TvEvent](#) [[▶ 128](#)]. The event object provide the full access to the signaled event. Each client receive a reference to the event object and no copy.

Return Values

S_OK

Function was successfully called

E_POINTER

evtObj was no valid pointer.

Visual Basic sample code

```
Option Explicit

Dim WithEvents evtLogger As TCEVENTLOGGERLib.TcEventLog

' form load
Private Sub Form_Load()
    ' get the one and only event logger
    Set evtLogger = New TcEventLog
End Sub

' event method
Private Sub evtLogger_OnSignalEvent(ByVal evtObj As Object)
    Dim evt As TcEvent
```

```

Set evt = evtObj
' print the date and time
Debug.Print evt.Date
End Sub

```

9.3.5.5 OnActiveEventsCleared

[ITcEventLogEvents](#) [[▶ 144](#)]

This event method is called on all clients that implement and connect this event interface when all active events are cleared. This method is called after a call of [ITcEventLog](#) [[▶ 129](#)][::ClearActiveEvents](#) [[▶ 131](#)] or from the PLC function block ADSCLEAREVENTS with the iMode set to TCEVENTLOGIOFFS_CLEARACTIVE or TCEVENTLOGIOFFS_CLEARALL and on shutdown of the TwinCAT system.

```
HRESULT OnActiveEventsCleared();
```

Parameters

S_OK

Function was successfully called

Visual Basic sample code

```

Option Explicit

Dim WithEvents evtLogger As TCEVENTLOGGERLib.TcEventLog

' form load
Private Sub Form_Load()
' get the one and only event logger
Set evtLogger = New TcEventLog
End Sub

' event method
Private Sub evtLogger_OnActiveEventsCleared()
Debug.Print "Active Event are cleared!"
End Sub

```

9.3.5.6 OnLoggedEventsCleared

[ITcEventLogEvents](#) [[▶ 144](#)]

This event method is called on all clients that implement and connect this event interface when all logged events are cleared. This method is called after a call of [ITcEventLog](#) [[▶ 129](#)][::ClearLoggedEvents](#) [[▶ 137](#)] or from the PLC function block ADSCLEAREVENTS with the iMode set to TCEVENTLOGIOFFS_CLEARLOGGED or TCEVENTLOGIOFFS_CLEARALL.

```
HRESULT OnLoggedEventsCleared();
```

Parameters

S_OK

Function was successfully called

Visual Basic sample code

```

Option Explicit

Dim WithEvents evtLogger As TCEVENTLOGGERLib.TcEventLog

' form load
Private Sub Form_Load()
' get the one and only event logger
Set evtLogger = New TcEventLog
End Sub

```

```
' event method
Private Sub evtLogger_OnLoggedEventsCleared()
    Debug.Print "Logged Event are cleared!"
End Sub
```

9.3.5.7 OnClearEvent

[ITcEventLogEvents](#) [[▶ 144](#)]

This event method is called on all clients that implement and connect this event interface when an event is cleared. This method is called on all events for one source id after a call of the PLC function block ADSCLEAREVENTS with the iMode set to TCEVENTLOGIOFFS_CLEAREVENTSRCID.

```
HRESULTOnClearEvent (ByVal
    evtObj As Object);
```

Parameters

evtObj

[in] IDispatch pointer to a new event object of the class TvEvent. The event object provides the full access to the cleared event. Each client receives a reference to the event object and no copy.

S_OK

Function was successfully called

Visual Basic sample code

```
Option Explicit

Dim WithEvents evtLogger As TCEVENTLOGGERLib.TcEventLog

' form load
Private Sub Form_Load()
    ' get the one and only event logger
    Set evtLogger = New TcEventLog
End Sub

' event method
Private Sub evtLogger_OnClearEvent (ByVal evtObj As Object)
    Dim evt As TcEvent
    Set evt = evtObj
    ' print the message string in English
    Debug.Print evt.GetMsgString(1033)
End Sub
```

9.3.5.8 OnShutdown

[ITcEventLogEvents](#) [[▶ 144](#)]

This event method is called when the operating system shuts down. This could be used by DCOM clients to release the connection.

```
HRESULTOnShutdown ([in] VARIANT
    shutdownParm);
```

Parameters

shutdownParm

[in] A variant that describes the shutdown reason (in our case always REASON_SYSSHUTDOWN).

The items of the array are explained in the following table.

Requirements

index	TcEventHeader [▶ 181] item	Description
0	REASON_INVALID	Invalid
1	REASON_TCSTOP	The TwinCAT system shuts down
2	REASON_SYSSHUTDOWN	The operating systems shuts down

S_OK

Function was successfully called

Visual Basic sample code

```
Option Explicit

Dim WithEvents evtLogger As TCEVENTLOGGERLib.TcEventLog

' form load
Private Sub Form_Load()
    ' get the one and only event logger
    Set evtLogger = New TcEventLog
End Sub

' event method
Private Sub evtLogger_OnShutdown(ByVal shutdownParm As Variant)
    Debug.Print shutdownParm
End Sub
```

9.3.5.9 OnNewEventConfiguration

[ITcEventLogEvents \[▶ 144\]](#)

This event method is called on all clients that implement and connect this event interface when all active events are cleared. This methods is called whenever the ITcEventLog::Reset Method was called and any Eventformatter has been reinitialized. The method is not called when ITcEventLog::Reset was called but no Formatter was reloaded.

```
HRESULT OnNewEventConfiguration();
```

Parameters

S_OK

Function was successfully called

C# sample code

```
using TCEVENTLOGGERLib;
...
TcEventLog tcEventLogger = new TcEventLog();
tcEventLogger.OnNewEventConfiguration += new
_ITcEventLogEvents_OnNewEventConfigurationEventHandler(tcEventLogger_OnNewEventConfiguration);
...
void tcEventLogger_OnNewEventConfiguration()
{
    Console.WriteLine("tcEventLogger_OnNewEventConfiguration");
}
```

9.3.6 ITcEnumEvents

The ITcEnumEvents interface is used to iterate through a list of events. The interface derives from **IUnknown** so that it could be used by all languages that support COM custom interfaces.

Table 20: Methods and Properties in Vtable Order

Unknown Methods	Description
QueryInterface	returns a pointer to the interface you query for
AddRef	Increment the reference counter
Release	Decrements the reference counter

ITcEnumEvents Methods	Description
Next [▶ 151]	Returns the next number of events in the enumeration sequence
Skip [▶ 152]	Skips several events in the enumeration sequence
Reset [▶ 153]	Resets the enumeration sequence to the first element
Clone [▶ 153]	Clones the enumeration object
VBNext [▶ 154]	Returns the next number of events wrapped in an safearray of the enumeration sequence

9.3.6.1 Next

[ITcEnumEvents \[▶ 150\]](#)

This method returns the next number of events in the enumeration sequence

```
HRESULT Next(
    [in] long celt,
    [out, size_is(celt), length_is(*pceltFetched)] IDispatch** ppElements,
    [out, retval] long *pceltFetched);
```

Parameter

celt

[in] Number of requested elements of the enumeration sequence

ppElements

[out, size_is(celt), length_is(*pceltFetched)] Pointer to the first element of an array of events. Returns the by pceltFetched given number of events

pceltFetched

Returns the number of returned events.

Return Values

S_OK

Function was successfully called.

S_FALSE

Number of returned elements was less than the requested

E_POINTER

Elements or pceltFetched were no valid pointer

Visual Basic sample code

```
' get the one and only event logger
Dim evtLogger As TCEVENTLOGGERLib.TcEventLog
Set evtLogger = New TCEVENTLOGGERLib.TcEventLog

' get the active event enumeration object
Dim enumEvt As ITcEnumEvents
Set enumEvt = evtLogger.EnumActiveEvents
' try to get max events an loop through the event list
Const coMax As Long = 20
Dim i As Long
Dim nFetched As Long
Dim evt As TcEvent
Dim arrEvt(1 To coMax) As Object

Do
    nFetched = enumEvt.Next(coMax, arrEvt(1))
    For i = 1 To nFetched
        Set evt = arrEvt(i)
        ' print the event message in english
        Debug.Print evt.GetMsgString(1033)
        ' release the evt object
        Set arrEvt(i) = Nothing
    Next i
Loop While (nFetched >= coMax)
```

Remarks

The **Next** method only works from languages which pass references to array types to the COM Interface. E.g. .NET languages only pass a pointer to the dereferenced first element of an array. From such languages use the [VBNext \[► 154\]](#) method.

9.3.6.2 Skip

[ITcEnumEvents \[► 150\]](#)

This method skips the next number of events in the enumeration sequence

```
HRESULT Skip( [in] long cSkipElem);
```

Parameter

cSkipElem

[in] Number of elements to skip

Return Value

S_OK

The method was successfully called

S_FALSE

The number of skipped elements was not equal to cSkipElem

E_NOTIMPL

The method is not implemented

Remarks

The method is not implemented for the enum object returned by [ITcEventLog \[► 129\]::EnumActiveEvents \[► 132\]](#) and [ITcEventLog \[► 129\]::EnumLoggedEvents \[► 133\]](#) of the class [TcEventLog \[► 128\]](#).

9.3.6.3 Reset

[ITcEnumEvents \[▶ 150\]](#)

This method resets the enumeration sequence to the first element

```
HRESULT Reset(  
    void);
```

Parameter

Return Value

S_OK

The method was successfully called

S_FALSE

The number of skipped elements was not equal to cSkipElem

E_NOTIMPL

The method is not implemented

Remarks

The method is not implemented for the enum object returned by [ITcEventLog \[▶ 129\]::EnumActiveEvents \[▶ 132\]](#) and [ITcEventLog \[▶ 129\]::EnumLoggedEvents \[▶ 133\]](#) of the class [TcEventLog \[▶ 128\]](#).

9.3.6.4 Clone

[ITcEnumEvents \[▶ 150\]](#)

This method clones the enumeration object

```
HRESULT Clone(  
    [out] ITcEnumEvents** ppEnum);
```

Parameter

[out] Pointer of the type [ITcEnumEvents \[▶ 150\]](#) to a pointer that receives a copy of the enumeration object

Return Value

S_OK

The method was successfully called

E_POINTER

ppEnum was no valid pointer

E_NOTIMPL

The method is not implemented

Remarks

The method is not implemented for the enum object returned by [ITcEventLog \[▶ 129\]::EnumActiveEvents \[▶ 132\]](#) and [ITcEventLog \[▶ 129\]::EnumLoggedEvents \[▶ 133\]](#) of the class [TcEventLog \[▶ 128\]](#).

9.3.6.5 VbNext

[ITcEnumEvents](#) [[▶ 150](#)]

This method returns the next number of events in the enumeration sequence

```
HRESULT VbNext(
    [in] long celt,
    [in,out] SAFEARRAY(VARIANT)* elements,
    [out, retval] long *pceltFetched);
```

Parameter

celt

[in] Number of requested elements of the enumeration sequence

ppElements

[in] Pointer to a safearray that is going to be filled with number of pceltFetched event objects

pceltFetched

Returns the number of returned events.

Return Values

S_OK

Function was successfully called.

S_FALSE

Number of returned elements was less than the requested

E_POINTER

Elements or pceltFetched were no valid pointer

Visual Basic sample code

```
' get the one and only event logger
Dim evtLogger As TCEVENTLOGGERLib.TcEventLog
Set evtLogger = New TCEVENTLOGGERLib.TcEventLog

' get the active event enumeration object
Dim enumEvt As ITcEnumEvents
Set enumEvt = evtLogger.EnumActiveEvents
' try to get max events an loop through the event list
Const coMax As Long = 20
Dim i As Long
Dim nFetched As Long
Dim evt As TcEvent
Dim arrEvt() As Variant
Do
    nFetched = enumEvt.VbNext(coMax, arrEvt)
    For i = LBound(arrEvt) To UBound(arrEvt)
        Set evt = arrEvt(i)
        ' print the event message in english
        Debug.Print evt.GetMsgString(1033)
        ' release the evt object
        Set arrEvt(i) = Nothing
    Next i
Loop While (nFetched >= coMax)
```

9.3.7 ITcEnumEventEx

The ITcEnumEventsEx interface is used to iterate through a list of events. This interface is provided to use better compatibility to the .NET programming languages. It implements the IEnumVARIANT interface to provide a StandardEnumerator. If you are programming in C++ you might want to use the previous [ITcEnumEvents \[▶ 150\]](#) interface

Table 21: Methods and Properties in Vtable Order

Unknown Methods	Description
QueryInterface	returns a pointer to the interface you query for
AddRef	Increment the reference counter
Release	Decrements the reference counter

ITcEnumEventsExMethods	Description
Item [▶ 155]	Get one item in the collection
Add [▶ 156]	Returns the next number of events in the enumeration sequence
Count [▶ 156]	Returns the next number of events in the enumeration sequence
Remove [▶ 156]	Returns the next number of events in the enumeration sequence
GetEnumerator	Returns a standard IEnumerator object

Visual Studio 2005 C# Sample Code:

```
foreach (TcEvent evt in new TcEventLog().EnumActiveEventsEx())
    evt.Reset();
```

9.3.7.1 Item

[ITcEnumEvents \[▶ 150\]](#)

This method returns an Item in the collection.

```
HRESULT Item(
    [in] int Index,
    [out, retval] ITcEvent** pVal );
```

Parameter

Index

[in] Id of the event to be returned

pVal

[out] To the element at the requested index.

return Values

S_OK

Function was successfully called.

Remarks

Depending on the programming language you are using, you might rather want to use the objects standard enumerator instead of its interface methods.

9.3.7.2 Add

[ITcEnumEvents \[▶ 150\]](#)

Add one event to the collection.

```
HRESULT Add([out] ITcEvent* pEvent);
```

Parameter

[out] Pointer of an [ITcEvent \[▶ 157\]](#) to be added to the collection.

Return Value

S_OK

The method was successfully called

Remarks

Do not manually add events to the enumerator. Instead use [ITcEventLog \[▶ 129\]::EnumActiveEventsEx \[▶ 133\]](#) or [ITcEventLog \[▶ 129\]::EnumLoggedEventsEx \[▶ 135\]](#) which return a completely constructed enum object.

9.3.7.3 Count

[ITcEnumEvents \[▶ 150\]](#)

This method returns number of events in the collection

```
HRESULT Count([out, retval] long* pVal );
```

Parameter

pVal

[out, retval] long pointer to receive the value

return Values

S_OK

Function was successfully called.

Remarks

Depending on the programming language you are using, you might rather want to use the objects standard enumerator instead of its interface methods.

9.3.7.4 Remove

[ITcEnumEvents \[▶ 150\]](#)

Remove one element from the collection

```
HRESULT Remove([in] int Index );
```

Parameter

Index

[in] Index of the Item to be removed

return Values

S_OK

Function was successfully called.

Remarks

Depending on the programming language you are using, you might rather want to use the objects standard enumerator instead of its interface methods.

9.3.8 ITcEvent

The interface ITcEvent gives access to the properties of one alarm. The interface derives from **IDispatch** so that it could be used by all languages that support COM automation interfaces. IDispatch itself derives from IUnknown.

Table 22: Methods and Properties in Vtable Order

IUnknown Methods	Description
QueryInterface	returns a pointer to the interface you query for
AddRef	Increments the reference counter
Release	Decrements the reference counter

IDispatch Methods	Description
GetIDsOfNames	Maps a single member name and an optional set of parameter names to a corresponding set of integer dispatch identifiers (DISPIDs), which can then be used on subsequent calls to IDispatch::Invoke .
GetTypeInfo	Retrieves the type information for an object.
GetTypeInfoCount	Retrieves the number of type information interfaces that an object provides, either 0 or 1.
Invoke	Provides access to properties and methods exposed by an object.

ITcEvent Methods and Properties	Description
GetMsgString [▶ 158]	returns the formatted message string for the requested language
Id [▶ 159]	returns the event id.
SrcId [▶ 159]	returns the id of the event source
Invokeld [▶ 160]	returns the invoke id
Class [▶ 161]	returns the event class
Priority [▶ 161]	returns the event priority
Flags [▶ 162]	returns the event flags
Ms [▶ 162]	returns the millisecond part of the event date
SourceName [▶ 163]	returns the name of the event source for the requested language
FmtProgId [▶ 164]	returns the ProgId of the event formatter.
Date [▶ 165]	returns the event date and time
GetData [▶ 165]	returns the event data parameters
GetHeader [▶ 166]	returns the event header as safearray
Confirm [▶ 167]	confirms the alarm
Reset [▶ 168]	resets the alarm

ITcEvent Methods and Properties	Description
State [▶ 169]	returns the event state.
MustConfirm [▶ 169]	returns if the alarm must be confirmed
DateConfirmed [▶ 170]	returns the date and time of the confirmation
MsConfirmed [▶ 171]	returns the millisecond part of confirmation time
DateReset [▶ 172]	returns the date and time when the event was reset
MsReset [▶ 172]	returns the millisecond part of reset time
UserData [▶ 173]	returns the user data
EnumDocLinks [▶ 174]	returns an enumeration object that can be used to enumerated through the list of document links.
TcId	not implemented
Signal [▶ 174]	signals the alarm

9.3.8.1 GetMsgString

[ITcEvent \[▶ 157\]](#)

The method GetMsgString returns the formatted message string for the requested language. Internally it calls the method `ITcLogFormatter::GetCompleteString` of the formatter that was assigned when the event was issued from a COM client or through an ADS call to the [TcEventLog \[▶ 128\]](#) object of the `TcEventLogger`.

```
HRESULT LoggedEvents([in] long langId,
[out,retval] BSTR* msg);
```

Parameters

langId

[in] The language Id of the requested language. The requested language should be tagged by *LCIDs. The next table shows a sample for some language ids. In the configuration of the event formatter the languages are tagged by the same language id

LCID	Description
1031	German
1033	US English
1034	Spanish - Spain
1036	French

msg

[out, retval] Pointer to an BSTR string that returns the formatted string for the requested language.

Return Values

S_OK

Function was successfully called

E_POINTER

msg was no valid pointer.

Visual Basic sample code

```
' get the one and only event logger
Dim evtLogger As TCEVENTLOGGERLib.TcEventLog
Set evtLogger = New TCEVENTLOGGERLib.TcEventLog

' get the most recent active event
Dim evt As TcEvent
Set evt = evtLogger.GetLastEvent

' get the message in English
Dim strMessage As String
strMessage = evt.GetMsgString(1033)
Debug.Print strMessage
```

Remarks

Most standard formatter like the XML base formatter (TcEventFormatter.TcXmlFormatter) will try to return a string for the default language if the requested language does not exist in the configuration of the formatter.

*LCID: for more information see MSDN Library

9.3.8.2 Id

[ITcEvent \[► 157\]](#)

The property returns the event id. Together with the source id the event id describes one event.

```
HRESULT Id([out, retval] long *pVal);
```

Parameters

pVal

[out, retval] Pointer to a long value that receives the event id.

Return Values

S_OK

Function was successfully called

E_POINTER

pVal was no valid pointer.

Visual Basic sample code

```
' get the one and only event logger
Dim evtLogger As TCEVENTLOGGERLib.TcEventLog
Set evtLogger = New TCEVENTLOGGERLib.TcEventLog

' get the most recent active event
Dim evt As TcEvent
Set evt = evtLogger.GetLastEvent

' get the event id
Dim id As Long
id = evt.id
Debug.Print id
```

9.3.8.3 SrcId

[ITcEvent \[► 157\]](#)

The property returns the event source id. Together with the event id the source id it describes one event.

```
HRESULT SrcId([out, retval] long *pVal);
```

Parameters

pVal

[out, retval] Pointer to a long value that receives the source id.

Return Values

S_OK

Function was successfully called

E_POINTER

pVal was no valid pointer.

Visual Basic sample code

```
' get the one and only event logger
Dim evtLogger As TCEVENTLOGGERLib.TcEventLog
Set evtLogger = New TCEVENTLOGGERLib.TcEventLog

' get the most recent active event
Dim evt As TcEvent
Set evt = evtLogger.GetLastEvent

' get the source id
Dim id As Long
id = evt.SrcId
Debug.Print id
```

9.3.8.4 Invokeld

[ITcEvent](#) [[▶ 157](#)]

The property returns the invoke id.

```
HRESULT InvokeId([out, retval] long *pVal);
```

Parameters

pVal

[out, retval] Pointer to a long value that receives the invoke id.

Return Values

S_OK

Function was successfully called

E_POINTER

pVal was no valid pointer.

Visual Basic sample code

```
' get the one and only event logger
Dim evtLogger As TCEVENTLOGGERLib.TcEventLog
Set evtLogger = New TCEVENTLOGGERLib.TcEventLog

' get the most recent active event
```



```
Dim evt As TcEvent
Set evt = evtLogger.GetLastEvent

' get the event id
Dim id As Long
id = evt.InvokeId
Debug.Print id
```

9.3.8.5 Class

[ITcEvent](#) [[▶ 157](#)]

The property returns the class of the alarm, like alarm, warning, hint. The alarm classes are described through the enum [TcEventClass](#) [[▶ 179](#)].

```
HRESULT Class([out, retval] long *pVal);
```

Parameters

pVal

[out, retval] pointer to a long value that receives the event class.

Return Values

S_OK

Function was successfully called

E_POINTER

pVal was no valid pointer.

Visual Basic sample code

```
' get the one and only event logger
Dim evtLogger As TCEVENTLOGGERLib.TcEventLog
Set evtLogger = New TCEVENTLOGGERLib.TcEventLog

' get the most recent active event
Dim evt As TcEvent
Set evt = evtLogger.GetLastEvent

' get the event class
Dim class As TcEventClass
class = evt.Class
```

9.3.8.6 Priority

[ITcEvent](#) [[▶ 157](#)]

The property returns the priority of the event. The event priority is described through the enum [TcEventPriority](#) [[▶ 161](#)].

```
HRESULT Priority([out, retval] long *pVal);
```

Parameters

pVal

[out, retval] Pointer to a long value that receives the event priority.

Return Values

S_OK

Function was successfully called

E_POINTER

pVal was no valid pointer.

Visual Basic sample code

```
' get the one and only event logger
Dim evtLogger As TCEVENTLOGGERLib.TcEventLog
Set evtLogger = New TCEVENTLOGGERLib.TcEventLog

' get the most recent active event
Dim evt As TcEvent
Set evt = evtLogger.GetLastEvent

' get the event priority
Dim prio As TcEventPriority
prio = evt.Priority
```

9.3.8.7 Flags

[ITcEvent](#) [► 157]

The property returns the event flag. The event flags are described through the enum [TcEventFlags](#) [► 180].

```
HRESULT Flags([out, retval] long *pVal);
```

Parameters

pVal

[out, retval] Pointer to a long value that receives the event flags.

Return Values

S_OK

Function was successfully called

E_POINTER

pVal was no valid pointer.

Visual Basic sample code

```
' get the one and only event logger
Dim evtLogger As TCEVENTLOGGERLib.TcEventLog
Set evtLogger = New TCEVENTLOGGERLib.TcEventLog

' get the most recent active event
Dim evt As TcEvent
Set evt = evtLogger.GetLastEvent

' get the event flags
Dim flags As TcEventFlags
flags = evt.Flags
```

9.3.8.8 Ms

[ITcEvent](#) [► 157]

The property returns the millisecond part of the event date and time. The event date and time is the time when the event was issued.

```
HRESULT Ms([out, retval] long *pVal);
```

Parameters

pVal

[out, retval] Pointer to a long value that receives the millisecond part of the event date and time.

Return Values

S_OK

Function was successfully called

E_POINTER

pVal was no valid pointer.

Visual Basic sample code

```
' get the one and only event logger
Dim evtLogger As TCEVENTLOGGERLib.TcEventLog
Set evtLogger = New TCEVENTLOGGERLib.TcEventLog

' get the most recent active event
Dim evt As TcEvent
Set evt = evtLogger.GetLastEvent

' get the millisecond part
Dim timeMs As Long
timeMs = evt.Ms
Debug.Print timeMs
```

9.3.8.9 SourceName[ITcEvent](#) [[▶ 157](#)]

The property SourceName returns the formatted source name for the requested language. Internally it calls the method `ITcLogFormatter::GetSourceName` of the formatter that was assigned when the event was issued from a COM client or through an ADS call to the [TcEventLog](#) [[▶ 128](#)] object of the TcEventLogger.

```
HRESULT SourceName([in] long langId, [out,retval] BSTR* szName);
```

Parameters

langId

[in] The language Id of the requested language. The requested language should be tagged by *LCIDs. The next table shows a sample for some language ids. In the configuration of the event formatter the languages are tagged by the same language id

LCID	Description
1031	German
1033	US English
1034	Spanish - Spain
1036	French

szName

[out, retval] Pointer to an BSTR string that returns the formatted string for the event source name.

Return Values

S_OK

Function was successfully called

E_POINTER

szName was no valid pointer.

Visual Basic sample code

```
' get the one and only event logger
Dim evtLogger As TCEVENTLOGGERLib.TcEventLog
Set evtLogger = New TCEVENTLOGGERLib.TcEventLog

' get the most recent active event
Dim evt As TcEvent
Set evt = evtLogger.GetLastEvent

' get the source name in English
Dim name As String
name = evt.SourceName(1033)
Debug.Print name
```

Remarks

Most standard formatter like the XML base formatter (TcEventFormatter.TcXmlFormatter) will try to return a string for the default language if the requested language does not exist in the configuration of the formatter.

*LCID: for more information see MSDN Library

9.3.8.10 FmtProgId

[ITcEvent](#) [[▶ 157](#)]

The property returns the Formatter ProgId. The ProgId is the link to the chosen formatter. The formatter was chosen when the event was issued from the PLC function block (or other ADS device) or from a call to a Report EventFunktion like [IT](#) [[▶ 137](#)][cEventLogC](#) [[▶ 137](#)][::ReportEvent](#) [[▶ 138](#)], [ITcEventC3](#) [[▶ 141](#)][::ReportEventEx](#) [[▶ 143](#)] or [ITcEventLog](#) [[▶ 129](#)][::ReportEvent](#) [[▶ 129](#)].

```
HRESULT FmtProgId([out, retval] BSTR *pVal);
```

Parameters

pVal

[out, retval] Pointer to an BSTR String that receives the formatter ProgId.

Return Values

S_OK

Function was successfully called

E_POINTER

pVal was no valid pointer.

Visual Basic sample code

```
' get the one and only event logger
Dim evtLogger As TCEVENTLOGGERLib.TcEventLog
Set evtLogger = New TCEVENTLOGGERLib.TcEventLog

' get the most recent active event
Dim evt As TcEvent
Set evt = evtLogger.GetLastEvent

' get the formatter ProgId
```

```
Dim progId As String
progId = evt.FmtProgId
Debug.Print progId
```

9.3.8.11 Date

[ITcEvent](#) [[▶ 157](#)]

The property returns the event date and time. The event date and time is the time when the event was issued.

```
HRESULT Date([out, retval] DATE *pVal);
```

Parameters

pVal

[out, retval] Pointer to a long value that receives the event date and time.

Return Values

S_OK

Function was successfully called

E_POINTER

pVal was no valid pointer.

Visual Basic sample code

```
' get the one and only event logger
Dim evtLogger As TCEVENTLOGGERLib.TcEventLog
Set evtLogger = New TCEVENTLOGGERLib.TcEventLog

' get the most recent active event
Dim evt As TcEvent
Set evt = evtLogger.GetLastEvent

' get the event date and time
Dim dateTime As Date
dateTime = evt.Date
Debug.Print dateTime
```

9.3.8.12 GetData

[ITcEvent](#) [[▶ 157](#)]

The property returns a safearray of data that was append to the event when it was logged. The data can be included into the event message by the formatter. The data was append when the event was issued from the PLC function block (or other ADS device) or from a call to a Report Event Function like [IT](#) [[▶ 137](#)][cEventLogC](#) [[▶ 137](#)][::ReportEvent](#) [[▶ 138](#)], [ITcEventC3](#) [[▶ 141](#)][::ReportEventEx](#) [[▶ 143](#)] or [ITcEventLog](#) [[▶ 129](#)][::ReportEvent](#) [[▶ 129](#)].

```
HRESULT GetData([in,out] SAFEARRAY(VARIANT) *
data);
```

Parameters

data

[out, retval] Pointer to a safearray of variants that receives the event data

Return Values

S_OK

Function was successfully called

E_POINTER

data was no valid pointer.

Visual Basic sample code

```
' get the one and only event logger
Dim evtLogger As TCEVENTLOGGERLib.TcEventLog
Set evtLogger = New TCEVENTLOGGERLib.TcEventLog

' get the most recent active event
Dim evt As TcEvent
Set evt = evtLogger.GetLastEvent

' get the event data
Dim data() As Variant
Call evt.GetData(data)

' loop through the list of data
Dim i As Long
For i = LBound(data) To UBound(data)
    Debug.Print data(i)
Next i
```

9.3.8.13 GetHeader

[ITcEvent](#) [[▶ 157](#)]

The property returns a safearray that represents the same information as [TcEventHeader](#) [[▶ 181](#)]. The header data was append when the event was issued from the PLC function block (or other ADS device) or from a call to a Report Event Function like [IT](#) [[▶ 137](#)][cEventLogC](#) [[▶ 137](#)][::ReportEvent](#) [[▶ 138](#)], [ITcEventC3](#) [[▶ 141](#)][::ReportEventEx](#) [[▶ 143](#)] or [ITcEventLog](#) [[▶ 129](#)][::ReportEvent](#) [[▶ 129](#)].

```
HRESULT GetHeader([in,out] SAFEARRAY(VARIANT)* data);
```

Parameters

data

[in] Pointer to a safearray with the bounds 0 to 9 that represents the event header. The array represents the same information as [TcEventHeader](#). The items of the array are explained in the following table.

index	TcEventHeader [▶ 181] item	Data Type	Description
0	nClass	TcEventClass [▶ 179]	The class of the alarm, like alarm, warning, hint
1	nPriority	TcEventPriority [▶ 180] (long)	The event priority. Since now the priority is always: TcEventPriority.TCEVENTPRIO_IMPLICIT=0
2	dwFlags	TcEventFlags [▶ 180] (long)	The flags control the behavior of the alarm. That could be combined by a OR junction.
3	dwUserData	long	A spare variable that can be used by the user.
4	nId	long	The event id, that unique represents this type of event for this one Event Source.
5	nInvokeld	long	
6	fDate	VARIANT-DATE	The date and time when this event occur
7	nMs	long	The millisecond part of the event date

index	TcEventHeader [▶ 181] item	DataType	Description
8	varSource	variant	The source id source name. The event source is used to distinguish different devices. Like I/O Event, or different parts of the PLC program.
9	szFmtProgId	BSTR string	The PrgId of the formatter. If we want to use the standard XML formatter the string should bet set to "TcEventFormatter.TcXmlFormatter"

Return Values

S_OK

Function was successfully called

E_POINTER

data was no valid pointer.

Visual Basic sample code

```
' get the one and only event logger
Dim evtLogger As TCEVENTLOGGERLib.TcEventLog
Set evtLogger = New TCEVENTLOGGERLib.TcEventLog

' get the most recent active event
Dim evt As TcEvent
Set evt = evtLogger.GetLastEvent

' get the event header
Dim data() As Variant
Call evt.GetHeader(data)

' loop though the list of data
Dim i As Long
For i = LBound(data) To UBound(data)
    Debug.Print data(i)
Next i
```

Remarks

For C++ and Visual Basic programmers, it's more convenient to use the method [ITcEventC \[▶ 137\]::GetHeader](#)

9.3.8.14 Confirm

[ITcEvent \[▶ 157\]](#)

The method is used to confirm an alarm. Alarm that had be issued from the PLC function block (or other ADS device) or from a call to a Report EventFunktion like [IT \[▶ 137\]cEventLogC \[▶ 137\]::ReportEvent \[▶ 138\]](#), [ITcEventC3 \[▶ 141\]::ReportEventEx \[▶ 143\]](#) or [ITcEventLog \[▶ 129\]::ReportEvent \[▶ 129\]](#) with the [TcEventFlags \[▶ 180\]](#) TCEVENTFLAG_REQMUSTCON set have to be confirmed by a **Confirm** method call from COM client or from the PLC.

```
HRESULT Confirm([in] long code);
```

Parameters

pVal

[in] The confirmation code that is described through the enum TcEventConCodes

Return Values

S_OK

Function was successfully called

TCEVENTERR_NOCONFIRM

The error was no confirmable error (TcEventFlags TCEVENTFLAG_REQMUSTCON was not set)

Visual Basic sample code

```
' get the one and only event logger
Dim evtLogger As TCEVENTLOGGERLib.TcEventLog
Set evtLogger = New TCEVENTLOGGERLib.TcEventLog

' get the most recent active event
Dim evt As TcEvent
Set evt = evtLogger.GetLastEvent

' confirm the alarm
Call evt.Confirm(TcEventConCodes.TCEVENTCON_OK)
```

Remarks

After this method call the Event Logger will raise the event [OnConfirmEvent \[▶ 145\]](#) on all client that implement the event interface [ITcEventLogEvents \[▶ 144\]](#) (VB: Dim WithEvents). If the alarm was issued by the PLC the TCEVENTSTATE_CONFIRMED will be flagged at the EventState output of the PLC function block.

9.3.8.15 Reset[ITcEvent \[▶ 157\]](#)

The method is used to reset an event. Events that had be issued from the PLC function block (or other ADS device) or from a call to a Report Event Function like [IT \[▶ 137\]cEventLogC \[▶ 137\]::ReportEvent \[▶ 138\]](#), [ITcEventC3 \[▶ 141\]::ReportEventEx \[▶ 143\]](#) or [ITcEventLog \[▶ 129\]::ReportEvent \[▶ 129\]](#) could be reset by this method call are directly from the PLC function block that issues the alarm.

If the event is reset the timestamps **DateReset + MsReset** will be assigned to the actual date + time.

```
HRESULT Reset();
```

Parameters**Return Values**

S_OK

Function was successfully called

Visual Basic sample code

```
' get the one and only event logger
Dim evtLogger As TCEVENTLOGGERLib.TcEventLog
Set evtLogger = New TCEVENTLOGGERLib.TcEventLog

' get the most recent active event
Dim evt As TcEvent
Set evt = evtLogger.GetLastEvent

' reset the alarm
Call evt.Reset
```


Remarks

After this method call the Event Logger will raise the event [OnResetEvent \[▶ 146\]](#) on all clients that implement the event interface [ITcEventLogEvents \[▶ 144\]](#) (VB: Dim WithEvents). If the alarm was issued by the PLC the TCEVENTSTATE_RESET will be flagged at the EventState output of the PLC function block.

9.3.8.16 State

[ITcEvent \[▶ 157\]](#)

The property returns the event status. The event state is described through the enum [TcEventStates \[▶ 181\]](#)

```
HRESULT State([out, retval] long *pVal);
```

Parameters

pVal

[out, retval] Pointer to a long value that receives the event state with the values of the enum TcEventStates

Return Values

S_OK

Function was successfully called

E_POINTER

pVal was no valid pointer.

Visual Basic sample code

```
' get the one and only event logger
Dim evtLogger As TCEVENTLOGGERLib.TcEventLog
Set evtLogger = New TCEVENTLOGGERLib.TcEventLog

' get the most recent active event
Dim evt As TcEvent
Set evt = evtLogger.GetLastEvent

' get the event state
Dim states As TcEventStates
states = evt.state
```

9.3.8.17 MustConfirm

[ITcEvent \[▶ 157\]](#)

The property returns if the event has to be confirmed. This flag is true if the event was issued from the PLC function block (or other ADS device) or from a call to a Report Event Function like [IT \[▶ 137\]cEventLogC \[▶ 137\]::ReportEvent \[▶ 138\]](#), [ITcEventC3 \[▶ 141\]::ReportEventEx \[▶ 143\]](#) or [ITcEventLog \[▶ 129\]::ReportEvent \[▶ 129\]](#) with the [TcEventFlags \[▶ 180\]](#) TCEVENTFLAG_REQMUSTCON set.

```
HRESULT MustConfirm([out, retval] BOOL *pVal);
```

Parameters

pVal

[out, retval] Pointer to a BOOL value that receives the confirmation flag. It's false if the value is 0, it's true if the value is not equal to 0.

Return Values

S_OK

Function was successfully called

E_POINTER

pVal was no valid pointer.

Visual Basic sample code

```
' get the one and only event logger
Dim evtLogger As TCEVENTLOGGERLib.TcEventLog
Set evtLogger = New TCEVENTLOGGERLib.TcEventLog

' get the most recent active event
Dim evt As TcEvent
Set evt = evtLogger.GetLastEvent

' get the MustConfirm flag
Dim mustConf As Long
mustConf = evt.mustConfirm
If mustConf <> 0 Then
    Debug.Print "Must Confirmed!"
Else
    Debug.Print "Must Not Confirmed!"
End If
```

Remarks

The return value is not from the type VARIANT_BOOL the Visual Basic type BOOLEAN. In Visual Basic it occurs as a long value. We must check if it's not equal 0 if the flag is set.

9.3.8.18 DateConfirmed

[ITcEvent](#) [[▶ 157](#)]

The property returns the event confirmation date and time. The event confirmation date and time is the time when the event was confirmed from the PLC function block by setting **EventQuit=TRUE** (or other ADS device) or the from a call to [ITcEvent](#) [[▶ 157](#)][::Confirm](#) [[▶ 167](#)]

```
HRESULT DateConfirmed([out, retval]
DATE *pVal);
```

Parameters

pVal

[out, retval] Pointer to a long value that receives the event confirmation date and time.

Return Values

S_OK

Function was successfully called

E_POINTER

pVal was no valid pointer.

Visual Basic sample code

```
' get the one and only event logger
Dim evtLogger As TCEVENTLOGGERLib.TcEventLog
Set evtLogger = New TCEVENTLOGGERLib.TcEventLog

' get the most recent active event
Dim evt As TcEvent
Set evt = evtLogger.GetLastEvent

' get the event confirmation date and time
Dim dateTime As Date
dateTime = evt.DateConfirmed
Debug.Print dateTime
```

Remarks

The event confirmation date and time is just set on those events that must be confirmed, see [ITcEvent \[► 157\]::MustConfirm \[► 169\]](#)

9.3.8.19 MsConfirmed

[ITcEvent \[► 157\]](#)

The property returns the millisecond part of the event confirmation date and time. The event confirmation date and time is the time when the event was confirmed from the PLC function block by setting **EventQuit=TRUE** (or other ADS device) or the from a call to [ITcEvent \[► 157\]::Confirm \[► 167\]](#)

```
HRESULT MsConfirmed([out, retval] long *pVal);
```

Parameters

pVal

[out, retval] Pointer to a long value that receives the millisecond part of the event confirmation date and time.

Return Values

S_OK

Function was successfully called

E_POINTER

pVal was no valid pointer.

Visual Basic sample code

```
' get the one and only event logger
Dim evtLogger As TCEVENTLOGGERLib.TcEventLog
Set evtLogger = New TCEVENTLOGGERLib.TcEventLog

' get the most recent active event
Dim evt As TcEvent
Set evt = evtLogger.GetLastEvent

' get the millisecond part of the event confirmation date and time
Dim timeMs As Long
timeMs = evt.MsConfirmed
Debug.Print timeMs
```

Remarks

The millisecond part of the event confirmation date and time is just set on those events that must be confirmed, see [ITcEvent \[► 157\]::MustConfirm \[► 169\]](#)

9.3.8.20 DateReset

[ITcEvent \[► 157\]](#)

The property returns the event reset date and time. The event reset date and time is the time when the event was reset from the PLC function block by setting **Event** from **TRUE** to **FALSE** (or other ADS device) or from a call to [ITcEvent \[► 157\]::Reset \[► 168\]](#)

```
HRESULT DateReset([out, retval] DATE *pVal);
```

Parameters

pVal

[out, retval] Pointer to a long value that receives the event reset date and time.

Return Values

S_OK

Function was successfully called

E_POINTER

pVal was no valid pointer.

Visual Basic sample code

```
' get the one and only event logger
Dim evtLogger As TCEVENTLOGGERLib.TcEventLog
Set evtLogger = New TCEVENTLOGGERLib.TcEventLog

' get the most recent active event
Dim evt As TcEvent
Set evt = evtLogger.GetLastEvent

' get the event reset date and time
Dim dateTime As Date
dateTime = evt.DateReset
Debug.Print dateTime
```

9.3.8.21 MsReset

[ITcEvent \[► 157\]](#)

The property returns the millisecond part of the event reset date and time. The event reset date and time is the time when the event was reset from the PLC function block by setting **Event** from **TRUE** to **FALSE** (or other ADS device) or from a call to [ITcEvent \[► 157\]::Reset \[► 168\]](#)

```
HRESULT MsReset([out, retval] long *pVal);
```

Parameters

pVal

[out, retval] Pointer to a long value that receives the millisecond part of the event reset date and time.

Return Values

S_OK

Function was successfully called

E_POINTER

pVal was no valid pointer.

Visual Basic sample code

```
' get the one and only event logger
Dim evtLogger As TCEVENTLOGGERLib.TcEventLog
Set evtLogger = New TCEVENTLOGGERLib.TcEventLog

' get the most recent active event
Dim evt As TcEvent
Set evt = evtLogger.GetLastEvent

' get the millisecond part of the event reset date and time
Dim timeMs As Long
timeMs = evt.MsReset
Debug.Print timeMs
```

9.3.8.22 UserData**[ITcEvent](#) [[▶ 157](#)]**

The property returns the user defined data of the event. The user data was assigned when the event was issued from the PLC function block (or other ADS device) or from a call to a Report EventFunction like [IT](#) [[▶ 137](#)][cEventLogC](#) [[▶ 137](#)][::ReportEvent](#) [[▶ 138](#)], [ITcEventC3](#) [[▶ 141](#)][::ReportEventEx](#) [[▶ 143](#)] or [ITcEventLog](#) [[▶ 129](#)][::ReportEvent](#) [[▶ 129](#)] with the [TcEventFlags](#) [[▶ 180](#)].

```
HRESULT UserData([out, retval]
long *pVal);
```

Parameters

pVal

[out, retval] Pointer to a BOOL value that receives the confirmation flag. It's false if the value is 0, it's true if the value is not equal to 0.

Return Values

S_OK

Function was successfully called

E_POINTER

pVal was no valid pointer.

Visual Basic sample code

```
' get the one and only event logger
Dim evtLogger As TCEVENTLOGGERLib.TcEventLog
Set evtLogger = New TCEVENTLOGGERLib.TcEventLog

' get the most recent active event
Dim evt As TcEvent
Set evt = evtLogger.GetLastEvent

' get the MustConfirm flag
Dim mustConf As Long
mustConf = evt.mustConfirm
If mustConf <> 0 Then
    Debug.Print "Must Confirmed!"
Else
    Debug.Print "Must Not Confirmed!"
End If
```

Remarks

The return value is not from the type of VARIANT_BOOL the Visual Basic type BOOLEAN. In Visual Basic it occurs as a long value. We must check if it's not equal 0 if the flag is set.

9.3.8.23 EnumDocLinks

[ITcEvent](#) [[▶ 157](#)]

Returns an enumeration object that is used to iterate through the list of document links. Document links are the path and file name to documents like HTML pages and pictures that could give more detailed information about one alarm for the chosen language. Internally EnumDocLinks is referred to the formatter method ITcLogFormatter::EnumDocLinks. The document links must be setup in the formatter configuration of the event.

```
HRESULTEnumDocLinks([in] long langId,
                    [out,retval] IUnknown** ppEnum
);
```

Parameters

langId

[in] The language Id of the requested language. The requested language should be tagged by *LCIDs. The next table shows a sample for some language ids. In the configuration of the event formatter the language are tagged by the same language id

LCID	Description
1031	German
1033	US English
1034	Spanish - Spain
1036	French

ppEnum

[out, retval] Pointer to the ITcEnumEventDocLink interface pointer that receives the enumeration object of the document links.

Return Values

S_OK

Function was successfully called

E_POINTER

ppEnum was no valid pointer.

Remarks

The Method does not work in Visual Basic

9.3.8.24 Signal

[ITcEvent](#) [[▶ 157](#)]

The method is used to signal a reset event again. This works only on an alarm that needs a confirmation see [ITcEvent \[▶ 157\]::MustConfirm \[▶ 169\]](#)

```
HRESULT Signal();
```

Parameters

Return Values

S_OK

Function was successfully called

Visual Basic sample code

```
' get the one and only event logger
Dim evtLogger As TCEVENTLOGGERLib.TcEventLog
Set evtLogger = New TCEVENTLOGGERLib.TcEventLog

' get the most recent active event
Dim evt As TcEvent
Set evt = evtLogger.GetLastEvent

' signal the alarm
Call evt.Signal
```

Remarks

After this method call the Event Logger will raise the event [OnSignalEvent \[▶ 147\]](#) on all client that implements the event interface [_ITcEventLogEvents \[▶ 144\]](#) (VB: Dim WithEvents).

9.3.9 ITcEventC

The interface ITcEvent gives access the properties of one alarm. The interface derives from **IUnknown** so that it could be use by all language that support COM custom interfaces.

Table 23: Methods and Properties in Vtable Order

IUnknown Methods	Description
QueryInterface	returns a pointer to the interface you query for
AddRef	Increment the reference counter
Release	Decrements the reference counter

ITcEventC Methods	Description
GetHeader [▶ 175]	returns the event Header that includes most of the event parameter in one bunch

9.3.9.1 GetHeader

[ITcEventC \[▶ 175\]](#)

This method returns the event Header that includes most of the event parameter in one bunch.

```
HRESULT GetHeader([out]TcEventHeader* pHead);
```

Parameters

pHead

[in] Pointer to an object of the type TcEventHeader. The object represents the alarm configuration.

Return Values

S_OK

Function was successfully called

E_POINTER

pVal was no valid pointer.

Visual Basic sample code

```
' get the one and only event logger
Dim evtLogger As TCEVENTLOGGERLib.TcEventLog
Set evtLogger = New TCEVENTLOGGERLib.TcEventLog

' get the most recent active event
Dim evt As ITcEventC
Set evt = evtLogger.GetLastEvent

' get the event header
Dim head As TcEventHeader
Call evt.GetHeader(head)
```

Also see about this

 TcEventHeader [[▶ 181](#)]

9.3.10 ITcEnumEventDocLink

The interface ITcEnumEventDocLink is used to enumerate the document links for one event for the chosen language. The interface derives from **IUnknown** so that it could be used by all languages that support COM custom interfaces.

Table 24: Methods in Vtable Order

IUnknown Methods	Description
QueryInterface	returns a pointer to the interface you query for
AddRef	Increments the reference counter
Release	Decrements the reference counter

ITcEnumEventDocLinkMethods	Description
Next [▶ 176]	Returns the next number of TcEventDocLink [▶ 181] objects in the enumeration sequence.
Skip [▶ 177]	Skips several TcEventDocLink [▶ 181] objects in the enumeration sequence
Reset [▶ 178]	Resets the enumeration sequence to the first element
Clone [▶ 178]	Clones the enumeration object

9.3.10.1 Next

Returns the next number of [TcEventDocLink](#) [[▶ 181](#)] objects in the enumeration sequence.


```
HRESULTNext(  
    [in] long celt,  
    [out, size_is(celt), length_is(*pceltFetched)] TcEventDocLink** ppElements,  
    [out, retval] long *pceltFetched);
```

Parameter

celt

[in] Number of requested elements of the enumeration sequence

ppElements

[out, size_is(celt), length_is(*pceltFetched)] Pointer to the first element of an array of TcEventDocLink objects. Returns the by pceltFetched given number of document links

pceltFetched

Returns the number of returned document links

Return Values

S_OK

Function was successfully called.

S_FALSE

Number of returned elements was less than the requested

E_POINTER

elements or pceltFetched were no valid pointer

Remarks

The method could not be called from Visual Basic

Also see about this

 ITcEnumEventDocLink [[▶ 176](#)]

9.3.10.2 Skip

This method skips the next number of [TcEventDocLink](#) [[▶ 181](#)] objects in the enumeration sequence

```
HRESULT Skip([in] long cSkipElem);
```

Parameter

cSkipElem

[in] Number of elements to skip

Return Value

S_OK

The method was successfully called

S_FALSE

The number of skipped elements was not equal to cSkipElem

E_NOTIMPL

The method is not implemented

Also see about this

 [ITcEnumEventDocLink \[▶ 176\]](#)

9.3.10.3 Reset

This method resets the enumeration sequence to the first element

```
HRESULT Reset(  
    void);
```

Parameter**Return Value**

S_OK

Function was successfully called

S_FALSE

The number of skipped elements was not equal to cSkipElem

E_NOTIMPL

The method is not implemented

Also see about this

 [ITcEnumEventDocLink \[▶ 176\]](#)

9.3.10.4 Clone

This method clones the enumeration object

```
HRESULT Clone([out] ITcEnumEventDocLink** ppEnum);
```

Parameter

[out] Pointer of the type [ITcEnumEventDocLink \[▶ 176\]](#) to a pointer that receives a copy of the enumeration object

Return Value

S_OK

The method was successfully called

E_POINTER

ppEnum was no valid pointer

E_NOTIMPL

The method is not implemented

Also see about this

 [ITcEnumEventDocLink \[▶ 176\]](#)

9.4 Enums

9.4.1 TcEventClass

This enum describes the event class. The event class can be set on issuing an alarm and can be used by the clients to distinguish different alarm classes. For e.g. the formatter displays for the different alarm class different icons.

```
enum TcEventClass
{
    TCEVENTCLASS_NONE =0,
    TCEVENTCLASS_MAINTENANCE =1,
    TCEVENTCLASS_MESSAGE =2,
    TCEVENTCLASS_HINT =3,
    TCEVENTCLASS_STATEINFO =4,
    TCEVENTCLASS_INSTRUCTION =5,
    TCEVENTCLASS_WARNING =6,
    TCEVENTCLASS_ALARM =7,
    TCEVENTCLASS_PARAMERROR =8,
    TCEVENTCLASS_MAX
};
```

Table 25: Parameters

Item	Description
TCEVENTCLASS_NONE	No special event class
TCEVENTCLASS_MAINTENANCE	The event is classified as an MAINTENANCE alarm
TCEVENTCLASS_MESSAGE	The event is classified as a message
TCEVENTCLASS_HINT	The event is classified as a hint
TCEVENTCLASS_STATEINFO	The event is classified as a state information
TCEVENTCLASS_INSTRUCTION	The event is classified as an instruction
TCEVENTCLASS_WARNING	The event is classified a warning
TCEVENTCLASS_PARAMERROR	The event is classified a parameter error
TCEVENTCLASS_MAX	The event class max is not used as classifier it just describes the max value of the enum

9.4.2 TcEventConCodes

This enum describes the event confirmation codes. The confirm code is used to describe how the the event is confirmed

```
enum TcEventConCodes
{
    TCEVENTCON_OK =0,
    TCEVENTCON_ABORT =1,
    TCEVENTCON_IGNORE =2,
    TCEVENTCON_RETRY =3
};
```

Table 26: Parameters

Item	Description
TCEVENTCON_OK	The confirmation was ok
TCEVENTCON_ABORT	The confirmation was aborted
TCEVENTCON_IGNORE	The confirmation was ignored
TCEVENTCON_RETRY	The confirmation was retried

9.4.3 TcEventFlags

This enum describes the event class. The event class can be set on issuing an alarm and can be used by the clients to distinguish different alarm classes. For e.g. the formatter displays for the different alarm class different icons.

```
enum TcEventClass
{
    TCEVENTFLAG_REQ=0x0001,
    TCEVENTFLAG_REQMUSTCON=0x0002,
    TCEVENTFLAG_CON=0x0004,
    TCEVENTFLAG_RESET=0x0008,
    TCEVENTFLAG_PRIOCLASS=0x0010,
    TCEVENTFLAG_FMTSELF=0x0020,
    TCEVENTFLAG_LOG=0x0040,
    TCEVENTFLAG_MSGBOX=0x0080,
    TCEVENTFLAG_SRCID=0x0100,
    TCEVENTFLAG_SELFRESET=0x0200,
    TCEVENTFLAG_TCID=0x0400,
    TCEVENTFLAG_SIGNAL=0x0800,
    TCEVENTFLAG_ADS=0x8000,
};
```

Table 27: Parameters

Item	Description
TCEVENTFLAG_REQ	sets the alarm that is not confirmable
TCEVENTFLAG_REQMUSTCON	sets the alarm that is confirmable
TCEVENTFLAG_CON	sets the confirmation
TCEVENTFLAG_RESET	resets the event
TCEVENTFLAG_PRIOCLASS	the event priority and class is read from the formatter configuration
TCEVENTFLAG_FMTSELF	not used yet
TCEVENTFLAG_LOG	writes the event to the list of logged events
TCEVENTFLAG_MSGBOX	Not used yet
TCEVENTFLAG_SRCID	We use a source id instead of a source name
TCEVENTFLAG_SELFRESET	the event resets itself directly. If TCEVENTFLAG_LOG is set the event is visible in the list of logged alarms
TCEVENTFLAG_TCID	not used
TCEVENTFLAG_SIGNAL	signals a alarm again
TCEVENTFLAG_ADS	indicates that the event is generated via ADS

9.4.4 TcEventPriority

This enum describes the event priority. Now only the implicit priority is available.

```
enum TcEventPriority
{
    TCEVENTPRIO_IMPLICIT=0,
};
```

Table 28: Parameters

Item	Description
TCEVENTPRIO_IMPLICIT	The standard priority

Remarks

The only priority that is defined by the enum is TCEVENTPRIO_IMPLICIT. For higher priorities the user uses an integer > TCEVENTPRIO_IMPLICIT. In every method where a priority is used the type long is used and not the type TcEventPriority.

9.4.5 TcEventStates

This enum describes the event status.

```
enum TcEventStates
{
    CEVENTSTATE_INVALID =0x00,
    TCEVENTSTATE_SINGALED =0x01,
    TCEVENTSTATE_RESET =0x02,
    TCEVENTSTATE_CONFIRMED =0x10,
    TCEVENTSTATE_RESETCON =0x12
}
```

Table 29: Parameters

Item	Description
CEVENTSTATE_INVALID	Invalid state some errors occurred
TCEVENTSTATE_SINGALED	The event is signaled
TCEVENTSTATE_RESET	The event is reset
TCEVENTSTATE_CONFIRMED	the event is confirmed
TCEVENTSTATE_RESETCON	the event is reset and confirmed

9.5 Structures

9.5.1 TcEventDocLink

This structure describes a document link.

```
struct TcEventDocLink
{
    BSTR strDocType;
    BSTR strDocLink;
}
```

Table 30: Parameters

TcEventHeader item	Description
strDocType	a BSTR string that describes the document link type
strDocLink	a BSTR string that holds the file path to the document

9.5.2 TcEventHeader

This structure describes the header of an event object. With the header the parameter of an event could be configured.

```
struct TcEventHeader
{
    TcEventClass nClass;
    TcEventPriority nPriority;
    long dwFlags;
    long dwUserData;
    long nId;
    long nInvokeId;
    DATE fDate;
    long nMs;
    VARIANT varSource;
    BSTR szFmtProgId;
};
```

Table 31: Parameters

TcEventHeader item	Data Type	Description
nClass	TcEventClass [▶ 179]	The class of the alarm, like alarm, warning, hint
nPriority	TcEventPriority [▶ 180] (long)	The event priority. Since now the priority is always: TcEventPriority.TCEVENTPRIO_IMPLICIT=0
dwFlags	TcEventFlags [▶ 180] (long)	The flags control the behavior of the alarm. That could be combined by a OR junction.
dwUserData	long	A spare variable that can be used by the user.
nId	long	The event id, that unique represent this type of event for this one Event Source.
nInvokeld	long	
fDate	VARIANT-DATE	The date and time when this event occur
nMs	long	The millisecond part of the event date
varSource	variant	The source id source name. The event source is used to distinguish different device. Like I/O Event, or different parts of the PLC program.
szFmtProgId	BSTR string	The PrgId of the formatter. If we want to use the standard XML formatter the string should bet set to "TcEventFormatter.TcXmlFormatter"

10 Samples

Event configuration

Description
Configuration of messagers with the TcEventConfigurator [▶ 184]

TwinCAT PLC interface (structured text)

Description	Source code
Issue a simple message [▶ 186]	https://infosys.beckhoff.com/content/1033/TcEventLogger/Resources/12333057163/.zip
Issue a complex message [▶ 187]	https://infosys.beckhoff.com/content/1033/TcEventLogger/Resources/12333058571/.zip
Issue a message with format parameters [▶ 189]	https://infosys.beckhoff.com/content/1033/TcEventLogger/Resources/12333059979/.zip

Microsoft Visual Studio C++

Description	Source code
Integration of TcEventViewer ActiveX control [▶ 190]	Sample01.zip (in preparation)
Console Application - Read logged events via DCOM [▶ 191]	https://infosys.beckhoff.com/content/1033/TcEventLogger/Resources/12333058571/.zip
Connection points (EventSink) [▶ 192]	https://infosys.beckhoff.com/content/1033/TcEventLogger/Resources/12333059979/.zip
Integration of TcEventConfigurator ActiveX [▶ 192]	Sample04.zip (in preparation)

Microsoft Visual Studio C#

Description	Source code
C# Visualisation - Logged events [▶ 193]	https://infosys.beckhoff.com/content/1033/TcEventLogger/Resources/12333057163/.zip
C# Visualisation - Active events [▶ 195]	https://infosys.beckhoff.com/content/1033/TcEventLogger/Resources/12333058571/.zip

Microsoft Visual Basic

Description	Source code
Integration of TcEventViewer ActiveX control [▶ 196]	Sample01.zip (in preparation)
View active alarms in user defined ListView [▶ 197]	https://infosys.beckhoff.com/content/1033/TcEventLogger/Resources/12333058571/.zip

CodeGear C++Builder 2009

Description	Source code
Integration in CodeGear C++Builder 2009 [▶ 198]	-

Description	Source code
Console Application - Read logged alarms via DCOM [► 206]	https://infosys.beckhoff.com/content/1033/TcEvent-Logger/Resources/12333057163/.zip
Console Application - Read logged alarms via ADS proxy [► 208]	https://infosys.beckhoff.com/content/1033/TcEvent-Logger/Resources/12333058571/.zip
View active alarms in user defined ListView [► 211]	https://infosys.beckhoff.com/content/1033/TcEvent-Logger/Resources/12333059979/.zip
Integration of TcEventViewer ActiveX control [► 214]	https://infosys.beckhoff.com/content/1033/TcEvent-Logger/Resources/12333061387/.zip

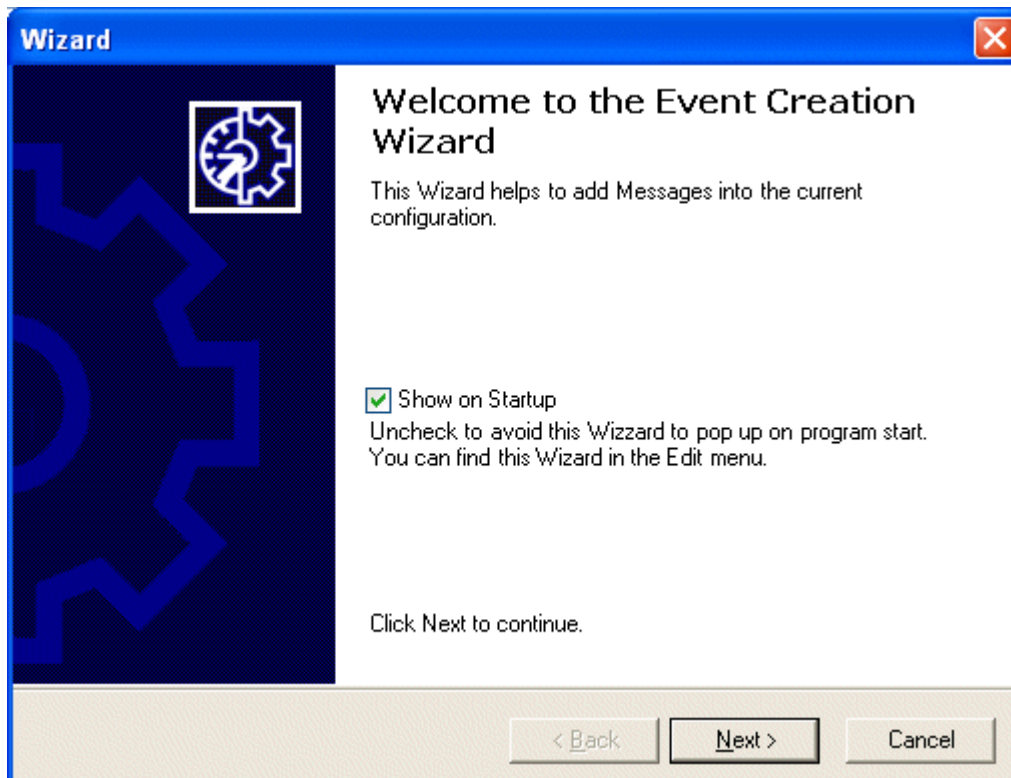
10.1 Configuration of Messages

This Sample describes the usage of the TcEventConfigurator.


The TcEventConfigurator helps to configure your message system and to create message texts in multiple languages. Its not necessary to edit the config files yourself so time is saved and an error source is eliminated.

After creating the Configuration a PLC Program will be created and the messages will be shown with the TcEventbar.

1. Install and start the TcEventConfigurator.
2. The Configurator starts with a simple Wizard for message creation, follow the steps as shown on the Screenshots.



Wizard [Close]

Sources 

Select a Source (When the Configuration already has one or more Sources) or enter a new Source ID.


In which Source shall the Message be created?

Select Source

New Source Source ID:

< Back Next > Cancel

Wizard [Close]

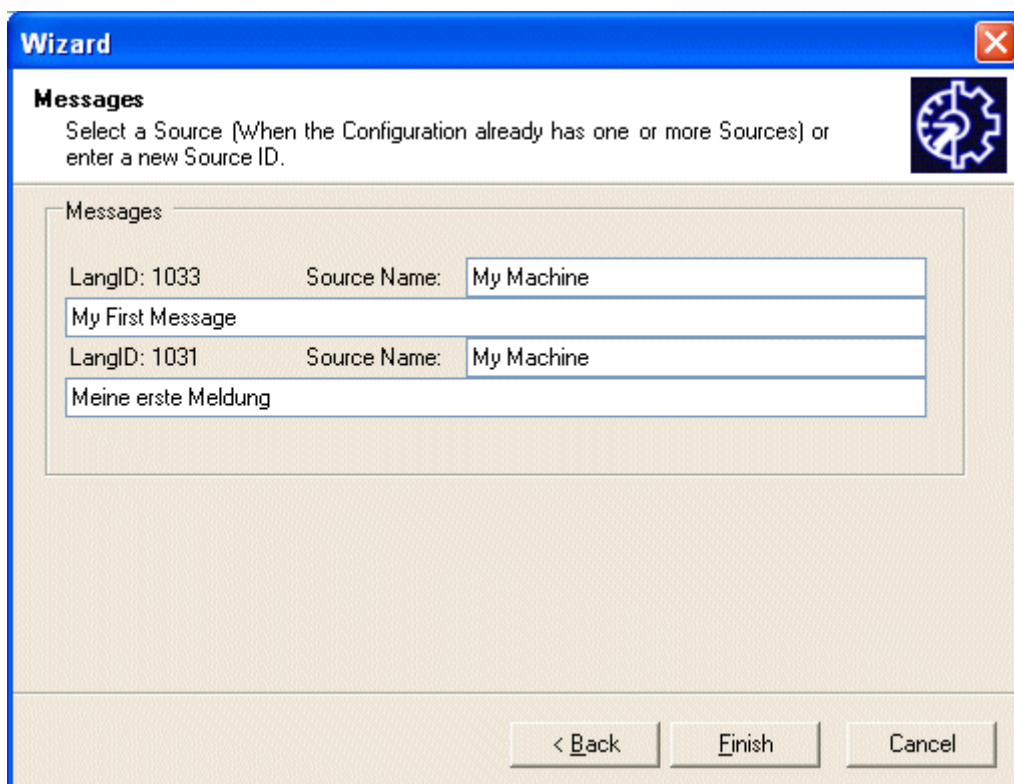
Language and Event ID 

In this step you can set for what Language IDs Messages and Sourcenames will be created. You can also change the default Event ID.

Use the Listbox to add or Remove Language IDs. **Event ID:**

Languages:

< Back Next > Cancel



3. After the Wizard has finished you have created your first Message. This message is given in german and english.
4. The TcEventConfigurator offers a clean surface to edit and create messages. With minimal effort you can add more languages and messages.
5. Save the Configuration. Chose TcXmlFormatter As file type and accept the following message box with Ok - your Configuration is now active.
6. With a right click on any message you can create PLC code to trigger the message.

Add this Code to a PLC Program and trigger the message.
Start the TcEventbar. The Message is shown.

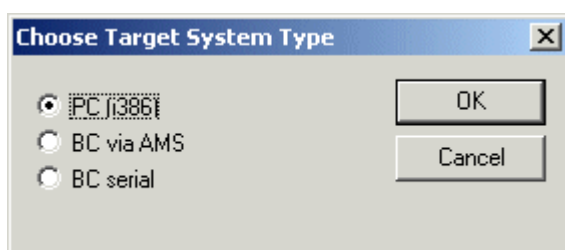
10.2 PLC interface

10.2.1 Issue a simple message

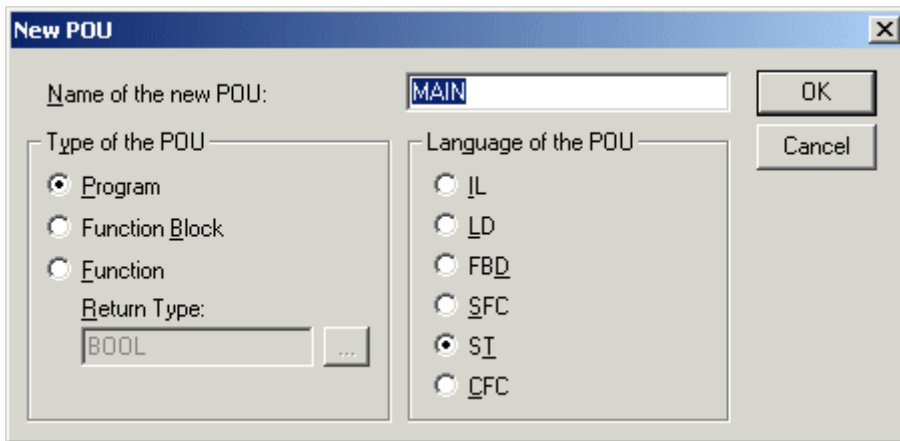
Messages can be triggered in 2 ways: simple and complex.

This Sample shows how a simple message is triggered in the PLC. The Code used for a simple message is shorter than for a complex message, but you will have less configuration possibilities.

1. Create a new PLC-Project.
 - Choose PC (i386) as Target System



- Keep the standard task-settings.
- Create a new ST-program.



- Open the menu *Window -> Library Manager* and add *TcSystem.lib* to the project. Now you have access to the *ADSLOGEVENT* function block.
2. Use the *TcEventConfigurator* [► 20] to create a quit-requiring message with the SourceID 17001 and EventID 1
 3. Add following code to the project:


```
(*Variable-declaration*)
VAR
    event : FB_SimpleAdsLogEvent;
    bEvent : BOOL;
    bQuitEvt : BOOL;
END_VAR

(*PLC Code*)
Event( SourceID := 17001,
      EventID := 1,
      bSetEvent := bEvent,
      bQuit := bQuitEvt);
```
 4. Start the PLC Program.
 5. Start the TcEventbar ('TwinCAT/Eventlogger' directory).
 6. By toggling *bSetEvent1* the message will be triggered. Rising edge starts the message, otherwise it will be reset.
 7. Toggling *bSetEvent* won't reset the message cause it is a quit-requiring message. It must be deleted before *bSetEvent* can be reset **and** *bQuitEvent* must be set.

Requirements

Language / IDE	Source code
TwinCAT PLC (Structured Text)	https://infosys.beckhoff.com/content/1033/TcEventLogger/Resources/12333057163/.zip

10.2.2 Issue a complex message

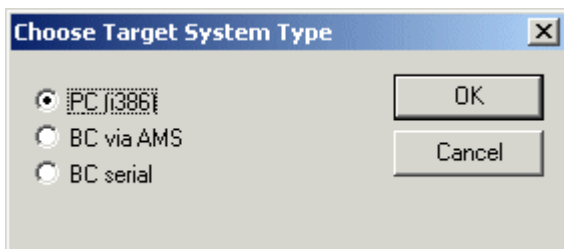
Messages can be triggered in 2 ways: simple and complex.

This Sample shows how a complex message is triggered in the PLC. The Code used for a complex message is bigger than for a simple message, but you will have more configuration possibilities.

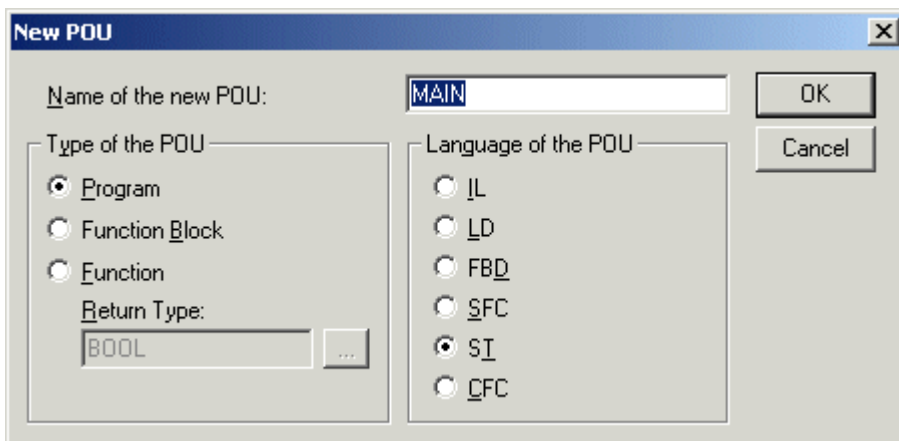
2 Messages will be created - a simple message and a quit-requiring.

They will be shown with the TcEventbar.

1. Create a new PLC-Project
 - Chose PC (i386) as Target System



- Keep the standard task-settings.
- Create a new ST-program.



- Open the menu *Window -> Library Manager* and add TcSystem.lib to the project. Now you have access to the ADSLOGEVENT function block.

2. Add following code to the project:

```
(*Variable-declaration*)
VAR
  bSetEvent1 : BOOL;
  bSetEvent2 : BOOL;
  bQuitEvent2 : BOOL;
```

```
event1: ADSLOGEVENT;
event2: ADSLOGEVENT;
```

```
CfgEvent1 : TcEvent;
CfgEvent2 : TcEvent;
END_VAR
```

```
(*PLC Code*)
(* Event 1 *)
CfgEvent1.Prio := 0;
CfgEvent1.bQuitRequired := FALSE;
CfgEvent1.Flags := TCEVENTFLAG_LOG OR TCEVENTFLAG_SRCID;
CfgEvent1.StreamType := TCEVENTSTREAM_SIMPLE;
CfgEvent1.ProgId := 'TcEventFormatter.TcXmlFormatter';
CfgEvent1.Id := 1; (* Meldung 1*)
CfgEvent1.Sourceld := 17001; (* 17001: Achsregelung *)
```

```

CfgEvent1.Class := TCEVENTCLASS_MESSAGE;

(* Event 2 *)
CfgEvent2.Prio := 0;
CfgEvent2.bQuitRequired := TRUE;
CfgEvent2.Flags := TCEVENTFLAG_LOG OR TCEVENTFLAG_SRCID;
CfgEvent2.StreamType := TCEVENTSTREAM_SIMPLE;
CfgEvent2.ProgId := 'TcEventFormatter.TcXmlFormatter';
CfgEvent2.Id := 2; (* Meldung 2*)
CfgEvent2.Sourceld := 17001; (* 17001: Achsregelung *)
CfgEvent2.Class := TCEVENTCLASS_ALARM;

event1(
NETID:= "",
PORT:= 110,
Event:= bSetEvent1,
EventConfigData:= CfgEvent1,
TMOUT:= t#10s);

event2(
NETID:= "",
PORT:= 110,
Event:= bSetEvent2,
EventQuit:= bQuitEvent2,
EventConfigData:= CfgEvent2,
TMOUT:= t#10s);

```

3. Start the PLC Program.
4. Start the TcEventbar ('TwinCAT/Eventlogger' directory).
5. By toggling bSetEvent1 the message will be triggered. Rising edge starts the message, otherwise it will be reset.
Cause no message texts are configured you will have a error in the fields Source and Message.
6. Toggling bSetEvent2 wont reset the message cause it is a quit-requiring message. It must be deleted before bSetEvent2 can be reset **and** bQuitEvent2 must be set.
7. [Continue with the message text configuration \[► 184\]](#)

Requirements

Language / IDE	Source code
TwinCAT PLC (Structured Text)	https://infosys.beckhoff.com/content/1033/TcEventLogger/Resources/12333058571/.zip

10.2.3 Issue a message with format parameters

Its possible to transfer one or more variables with a message. This example shows how to do it.

1. Create a PLC Project like in the [previous Sample \[► 187\]](#)
2. Add following variables to the variable configuration:
The value 'value' will be transferred; The string 'format' describes the data format.

```

value : DWORD := 0;
format : STRING := '%d';

```
3. Now Change the configuration for the Event:

```

(* Event 2 *)
CfgEvent2.Prio := 0;
CfgEvent2.bQuitRequired := TRUE;
CfgEvent2.Flags := TCEVENTFLAG_LOG OR TCEVENTFLAG_SRCID;
CfgEvent2.StreamType := TCEVENTSTREAM_SIMPLE;
CfgEvent2.ProgId := 'TcEventFormatter.TcXmlFormatter';
CfgEvent2.Id := 2; (* Meldung 2*)
CfgEvent2.Sourceld := 17001; (* 17001: Achsregelung *)
CfgEvent2.DataFormatStrAddress := ADR(format);
CfgEvent2.Class := TCEVENTCLASS_ALARM;

event2(
NETID:= "",
PORT:= 110,
Event:= bSetEvent2,
EventQuit:= bQuitEvent2,
EventConfigData:= CfgEvent2,
EventDataAddress := ADR(value),
EventDataLength := SIZEOF(value),
TMOUT:= t#10s);

```

4. Trigger the event

You can transfer multiple variables by wrapping them inside a structure. Then the format string must define the order of the variables inside the struct.

When a string is transferred with the eventdata, this string must be the last element in the structure. Only one string at a time may be transferred.

```

(*data types*)
TYPE ST_EventData1 :
STRUCT
    axis    : UDINT;
    pos     : LREAL;
    text    : STRING;
END_STRUCT
END_TYPE

```

```

(*variable-declaration*)
...
format      : STRING := '%d%f%s'
Eventdata1  : ST_EventData1;
...

(*PLC ode*)
...
EventDataAddress := ADR(Eventdata1),
EventDataLength :=SIZEOF(Eventdata1),
...

```

Requirements

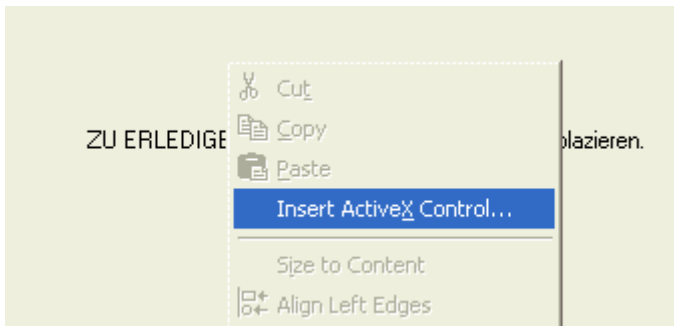
Language / IDE	Source code
TwinCAT PLC (Structured Text)	https://infosys.beckhoff.com/content/1033/TcEventLogger/Resources/12333059979.zip

10.3 Visual Studio C++

10.3.1 Integration of TcEventViewer ActiveX control

1. Create a new MFC exe Project.

2. Choose 'dialog based' as application type.
3. Right click on the dialog and choose 'Insert ActiveX Control'



4. Choose the 'BECKHOFF TcEventView Class'

Language / IDE	Source code
Visual Studio 6 C++	Sample01.zip (in preparation)

10.3.2 Console Application - Read logged events via DCOM

The following example shows how to view a list of logged messages in C++.

1. Create a standard Console Application.
2. Add this Sourcecode:

```

if( SUCCEEDED(CoInitialize( 0 )) )
{
    ITcEventLogPtr    spEventLogger;
    ITcEnumEventsPtr spEnumEvts;
    ITcEventPtr      spEvent;
    IDispatchPtr     spDisp;

    if( SUCCEEDED( spEventLogger.CreateInstance(CLSID_TcEventLog)) )
    {
        spEnumEvts =spEventLogger->EnumLoggedEvents();
        if(spEnumEvts)
        {
            while(spEnumEvts->Next(1, &spDisp) == 1 )
            {
                spEvent = spDisp;
                spEvent->GetMsgString(1033);
                printf( "Event %i; Source %i\n Message = %s\n",spEvent->GetId(), spEvent-
>GetSrcId(), (char*) spEvent->GetMsgString(1033) );
                spDisp = NULL;
            }
        }
    }

    spEnumEvts = NULL;
    spEvent = NULL;
    spEventLogger = NULL;

    CoUninitialize();
}

return 0;

```

The Program uses the ITcEnumEvents Interface to enumerate logged messages. The text of all logged messages is shown in the Console.

Language / IDE	Source code
Visual Studio 6 C++	https://infosys.beckhoff.com/content/1033/TcEventLogger/Resources/12333058571/.zip

10.3.3 Connection Points

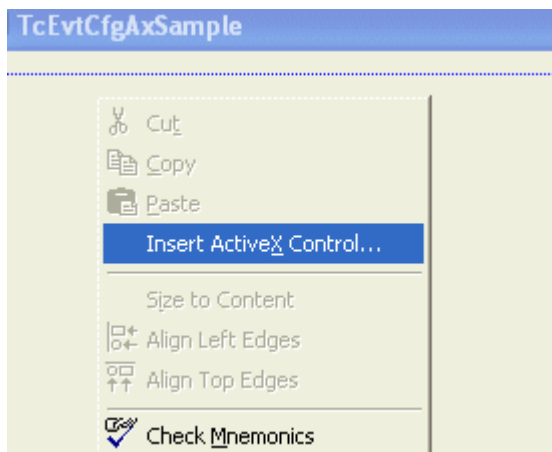
See the sample program for a complete implementatin of a connection point to the TcEventLogger.

Language / IDE	Source code
Visual Studio 6 C++	https://infosys.beckhoff.com/content/1033/TcEventLogger/Resources/12333059979/.zip

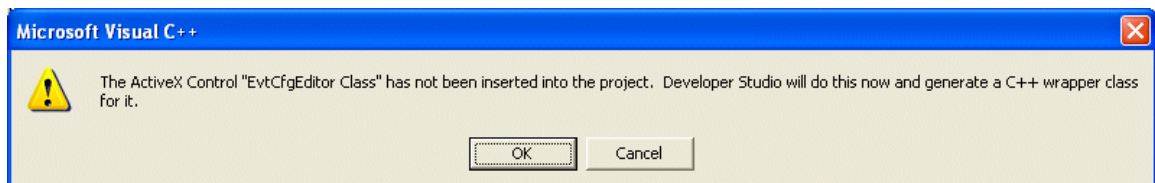
10.3.4 Add the TcEvtCfgEditor ActiveX into an Dialog

To insert the TcEvtCfgEditor into an dialog based MFC Application simply follow this step by step tutorial.

1. Start MSVisual Studio 6.
2. Create a new dialog based MFC Application.
3. In design mode rightclick on the dialog resource and choose 'Insert ActiveX Control'.



4. Select the Beckhoff EvtCfgEditor and place it on your Dialog.
5. Open the Class Wizzard and switch to the 'Member Variables' Tab
6. Select IDC_EVTFCGEDITOR1 and click on AddVariable
7. In the following Dialog choose OK to get an automaticly generated wrapper class



8. In the following Dialog keep all items selected and choose OK.
9. Now enter a name for the member variable.
10. The ActiveX is ready to use.
Call [GetConfiguration](#) [▶ 63] to get access to the underlying [Configuration Object](#) [▶ 26]

Language / IDE	Source code
Visual Studio 6 C++	Sample04.zip (in preparation)

10.4 Visual Studio C#

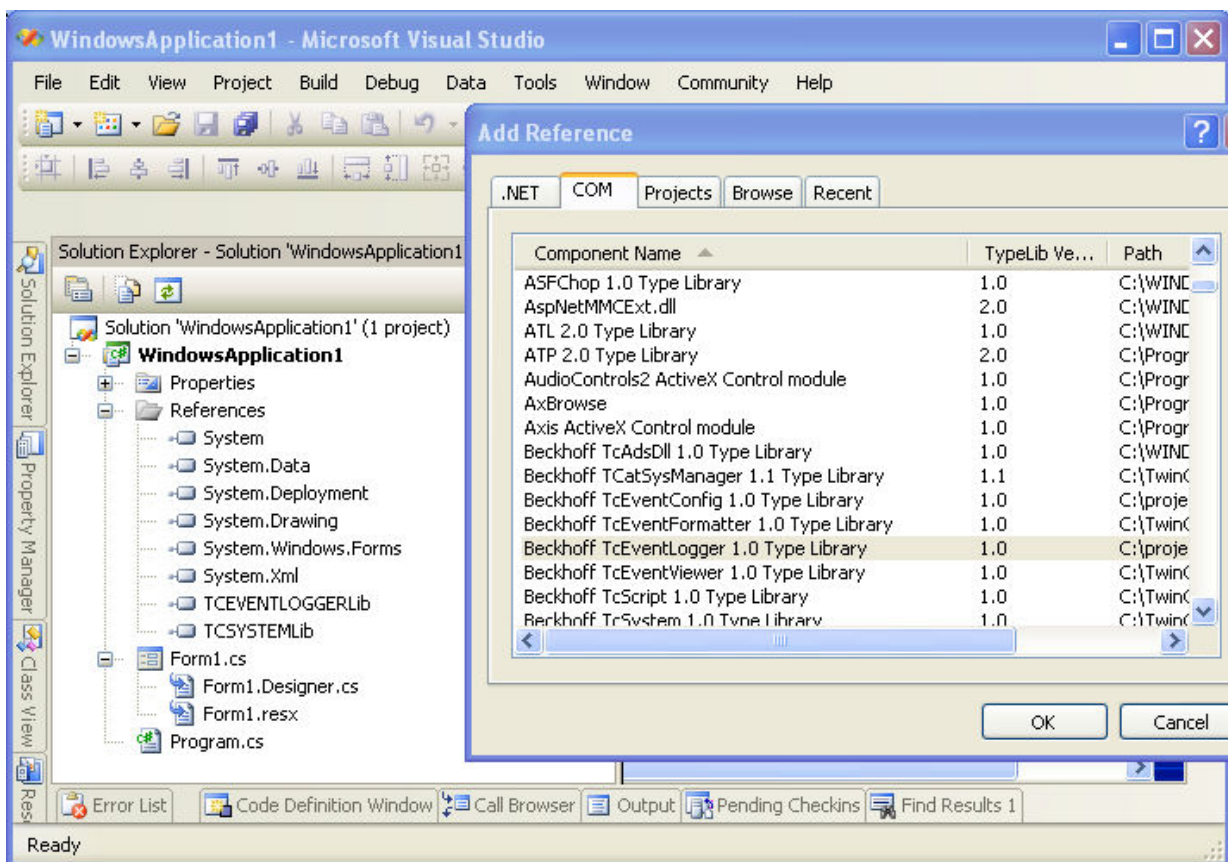
10.4.1 Displaying Logged Events in C#

This sample applies to

- Visual Studio 2005
- TcEventLogger 2.9.0.90 and above (Shipped since TwinCAT 2.10. Build1244)

Before starting with the sample be aware that the TcEventViewer ActiveX does provide a fully featured implementation of an event HMI which can easily be added to your application. If you have special needs that cant be realized using the standard TcEventViewer proceed with the sample to display a list of logged events.

1. Create a Win32 C# Application.
2. Add a reference to the Beckhoff TcEventLogger:



3. On top of the Form1.cs add

```
using TCEVENTLOGGERLib;
```
4. Create a global instance on the TcEventlogger.

```
TcEventLog tcEventLogger = new TcEventLog();
```
5. Add a ListView control to your Main Form and name it eventView1
6. In Form1_Load initialize the ListView:

```
eventView1.View = View.Details;
eventView1.Columns.Add("Time", "Time");
eventView1.Columns.Add("Type", "Type");
eventView1.Columns.Add("Source", "Source");
eventView1.Columns.Add("Message", "Message");
```
7. Add code to display events in the ListView control:

```
///

```

```

///</summary>
///<param name="evt">Event to be added</param>
privatevoidAddEvent(TcEventevt)
{
    ListViewItemlvi =newListViewItem();
    lvi.Name ="Time";
    try
    { lvi.Text =Convert.ToString(evt.Date); }
    catch(Exception)
    { lvi.Text =""; }
    eventView1.Items.Insert(0, lvi);
    PrepareListViewItem(lvi);

    try
    { AddListViewSubItem(lvi,"Type", GetEventClassName(evt)); }
    catch(Exception)
    { }
    try
    { AddListViewSubItem(lvi,"Source", evt.get_SourceName(langId)); }
    catch(Exception)
    { }

    try
    { AddListViewSubItem(lvi,"Message", evt.GetMsgString(langId)); }
    catch(Exception)
    { }
}
///<summary>
///Prepare the listview item by padding its subitems with empty items
///</summary>
///<param name="lvi">ListViewItem to be prepared</param>
privatevoidPrepareListViewItem(ListViewItemlvi)
{
    for(inti = 0; i < eventView1.Columns.Count; i++)
        lvi.SubItems.Add("???");// pad items with question marks.
}
///<summary>
///Add one sub item at an indexed position to the ListViewItem
///</summary>
///<param name="lvi">Parent item. The item must have been inserted to a
ListView control</param>
///<param name="name">Item name (index) of the item. A coulumn with such
index must exist in the ListView control</param>
///<param name="text">Display text</param>
privatevoidAddListViewSubItem(ListViewItemlvi,stringname,stringtext)
{
    ListViewItem.ListViewSubItemlvsi =newListViewItem.ListViewSubItem();
    lvsi.Name = name;
    lvsi.Text = text;
    lvi.SubItems.Insert(lvi.ListView.Columns.IndexOfKey(lvsi.Name), lvsi);
}
///<summary>
///Get a textual representation for the event class.
///</summary>
///<param name="evt">Event object</param>
///<returns>Classname</returns>
privatestringGetEventClassName(TcEventevt)
{
    switch( (TcEventClass)evt.Class)

```

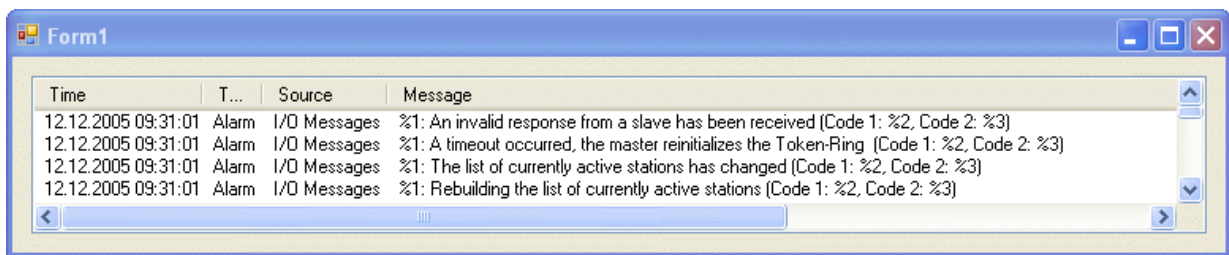
```

{
    caseTcEventClass.TCEVENTCLASS_ALARM:
        return"Alarm";
    caseTcEventClass.TCEVENTCLASS_HINT:
        return"Hint";
    caseTcEventClass.TCEVENTCLASS_INSTRUCTION:
        return"Instruction";
    caseTcEventClass.TCEVENTCLASS_MAINTENANCE:
        return"Maintenance";
    caseTcEventClass.TCEVENTCLASS_MESSAGE:
        return"Message";
    caseTcEventClass.TCEVENTCLASS_PARAMERROR:
        return"Paramerror";
    caseTcEventClass.TCEVENTCLASS_STATEINFO:
        return"State info";
    caseTcEventClass.TCEVENTCLASS_WARNING:
        return"Warning";
    default:
        return"?";
}
}

```

8. Finally, in Form1_Load, read the list of active alarms from the TcEventLogger


```
foreach(TcEvent evt in tcEventLogger.EnumLoggedEventsEx() )
    AddEvent(evt);
```
9. Start a PLC program to issue some events.



Language / IDE	Source code
Visual Studio 2005	https://infosys.beckhoff.com/content/1033/TcEventLogger/Resources/12333057163/.zip

10.4.2 Displaying Active Events in C#

This sample applies to

- Visual Studio 2005
- TcEventLogger 2.9.0.90 and above (Shipped since TwinCAT 2.10. Build1244)

Before starting with the sample be aware that the TcEventViewer ActiveX does provide a fully featured implementation of an event HMI which can easily be added to your application. If you have special needs that cant be realized using the standard TcEventViewer proceed with the sample to display a list of active events.

- This sample builds on the sample '[Displaying Logged Events in C# \[► 193\]](#)'
- To implement a program capable of displaying active events we will need to extend the previous sample with the following features:
 - Add events dynamically as they occur
 - Remove inactive events from the listview.
 - Associate an event with one entry in the listview.

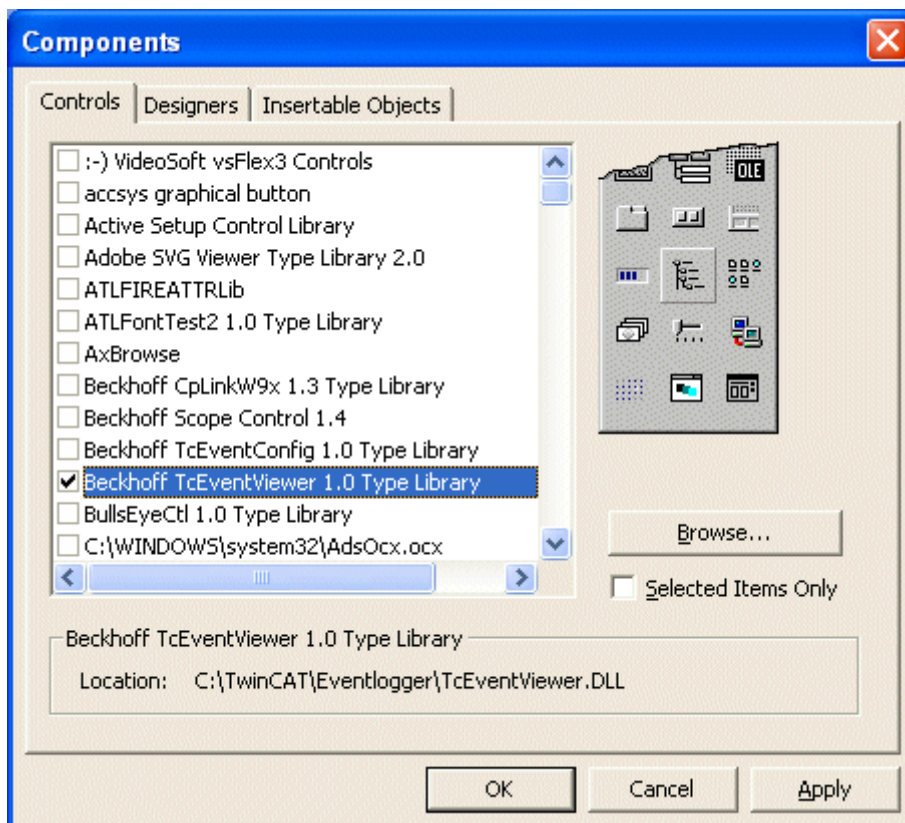
- If we use the Eventloggers callback methods to be notified about event state changes we need to make our application threadsafe. This will include defining some critical sections, using the lock directive on the one hand and access to the ListView on the other hand. Accessing Windows controls from different threads does not work in general. You can use delegate methods and the ListViews Invoke method for synchronization.
- See the complete project for a very basic implementation of the points mentioned above. This implementation does not deal with event confirmations.

Language / IDE	Source code
Visual Studio 2005	https://infosys.beckhoff.com/content/1033/TcEventLogger/Resources/12333058571/.zip

10.5 Visual Basic

10.5.1 Integration of TcEventViewer ActiveX control

1. Create a new standard exe project
2. Right-click on the Toolbox and choose 'Components...'
3. Add the TcEventViewer



4. The TcEventviewer is now in the Toolbox. Add it to the form.

Language / IDE	Source code
Visual Basic 6.0	Sample01.zip (in preparation)

10.5.2 View active alarms in user defined ListView

The following example shows how to create a user defined List View in Visual Basic for the TcEventLogger, that shows active events.

Do following steps to create the sample:

1. Create a standard EXE project with a form.
2. Add MS Standard Controls 6 to the Control list.
3. Add the Beckhoff TcEventLogger Library to the references.
4. Add the List View ActiveX Control to the form.
5. Rename the List View to lwvAlarm
6. Copy following Code to the form:

```
Option Explicit
Dim WithEvents objEventLogger As TcEventLog

'-----
' this method is called when the form is loaded
'
Private Sub Form_Load()
    Set objEventLogger = New TcEventLog
    Call initListView
    Call DisplayActiveAlarms
End Sub

'-----
' this method is called when a new alarm is issued
'
Private Sub objEventLogger_OnNewEvent(ByVal evtObj As Object)
    ' cast to TcEvent interface
    Dim objEvent As TcEvent
    Set objEvent = evtObj
    ' get the key
    Dim newKey As String
    newKey = GetKeyFromIDs(objEvent.SrcId, objEvent.Id)
    'get the Event Message Text
    Dim strMessage As String
    strMessage = objEvent.GetMsgString(1033)
    'add the Message to the listview
    Dim objListItem As ListItem
    Set objListItem = lwvAlarm.ListItems.Add(, newKey, strMessage & objEvent.Date)
End Sub

'-----
' this method is called when a alarm is reset
'
Private Sub objEventLogger_OnResetEvent(ByVal evtObj As Object)
    ' cast to TcEvent interface
    Dim objEvent As TcEvent
    Set objEvent = evtObj
    ' get the key
    Dim newKey As String
    newKey = GetKeyFromIDs(objEvent.SrcId, objEvent.Id)
    'remove the event from the listview
    Call lwvAlarm.ListItems.Remove(newKey)
End Sub

'-----
' initialize the list view
'
Sub initListView()
    lwvAlarm.View = lvwReport
    lwvAlarm.GridLines = True
    'init the Header
    Dim objColHdr As ColumnHeaders
    Set objColHdr = lwvAlarm.ColumnHeaders
    Debug.Assert Not objColHdr Is Nothing
    Call objColHdr.Clear
    Call objColHdr.Add(1, "Message", "Message", 5000)
End Sub

'-----
' display all active alarms
'
Private Sub DisplayActiveAlarms()
```

```

'clear the listview
Call lvwAlarm.ListItems.Clear

Debug.Assert Not objEventLogger Is Nothing

Dim objEvent As TcEvent
'get the number of active Events
For Each objEvent In objEventLogger.EnumActiveEventsEx
  If (Not objEvent Is Nothing) Then
    'here we get one Event from the collection
    Dim strMessage As String
    'get the Event Message Text
    strMessage = objEvent.GetMsgString(1033)
    'get the key
    Dim newKey As String
    newKey = GetKeyFromIDs(objEvent.SrcId, objEvent.Id)
    'add the Message to the listview
    Dim objListItem As ListItem
    Set objListItem = lvwAlarm.ListItems.Add(, newKey, strMessage & objEvent.Date)
  End If
Next
End Sub

'-----
' create unique key from event group and event id
'
Public Function GetKeyFromIDs(ByVal SourceId As Long, ByVal EventId As Long) As String
  GetKeyFromIDs = SourceId & "-" & EventId
End Function

```

The List View has a column with the Event message. The Method DisplayActiveAlarms gets the active alarms from the EventLogger and shows them. When a new Event is shown or a actual event is reset the OnNewEvent event-function adds events to the List View and the OnResetEvent event-function removes events from the List View. The method GetKeyFromIDs created a key of the event group and the event id.

The user defined Event View only supports events that are not quit-requiring. To use it for quit-requiring events you have to add functions for [ITcEventLogEvents \[▶ 144\]::OnSignalEvent \[▶ 147\]](#) and [ITcEventLogEvents \[▶ 144\]::OnConfirmEvent \[▶ 145\]](#).

Language / IDE	Source code
Visual Basic 6.0	https://infosys.beckhoff.com/content/1033/TcEventLogger/Resources/12333058571/.zip

10.6 C++Builder 2009

10.6.1 Integration in CodeGear C++Builder 2009

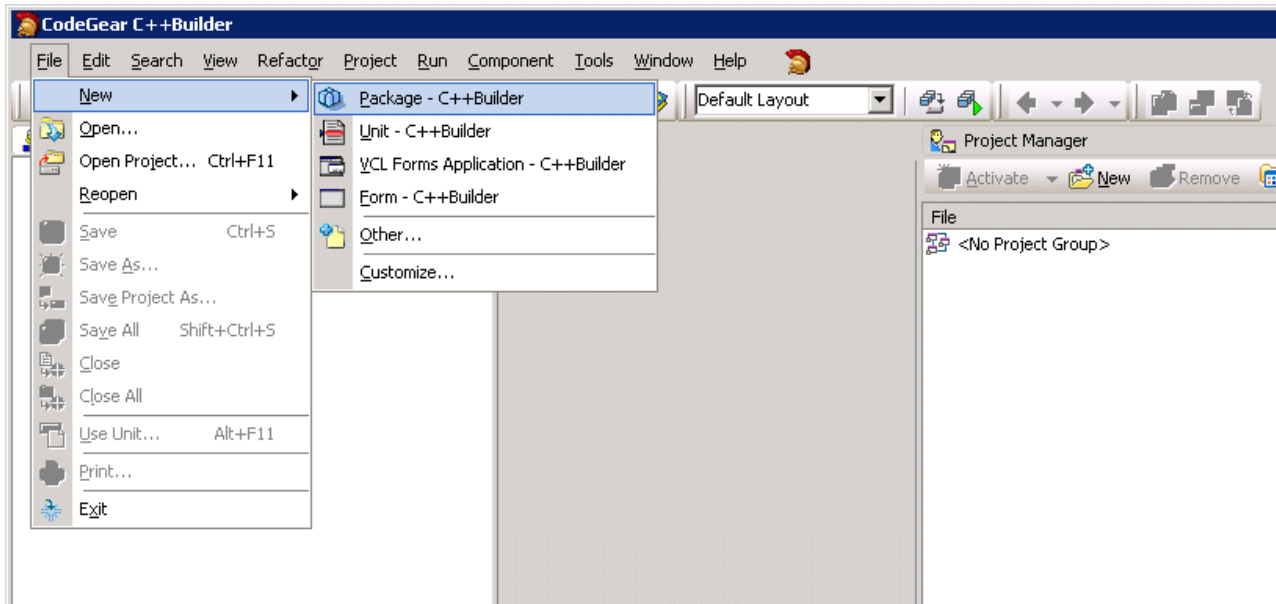
Die Komponenten der EventLogger-Typbibliotheken werden importiert und einem neuen Package hinzugefügt. Das Komponenten-Package wird anschliessend in der ActiveX-Komponentenpalette installiert.

Systemvoraussetzungen

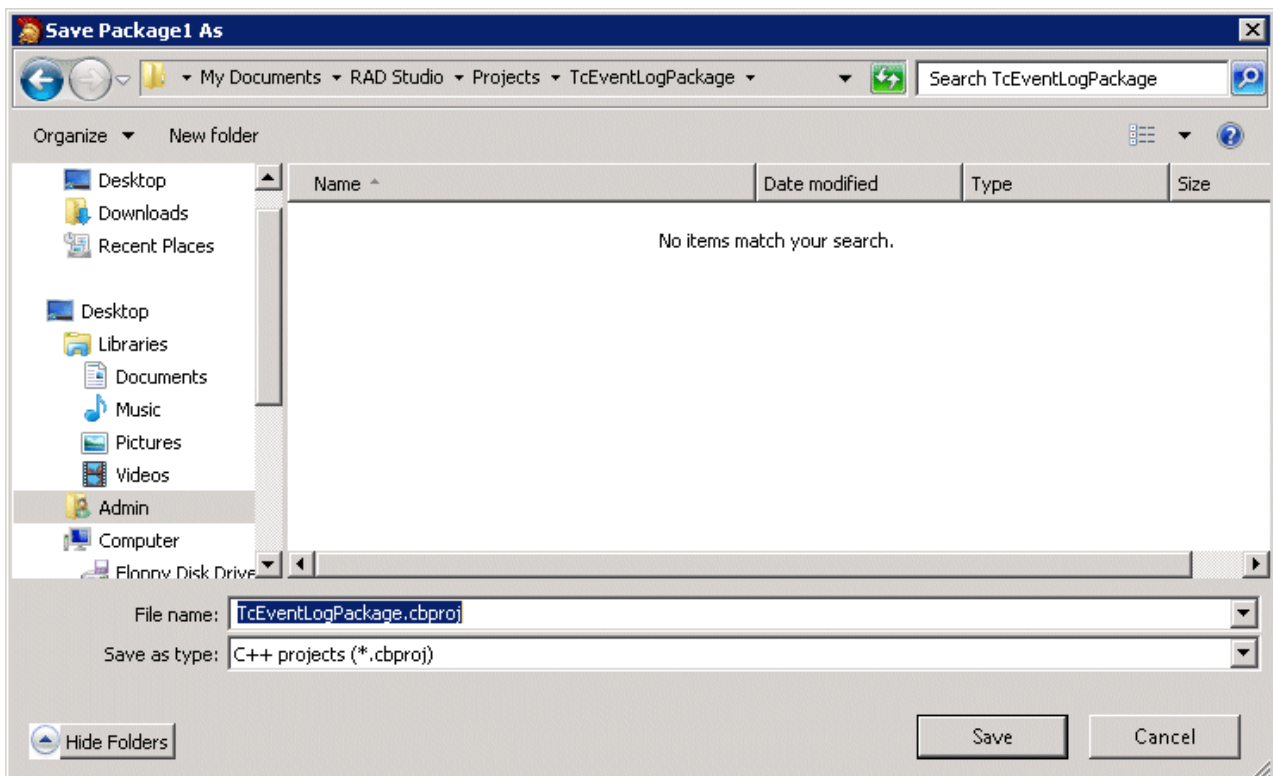
- CodeGear C++Builder 2009;
- Delphi and C++Builder 2009 Update 3;
- Delphi and C++Builder 2009 Update 4 (Database Pack Update);
- C++Builder 2009 Boost Update;
- RAD Studio 2009 Hotfix 2 (dieser Hotfix behebt u.a. folgende Probleme beim Debuggen unter Windows 7: Assertion :... "(!SetThreadContext fehlgeschlagen)");
- TwinCAT v2.11 oder höher;

Neues Package-Projekt erstellen

Erstellen Sie in CodeGear C++Builder 2009 ein neues Projekt vom Typ: "Package - C++Builder".

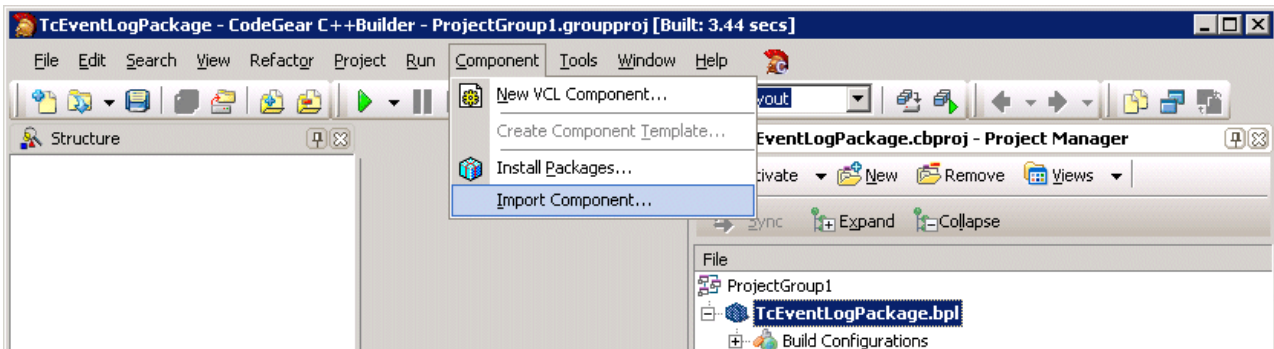


Speichern Sie das neue Projekt mit dem Menübefehl: *File->Save Project as...* unter einem neuen Namen z.B. *TcEventLogPackage.cbproj*.

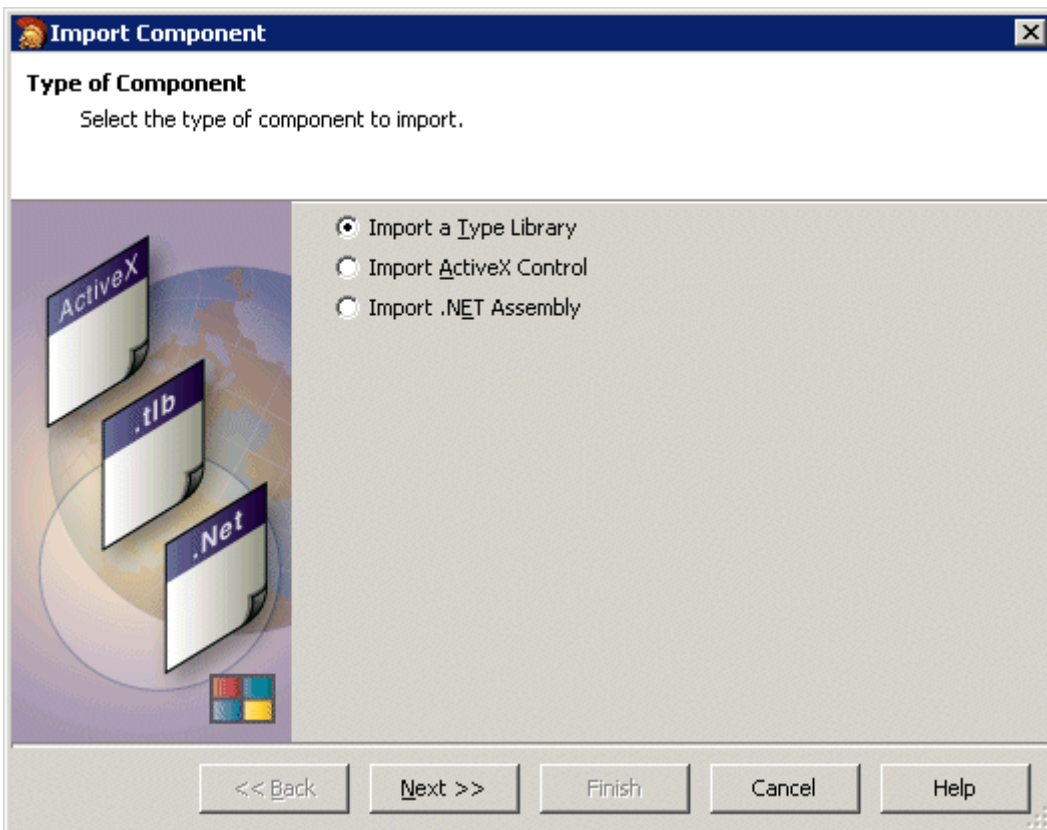


Typbibliotheken importieren

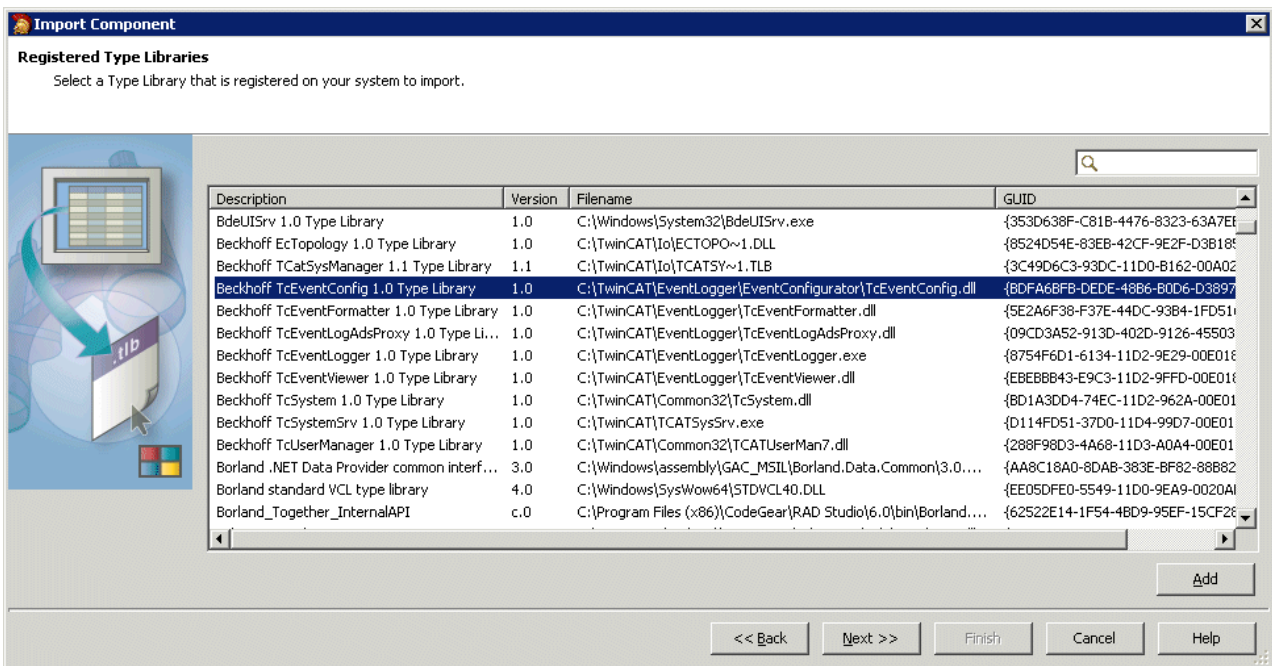
Schritt 1: Im Menue rufen Sie den Befehl: *Component -> Import Component* auf.



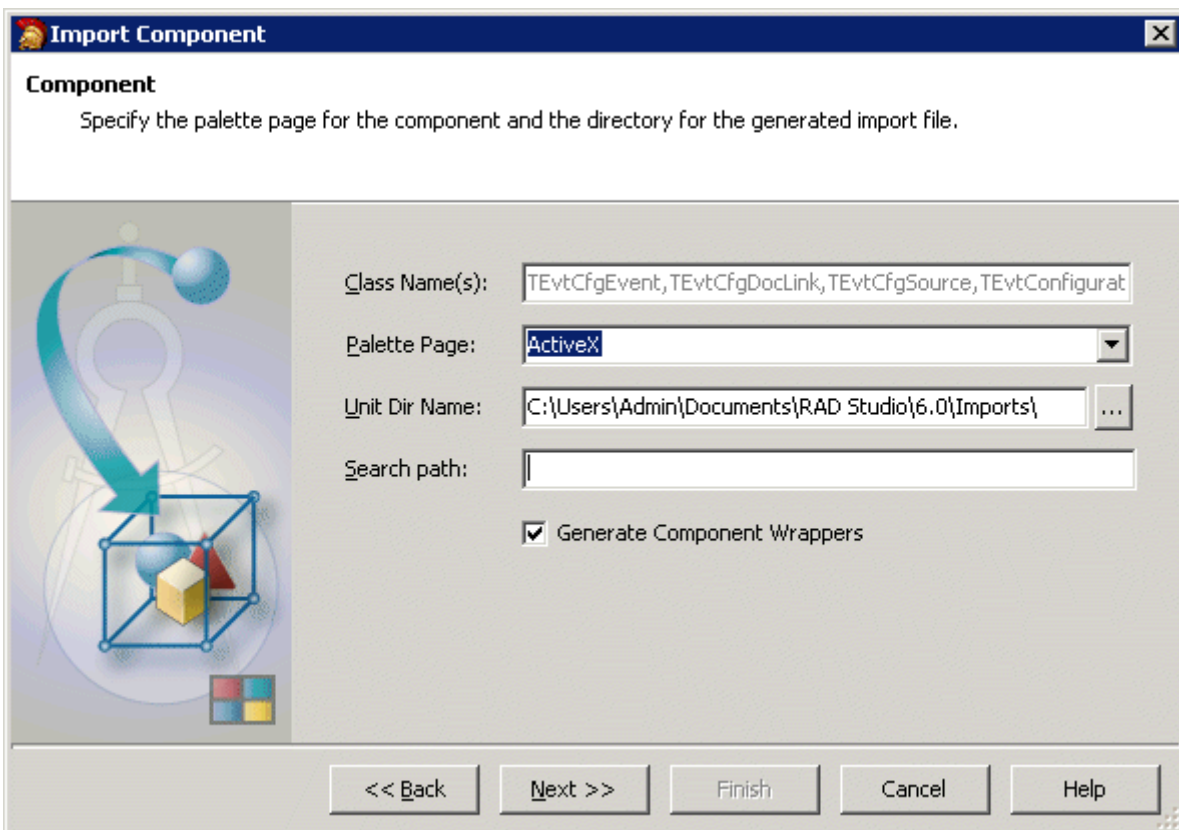
Schritt 2: Wählen Sie die Option: *Import a Type Library* aus und bestätigen Sie mit *Next*.



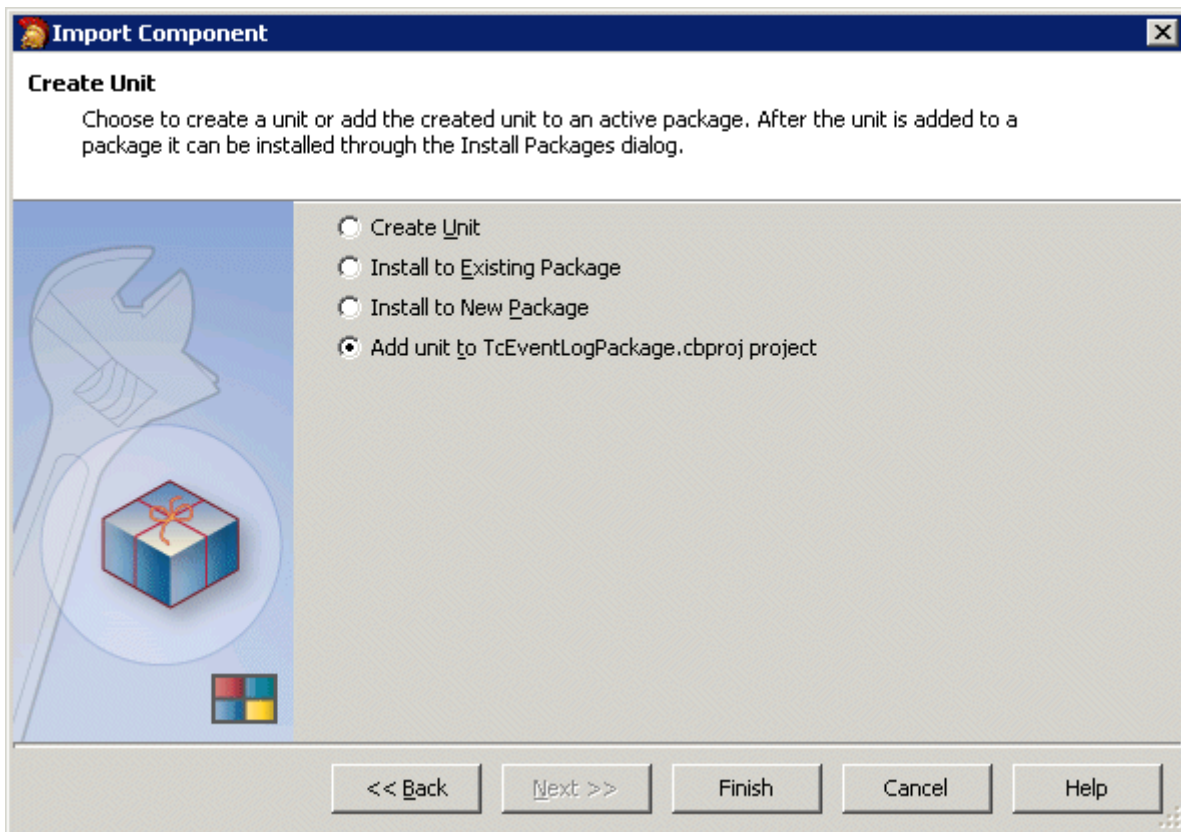
Schritt 3: In der Komponentenliste wählen Sie die erste EventLogger Komponente: *Beckhoff TcEventConfig 1.0 Type Library* aus und bestätigen Sie mit *Next*.



Schritt 4: Konfigurieren Sie als Komponentenpalettenseite: *ActiveX* aus und bestätigen Sie mit *Next*.

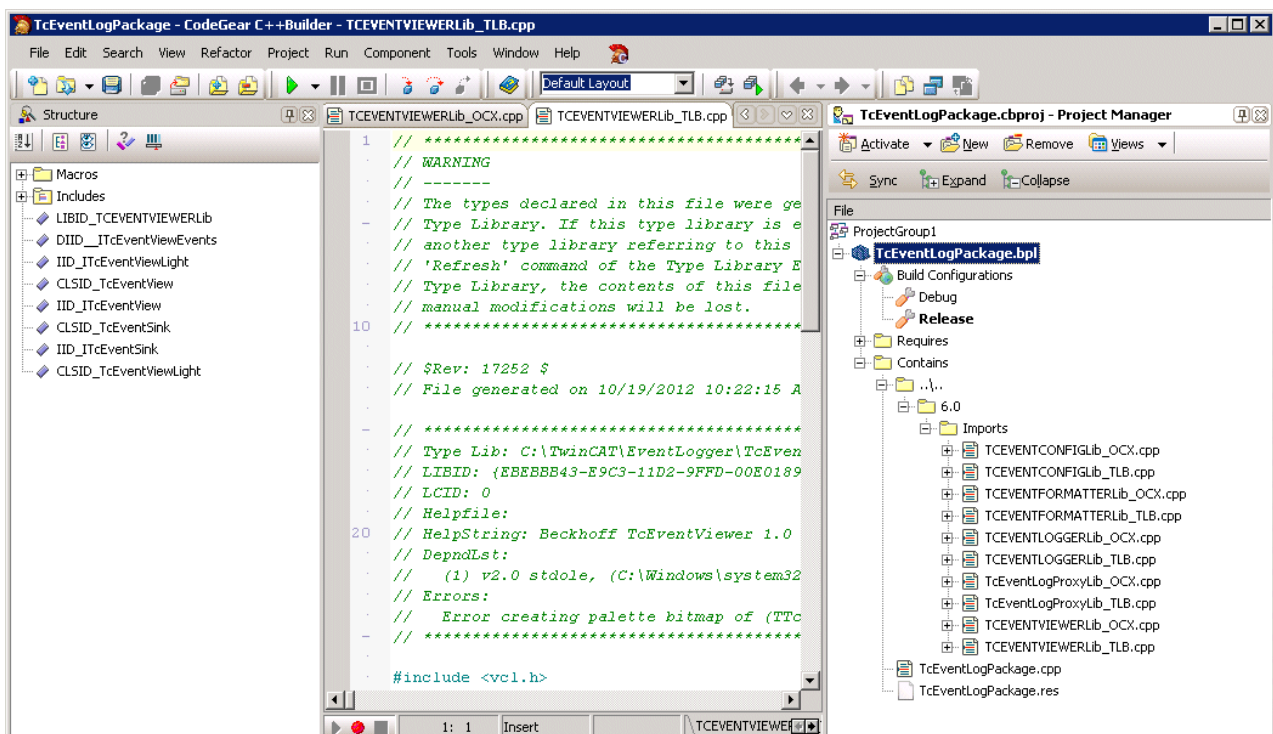


Schritt 5: Im nächsten Dialog wählen Sie die Option: *Add unit to TcEventLogPackage.cbproj project* und bestätigen Sie mit *Finish*.



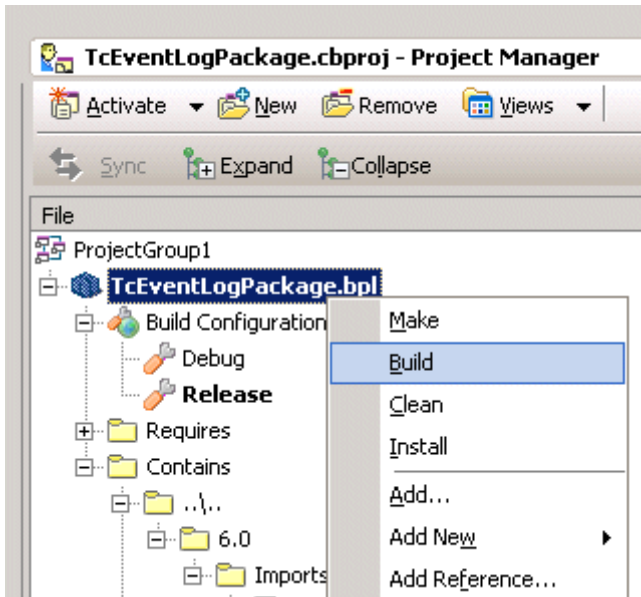
Schritt 6: Wiederholen Sie die Schritte 1 bis 5 für alle anderen EventLogger-Komponenten:

- Beckhoff TcEventFormatter 1.0 Type Library;
- Beckhoff TcEventLogAdsProxy 1.0 Type Library;
- Beckhoff TcEventLogger 1.0 Type Library;
- Beckhoff TcEventViewer 1.0 Type Library;

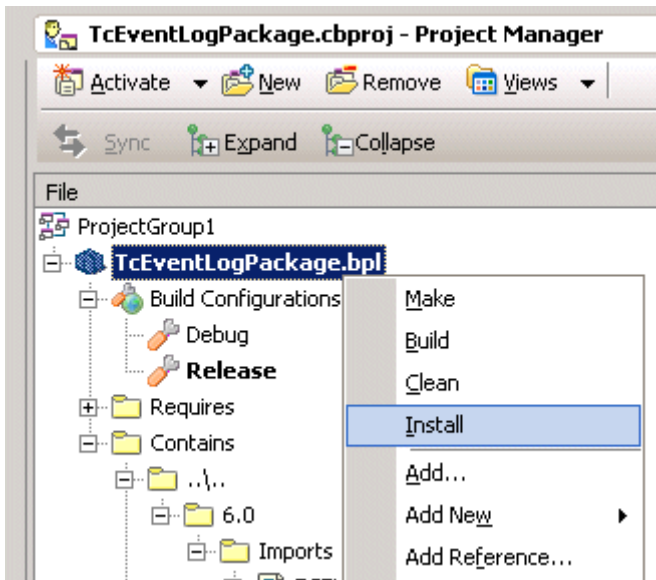


Package installieren

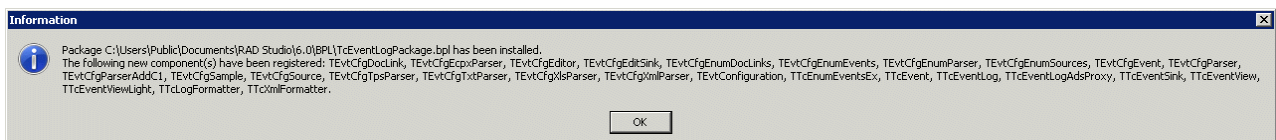
Wählen Sie aus dem Kontextmenu den Befehl: *Build* aus und übersetzen Sie das Package. Das Package muss fehlerfrei übersetzbar sein.



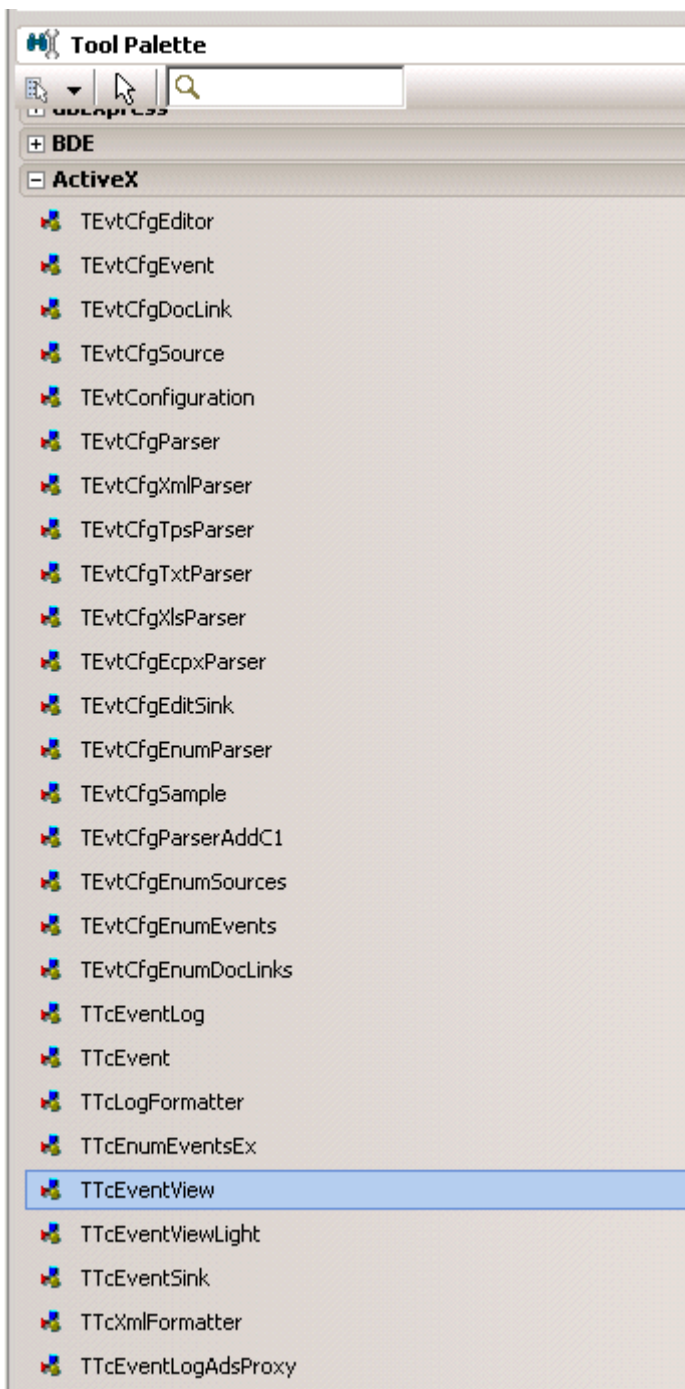
Als nächstes wählen Sie aus dem Kontextmenu den Befehl: *Install* aus und installieren Sie die Komponenten in der Komponententpalettenseite (in unserem Fall unter ActiveX).



Die erfolgreiche Installation der Komponenten wird im folgenden Dialog bestätigt. Schliessen Sie den Dialog mit OK.

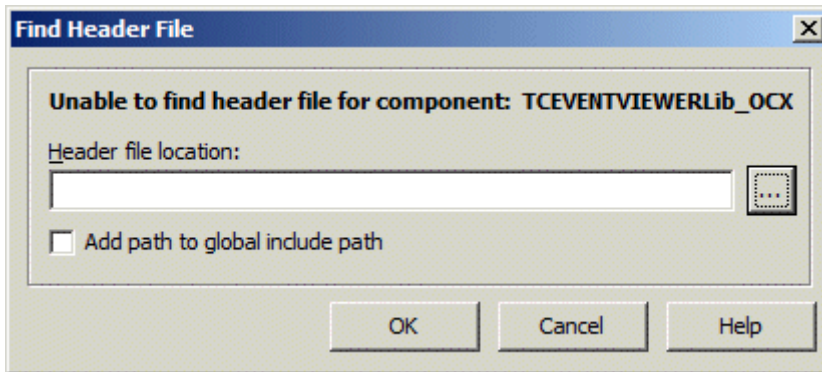


Speichern Sie die letzten Projektänderungen mit dem Befehl: *File->Save all*. Schliessen Sie das Project mit dem Befehl: *File->Close all*. Die EventLogger-Komponenten können ab jetzt von der Komponentenpalette: ActiveX auf die Form gezogen und verwendet werden.

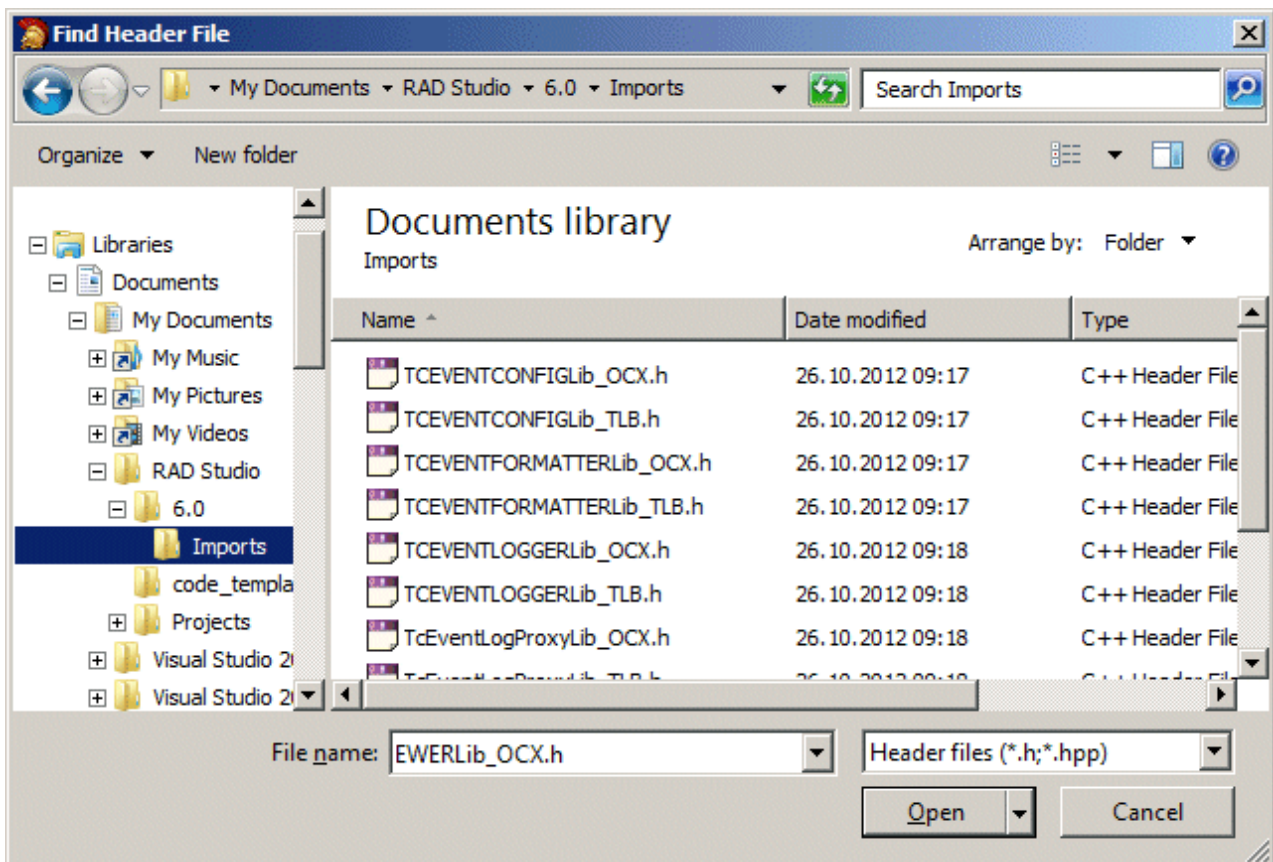


Imports-Pfad zum globalen Include-Pfad hinzufügen

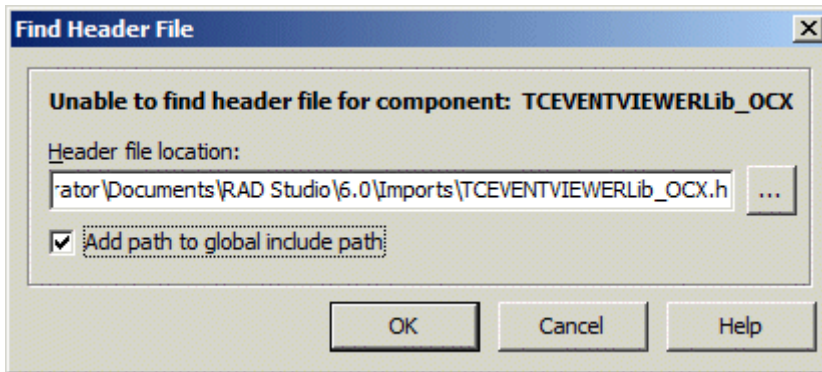
Fügen Sie den Imports-Pfad zum globalen Include-Pfad hinzu wenn beim Zugriff auf die neuen Komponenten die Headerdateien nicht gefunden werden können.



Dies muss normalerweise nur einmalig gemacht werden. Navigieren Sie hierfür zum Imports-Ordner in dem sich die Headerdateien der importierten Typbibliotheken befinden.



Wählen Sie die Option an: *Add path to global include path* und bestätigen Sie mit *OK*.

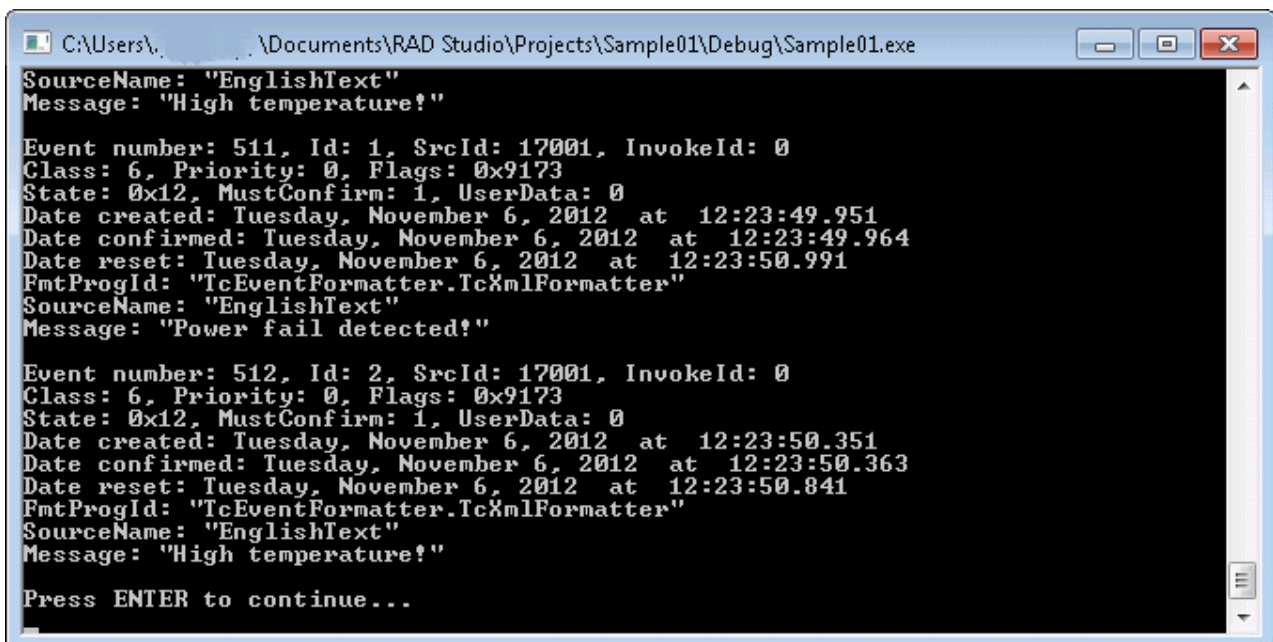


10.6.2 Console Application - Read logged alarms via DCOM

System requirements:

- CodeGear C++Builder 2009;
- TwinCAT v2.11 B2228 or higher;
- Import of the library TcEventLogger.exe (TCEVENTLOGGERLib_TLB.h);

The sample creates DCOM connection to a Remote-EventLogger and shows all logged EventLogger messages in the console.



```
#include <vcl.h>
#pragma hdrstop

#include <tchar.h>
#include <stdio.h>
#include <string.h>
#include <iostream.h>
#include <TCEVENTLOGGERLib_TLB.h>
```

```

//-----
#pragma argsused
int _tmain(int argc, _TCHAR* argv[])
{
    // Create connection via DCOM
    ITcEventLogPtr spTcEventLog;
    COSERVERINFO comServerInfo={0};
    // comServerInfo.pwszName = SysAllocString( L"172.17.60.234" );// ToDo: Configure the IP address i
f connecting to remote TwinCAT System
    comServerInfo.pwszName = SysAllocString( L"localhost" );
    long langID = 1033;// ToDo: Select language ID (e.g. english)
    MULTI_QI mQI={&IID_ITcEventLog, NULL, 0};
    HRESULT hr = ::CoCreateInstanceEx( CLSID_TcEventLog, NULL, CLSCTX_SERVER, &comServerInfo, 1, &mQ
I);
    if (SUCCEEDED(hr) && mQI.pItf )
    {
        spTcEventLog = mQI.pItf;

        long nLoggedEvents = 0;
        long nActiveEvents = 0;
        hr = spTcEventLog->get_LoggedEvents(&nLoggedEvents);
        hr = spTcEventLog->get_ActiveEvents(&nActiveEvents);
        wprintf( L"Max. number of events: %d logged, %d active.\n", nLoggedEvents, nActiveEvents );

        ITcEnumEventsExpPtr spEnumEvts;
        hr = spTcEventLog->EnumLoggedEventsEx(&spEnumEvts);// get collection of logged events
        if (SUCCEEDED(hr))
        {
            if (spEnumEvts->Count > 0)
            {
                for (long i = 0; i < spEnumEvts->Count; i++)
                {
                    ITcEventPtr spEvent;
                    hr = spEnumEvts->Item(i, &spEvent); // get event
                    if (SUCCEEDED(hr)) {

                        BSTR msgString = 0;
                        BSTR fmtProgId = 0;
                        BSTR srcName = 0;
                        DATE dtCreated = 0;
                        DATE dtConfirmed = 0;
                        DATE dtReset= 0;
                        long msCreated = 0;
                        long msConfirmed = 0;
                        long msReset = 0;

                        hr = spEvent->get_Date(&dtCreated);
                        hr = spEvent->get_DateConfirmed(&dtConfirmed);
                        hr = spEvent->get_DateReset(&dtReset);
                        hr = spEvent->get_Ms(&msCreated);
                        hr = spEvent->get_MsConfirmed(&msConfirmed);
                        hr = spEvent->get_MsReset(&msReset);

                        wprintf( L"Event number: %d, Id: %d, SrcId: %d, InvokeId: %d\n", i+1, spEven
t->Id, spEvent->SrcId, spEvent->InvokeId );
                        wprintf( L"Class: %d, Priority: %d, Flags: 0x%X\n", spEvent->Class, spEvent-
>Priority, spEvent->Flags );
                        wprintf( L"State: 0x%X, MustConfirm: %d, UserData: %d\n", spEvent-
>State, spEvent->MustConfirm, spEvent->UserData );
                        wprintf( L"Date created: %s.
%d\n", FormatDateTime( L"dddd, mmmm d, yyyy ' at ' hh:mm:ss", dtCreated ), msCreated );
                        wprintf( L"Date confirmed: %s.
%d\n", FormatDateTime( L"dddd, mmmm d, yyyy ' at ' hh:mm:ss", dtConfirmed ), msConfirmed );
                        wprintf( L"Date reset: %s.
%d\n", FormatDateTime( L"dddd, mmmm d, yyyy ' at ' hh:mm:ss", dtReset ), msReset );

                        hr = spEvent->get_FmtProgId(&fmtProgId);
                        if (SUCCEEDED(hr)) {
                            wprintf( L"FmtProgId: \"%s\"\n", fmtProgId );
                            SysFreeString( fmtProgId );
                        }

                        hr = spEvent->get_SourceName( langID, &srcName );
                        if (SUCCEEDED(hr)) {
                            wprintf( L"SourceName: \"%s\"\n", srcName );
                            SysFreeString( srcName );
                        }

                        hr = spEvent->GetMsgString( langID, &msgString );

```

```

        if (SUCCEEDED(hr)) {
            wprintf( L"Message: \"%s\"\n\n", msgString );
            SysFreeString( msgString );
        }

        }// if (SUCCEEDED(hr)), spEvent
    }// for (long i ...
} // if (spEnumEvts->Count > 0)
} // if (SUCCEEDED(hr)), spEnumEvts
} // if (SUCCEEDED(hr)), spTcEventLog

SysFreeString(comServerInfo.pwszName);
wprintf( L"Press ENTER to continue...\n" );
_gettchar();
return 0;
}

```

Language / IDE	Source code
CodeGear C++Builder 2009	https://infosys.beckhoff.com/content/1033/TcEventLogger/Resources/12333057163/.zip

10.6.3 Console Application - Read logged alarms via ADS proxy

System requirements:

- CodeGear C++Builder 2009;
- TwinCAT v2.11 B2228 or higher;
- Import the library TcEventLogger.exe (TCEVENTLOGGERLib_TLB.h);
- Import the library TcEventLogAdsProxy.dll (TCEVENTLOGPROXYLib_TLB.h):

The samples creates an ADS proxy connection to a Remote-EventLogger and shows all logged EventLogger messages in the console.


```

C:\Users\..._nDocuments\RAD Studio\Projects\Sample02\Release\Sample02.exe
SourceName: "EnglishText"
Message: "High temperature!"

Event number: 511, Id: 1, SrcId: 17001, InvokeId: 0
Class: 6, Priority: 0, Flags: 0x9173
State: 0x12, MustConfirm: 1, UserData: 0
Date created: Tuesday, November 6, 2012 at 12:23:49.951
Date confirmed: Tuesday, November 6, 2012 at 12:23:49.964
Date reset: Tuesday, November 6, 2012 at 12:23:50.991
FmtProgId: "TcEventFormatter.TcXmlFormatter"
SourceName: "EnglishText"
Message: "Power fail detected!"

Event number: 512, Id: 2, SrcId: 17001, InvokeId: 0
Class: 6, Priority: 0, Flags: 0x9173
State: 0x12, MustConfirm: 1, UserData: 0
Date created: Tuesday, November 6, 2012 at 12:23:50.351
Date confirmed: Tuesday, November 6, 2012 at 12:23:50.363
Date reset: Tuesday, November 6, 2012 at 12:23:50.841
FmtProgId: "TcEventFormatter.TcXmlFormatter"
SourceName: "EnglishText"
Message: "High temperature!"

Press ENTER to continue...

```

```

#include <vcl.h>
#pragma hdrstop

#include <tchar.h>
#include <stdio.h>
#include <string.h>
#include <iostream.h>
#include <TCEVENTLOGGERLib_TLB.h>
#include <TCEVENTLOGPROXYLib_TLB.h>

//-----
#pragma argsused
int _tmain(int argc, _TCHAR* argv[])
{
    // Connect via ADS
    // BSTR netID = SysAllocString( L"10.1.128.220.1.1" );// ToDo: Configure ams net id if connection to
    remote TwinCAT SystemBSTR netID = SysAllocString( L"" );
    long langID = 1033;
    ITcEventLogAdsProxyPtr spAdsProxy;
    HRESULT hr = spAdsProxy.CreateInstance(CLSID_TcEventLogAdsProxy);
    if (SUCCEEDED(hr))
    {
        hr = spAdsProxy->Connect( netID );// connect to the remote/local TwinCAT system
        if (SUCCEEDED(hr))
        {
            ITcEventLogPtr spTcEventLog;
            hr = spAdsProxy->QueryInterface( IID_ITcEventLog, (void*)&spTcEventLog);
            if (SUCCEEDED(hr))
            {
                long nLoggedEvents = 0;
                long nActiveEvents = 0;
                hr = spTcEventLog->get_LoggedEvents(&nLoggedEvents);
                hr = spTcEventLog->get_ActiveEvents(&nActiveEvents);
                wprintf( L"Max. number of events: %d logged, %d active.
\n", nLoggedEvents, nActiveEvents );

                ITcEnumEventsExPtr spEnumEvts;
                hr = spTcEventLog-
>EnumLoggedEventsEx(&spEnumEvts);// get collection of logged events
                if (SUCCEEDED(hr))
                {
                    if (spEnumEvts->Count > 0)
                    {
                        for (long i = 0; i < spEnumEvts->Count; i++)
                        {
                            ITcEventPtr spEvent;
                            hr = spEnumEvts->Item(i, &spEvent); // get event
                            if (SUCCEEDED(hr))
                            {

```

```

        BSTR msgString = 0;
        BSTR fmtProgId = 0;
        BSTR srcName = 0;
        DATE dtCreated = 0;
        DATE dtConfirmed = 0;
        DATE dtReset = 0;
        long msCreated = 0;
        long msConfirmed = 0;
        long msReset = 0;

        hr = spEvent->get_Date(&dtCreated);
        hr = spEvent->get_DateConfirmed(&dtConfirmed);
        hr = spEvent->get_DateReset(&dtReset);
        hr = spEvent->get_Ms(&msCreated);
        hr = spEvent->get_MsConfirmed(&msConfirmed);
        hr = spEvent->get_MsReset(&msReset);

        wprintf( L"Event number: %d, Id: %d, SrcId: %d, InvokeId: %d\n", i+1
, spEvent->Id, spEvent->SrcId, spEvent->InvokeId );
        wprintf( L"Class: %d, Priority: %d, Flags: 0x%X\n", spEvent-
>Class, spEvent->Priority, spEvent->Flags );
        wprintf( L"State: 0x%X, MustConfirm: %d, UserData: %d\n", spEvent-
>State, spEvent->MustConfirm, spEvent->UserData );
        wprintf( L"Date created: %s.
%d\n", FormatDateTime( L"dddd, mmmm d, yyyy ' at ' hh:mm:ss", dtCreated ), msCreated );
        wprintf( L"Date confirmed: %s.
%d\n", FormatDateTime( L"dddd, mmmm d, yyyy ' at ' hh:mm:ss", dtConfirmed ), msConfirmed );
        wprintf( L"Date reset: %s.
%d\n", FormatDateTime( L"dddd, mmmm d, yyyy ' at ' hh:mm:ss", dtReset ), msReset );

        hr = spEvent->get_FmtProgId(&fmtProgId);
        if (SUCCEEDED(hr)) {
            wprintf( L"FmtProgId: \"%s\"\n", fmtProgId );
            SysFreeString( fmtProgId );
        }

        hr = spEvent->get_SourceName( langID, &srcName );
        if (SUCCEEDED(hr)) {
            wprintf( L"SourceName: \"%s\"\n", srcName );
            SysFreeString( srcName );
        }

        hr = spEvent->GetMsgString( langID, &msgString );
        if (SUCCEEDED(hr)) {
            wprintf( L"Message: \"%s\"\n\n", msgString );
            SysFreeString( msgString );
        }
    } // if (SUCCEEDED(hr)), spEvent
} // for (long i ...
} // if (spEnumEvts->Count > 0)
} // if (SUCCEEDED(hr)), spEnumEvts
} //if (SUCCEEDED(hr)), spTcEventLog

spAdsProxy->Disconnect();

} //if (SUCCEEDED(hr)), Connect(...)
} // if (SUCCEEDED(hr)), CreateInstance(..

SysFreeString(netID);
wprintf( L"Press ENTER to continue...\n" );
_gettchar();
return 0;
}

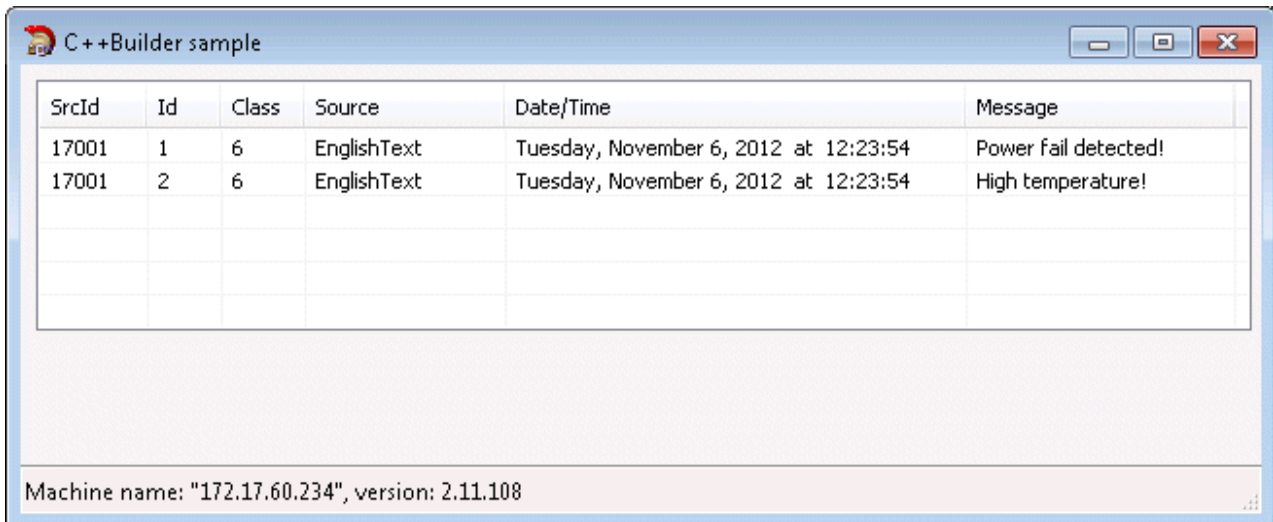
```

Language / IDE	Source code
CodeGear C++Builder 2009	https://infosys.beckhoff.com/content/1033/TcEventLogger/Resources/12333058571/.zip

10.6.4 View active alarms in user defined ListView

System requirements:

- CodeGear C++Builder 2009;
- TwinCAT v2.11 B2228 or higher;
- Import the library TcEventLogger.exe (TCEVENTLOGGERLib_OCX.h);



The sample shows the active alarms in a user-defined C++Builder list view (TListView-Control).

The following steps are needed to create the sample:

1. Create a new VCL form application
2. Move the TTcEventLog and TStatusBar activeX components from the component palette to the form.
3. Implement the methods: FormCreate(), FormClose(). With FormCreate() the connection to EventLogger is configured / created. With FormClose() the connection is separated;
4. Implement the event routines: OnNewEvent, OnResetEvent. The example project supports thereby alarms without acknowledgement. To receive alarms with acknowledge you have to implement further event routines: OnSignalEvent() and OnConfirmEvent();
5. Implement the event routine OnShutdown(). This routine is called if the last instance of the Eventloggers (e.g. at shutdown of the system) is discharged completely;
6. Implement the help methods: AddNewEvent() and DisplayActiveAlarms(). These methods are used to read the active messages and to show them in the list;
7. Implement the help method: GetEventPosByID(). These method searches in the list view for a specific message with the help of the SourceId and ID and supplies in case of success the row number (itemIndex). You can delete the message from the list with the help of the row number;

```
//-----
#include <vcl.h>
#pragma hdrstop

#include "Unit2.h"
//-----
#pragma package(smart_init)
#pragma link "TCEVENTLOGGERLib_OCX"
#pragma resource "*.dfm"
TForm2 *Form2;

//-----
__fastcall TForm2::TForm2(TComponent* Owner)
```

```

: TForm(Owner)
{
}

const char ColH[6][10] = { "SrcId", "Id", "Class", "Source", "Date/Time", "Message" };

//-----
// This method is called when the form is loaded
void __fastcall TForm2::FormCreate(TObject *Sender)
{
    bool bRemote = true;
    if (bRemote) {
        TcEventLog1->ConnectKind = ckRemote;
        TcEventLog1->RemoteMachineName = "172.17.60.234";
        TcEventLog1->AutoConnect = true;
    }
    langID = 1033; // language id // initialize ListView
    ListView1->ViewStyle = vsReport;
    ListView1->GridLines = true;
    ListView1->RowSelect = true;

    // init/add column header
    ListView1->Columns->Clear();
    for (int i = 0; i < 6; i++)
    {
        TListColumn *pCol = ListView1->Columns->Add();
        if (pCol)
        {
            pCol->Caption = ColH[i];
            pCol->AutoSize = true;
        }
    }

    long v = 0, r = 0, b = 0;
    TcEventLog1->GetVersion(&v, &r, &b);

    StatusBar1->SimplePanel = true;
    StatusBar1->SimpleText = Format( "Machine name: \"%s\", version: %d.%d.%d",
    ARRAYOFCONST((TcEventLog1->RemoteMachineName, v, r, b)));

    DisplayActiveAlarms();
}
//-----
// Display all active alarms
void TForm2::DisplayActiveAlarms()
{
    // Clear the ListView
    ListView1->Items->Clear();
    // Get collection of active events
    ITcEnumEventsExPtr spEnumEvts = TcEventLog1->EnumActiveEventsEx();
    if (spEnumEvts)
    {
        if (spEnumEvts->Count > 0)
        {
            for (long i = 0; i < spEnumEvts->Count; i++)
            {
                // Here we get one event from the collection
                ITcEventPtr spEvent;
                HRESULT hr = spEnumEvts->Item(i, &spEvent);
                if (SUCCEEDED(hr))
                {
                    // Add event to the ListView
                    AddNewEvent( spEvent );
                }
            } // for (long i = 0 ...
        } // if (spEnumEvts->Count ...
    } // if (SUCCEEDED(hr))
}
//-----
// This method adds new event to the list
void TForm2::AddNewEvent( ITcEventPtr spEvent )
{
    TListItem *pItem = ListView1->Items->Add();
    if (pItem) {
        // Save Id to Data property for later use
        pItem->Data = (void*)spEvent->Id;

        // Get event SrcId
        pItem->Caption = spEvent->SrcId;
    }
}

```

```

// Get event Id
pItem->SubItems->Add( spEvent->Id );

// Get event Class
pItem->SubItems->Add( spEvent->Class );

// Get source SourceName
BSTR srcName = 0;
HRESULT hr = spEvent->get_SourceName( langID, &srcName );
if (SUCCEEDED(hr)) {
    pItem->SubItems->Add( srcName );
    SysFreeString( srcName );
}

// Get event creation date/time
DATE dtCreated = 0;
hr = spEvent->get_Date(&dtCreated);
if (SUCCEEDED(hr)) {
    pItem->SubItems-
>Add( FormatDateTime( L"dddd, mmmm d, yyyy ' at ' hh:mm:ss", dtCreated ) );
}

// Get the event message text
BSTR msgString = 0;
hr = spEvent->GetMsgString( langID, &msgString );
if (SUCCEEDED(hr)) {
    pItem->SubItems->Add( msgString );
    SysFreeString( msgString );
}
} // if (pItem) ...
}
//-----
// This method returns ListView event index
// Return value: true => Success, false => event entry not found in the list
bool TForm2::GetEventPosByID( long SrcId, long Id, int &itemIndex )
{
    itemIndex = 0;
    TListItem *pItem;
    int startIndex = 0;
    do{
        // search for the event in the list (by SrcId + Id)
        pItem = ListView1->FindCaption(startIndex, SrcId, true, true, true );
        if (pItem) {
            startIndex = pItem->Index + 1;
            if ( Id == (long)pItem->Data )
            {
                itemIndex = pItem->Index;
                return true;
            }
        }
    }while (pItem);
    return false;
}
//-----
// This method is called when a new alarm is issued
void __fastcall TForm2::TcEventLog1NewEvent(TObject *Sender, LPDISPATCH srcObj)
{
    AddNewEvent( srcObj );
}
//-----
// This method is called when a alarm is reset
void __fastcall TForm2::TcEventLog1ResetEvent(TObject *Sender, LPDISPATCH srcObj)
{
    ITcEventPtr spEvent = srcObj;
    int itemIndex = 0;
    if( GetEventPosByID( spEvent->SrcId, spEvent->Id, itemIndex ) )
        ListView1->Items->Delete(itemIndex); //removes event from the list
}
//-----
void __fastcall TForm2::FormClose(TObject *Sender, TCloseAction &Action)
{
    TcEventLog1->Disconnect();
}
//-----
void __fastcall TForm2::TcEventLog1Shutdown(TObject *Sender, Variant shutdownParm)
{
    TcEventLog1->Disconnect();
}

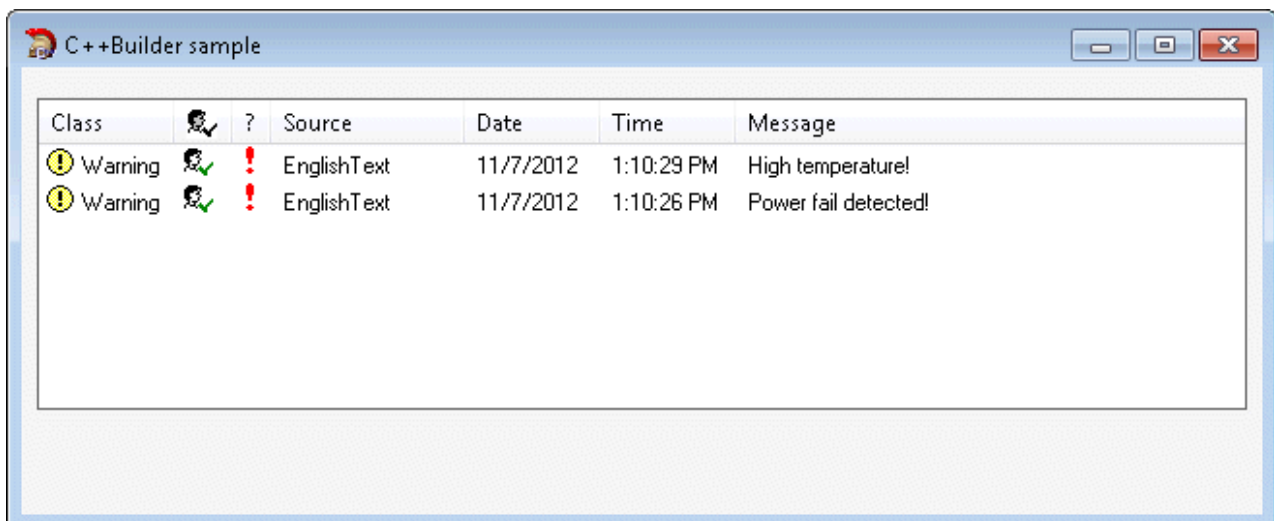
```

Language / IDE	Source code
CodeGear C++Builder 2009	https://infosys.beckhoff.com/content/1033/TcEventLogger/Resources/12333059979/.zip

10.6.5 Einbinden von TcEventViewer-ActiveX-Control

System requirements:

- CodeGear C++Builder 2009;
- TwinCAT v2.11 B2228 or higher;
- Import the type library of the TcEventViewer.dll (TCEVENTVIEWERLib_OCX);



The sample below uses the TcEventViewer-activeX control to display active alarms on a TwinCAT System.

The following steps are necessary to create the sample:

1. Create a new VCL formula application;
2. Move TTcEventView from the component palette to the form;
3. Implement the methods: FormCreate(), FormClose();

With FormCreate() the connection to EventLogger is configured/created. With FormClose() the connection is separated. If you want to access to a remote PC you have to configure the network address of the TwinCAT System (AmsNetID).

```
#include <vcl.h>
#pragma hdrstop

#include "Unit2.h"
//-----
#pragma package(smart_init)
#pragma link "TCEVENTVIEWERLib_OCX"
#pragma resource "*.dfm"
TForm2 *Form2;
//-----
```

```
__fastcall TForm2::TForm2(TComponent* Owner)
: TForm(Owner)
{
    address = 0;
    bRemote = false;
}

//-----
// This method is called when the form is loaded
void __fastcall TForm2::FormCreate(TObject *Sender)
{
    if (bRemote) {
        address = ::SysAllocString(L"ADS://10.1.128.220.1.1");
        TcEventView1->AddConnection(address);
    }

    TcEventView1->ShowActiveEvents();
}
//-----
void __fastcall TForm2::FormClose(TObject *Sender, TCloseAction &Action)
{
    if (bRemote) {
        TcEventView1->DeleteConnection(address);
        if (address) {
            ::SysFreeString(address);
            address = 0;
        }
    }
}
}
```

Language / IDE	Source code
CodeGear C++Builder 2009	https://infosys.beckhoff.com/content/1033/TcEventLogger/Resources/12333061387/.zip

More Information:
www.beckhoff.com/automation

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

