BECKHOFF New Automation Technology

Handbuch | DE

TcEventLogger

TwinCAT 2 | System

22.09.2022 | Version: 1.1

Inhaltsverzeichnis

1 Vorwort

1.1 Hinweise zur Dokumentation

Diese Beschreibung wendet sich ausschließlich an ausgebildetes Fachpersonal der Steuerungs- und Automatisierungstechnik, das mit den geltenden nationalen Normen vertraut ist.
Zur Installation und Inbetriebnahme der Komponenten ist die Beachtung der Dokumentation und der nachfolgenden Hinweise und Erklärungen unbedingt notwendig.
Das Fachpersonal ist verpflichtet, für jede Installation und Inbetriebnahme die zu dem betreffenden Zeitpunkt veröffentliche Dokumentation zu verwenden.

Das Fachpersonal hat sicherzustellen, dass die Anwendung bzw. der Einsatz der beschriebenen Produkte alle Sicherheitsanforderungen, einschließlich sämtlicher anwendbaren Gesetze, Vorschriften, Bestimmungen und Normen erfüllt.

Disclaimer

Diese Dokumentation wurde sorgfältig erstellt. Die beschriebenen Produkte werden jedoch ständig weiter entwickelt.
Wir behalten uns das Recht vor, die Dokumentation jederzeit und ohne Ankündigung zu überarbeiten und zu ändern.
Aus den Angaben, Abbildungen und Beschreibungen in dieser Dokumentation können keine Ansprüche auf Änderung bereits gelieferter Produkte geltend gemacht werden.

Marken

Beckhoff®, TwinCAT®, TwinCAT/BSD®, TC/BSD®, EtherCAT®, EtherCAT G®, EtherCAT G10®, EtherCAT P®, Safety over EtherCAT®, TwinSAFE®, XFC®, XTS® und XPlanar® sind eingetragene und lizenzierte Marken der Beckhoff Automation GmbH.
Die Verwendung anderer in dieser Dokumentation enthaltenen Marken oder Kennzeichen durch Dritte kann zu einer Verletzung von Rechten der Inhaber der entsprechenden Bezeichnungen führen.

Patente

Die EtherCAT-Technologie ist patentrechtlich geschützt, insbesondere durch folgende Anmeldungen und Patente:
EP1590927, EP1789857, EP1456722, EP2137893, DE102015105702
mit den entsprechenden Anmeldungen und Eintragungen in verschiedenen anderen Ländern.

EtherCAT®

EtherCAT® ist eine eingetragene Marke und patentierte Technologie lizenziert durch die Beckhoff Automation GmbH, Deutschland

Copyright

1.2 Sicherheitshinweise

Sicherheitsbestimmungen

Beachten Sie die folgenden Sicherheitshinweise und Erklärungen!
Produktspezifische Sicherheitshinweise finden Sie auf den folgenden Seiten oder in den Bereichen Montage, Verdrahtung, Inbetriebnahme usw.

Haftungsausschluss

Die gesamten Komponenten werden je nach Anwendungsbestimmungen in bestimmten Hard- und Software-Konfigurationen ausgeliefert. Änderungen der Hard- oder Software-Konfiguration, die über die dokumentierten Möglichkeiten hinausgehen, sind unzulässig und bewirken den Haftungsausschluss der Beckhoff Automation GmbH & Co. KG.

Qualifikation des Personals

Diese Beschreibung wendet sich ausschließlich an ausgebildetes Fachpersonal der Steuerungs-, Automatisierungs- und Antriebstechnik, das mit den geltenden Normen vertraut ist.

Erklärung der Symbole

In der vorliegenden Dokumentation werden die folgenden Symbole mit einem nebenstehenden Sicherheitshinweis oder Hinweistext verwendet. Die Sicherheitshinweise sind aufmerksam zu lesen und unbedingt zu befolgen!

⚠ GEFAHR

Akute Verletzungsgefahr!

Wenn der Sicherheitshinweis neben diesem Symbol nicht beachtet wird, besteht unmittelbare Gefahr für Leben und Gesundheit von Personen!

⚠ WARNUNG

Verletzungsgefahr!

Wenn der Sicherheitshinweis neben diesem Symbol nicht beachtet wird, besteht Gefahr für Leben und Gesundheit von Personen!

⚠ VORSICHT

Schädigung von Personen!

Wenn der Sicherheitshinweis neben diesem Symbol nicht beachtet wird, können Personen geschädigt werden!

HINWEIS

Schädigung von Umwelt oder Geräten

Wenn der Hinweis neben diesem Symbol nicht beachtet wird, können Umwelt oder Geräte geschädigt werden.

Tipp oder Fingerzeig

Dieses Symbol kennzeichnet Informationen, die zum besseren Verständnis beitragen.

1.3 Hinweise zur Informationssicherheit

Die Produkte der Beckhoff Automation GmbH & Co. KG (Beckhoff) sind, sofern sie online zu erreichen sind, mit Security-Funktionen ausgestattet, die den sicheren Betrieb von Anlagen, Systemen, Maschinen und Netzwerken unterstützen. Trotz der Security-Funktionen sind die Erstellung, Implementierung und ständige Aktualisierung eines ganzheitlichen Security-Konzepts für den Betrieb notwendig, um die jeweilige Anlage, das System, die Maschine und die Netzwerke gegen Cyber-Bedrohungen zu schützen. Die von Beckhoff verkauften Produkte bilden dabei nur einen Teil des gesamtheitlichen Security-Konzepts. Der Kunde ist dafür verantwortlich, dass unbefugte Zugriffe durch Dritte auf seine Anlagen, Systeme, Maschinen und Netzwerke verhindert werden. Letztere sollten nur mit dem Unternehmensnetzwerk oder dem Internet verbunden werden, wenn entsprechende Schutzmaßnahmen eingerichtet wurden.

Zusätzlich sollten die Empfehlungen von Beckhoff zu entsprechenden Schutzmaßnahmen beachtet werden. Weiterführende Informationen über Informationssicherheit und Industrial Security finden Sie in unserem https://www.beckhoff.de/secguide.

Die Produkte und Lösungen von Beckhoff werden ständig weiterentwickelt. Dies betrifft auch die Security-Funktionen. Aufgrund der stetigen Weiterentwicklung empfiehlt Beckhoff ausdrücklich, die Produkte ständig auf dem aktuellen Stand zu halten und nach Bereitstellung von Updates diese auf die Produkte aufzuspielen. Die Verwendung veralteter oder nicht mehr unterstützter Produktversionen kann das Risiko von Cyber-Bedrohungen erhöhen.

Um stets über Hinweise zur Informationssicherheit zu Produkten von Beckhoff informiert zu sein, abonnieren Sie den RSS Feed unter https://www.beckhoff.de/secinfo.

2 Einführung

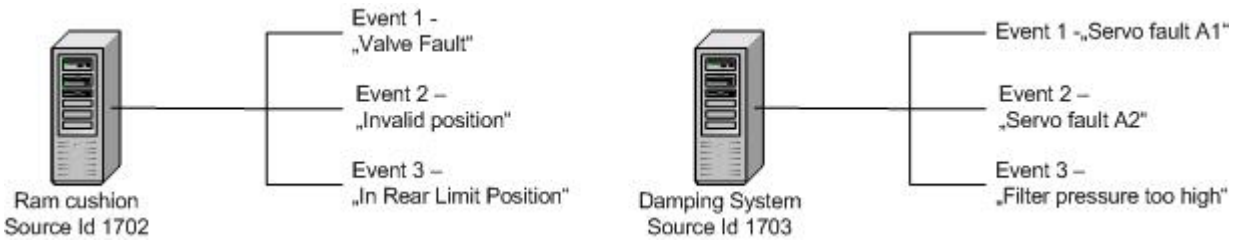
Der TcEventLogger ist Beckhoffs Standardmechanismus zum Verwalten von Meldungen aus der SPS. Mithilfe des TcEventLoggers können Warnungen, Fehlermeldungen, Informationen oder Statusmeldungen in einem SPS Programm ausgelöst und durch Visualisierungen angezeigt werden. Durch die Verwendung des TcEventLoggers können sie folgende Anforderungen realisieren:

1. Meldungen und Alarme in der SPS auslösen oder zurücksetzen
2. Meldungen durch den Eventlogger verwalten lassen
3. COM APIs zur HMI Entwicklung
4. Umfangreiches ActiveX Control zur Anzeige von Meldungen
5. Unterstützung für die Realisierung verteilter Systeme: Das HMI kann mühelos die Meldungen eines remote Systems anzeigen
6. Meldungstexte werden sprachabhängig aus einer Datenbank geladen

Das TwinCAT Meldungs-Konzept

Meldungen (Events)

Die Meldungen des TcEventLoggers werden anhand einer Source- und einer EventID identifiziert. Hierdurch wird eine möglichst realitätsnahe Sicht auf Meldungen ermöglicht. Die SourceID wird einer Maschine und die EventID einer Meldung dieser Maschine zugeordnet.



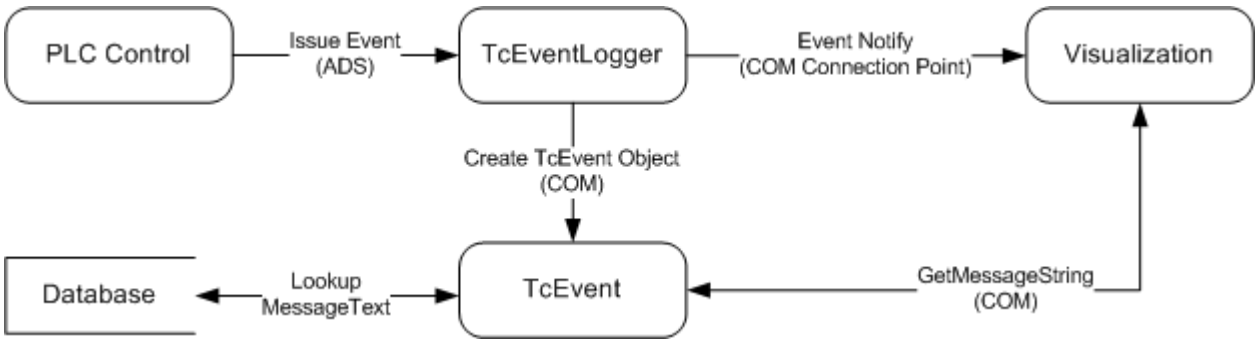
Zusätzlich können zu jeder Meldung beliebig viele DocLinks gespeichert werden.
Ein DocLink wird durch einen Namen identifiziert und enthält einen Link zu einer Datei. Diese verlinkten Dateien können zusätzliche Informationen enthalten, z.B. Maßnahmen zum Beheben von Fehlern . Der TcEventlogger verarbeitet diese DocLinks nicht! Es bleibt letztlich den Clients überlassen die DocLinks abzufragen und zu verarbeiten. Die DocLinks werden mit den Meldungstexten gespeichert und durch den entsprechenden Formatter (siehe 'Formatieren von Meldungen [▶ 10]') verarbeitet.

Auslösen von Meldungen

Aus einem SPS Programm heraus kann ein einfacher Funktionsbaustein aufgerufen werden, welcher dem TcEventlogger intern eine Meldung signalisiert. Diese Meldung wird nur anhand ihrer Source- und EventID identifiziert

Sobald der TcEventLogger eine Meldung erhält, informiert er alle verbundenen Clients hierüber und verarbeitet die Meldung intern. Die Clients erhalten mit dem Rückruf ein Meldungsobjekt - ein COM-Objekt von Typ TcEvent [▶ 159].

Die Visualisierung erhält nun das Meldungsobjekt, welches die Event- und SourceID enthält. Sie kann an dem Event Objekt den zugehörigen Meldungstext abfragen. Der Meldungstext wird dann in der gewünschten Sprache aus einer Datenbank geladen.



Formatieren von Meldungen

Zum Laden der Meldungstexte bietet der TcEventlogger das TcEventFormatter [▶ 117] Konzept:
Ein Formatter ist für das Laden der Meldungstexte aus der entsprechenden Datenbank zuständig.

Formatter können kundenspezifisch implementiert und in das System integriert werden. Mit dem TcEventlogger werden zwei Standardlösungen angeboten:

Formatter	Beschreibung
TcEventFormatter	Laden von Meldungstexten aus dem Projekt Storage.
TcXMLFormatter	Datensätze werden aus XML Dateien geladen.

Der zu verwendende Formatter wird schon beim Auslösen der Meldung (am PLC Baustein) ausgewählt.

Der Formatter ist auch für das Laden der entsprechenden DocLinks verantwortlich.

Konfigurieren von Meldungen

Um Meldungstexte aus einer Datenbank zu laden, muss die Datenbank zunächst gefüllt werden. Hierfür bietet Beckhoff den TcEventKonfigurator [▶ 21] als grafische Benutzeroberfläche an.

Der TcEventKonfigurator unterstützt die Formate der beiden Standard-Formatter sowie ein Plain Text Format. Dieses Plain Text Format kann z.B. an Übersetzungsbüros weitergegeben werden.

Mithilfe des TcEventkonfigurators kann der gesamte Vorgang zum Generieren von Meldungen realisiert werden:

- Meldungen werden konfiguriert
- Daten werden in einem wählbaren Format gespeichert
- SPS Sample Code wird generiert.

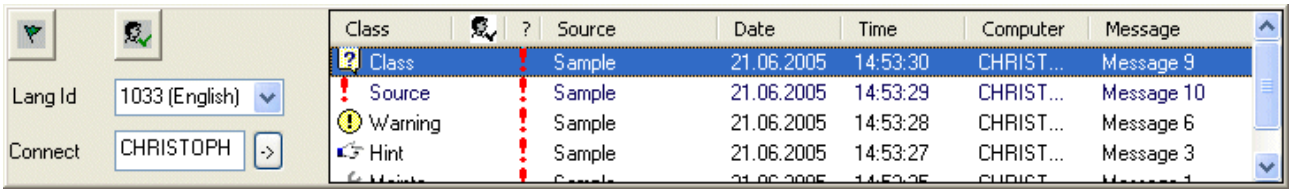
Der Funktionskern des TcEventKonfigurators ist in Form von COM Komponenten realisiert, so dass er auch in eine Kundenapplikation integriert werden kann.

Visualisierungen

Wenn sich auch eine minimale Implementierung einer Visualisierung mit einigen wenigen Zeilen Quellcode erreichen lässt, so fällt für die Implementierung einer voll funktionalen Visualisierung trotzdem ein verhältnismäßig großer Programmieraufwand an.

Da sich die Implementierungen oft ähneln, bietet Beckhoff seinen Kunden hier eine Standard Implementierung zur Anzeige von Meldungen an : den TcEventViewer [▶ 68]. Der TcEventViewer ist ein umfassend konfigurierbares ActiveX Control zur Einbindung in die Kundenapplikation.

Als beispielhafte Implementierung und zum Betrachten von Meldungen bietet Beckhoff die TcEventbar an. Die TcEventbar ist eine Visualisierung, die den TcEventViewer nutzt um Meldungen anzuzeigen.



Die TcEventBar liegt im 'TwinCAT/EventLogger' Verzeichniss.

Einstellungen der TcEventBar

Die TcEventBar nutzt (seit der v.2.9.0.11) einen Registry Key (`HKEY_CURRENT_USER\Software\TwinCAT\TcEventBar`), um die Einstellungen des Kontext Menüs zu speichern. Die folgenden Flags können in "ContextMenuFlags" gesetzt werden um die Menü Einträge zu deaktivieren.

TcEventView:	Beschreibung
0x00000001	"Aktive Meldungen löschen"
0x00000002	"Logbuch löschen"
0x00000004	"Filter setzen"
0x00000008	"Quittieren"
0x00000010	"Rücksetzen"
0x00000020	"Details"
0x0000FFFF	Deaktivierung des gesamten Kontextmenüs

Hauptfenster Kontextmenü:	Beschreibung
0x00010000	"Always on top"
0x00020000	"Autohide"
0x00040000	"Exit"
0xFFFF0000	Deaktivierung des gesamten Kontextmenüs

Connectivity

Der TcEventLogger bietet zwei Wege zur Realisierung von verteilten Systemen:

DCOM

DCOM wird nur auf NT / XP Plattformen unterstützt. Die Konfiguration von DCOM ist oft problematisch in Hinblick auf Konfiguration und Netzwerk-Timeouts

ADS Stream Interface

The TcEventLogger bietet ein binäres ADS basiertes Netzwerk Interface.

Um den Aufwand zur Entwicklung gering zu halten bietet Beckhoff eine Proxy [▶ 19] Implementierung welche binäre Codierung und Datenaustauch kapselt. Die TcEventLogAdsProxy [▶ 19] Klasse bietet alle Methoden der TcEventLogger COM API [▶ 128], benutzt statt DCOM jedoch ADS um die Verbindung zu remote Systemen herzustellen.

3 Alarm & Event für Windows NT/2000/XP/Vista

TwinCAT Systemfehler Loggen

Ab TwinCAT 2.8 kann das TwinCAT System NC Fehler und I/O Fehler im TcEventLogger protokollieren.

Das Protokollierverhalten kann durch den folgenden Registereintrag kontrolliert werden:

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

Mögliche Werte für LogMesssageType sind:

Wert	Beschreibung
0	TwinCAT System Meldung in das Betriebssystem eingeloggt.
1	TwinCAT System Meldung in den TcEventLogger [▶ 128] eingeloggt. Sie sind für das Benutzerprogramm wie das HMI in der Liste der protokollierten Events verfügbar. System Meldungen verwenden den TcXmlFormatter [▶ 117].
2	Defaultmodus , wie 0 und 1. TwinCAT System Meldung in das Betriebssystem und den TcEventLogger [▶ 128] eingeloggt.

Meldungsquellen, Datenhaltung und Anzeige auf einem TwinCAT System in Übersicht:

Meldungen auf einem TwinCAT (PC) System

Quelle	Meldungstext als XML-Datei	Datenbank		Anzeige		
		TcEventLogger (TwinCAT Projektspeicher *.tps - Datei)	Ereignisanzeige des Betriebssystems	TcEventViewer-Client z.B. TcEventBar.exe	Ereignisanzeige des Betriebssystems	TwinCAT System Manager Logger
Betriebssystem	-	-	**X**	-	**X**	-
ADS-LOGSTR, ADSLOGDINT, ADSLOGLREAL Funktionen	-	-	**X**	-	**X**	**X**
ADSLOGEVENT Funktionsbaustein	**X***	**X**	-	**X**	-	-
TwinCAT System NC und IO-Meldungen	**X** ab TwinCAT 2.8	**X**** nur bei LogMessageType 1 oder 2	**X** nur bei LogMessageType 0 oder 2	**X**** nur bei LogMessageType 1 oder 2	**X** nur bei LogMessageType 0 oder 2	**X**

- : Nicht möglich

X : Ja, möglich

***** Nur bei der Benutztung vom TcXmlFormatter.

****** Es werden nicht alle Meldungen gelogt um die Systemauslastung zu minimieren.

Logbuch abschalten

Der TcEventlogger hält sich ein kleines Logbuch von max 128 Meldungen. Dieses Logbuch speichert der TcEventlogger mit seiner Konfiguration im Projekt Storage. Um das Speichern im Storage zu unterbinden und auf temporäre Log Files umzuschalten kann ein Wert in der Registry gesetzt werden:

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

- Nicht vorhanden, oder = 0x1 lässt in den Storage Loggen.
- Vorhanden und Wert = 0x0 verhindert das Loggen in den Storage.

DCOM Security

Ab TwinCAT 2.11. werden die Rechte des TcEventLoggers implizit gesetzt. Um die Rechte (z.B. für remote Zugriff) mittels dcomcnfg zu konfigurieren muss folgender Wert in der Registry gesetzt werden:

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

- Nicht vorhanden oder = 0 : implizite Konfiguration der Zugriffsrechte
- >= 1 : Konfiguration via Registry/ Dcomcnfg

4 Alarm & Events für Windows CE

Der TcEventLoggerCE ist auf Beckhoffs WindowsCE Plattformen verfügbar seit Image Version

- 2.20a
- 3.06a

Unter WindowsCe wird nur der ADS Kommunikationskanal [▶ 17] unterstützt. Benutzen sie die TcEventLogAdsProxy [▶ 19] Library um eventlogging Lösungen zu realisieren.

5 Connectivity

Certain pruduction 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 propietary 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 avalable 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

Allgemein

Der TcEventLogger unterstützt DCOM Verbindungen zu Remote Systemen. DCOM erlaubt die Erzeugung virtueller lokaler Instanzen eines Objekts, welches physikalisch auf einem Remote PC läuft. Durch die Verwendung von DCOM können Sie eine Instanz auf einem Remote Evenlogger erzeugen und seine aktiven oder geloggten Events programmtechnisch abfragen.

EventViewer/ HMI

Der TcEventViewer ActiveX unterstützt DCOM Remote Verbindungen. Verwenden Sie die Methode AddConnection um einen Remote Eventlogger hinzu zu fügen. Der TcEventViewer unterstützt Verbindungen zu einer unbegrenzten Anzahl von virtuellen Remote Maschinen.

Hinweise

DCOM muss konfiguriert werden, um den Zugriff auf einen anderen Rechner zu erlauben. Weitere Informationen finden Sie im MSDN.

DCOM wird nicht von der Beckhoff Windows CE Plattform unterstützt.

5.1.1 Beispiele

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 neccesary 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 reqired 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

   ```
   TcEventLog m_logger = null;
   ```

3. Use theese 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 theese 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() [▶ 78] 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 [▶ 130]	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 [▶ 139]	This is the basic interface of ITcEventLogC3 and ITcEventLogC2.
ITcEventLogC2 [▶ 140]	This interface derives from ITcEventLogC and is the basic interface for ITcEventLogC3.
ITcEventLogC3 [▶ 142]	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 [▶ 146]	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 Methods and Properties	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.

ITcEventLog Methods and Properties	Description
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 Konfiguration

6.1 Beckhoff TcEventConfigurator

Der TcEventConfigurator ist ein Tool zur Einrichtung des Event Systems ohne Kenntnisse über das gesamte komplexe TcEventLogger System zu besitzen.
Die Dokumentation deckt folgende Themen ab:

1. First Steps [▶ 21] - Einführung in den TcEventConfigurator (Erstellen von Konfigurationen mit dem Wizard).
2. Referenz [▶ 22] für die UI des Konfigurators.
3. COM Interface [▶ 26] Dokumentation (für Entwickler die den Konfigurator in ihren eigenem Programm nutzen möchten).
4. Formate [▶ 24] des Konfigurators.

Der Beckhoff EventConfirator ist seit TwinCAT Version 2.10.0 (Build 1253) verfügbar.
Windows 7 support wird seit Version 2.0.0 (Build 68) des EventConfirators unterstützt.

Anforderungen:
.NET Framework 2.0

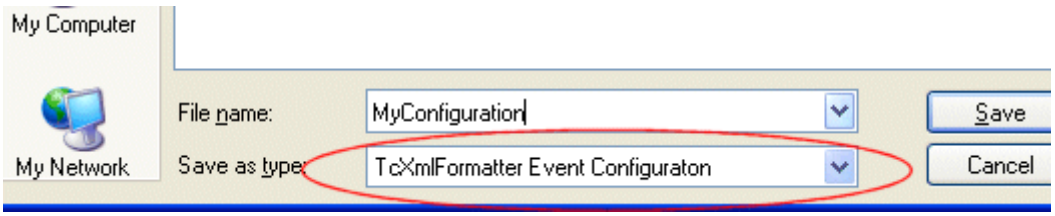
6.2 First Steps

Dieses Tutorial gibt Ihnen Schritt für Schritt Anweisungen zum Erstellen Ihrer ersten Meldung. Mit Hilfe eines übersichtlichen WIzards können Sie die Meldung konfigurieren und danach mit dem Konfigurator bearbeiten und weitere Meldungen hinzufügen.

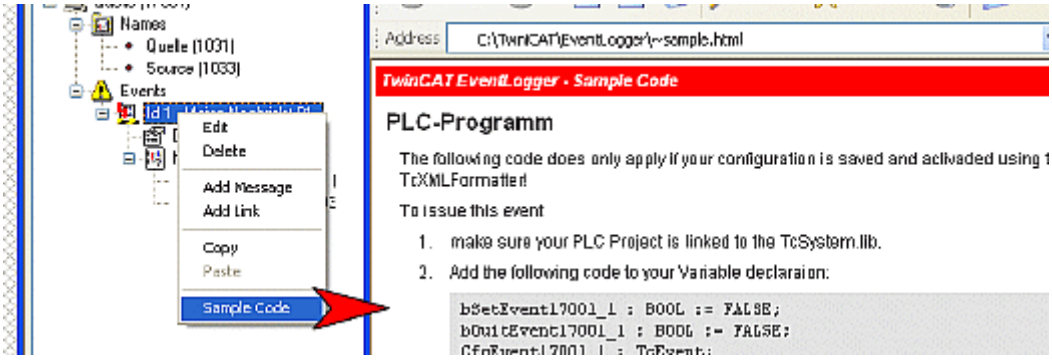
Wenn die Meldungen konfiguriert sind können Sie sich von dem EventKonfigurator vollständigen SPS Sample Code erzeugen lassen um die Meldung auszulösen.

1. Starten Sie den TcEventConfigurator.

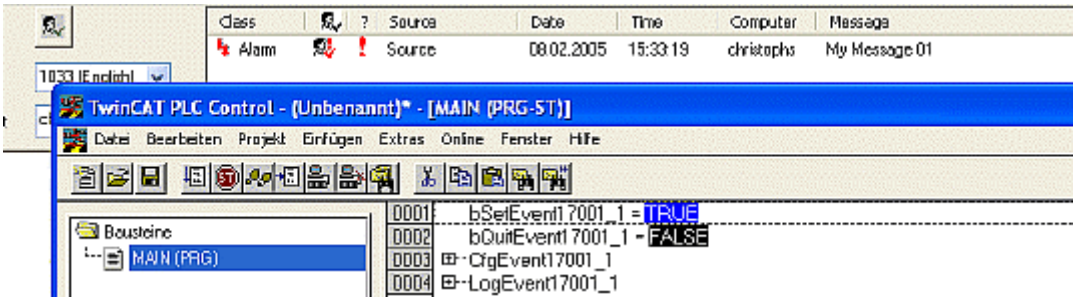
2. Falls Sie den Configurator zum ersten mal starten wird der Wizard automatisch starten.
   Anderenfalls müssen Sie den Wizard aus der Toolbar starten.
   Folgen Sie den Anweisungen des Wizards um eine Source und einen Event zu erstellen.
3. Falls Sie mehrere Events erstellen wollen wiederholen Sie Schritt 2.
4. Speichern Sie ihre Konfiguration im .ecp oder .xml Format.



5. Den SPS Code zum Auslösen einer Meldung können Sie sich nun vom Konfigurator erzeugen lassen, indem Sie die entsprechende Meldung auswählen und im Kontext Menü den Punkt '*Sample Code*' wählen.

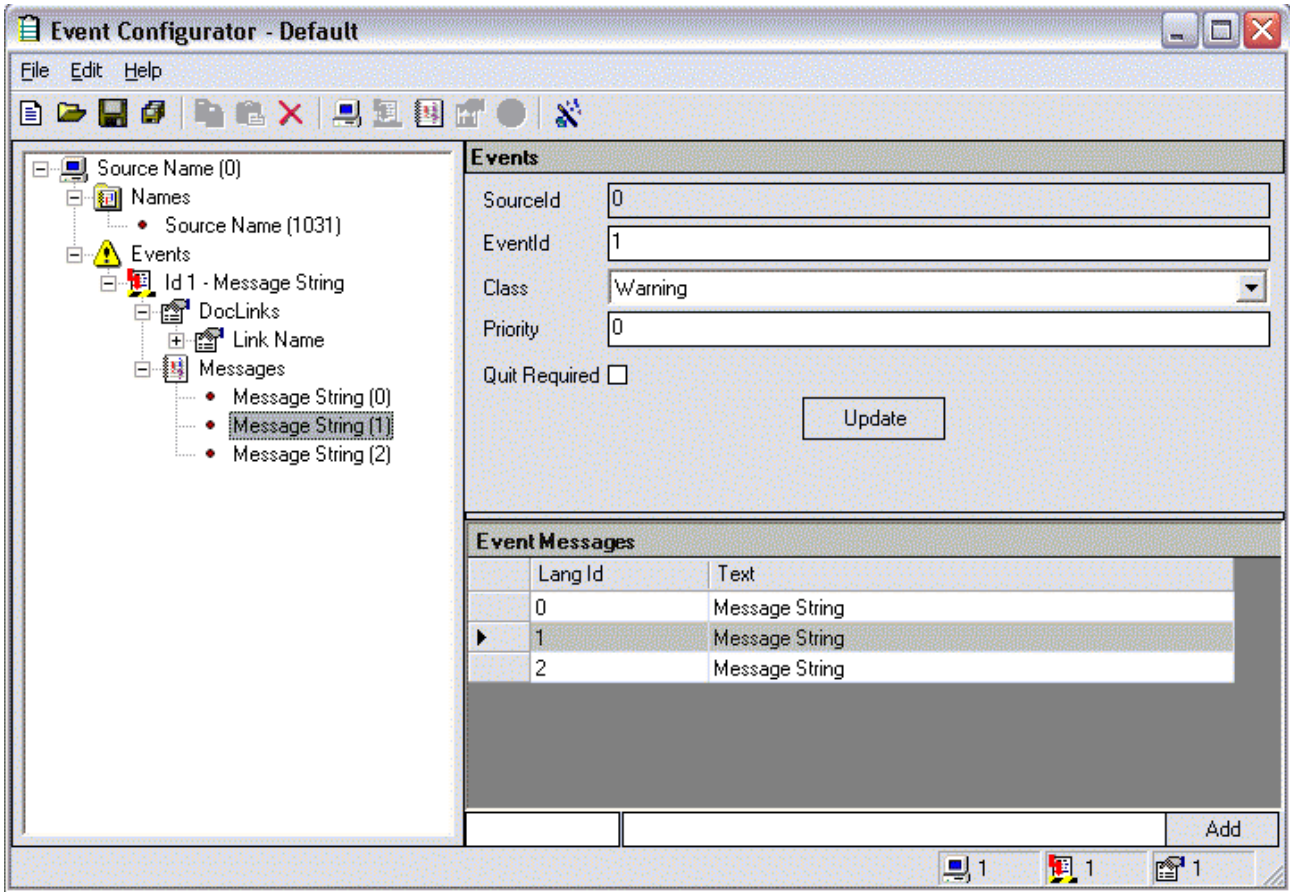


6. Fügen Sie den Code in Ihr SPS Programm ein und lösen Sie durch toggeln der Variablen '*bSetEventXXX*' die Meldung aus.
   Zur Anzeige der Meldungen können Sie die TcEventbar oder den EventKonfigurator nutzen.



6.3 TcEventConfigurator Referenz

Der TcEventConfigurator besteht aus einen TreeView links und einer vom Kontext abhängigen Eigenschaftsseite rechts.



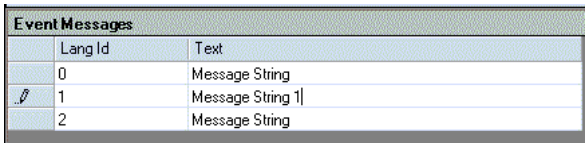
Die TreeView zeigt die Struktur der Event Konfiguration an.
Per Rechtsclick kann man alle Items bearbeiten/löschen.

	Source. Jede Source wird durch ein Source Icon dargestellt
	Der Ordner enthält alle Sourcenamen
	Der Event tag. Erweitern um alle Events zu sehen
	Message Tag. Erweitern um alle Message Strings des aktuellen Events zu sehen
	Document links zum aktuellen Event

Das rechte Panel zeigt detaillierte Eigenschaften zum ausgewählten Item.

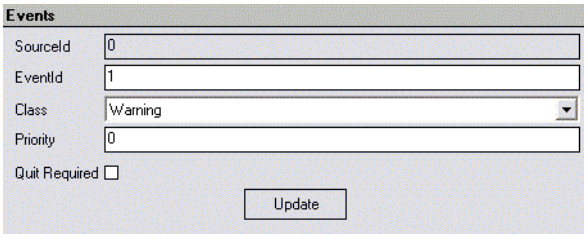
Hinzufügen/Bearbeiten vonSource Names, Messages oder URLs

Wählen sie im Tree aus was sie hinzufügen wollen, wenn eine Source ausgewählt ist kann ein Sourcename hinzugefügt werden, wenn ein Event ausgewählt ist können Messages hinzugefügt werden und wenn ein DokLink ausgewählt ist können URLs hinzugefügt werden. Sie bekommen eine editierbare Liste mit Einträgen für das ausgewählte Item. Geben sie eine LangID ins erste Feld und den text/namen ins zweite Feld ein und klicken sie auf 'Add'.



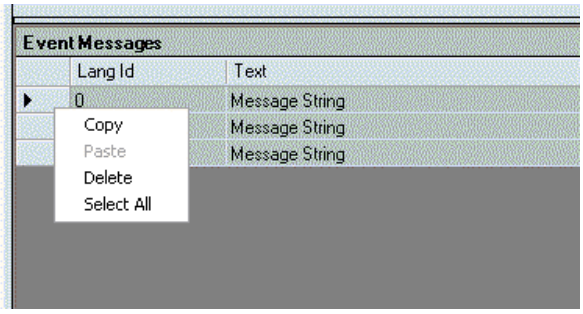
Events Bearbeiten

Wenn sie die Eigenschaften eines events ändern wollen müssen sie einen Event im Tree auswählen. Sie werden eine Eigenschaftsseite im rechten Panel bekommen in dem sie die Eigenschaften ändern können. Vergessen sie nicht auf 'Update' zu klicken, nachdem sie die eigenschaften geändert haben.



Löschen und kopieren von Einträgen

Sie können die Liste der messages benutzen um Messages zu kopieren oder zu löschen. Einfach eine oder mehrere Zeilen auswählen und Rechtsklicken.



6.4 TcEventConfigurator - Dateiformate

Der TcEventConfigurator bietet die folgenden Dateiformate an:

Format	Datei Endung	Beschreibung
Event Configuration Project Ex	ecpx	Natives TcEventConfigurator Format. Verbesserte Variante des ecp Formates
Event Configuration Project	ecp	Natives TcEventConfigurator Format. Dieses Format wurde duch das ecpx Format ersetzt.
TcXMLFormatter	xml	Lädt/Speichert Daten im XML Format.
TcEventFormatter	tps	Lädt/Speichert Daten im TwinCAT Projekt Storage
Excel	xls	Lädt/Speichert Daten im Excel xls Format zur Weitergabe an Übersetztungsbüros
Plain Text Format	txt	Lädt/Speichert Daten in txt Dateien

Format features

Nicht jedes Format unterstützt alle Features des TcEventKonfigurators. Die untenstehende Tabelle zeigt wellches Format welche Features unterstützt.

Wenn ihr Projekt keine besonderen Ansprüche hat welche das die Verwendung eines anderen Formats als des ecp Formats nötig machen, so sollten sie immer das Native Format bevorzugen. Das ecp format wird zu jeder Zeit alle Fautures des TcEventKonfiguartors unterstützen.

Format	Aktivierung *	Speichern der Meldungs Eigenschaften **	Speichern von multiplen Quellen IDs ***
ecpx	x	x	x
ecp	x	x	x

Format	Aktivierung *	Speichern der Meldungs Eigenschaften **	Speichern von multiplen Quellen IDs ***
xml	x	x	x
tps	x	x	
xls		x	x
txt			

* Wenn eine Konfiguration aktiviert ist wird der TcEventlogger diese Datei nach Meldungstexten durchsuchen.
** Meldungseigenschaften sind der Meldungs Typ(Warnung/Nachricht/...), Priorität usw.
*** Das Multiple Source ID Feature bedeutet das eine tatsächliche Quelle mit mehreren Quellen IDs verknüpft wird

Plain Text Format Beschreibung

Das PlainText Format kann zur Weitergabe an Übersetzungs Büros verwendet werden. Die Daten werden hier in Menschlich lesbarer Form gespeichert und die Datei kann händich um weitere Sprachen erweitert werden.

Dieses Beispiel zeigt eine kleine Event-Konfiguration mit dem Plain Text Format und den Sprachen Deutsch and Englisch.

```
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'
```

Um eine dritte Sprache zur Konfiguration hinzuzufügen müssen sie eine Zeile kopieren und die Language ID (blau) und den dazugehörigen Text (grün) ändern.

Genaueres wird im folgendem Beispiel beschrieben:

```
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'
```

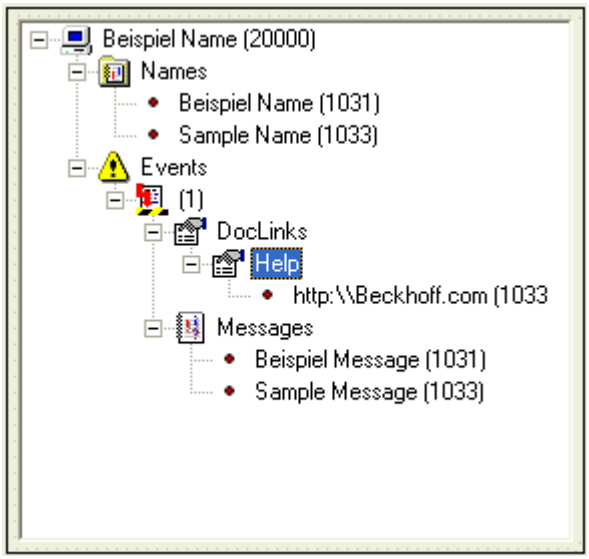
Die folgende Tabelle gibt Informationen über einige language *LCIDs.

LCID	Beschreibung
1031	Deutsch
1033	US Englisch
1034	Spanisch
1036	Französisch

*LCID: mehr Informationen gibt es in der MSDN Library.

6.5 TcEventConfigEditor ActiveX

Der TcEvtCfgEditor ist ein simples TreeView Control das eine Konfiguration anzeigen und bearbeiten kann. Es bietet Basis Funktionen um eine komplette Konfiguration für den TcEventLogger zu erstellen.



Für weitergehende Funktionen werden sie die TcEvtConfiguration COM Interfaces [▶ 26] benutzen müssen, so wie es die EventConfigApp macht.
Um das Verhalten des Editors anzupassen kann das IEvtCfgEditor [▶ 63] Interface benutzt werden.

6.6 EventConfiguration on Windows CE

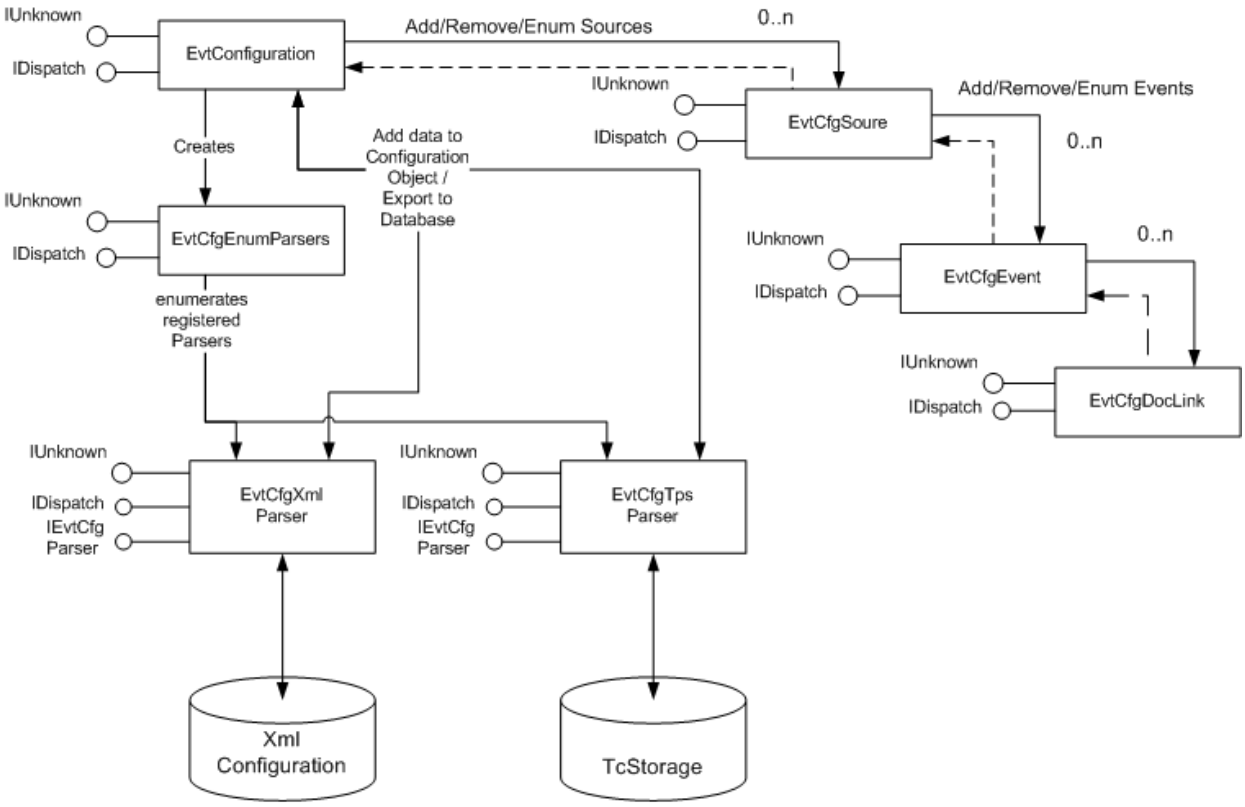
The TcEventConfigurator is not available for Windows CE. You can create event configurations on your Desktop System and manually activate these files on the embedded device.

To manually activate a configuration:

1. Save it in Xml or Ecpx format.
2. Activate the file(s) as described in the TcXmlFormatter section [▶ 123].

6.7 COM

Das TcEvtConfiguration Objekt ist die Kernkomponente des Konfigurators. Es stellt seine funktionalität durch COM mit dem ITcEvtConfig Interface zur Verfügung. Durch die Methoden kann man Konfigurationen erstellen, öffnen und bearbeiten.
Mit der TcEventConfig Library werden einige standard import und export Filter zur Verfügung gestellt. Wenn sie eine eigene Datanbank zum speichern der Informationen wollen müssen sie einfach einen eigenen Parser implementieren. Für genauere Informationen schauen sie sich das ITcEvtCfgParser Interface an.



Benutzen sie diese Interfaces um die Konfiguration zu bearbeiten:

Interface	Beschreibung
IEvtConfiguration [▶ 27]	Kernkomponente des Konfigurators. Von hier werden Dateien geladen, gespeichert und verändert.
IEvtCfgSource [▶ 33]	Diese Komponente repräsentiert eine Source. Damit können Sourcenamen eingestellt werden und zur Source gehörende events angesprochen werden.
IEvtCfgEvent [▶ 42]	Diese Komponente repräsentiert einen Event. Damit können zB Messages hinzugefügt werden. Außerdem kann auf DocLinks die zum Event gehören angesprochen werden.
IEvtCfgDocLink [▶ 51]	Diese Komponente repräsentiert einen DocLink.
IEvtCfgEnumParsers [▶ 56]	Listet alle registrierten Parser auf.
IEvtCfgParser [▶ 56]	Importiert und exportiert Konfigurationen in verschiedenen Formaten.

6.7.1 IEvtConfiguration

Das IEvtConfiguration Interface ist die Kernkomponente des Konfigurators. Durch seine Methoden können Konfigurationen gespeichert, geladen und bearbeitet werden.
Es implementiert ausserdem das _IEvtCfgEvents Interface um die Applikation über änderungen in der Konfiguration zu benachrichten.

Tab. 1: IEvtConfiguration Methoden und Eigenschaften nach Vtable sortiert

IEvtConfiguration Methods	Description
AddSource [▶ 28]	Fügt eine Source zur Konfigutation hinzu
GetSource [▶ 31]	Bekommt eine Source von der Konfiguration
RemoveSource [▶ 32]	Entfernt eine Source aus der Konfiguration

IEvtConfiguration Methods	Description
GetFirstSource [▶ 30]	Startet eine Auflistung aller Sources in der Konfiguration
GetNextSource [▶ 31]	Führt die Auflistung der Sources fort
EnumParsers [▶ 29]	Bekommt ein enum Objekt um die registrierten Parser aufzulisten
SourceCount [▶ 32]	Zählt die Anzahl der Sources in der Konfiguration
ClearConfig [▶ 29]	Setzt die ganze Konfiguration zurück
GetConfigSize [▶ 30]	Bekommt die Anzahl der Sources, Events und DocLinks in der Konfiguration
CopyTo [▶ 29]	Kopiert die Konfiguration zu einer anderen oder zu einer neuen Konfiguration

Tab. 2: _IEvtCfgEvents Methoden und Eigenschaften nach Vtable sortiert

_IEvtCfgEvents Methods	Description
OnNewSource	Eine neue Source wurde hinzugefügt
OnEditSource	Eine Source wurde bearbeitet
OnDeleteSource	Eine Source wurde gelöscht
OnNewEvent	Ein Event wurde hinzugefügt
OnEditEvent	Ein Event wurde bearbeitet
OnDeleteEvent	Ein Event wurde gelöscht
OnNewLink	Ein Link wurde hinzugefügt
OnEditLink	Ein Link wurde bearbeitet
OnDeleteLink	Ein Link wurde gelöscht

6.7.1.1 AddSource

Fügt eine Source zur Konfigutation hinzu.

Sources werden anhand ihrer Id identifiziert, doppelte Ids sind nicht erlaubt.

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

Parameter

pSource

[in] Pointer zu einem neuen Source Objekt

Rückgabe Werte

S_OK

Die Source wurde erfolgreich zur Konfiguration hinzugefügt. Der OnNewSource Event wurde ausgelöst.

E_FAIL

pSource konnte nicht zur Konfiguration hinzugefügt werden. Das kann passieren wenn die Source bereits Child einer anderen Konfiguration ist oder die Id bereits verwendet wird.

E_POINTER

pSource ist kein gültiger Pointer

E_ACCESSDENIED

Das Objekt ist wegen einer asynchronen Operation gesperrt.

6.7.1.2 ClearConfig

Setzt die ganze Konfiguration zurück.

```
HRESULT ClearConfig();
```

Parameter

Rückgabe Werte

S_OK

Die Konfiguration wurde geleert

E_ACCESSDENIED

Das Objekt ist wegen einer asynchronen Operation gesperrt.

6.7.1.3 CopyTo

ITcEvtConfig

Kopiert die Konfiguration zu einer anderen oder zu einer neuen Konfiguration.

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

Parameter

ppConfiguration

[out, retval] Pointer zu einem Pointer zur Zielkonfiguration. Wenn *ppConfiguration NULL ist wird eine neue Konfiguration erstellt.

Rückgabe Werte

S_OK

ppConfiguration enthält eine Kopie der Konfiguration

E_POINTER

pSource ist kein gültiger Pointer

E_ACCESSDENIED

Das Objekt ist wegen einer asynchronen Operation gesperrt.

6.7.1.4 EnumParsers

Bekommt ein IEvtCfgEnumParser Objekt um die registrierten Parser aufzulisten.

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

Parameter

ppEnum

[out, retval] Pointer zu einem Pointer der das zurückgegebene Aufzählungsobjekt speichern soll.

Rückgabe Werte

S_OK

ppEnum enthält einen gültigen pointer zu einem IDispatch Pointer, mit dem sie auf das IEvtCfgEnumParser Interface zugreifen können.

E_ACCESSDENIED

Das Objekt ist wegen einer asynchronen Operation gesperrt.

6.7.1.5 GetConfigSize

Bekommt die Anzahl der Sources, Events und DocLinks in der Konfiguration.

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

Parameter

nSources

[out] Pointer zu einer Variablen die die Anzahl der Sourcen speichern soll. Falls null wird die Anzahl nicht zurückgegeben.

nEvents

[out] Pointer zu einer Variablen die die Anzahl der Events speichern soll. Falls null wird die Anzahl nicht zurückgegeben.

nDocLinks

[out] Pointer zu einer Variablen die die Anzahl der DocLinks speichern soll. Falls null wird die Anzahl nicht zurückgegeben.

Rückgabe Werte

S_OK

Die Funktion wurde erfolgreich ausgeführt

E_ACCESSDENIED

Das Objekt ist wegen einer asynchronen Operation gesperrt.

6.7.1.6 GetFirstSource

Bekommt eine Source von der Konfiguration.
Diese Methode resettet den internen Iterator und ruft GetNextSource [▶ 31] auf.
Um die Aufzählung fortzusetzen rufen sie GetNextSource [▶ 31] auf.

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

Parameter

[out, retval] Pointer zu einem Pointer der die zurückgegebene Source speichert.

Rückgabe Werte

S_OK

ppSource enthält einen gültigen Pointer zu einem IEvtCfgSource Objekt.

S_FALSE

Die Konfiguration hat keine Sourcen. *ppSource wurde auf NULL gesetzt.

E_POINTER

ppSource ist kein gültiger Pointer

E_ACCESSDENIED

Das Objekt ist wegen einer asynchronen Operation gesperrt.

6.7.1.7 GetNextSource

Bekommt die nächste Source der Konfiguration im bezug auf den internen Iterator.
Bevor sie eine Auflistung starten rufen sie GetFirstSource [▶ 30] auf.
Nachdem Sourcen hinzugefügt oder entfernt wurden müssen sie die Aufzählung mit GetFirstSource [▶ 30] neu starten.

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

Parameter

[out, retval] Pointer zu einem Pointer der die zurückgegebene Source speichert.

Rückgabe Werte

S_OK

ppSource enthält einen gültigen Pointer zu einem IEvtCfgSource Objekt.

S_FALSE

Die Konfiguration hat keine weiteren Sourcen. *ppSource wurde auf NULL gesetzt.

E_POINTER

ppSource ist kein gültiger Pointer

E_ACCESSDENIED

Das Objekt ist wegen einer asynchronen Operation gesperrt.

6.7.1.8 GetSource

Bekommt eine Source von der Konfiguration
Die Source wird anhand ihrer Id identifiziert. Wenn die Id unbekannt ist können sie die Sourcen mit GetFistSource [▶ 30] und GetNextSource [▶ 31] auflusten.

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

Parameter

nId

[in] Id der Source

ppSource

[out, retval] Pointer zu einem Pointer der die Source speichern soll.

Rückgabe Werte

S_OK

ppSource enthält einen gültigen Pointer zu einem IEvtCfgSource Objekt.

E_INVALIDARG

Es existiert keine Source mit dieser Id in der Konfiguration.

E_ACCESSDENIED

Das Objekt ist wegen einer asynchronen Operation gesperrt.

6.7.1.9 RemoveSource

Entfernt eine Source aus der Konfiguration.
Die Source wird anhand ihrer Id identifiziert.

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

Parameter

nId

[in] Id der Source die entfernt werden soll

Rückgabe Werte

S_OK

Die angegebene Source wurde aus der Konfiguration entfernt.

E_INVALIDARG

Es existiert keine Source mit dieser Id in der Konfiguration.

E_ACCESSDENIED

Das Objekt ist wegen einer asynchronen Operation gesperrt.

6.7.1.10 SourceCount

Zählt die Anzahl der Sources in der Konfiguration.

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

Parameter

pVal

[out, retval] Pointer zu einer Variable die den Rückgabewert speichern soll.

Rückgabe Werte

S_OK

ppSource enthält ein gültiges IEvtCfgSource Objekt.

E_POINTER

pVal ist kein gültiger Pointer

E_ACCESSDENIED

Das Objekt ist wegen einer asynchronen Operation gesperrt.

6.7.2 IEvtCfgSource

Das ITcEvtCfgSource Interface bietet Zugriff auf Objekte, die eine Source in der Konfiguration repräsentieren. Jede Source wird an ihre eindeutigen Id identifiziert und kann Namen in mehreren Sprachen haben. Wenn Sources Teil einer Konfiguration sind haben sie genau einen Parent vom Typ IEvtConfiguration [▶ 27]. Eine Source kann nicht Child von mehreren Konfigurationen sein. Eine source kann mehrere Childs vom Typ ITcEvtCfgEvent [▶ 42] haben.

Tab. 3: Methoden und Eigenschaften nach Vtable sortiert

ITcEvtConfig Methods	Description
Name [▶ 40]	Setzt/Bekommt den Sourcenamen in der angegebenen Sprache
AddEvent [▶ 33]	Fügt einen Event zur Source hinzu
GetEvent [▶ 36]	Bekommt den Event mit der angegebenen Id
RemoveEvent [▶ 41]	Entfernt einen Event
Id [▶ 39]	Setzt oder bekommt die Source Id
GetFirstName [▶ 37]	Startet eine Aufzählung der Sourcenamen
GetNextName [▶ 38]	Setzt die Aufzählung der Sourcenamen fort
GetFirstEvent [▶ 36]	Startet eine Aufzählung der Events
GetNextEvent [▶ 37]	Setzt die Aufzählung der Events fort
GetDefaultName [▶ 35]	Holt den standard Sourcenamen im Bezug auf die Systemsprache
RemoveName [▶ 42]	Entfernt den Sourcename in der angegebenen Sprache
Parent [▶ 41]	Setzt oder bekommt den Parent der Source
IsParentOnlyReference [▶ 39]	Prüft, ob die Source die einzige Referenz ist
EventCount [▶ 34]	Zählt die Anzahl der Events die Children der aktuellen Source sind
CopyTo [▶ 34]	Kopiert die Source zu einem anderen EvtCfgSource Objekt

6.7.2.1 AddEvent

ITcEvtCfgSource [▶ 33]

Fügt einen Event zur Source hinzu.
Events werden anhand ihrer Id identifiziert - doppelte Ids sind nicht möglich. Events dürfen auch nur einen Parent haben, wenn der Event bereits Child einer anderen Source ist wird die Funktion fehlschlagen.

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

Parameter

pEvent

[in] Pointer zum Event Objekt das hinzugefügt werden soll

Rückgabe Werte

S_OK

Der neue event wurde erfolgreich hinzugefügt. Der OnNewEvent Event wurde ausgelöst.

E_FAIL

pEvent konnte nicht zur Konfiguration hinzugefügt werden. Das kann passieren, weil die Source bereits einen Event mit dieser Id hat oder der Event zu einer anderen Source gehört.

E_POINTER

pEvent ist kein gültiger Pointer

E_ACCESSDENIED

Das Objekt ist wegen einer asynchronen Operation gesperrt.

6.7.2.2 CopyTo

ITcEvtCfgSource [▶ 33]

Kopiert die Source zu einem anderen EvtCfgSource Objekt.

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

Parameter

ppSource

[out, retval] Pointer zu einem Pointer zur Zielsource. Wenn *ppSource NULL ist wird eine neue Source erstellt.

Rückgabe Werte

S_OK

ppSource enthält eine Kopie der Source

E_POINTER

ppSource ist kein gültiger Pointer

E_ACCESSDENIED

Das Objekt ist wegen einer asynchronen Operation gesperrt.

6.7.2.3 EventCount

ITcEvtCfgSource [▶ 33]

Zählt die Anzahl der Events die Children der aktuellen Source sind.

Property get

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

Parameter

pVal

[out, retval] Pointer zu einer long Variable die die Anzahl der Children bekommt.

Rückgabe Werte

S_OK

Funktion erfolgreich ausgeführt

E_POINTER

pVal ist kein gültiger Pointer

E_ACCESSDENIED

Das Objekt ist wegen einer asynchronen Operation gesperrt.

6.7.2.4 GetDefaultName

ITcEvtCfgSource [▶ 33]

Holt den Standard-Sourcenamen.

Es wird in folgender Reihenfolge geprüft und der erste 'Treffer' wird übergeben.

1. Language des Systems
2. Englisch (1031 / 9)
3. Deutsch (1033 / 7)
4. Niedringste lcid

Property get

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

Parameter

name

[out, retval] Pointer zu einer BSTR Variable die den Sourcenamen bekommen soll

Rückgabe Werte

S_OK

Funktion wurde erfolgreich ausgeführt

S_FALSE

Es ist kein Name für die Source vorhanden, name ist NULL

E_POINTER

name ist kein gültiger Pointer

E_ACCESSDENIED

Das Objekt ist wegen einer asynchronen Operation gesperrt.

6.7.2.5 GetEvent

Bekommt den Event mit der angegebenen Id.
Wenn die event Id unbekannt ist können sie alle Events der Source mit GetFistEvent und GetNextEvent auflisten.

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

Parameter

nId

[in] Id des Events

ppSource

[out, retval] Pointer zu einem Pointer der das zurückgegebene Objekt speichern soll

Rückgabe Werte

S_OK

ppSource enthält einen gütigen Pointer auf ein IEvtCfgSource Objekt.

E_INVALIDARG

Es existiert kein Event mit der angegebenen Id in der Source-

E_POINTER

ppEvent ist kein gültiger Pointer

E_ACCESSDENIED

Das Objekt ist wegen einer asynchronen Operation gesperrt.

6.7.2.6 GetFirstEvent

Bekommt den ersten Event der Source.
Diese Funktion resetet den internen Iterator und ruft GetNextEvent [▶ 37] auf.
Um die Aufzählung fortzusetzen rufen sie GetNextEvent [▶ 37] auf.
Wenn die Source von einer asynchronen Operation gesperrt ist, wird nicht E_FAIL zurückgegeben sondern gewartet bis die sperrung aufgehoben ist.

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

Parameter

ppEvent

[out, retval] Pointer zu einem IEvtCfgEvent interface Pointer in dem das erste Child gespeichert wird.

Rückgabe Werte

S_OK

Funktion erfolgreich ausgeführt.

S_FALSE

Die Source hat keine Events. *ppEvent ist NULL.

E_POINTER

ppEvent ist kein gültiger Pointer

E_ACCESSDENIED

Das Objekt ist wegen einer asynchronen Operation gesperrt.

6.7.2.7 GetFirstName

ITcEvtCfgSource

Bekommt den ersten Namen der Source.

Diese Funktion resettet den internen Iterator und ruft GetNextName [▶ 38] auf.
Um die Aufzählung fortzusetzen rufen sie GetNextName [▶ 38] auf.

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

Parameter

name

[out] Pointer zu einer BSTR Variable die den Namen bekommen soll.

pLangId

[out, retval] Pointer zu einer long Variable die die lang id bekommen soll

Rückgabe Werte

S_OK

name enthält den ersten Namen der Source

S_FALSE

Die Source hat keine Namen. *name ist NULL.

E_POINTER

name ist kein gültiger Pointer

E_ACCESSDENIED

Das Objekt ist wegen einer asynchronen Operation gesperrt.

6.7.2.8 GetFirstName

Bekommt das nächste Child ausgehend vom internen Iterator.

Vor dem start einer Aufzählung müssen sie GetFirstEvent [▶ 36] aufrufen.
Nachdem Events hinzugefügt oder entfernt wurden müssen sie den internen Iterator resetten indem sie GetFirstEvent [▶ 36] aufrufen.
Wenn die Source von einer asynchronen Operation gesperrt ist, wird nicht E_FAIL zurückgegeben sondern gewartet bis die sperrung aufgehoben ist.

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

Parameter

ppEvent

[out, retval] Pointer zu einem IEvtCfgEvent interface Pointer in dem das erste Child gespeichert wird.

Rückgabe Werte

S_OK

Funktion erfolgreich ausgeführt

S_FALSE

Die Source hat keine Events. *ppEvent ist NULL.

E_POINTER

ppEvent ist kein gültiger Pointer

E_ACCESSDENIED

Das Objekt ist wegen einer asynchronen Operation gesperrt.

6.7.2.9 GetNextName

Bekommt den nächsten Namen ausgehend vom internen Iterator.

Vor dem Start einer Aufzählung müssen sie GetFirstName [▶ 37] aufrufen.
Nachdem Events hinzugefügt oder entfernt wurden müssen sie den internen Iterator resetten indem sie GetFirstName [▶ 37] aufrufen.

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

Parameter

name

[out] Pointer zu der BSTR Variable die den Sourcenamen bekommen soll

pLangId

[out, retval] Pointer zu der long Variable die die lang id bekommen soll

Rückgabe Werte

S_OK

Funktion erfolgreich ausgeführt

S_FALSE

Die Source hat keine Namen. *name ist NULL.

E_POINTER

name ist kein gültiger Pointer

E_ACCESSDENIED

Das Objekt ist wegen einer asynchronen Operation gesperrt.

6.7.2.10 Id

Setzt oder bekommt die Source Id.

Wenn die Source bereits Child einer Konfiguration ist, wird sie entfernt und mit der neuen Id hinzugefügt. Der SourceRemoved und der NewSource Event werden ausgelöst.

Property get

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

Parameter

pVal

[out, retval] pointer zu einer long Variable die die Source Id bekommen soll

Rückgabe Werte

S_OK

Funktion wurde erfolgreich ausgeführt

E_POINTER

pVa ist kein gültiger Pointer

Property set

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

Parameter

newVal

[in] long Variable mit der neuen Source Id

Rückgabe Werte

S_OK

Funktion wurde erfolgreich ausgeführt

E_FAIL
Die Source ist Child einer Konfiguration, die bereits eine andere Source mit der angegebenen Id hat.
Die Id bleibt unverändert.

E_ACCESSDENIED

Das Objekt ist wegen einer asynchronen Operation gesperrt.

6.7.2.11 IsParentOnlyReference

Prüft ob die Source die einzige Referenz ist.

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

Parameter

pVal

[out, retval] Pointer zu einer BOOL Variable die das Ergebnis speichern soll.

Rückgabe Werte

S_OK

Erfolg.

S_FALSE

Die Source hat keinen Parent. *pVal ist False.

E_POINTER

pVal ist kein gültiger Pointer

6.7.2.12 Name

Setzt/Bekommt den Sourcenamen in der angegebenen Sprache

Property get

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

Parameter

langId

[in] long Variable die die lang id für den gewünschten String enthält.

pVal

[out, retval] Pointer zu einer BSTR Variable die den Sourcenamen bekommen soll.

Rückgabe Werte

S_OK

Funktion erfolgreich ausgeführt.

E_FAIL

Es existiert kein Sourcename für die angegebene Sprache

E_POINTER

pVal ist kein gültiger Pointer

E_ACCESSDENIED

Das Objekt ist wegen einer asynchronen Operation gesperrt.

Property set

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

Parameter

langId

[in] long Value mit der lang id die der Sourcename bekommen soll.

pVal

[in] BSTR Variable, die den Namen der hinzugefügt werden soll enthält.

Rückgabe Werte

S_OK

Funktion erfolgreich ausgeführt.

E_FAIL

A name for the language Id does already exist

E_ACCESSDENIED

Das Objekt ist wegen einer asynchronen Operation gesperrt.

6.7.2.13 Parent

Setzt oder bekommt den Parent der Source.

Den Parent zu setzen ist nur für internen Gebrauch. Wenn man den Parent auf ungültige Werte setzt, kann das zu Nebeneffekten führen. Wenn sie den Parent ändern wollen benutzen sie IEvtConfiguration::AddSource [▶ 28] oder IEvtConfiguration::RemoveSource [▶ 32].

Property get

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

Parameter

ppVal

[out, retval] Pointer zu einer IDispatch* Variable die den Parent der Source bekommt

Rückgabe Werte

S_OK

Funktion erfolgreich, wenn die source einen Parent hat ist *ppVal gültig, sonst NULL

E_POINTER

ppVal ist kein gültiger Pointer

E_ACCESSDENIED

Das Objekt ist wegen einer asynchronen Operation gesperrt.

6.7.2.14 RemoveEvent

ITcEvtCfgSource [▶ 33]

Entfernt einen Event.

Der zu entfernende Event wird anhand seiner Id Identifiziert.

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

Parameter

nId

[in] Id des evetns der entfernt werden soll.

Rückgabe Werte

S_OK

Der angegebene Event wurde aus der Konfiguration gelöscht.

E_INVALIDARG

Es existiert kein Event mit der angegebenen Id in der Source

E_ACCESSDENIED

Das Objekt ist wegen einer asynchronen Operation gesperrt.

6.7.2.15 RemoveName

ITcEvtCfgSource [▶ 33]

Entfernt den Sourcename in der angegebenen Sprache.

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

Parameter

nId

[in] Lang Id des Namens der gelsöcht werden soll.

Rückgabe Werte

S_OK

Der angegebene Name wurde entfernt

E_INVALIDARG

Der Name mit der angegebenen Lang Id existiert nicht.

E_ACCESSDENIED

Das Objekt ist wegen einer asynchronen Operation gesperrt.

6.7.3 IEvtCfgEvent

Das IEvtCfgEvent Interface repräsentiert einen Event der Konfiguration. Durch seine Methoden können Events erstellt und verändert werden.

Tab. 4: Methoden und eigenschaften nach Vtable sortiert

IEvtCfgEvent Methods	Description
Id [▶ 47]	Setzt oder holt die Event Id
Class [▶ 43]	Setzt oder holt die Event Klasse
Priority [▶ 50]	Setzt oder holt die Event Priority
MustConfirm [▶ 49]	Prüft ob der Event Bestätigt werden muss

IEvtCfgEvent Methods	Description
MsgString [▶ 49]	Setzt oder holt den Message String in einer bestimmten Sprache
AddDocLink [▶ 43]	Fügt einen DocLink zum Event hinzu
RemoveDocLink [▶ 51]	Entfernt einen DocLink vom Event
GetDocLink [▶ 45]	Holt einen DocLink
GetFirstMessage [▶ 46]	Startet die Auflistung der Message Strings
GetNextMessage [▶ 46]	Führt die Auflistung der Message Strings fort
GetFirstFormatString [▶ 46]	Startet die Auflistung der Format Strings
GetNextFormatString [▶ 46]	Führt die Auflistung der format Strings fort
GetFirstDocLink [▶ 46]	Startet die Auflistung der DocLinks
GetNextDocLink [▶ 46]	Führt die Auflistung der DocLinks fort
Parent [▶ 50]	Setzt oder holt den Parent des Events
IsParentOnlyReference [▶ 48]	Prüft ob der Event die einzige Referenz ist
DocLinkCount [▶ 45]	Zählt die DocLinks vom Event
CopyTo [▶ 44]	Erstellt eine Kopie des Events

6.7.3.1 AddDocLink

Fügt einen DocLink zum Event hinzu.

DocLinks werden durch ihren Namen identifiziert. DocLinks können nur einen Parent haben. Wenn ein DocLink bereits Child eines anderen Events ist wird die Methode fehlschlagen.

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

Parameter

pLink

[in] Pointer zu dem neuen DocLink Objekt das hinzugefügt werden soll.

Rückgabe Werte

S_OK

Der neue DocLink wurde erfolgreich hinzugefügt

E_FAIL

pLink konnte nicht zum Event hinzugefügt werden. Das kann passieren, wenn der Link bereits Child eines anderen Events ist.

E_POINTER

pLink ist kein gültiger Pointer

E_ACCESSDENIED
Das Objekt ist wegen einer asynchronen Operation gesperrt.

6.7.3.2 Class

Setzt oder holt die Event Klasse.

Property get

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

Parameter

pVal

[out, retval] Pointer zu einer long Variable die die Event Klasse bekommt.

Rückgabe Werte

S_OK

Funktion wurde erfolgreich ausgeführt.

E_POINTER

pVal ist kein gültiger Pointer

Property set

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

Parameter

newVal

[in] long Variable mit der neuen Event Klasse.

Rückgabe Werte

S_OK

Die Event Klasse wurde erfolgreich geändert.

6.7.3.3 CopyTo

Erstellt eine Kopie des Events.

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

Parameter

ppEvent

[out, retval] Pointer zu einem Pointer zum Zielevent. Wenn *ppEvent NULL ist wird ein neuer Event erstellt.

Rückgabe Werte

S_OK

ppEvent erhält eine exakte Kopie des Events

E_POINTER

ppEvent ist kein gültiger Pointer

E_ACCESSDENIED

Das Objekt ist wegen einer asynchronen Operation gesperrt.

6.7.3.4 DocLinkCount

ITcEvtCfgEvent

Zählt die DocLinks vom Event.

Property get

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

Parameter

pVal

[out, retval] Pointer zu einer long Variable die die Anzahl der DocLinks bekommt

Rückgabe Werte

S_OK

Funktion wurde erfolgreich ausgeführt.

E_POINTER

pVal ist kein gültiger Pointer

6.7.3.5 GetDocLink

Holt einen DocLink.

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

Parameter

name

[in] Name des Links der geholt werden soll.

ppLink

[out, retval] Pointer zu einem Pointer der das zurückgegebene DocLink Objekt.

Rückgabe Werte

S_OK

ppLink enthält einen gültigen pointer zu einem IEvtCfgSource Objekt.

E_INVALIDARG

Es existiert kein DocLink mit dem angegebenen Namen im Event.

E_POINTER

ppLink ist kein gültiger Pointer

6.7.3.6 GetFirstMessage

Startet die Auflistung der Message Strings.

Diese Funktion resettet den internen Iterator und ruft GetNextMessage [▶ 46] auf.
Um die Aufzählung fortzusetzen rufen sie GetNextMessagee [▶ 46] auf.

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

Parameter

message

[out] Pointer zu einer BSTR Variable die die erste Message des Events bekommen soll.

pLangId

[out, retval] Pointer zu einer long Variable die die lang Id der Message speichert.

Rückgabe Werte

S_OK

message enthält die erste Message des Events.

S_FALSE

Die Source hat keine Messages. *message ist NULL.

E_POINTER

message ist kein gültiger Pointer

E_ACCESSDENIED

Das Objekt ist wegen einer asynchronen Operation gesperrt.

6.7.3.7 GetNextMessage

Führt die Auflistung der Message Strings fort.
Bevor sie die Auflustung starten, rufen sie GetFirstMessage [▶ 46] auf.
Nachdem Messages hinzugefügt oder entfernt wurden muss die Aufzählung mit GetFirstMessage [▶ 46] neu gestartet werden.

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

Parameter

message

[out] Pointer zu einer BSTR Variable die die nächste Message des Events bekommen soll.

langId

[out, retval] Pointer zu einer long Variable die die lang Id der Message speichert.

Rückgabe Werte

S_OK

message enthält die nächste Message des Events.

S_FALSE

Die Source hat keine Messages. *message ist NULL.

E_POINTER

message ist kein gültiger Pointer

E_ACCESSDENIED

Das Objekt ist wegen einer asynchronen Operation gesperrt.

6.7.3.8 Id

Setzt oder holt die Event Id.

Wenn der Event bereits das Child einer Source ist, wird er entfernt und neu hinzugefügt. die EventRemoved und NewEvent events werden ausgelöst.

Property get

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

Parameter

pVal

[out, retval] Pointer zu einer long Variable die die Event Id bekommt

Rückgabe Werte

S_OK

Funktion wurde erfolgreich ausgeführt.

E_POINTER

pVal ist kein gültiger Pointer

Property set

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

Parameter

newVal

[in] long Variable mit der neuen Event Id

Rückgabe Werte

S_OK

The Event Id wurde erfolgreich auf newVal gesetzt

E_FAIL

Der Event ist das child eines Source Objekts welches bereits einen anderen Event mit dieser id enthält.
Die id wurde nicht verändert.

E_ACCESSDENIED

Das Objekt ist wegen einer asynchronen Operation gesperrt.

6.7.3.9 IsParentOnlyReference

Prüft ob der Event die einzige Referenz ist.

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

Parameter

pVal

[out, retval] Pointer auf eine BOOL Variable die das Ergebnis speichert.

Rückgabe Werte

S_OK

Erfolg.

S_FALSE

Der Event hat keinen Parent. *pVal ist False.

E_POINTER

pVal ist kein gültiger Pointer

6.7.3.10 MsgString

Setzt oder holt den Message String in einer bestimmten Sprache.

Property get

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

Parameter

langId

[in] Die language id die Message die zurückgegeben werden soll.

pVal

[out, retval] Pointer zu einer BSTR Variable die den String bekommt.

Rückgabe Werte

S_OK

Funktion wurde erfolgreich ausgeführt.

E_POINTER

pVal ist kein gültiger Pointer

E_FAIL

Es existiert keine Message für die angegebene Sprache

Property set

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

Parameter

langId

[in] Die language id für den message String der gesetzt werden soll.

newVal

[in] BSTR Variable die den String der geschrieben werden soll enthält.

Rückgabe Werte

S_OK

Funktion wurde erfolgreich ausgeführt.

E_FAIL

Ein Message String für die angegebene Sprache existiert bereits

6.7.3.11 MustConfirm

Prüft, ob der Event bestätigt werden muss.

Property get

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

Parameter

pVal

[out, retval] Pointer zu einer BOOL Variable die den aktuellen Status bekommt

Rückgabe Werte

S_OK

Funktion wurde Erfolgreich ausgeführt

E_POINTER

pVal ist kein gültiger Pointer

Property set

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

Parameter

newVal

[in] BOOL Variable mit dem neuen Wert

Rückgabe Werte

S_OK

Funktion wurde erfolgreich ausgeführt

6.7.3.12 Parent

Setzt oder holt den Parent des Events.

Den Parent setzen ist nur für den internen Gebrauch. Den Wert auf unpassende Werte setzen kann zu Nebeneffekten führen.

Um den Parent eines Events zu setzen benutzen Sie die Funktionen IEvtCfgSource::AddEvent [▶ 33] oder IEvtCfgSource::RemoveEvent. [▶ 41]

Property get

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

Parameter

ppVal

[out, retval] Pointer zu einer IDispatch* Variable die den Parent des Events bekommt.

Rückgabe Werte

S_OK

Funktion wurde erfolgreich ausgeführt. Wenn der Event einen Paerent hat ist *ppVal gültig, sonst NULL

E_POINTER

ppVal war kein gültiger Pointer

E_ACCESSDENIED
Das Objekt ist wegen einer asynchronen Operation gesperrt.

6.7.3.13 Priority

Setzt oder holt die Event Priority.

Property get

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

Parameter

pVal

[out, retval] Pointer zu einer long Variable die die Event Priority bekommt.

Rückgabe Werte

S_OK

Funktion wurde erfolgreich ausgeführt

E_POINTER

pVal ist kein gültiger Pointer

Property set

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

Parameter

newVal

[in] long Variable mit der neuen Event Priority

Rückgabe Werte

S_OK

Funktion wurde erfolgreich ausgeführt

6.7.3.14 RemoveDocLink

Entfernt einen DocLink vom Event.

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

Parameter

pLink

[in] the DocLink der entfernt werden soll.

Rückgabe Werte

S_OK

Der angegebene Link wurde vom Event entfernt.

E_INVALIDARG

Der link ist kein child des Events

E_POINTER

pLink ist kein gültiger Pointer

E_ACCESSDENIED

Das Objekt ist wegen einer asynchronen Operation gesperrt.

6.7.4 IEvtCfgDocLink

Das IEvtCfgDocLink Interface ermöglicht Zugriff auf die DocLink Eigenschaften.
DocLinks können benutzt werden um zusätzliche Informationen zum Event anzugeben. Ein DocLink besteht aus einen einzugartigen Namen und mehreren Links.

Tab. 5: Methoden und Eigenschaften nach Vtable sortiert

IEvtCfgDocLink Methods	Description
Name [▶ 55]	Setzt oder bekommt den Namen des DocLinks
Link [▶ 54]	Setzt oder bekommt einen Link in der angegebenen Sprache
GetFirstLink [▶ 52]	Startet Auflistung der Links
GetNextLink [▶ 53]	Führt die Auflistung fort

IEvtCfgDocLink Methods	Description
RemoveLink [▶ 56]	Entfernt den Link in der angegebenen Sprache
Parent [▶ 55]	Setzt oder Bekommt den Parent des DocLinks
IsParentOnlyReference [▶ 53]	Prüft ob der DocLink die einzige Referenz ist
CopyTo [▶ 52]	Erstellt eine neue Kopie des DocLinks

6.7.4.1 CopyTo

Erstellt eine Kopie des DocLink Objekts.

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

Parameter

ppLink

[out, retval] Pointer zu einem Pointer zum Ziellink. Wenn *ppLink NULL ist wird ein neue DocLink erstellt.

Rückgabe Werte

S_OK

ppLink enthät eine genaue Kopie des DocLinks

E_POINTER

ppLink war ein ungültiger Pointer

6.7.4.2 GetFirstLink

Gibt den ersten Link des Objekts zurück.

Diese Funktion setzt den internen Iterator zurück und rugt GetNextLink [▶ 53] auf.
Um die Aufzählung fortzusetzen rufen sie GetNextLink [▶ 53] auf.

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

Parameter

name

[out] Pointer zu einer BSTR Variable die die erste URL vom aktuellen Event enthält.

pLangId

[out, retval] Pointer zu einer long Variable die die lang Id URL enthält.

Rückgabe Werte

S_OK

name enthält die erste URL des DocLinks.

S_FALSE

Der DocLink hat keine URLs. *name wurde auf NULL gesetzt.

E_POINTER

Name war kein gültiger Pointer.

6.7.4.3 GetNextLink

Holt dem nächsten Link ausgehend vom internen Iterator.

Bevor sie eine Auflistung der Links starten, rufen sie GetFirstLink [▶ 52] auf.
Nachdem sie Links hinzugefügt oder entfernt haben müssen sie die aufzählung neu starten indem sie GetFirstLink [▶ 52] aufrufen.

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

Parameter

link

[out] Pointer zu einer BSTR Variable die den nächsten Link speichert.

langId

[out, retval] Pointer zu einer long Variable die die lang Id des Links speichert.

Rückgabe Werte

S_OK

link enthält den nächsten Link.

S_FALSE

Der DocLink hat keine weiteren Links. *link wurde auf NULL gesetzt.

E_POINTER

link war ein ungültiger Pointer.

6.7.4.4 IsParentOnlyReference

Prüft, ob der DocLink die einzige Referenz ist.

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

Parameter

pVal

[out, retval] Pointer zu einer BOOL Variable.

Rückgabe Werte

S_OK

Erfolg.

S_FALSE

Der DocLink hat keinen Parent. *pVal ist False.

E_POINTER

pVal ist kein gültiger Pointer

6.7.4.5 Link

Setzt oder bekommt den link für eine bestimmte Sprache.

Property get

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

Parameter

langId

[in] Dielanguage id für den Link

pVal

[out, retval] Pointer zu einer BSTR Variable in der der Link gespeichert wird.

Rückgabe Werte

S_OK

Funktion wurde erfolgreich ausgeführt

E_POINTER

pVal war kein gültiger Pointer

E_FAIL

für die angegebene language id existiert kein Link

Property set

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

Parameter

langId

[in] Die language id für den Link der gespeichert werden soll

newVal

[in] BSTR Variable in der der neue Link gespeichert ist.

Rückgabe Werte

S_OK

Funktion wurde erfolgreich ausgeführt

E_FAIL

Ein Link für die angegebenen language id existiert bereits

6.7.4.6 Name

Setzt oder bekommt den Namen des DocLinks.

Property get

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

Parameter

pVal

[out, retval] Pointer zu einer BSTR Variable in der der Name gespeichert wird.

Rückgabe Werte

S_OK

Funktion wurde erfolgreich ausgeführt

E_POINTER

pVal war kein gültiger Pointer

Property set

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

Parameter

pVal

[in] BSTR Variable die den neuen Namen enthält

Rückgabe Werte

S_OK

Funktion wurde erfolgreich ausgeführt

6.7.4.7 Parent

Setzt oder bekommt den Parent des DocLinks.

Den Parent setzen ist nur für den internen Gebrauch. Den Wert auf unpassende Werte setzen kann zu Nebeneffekten führen.
Um den Parent eines Events zu setzen benutzen Sie die Funktionen IEvtCfgDocLink::AddDocLink und IEvtCfgIEvtCfgDocLink::RemoveDocLink.

Property get

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

Parameter

ppVal

[out, retval] Pointer zu einer IDispatch* Variable die den Parent des DocLinks bekommt. The Parent ist immer vom Typ IEvtCfgEvent [▶ 42]

Rückgabe Werte

S_OK

Funktion wurde erfolgreich ausgeführt. Wenn der DocLink einen Parent hat ist *ppVal gültig, sonst NULL

E_POINTER

ppVal war kein gültiger Pointer

E_ACCESSDENIED
Das Objekt ist wegen einer asynchronen Operation gesperrt.

Sehen Sie dazu auch

- AddDocLink [▶ 43]
- RemoveDocLink [▶ 51]

6.7.4.8 RemoveLink

Entfernt den Link in der angegebenen Sprache

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

Parameter

langId

[in] Die langId vom Link der entfernt werden soll

Rückgabe Werte

S_OK

Der angegebene Link wurde entfernt.

E_INVALIDARG

Es existiert kein Link mit der angegebenen langid.

6.7.5 IEvtCfgEnumParser

Das IEvtCfgEnumParser Interface wird benutzt um alle registrierten Parser aufzulisten.
EvtCfgParser Objekte werden durch die IEvtConfiguration::EnumParsers [▶ 29] Funktion erstellt.

Tab. 6: Methoden und Eigenschaften nach Vtable sortiert

IEvtCfgEnumParser Methods	Description
Next	Listet die nächsten X Parser Objekte auf
Reset	Resettet den Zähler
Skip	Überspringe die nächsten X Parser Objekte
VbNext	VB und .NET kompatible Next() Funktion

6.7.6 IEvtCfgParser

Die TcEventConfig library liefert einige standard Parser, zB den Xml und den Tps Parser.
Jeder dieser Parser implementiert das IEvtCfgParser Interface.
Mit diesen System können auch eigene Parser hinzugefügt werden. Dazu sind nur die Implementierung des

Interfaces und eine Registrierung nötig. (Beispiel)
Die Folgende Interface Beschreibung zeigt wie das EvtCfgParser Interface definiert ist und wie jede Implementation auf Funktionsaufrufe reagieren muss.

Tab. 7: IEvtCfgParser Methoden und Eigenschaften nach Vtable sortiert

IEvtCfgParser Methods	Description
OpenConfiguration [▶ 61]	Öffnet eine existierende Konfiguration
Import [▶ 61]	Importiert von einer geöffneten Datei
Export [▶ 60]	Exportiert eine geöffnete Datei
CloseConfiguration [▶ 58]	Schliesst eine Konfigurationsdatei
DefFileExt [▶ 59]	Bekommt die standard Dateiendung für das implementierte import Format
DefFileType [▶ 59]	Bekommt eine kurze Beschreibung des File Typs für das implementierte Format
CreateConfiguration [▶ 58]	Erstellt und öffnet eine neue Konfigurationsdatei
ActivateConfiguration [▶ 57]	Aktiviert die aktuelle Konfiguration
GetConfigSize [▶ 60]	Bekommt die grösser der aktuellen Konfiguration
AsyncImport [▶ 58]	Importiert eine Konfiguration asynchron
AsyncExport [▶ 58]	Exportiert eine Konfiguration asynchron

Tab. 8: _IEvtCfgParserEvents Methoden und Eigenschaften nach Vtable sortiert

_IEvtCfgParserEvents Methods	Description
SourceLoaded	Source wurde geladen
SourceSaved	Source wurde gespeichert
EventLoaded	Event wurde geladen
EventSaved	Event wurde gespeichert
LinkLoaded	Link wurde geladen
LinkSaved	Link wurde gespeichert
LoadComplete	Laden abgeschlossen
SaveComplete	Speichern abgeschlossen

Tab. 9: Parser-Implementationen

Parser	Description
EvtCfgTpsParser [▶ 62]	Nach TwinCAT Stroage exportieren/importieren
EvtCfgXMLParser [▶ 62]	Nach XML exportieren/importieren

6.7.6.1 ActivateConfiguration

Aktiviert die aktuelle Konfigutration.

Dieses Verhalten ist Parser spezifisch.

```
HRESULT ActivateConfiguration();
```

Parameter

Rückgabe Werte

S_OK

Die Konfiguration wurde aktiviert.

E_ACCESSDENIED

Der Parser ist wegen einer asynchronen Operation gesperrt.

E_FAIL

Die Konfiguration konnte nicht aktiviert werden.

6.7.6.2 AsyncExport

Asynchrone Alternative zu Export [▶ 60]().

Die Funktion wird sofort beended und sie müssen auf IEvtCfgParser Events achten um den Fortschritt zu prüfen.

Sehen Sie dazu auch

IEvtCfgParser [▶ 56]

6.7.6.3 AsyncImport

Asynchrone Alternative zu Import [▶ 61]().

Die Funktion wird sofort beendet und sie müssen auf IEvtCfgParser [▶ 56] Events achten um den Vortschritt zu prüfen.

6.7.6.4 CloseConfiguration

Schließt eine Konfigurationsdatei

```
HRESULT CloseConfiguration();
```

Parameter

Rückgabe Werte

S_OK

Die Konfiguration wurde geschlossen.

E_ACCESSDENIED

Der Parser ist wegen einer asynchronen Operation gesperrt.

6.7.6.5 CreateConfiguration

Erstellt und öffnet eine neue Konfigurationsdatei

Die angegebene Datei muss nicht existieren.

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

Parameter

filename

[in] Pfad und Dateiname

Rückgabe Werte

S_OK

Die Konfiguration wurde erstellt und geöffnet

E_ACCESSDENIED

Der Parser ist wegen einer asynchronen Operation gesperrt.

E_FAIL

Die Konfiguration konnte nicht erstellt werden.

6.7.6.6 DefFileExt

Bekommt die Standard Dateiendung für das implementierte Import Format

Property get

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

Parameter

pVal

[out, retval] Pointer zu einer BSTR Variable die die Dateiendung bekommt.

Rückgabe Werte

S_OK

Funktion wurde erfolgreich ausgeführt.

E_POINTER

pVal ist kein gültiger Pointer.

6.7.6.7 DefFileType

Bekommt eine kurze Beschreibung des File Typs für das implementierte Format

Property get

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

Parameter

pVal

[out, retval] Pointer zu einer BSTR Variable die die Beschreibung bekommt.

Rückgabe Werte

S_OK

Funktion wurde erfolgreich ausgeführt.

E_POINTER

pVal ist kein gültiger Pointer.

6.7.6.8 Export

Exportiert von dem übergebenen IEvtConfiguration [▶ 27] Objekt in die geöffnete Konfigurationsdatei.

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

Parameter

pConfig

[in] Konfiguration.

Rückgabe Werte

S_OK

pConfig wurde erfolgreich exportiert.

E_ACCESSDENIED

Der Parser ist wegen einer asynchronen Operation gesperrt.

E_FAIL

Die Konfiguration konnte nicht exportiert werden

6.7.6.9 GetConfigSize

ITcEvtCfgParser

Bekommt die Anzahl der Sourcen, Events und DocLinks der Konfigurationsdatei.

Diese Methode funktioniert nur mit geöffneten Dateien. Um die Grösse einer Konfiguration bekommen bevor sie epxortiert wurde rufern sie IEvtConfiguration::GetConfigSize() [▶ 30] auf.

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

Parameter

nSources

[out] Anzahl der Sourcen

nEvents

[out] Anzahl der Events

nLinks

[out] Anzahl der DocLinks

Rückgabe Werte

S_OK

Erfolg.

E_ACCESSDENIED

Der Parser ist wegen einer asynchronen Operation gesperrt.

E_FAIL

Fehler.

Anmerkungen

Die wiedergegebenen werte müssen nicht richtig sein, Parser können die grösser der Konfiguration aus Performancegründen schätzen.
Die XML und Tps Parser geben zB 0 zurück, wenn sie eine selbst erstellte Konfiguration zum ersten Mal öffnen. Diese Methode funktioniert nur mit Dateien, die mit dem Parser erstellt wurden.

6.7.6.10 Import

CreateAutoIndex.exe

Importiert die geöffnete Konfiguration in ein übergebenes Konfigurationsobjekt.

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

Parameter

pConfig

[in] Zielkonfiguration

Rückgabe Werte

S_OK

Konfiguration wurde erfolgreich in pConfig importiert.

E_ACCESSDENIED

Der Parser ist wegen einer asynchronen Operation gesperrt.

E_FAIL

Die Konfiguration konnte nicht importiert werden.

Sehen Sie dazu auch

IEvtConfiguration [▶ 27]

6.7.6.11 OpenConfiguration

Öffnet eine Konfigurationsdatei. Alle Nachfolgenden Aufrufe sind an diese Datei gerichtet.
Wenn der Vorgang abgeschlossen wurde, rufen Sie CloseConfiguration [▶ 58] auf.

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

Parameter

name

[in] Pfad und Dateiname

Rückgabe Werte

S_OK

Die Konfiguration wurde geöffnet

E_ACCESSDENIED

Der Parser ist wegen einer asynchronen Operation gesperrt.

E_FAIL

Die Datei konnte nicht geöffnet werden.

6.7.7 EvtCfgTpsParser

Der EvtCfgTpsParser arbeitet mit dem TwinCAT Storage Dateiformat.

Die standard Storage Datei ist 'TwinCAT\default.tps'.

Wenn sie Event Messages aus der TwinCAT Storage auslesen wollen müssen sie den TcEventFormatter wählen, wenn sie Events aus der PLC auslösen.
Dies können sie durch das benutzen von 'TcEventLogger.TcEventFormatter' als progid in der TcEvent Struktur.

Wenn sie ActivateConfiguration aufrufen, wird der Parser die geöffnete Konfigurationsdatei als TwinCATs aktuelles Projekt festlegen.

Wenn sie eine neue Konfiguration erstellen, und Registrieren haben sie 2 Nebeneffekte:

- Die Liste der Geloggten Events des EventLoggers geht verloren.
- Zusätzlich TwinCAT server sind nicht mehr vorhanden.

Wenn sie die vorherige Konfiguration aktivieren, sind die Nebeneffekte nicht mehr vorhanden.
Sie müssen TwinCAT neu starten damit das wechseln der aktiven Konfiguration wirksam wird.

6.7.8 EvtCfgXmlParser

DerEvtCfgTpsParser arbeitet mit dem XML Format.

Jede Quelle und ihre Dateien werden in Ihrer eigenen Datei gespeichert.
Alle vorhandenen source ids werden mit Links zu den source XML files in der Projektdatei gespeichert.

Wenn sie Event Messages aus der XML-Datei auslesen wollen müssen sie den TcEventFormatter wählen wenn sie Events aus der PLC auslösen.
Dies können sie durch das benutzen von 'TcEventLogger.TcXmlFormatter' als progid in der TcEvent Struktur.

Wenn sie ActivateConfiguration aufrufen, wird der Parser ihr Projekt in die TcEventSourceLocation.xml im TwinCAT\Resource Verzeichnis hinzufügen.
Alle vorher gespeicherten Informationen werden nicht verändert, neue einträge werden als neu markiert. Nur die markierten Einträge werden gelöscht, wenn sie eine andere Konfiguration aktivieren.

6.7.9 IEvtCfgEditor

The IEvtCfgEditor Iterface provides access to the EventEditors properties.

Tab. 10: Methoden und Eigenschaften nach Vtable sortiert

IEvtCfgEditor Methods	Description
GetConfiguration [▶ 65]	Holt das EventConfiguration [▶ 27]
SelectItem [▶ 66]	Wählt ein item in der TreeView aus
GetSelectedItem [▶ 65]	Holt das ausgewählte TreeView Item
ExpandSelection [▶ 64]	Klappt das gewählte Item auf
CollapseSelection [▶ 64]	Klappt das gewählte Item zu
Copy [▶ 64]	Kopiert den ausgewählten Text
Paste [▶ 66]	Fügt den kopierten Text ein
CanCopy [▶ 63]	Prüft ob das aktuelle Item Kopierbar ist
CanPaste [▶ 63]	Prüft ob eingefügt werden kann
_IEvtCfgEditorEvents Methods	Description
SelChanged	Es wurde ein anderes Item ausgewählt

Tab. 11: Enums

Name	Description
EVTCFGEDITOR_ITEMTYPES [▶ 67]	Stellt fest welches Item ausgewählt ist.

6.7.9.1 CanCopy

Prüft, ob das aktuelle Item Kopierbar ist.

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

Parameter

pVal

[out, retval] TRUE wenn kopieren möglich ist, sonst FALSE.

Rückgabe Werte

S_OK

Erfolg.

6.7.9.2 CanPaste

Prüft ob eingefügt werden kann.

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

Parameter

pVal

[out, retval] TRUE wenn einfügen möglich ist, sonst FALSE.

Rückgabe Werte

S_OK

Erfolg.

6.7.9.3 CollapseSelection

Klappt das gewählte Item zu.

```
HRESULT CollapseSelection();
```

Parameter

Rückgabe Werte

S_OK

Das Item wurde zugeklappt.

6.7.9.4 Copy

Kopiert den ausgewählten Text.

Die Daten des Kopiervorgangs sind nicht Systemweit gespeichert - sie gehen verloren wenn der Editor geschlossen wird und können nicht in mehreren Instanzen genutzt werden.

```
HRESULT Copy();
```

Parameter

Rückgabe Werte

E_FAIL

Das aktuelle Item kann nicht kopiert werden. Rufen sie CanCopy [▶ 63] auf um zu prüfen ob ein Item kopiert werden kann oder nicht.

S_OK

Das Item wurde kopiert.

6.7.9.5 ExpandSelection

Klappt das gewählte Item auf.

```
HRESULT ExpandSelection();
```

Parameter

Rückgabe Werte

S_OK

Das Item wurde aufgeklappt.

6.7.9.6 GetConfiguration

Holt das EventConfiguration [▶ 27] Objekt

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

Parameter

ppConfig

[out, retval] Pointer zu einem Pointer zur Konfiguration.

Rückgabe Werte

S_OK

*ppConfig enthält einen gültigen Pointer.

E_FAIL
Der Editor wurde nicht initialisiert und enthält kein Konfigurationsobjekt.

E_POINTER

ppConfig ist kein gültiger Pointer

6.7.9.7 GetSelectedItem

Holt das ausgewählte TreeView Item.

Das Item wird am Typ des COM Onjekts identifiziert (IEvtCfgSource [▶ 33], IEvtCfgEvent [▶ 42] or IEvtCfgDocLink [▶ 51]).
Unteritems werden mit dem enum EVTCFGEDITOR_ITEMTYPES [▶ 67] und der language Id identifiziert.

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

Parameter

ppItem

[out] Pointer zu einem IDispatch Pointer. Es enthält einen Pointer zu einer der oben genannten Instanzen.

subItem

[out] Das ausgewählte Unteritem. Wenn TYPE_NULL ist das item selbst ausgewählt. Hier [▶ 67] gibt es mehr Details.
Wenn das Unteritem TYPE_SRCNAME, TYPE_EVTMSG oder TYPE_URL ist dann enthält langId gültige Daten.

langId

[out] Die language id für das gewählte Item.

Rückgabe Werte

S_OK

Funktion wurde erfolgreich ausgeführt.

E_POINTER

ppItem ist kein gültiger Pointer.

6.7.9.8 Paste

Fügt den kopierten Text ein. Wenn das kopierte Item nicht als Child vom aktuellem Item eingefügt werden kann schlägt die Funktion fehl.

```
HRESULT Paste();
```

Parameter

Rückgabe Werte

E_FAIL

Item kann nicht als Child vom aktuellem Item eingefügt werden. Rufen sie CanPase [▶ 63] auf um zu prüfen ob Einfügen möglich ist.

S_OK

Das Item wurde eingefügt.

6.7.9.9 SelectItem

IEvtCfgEditor [▶ 63]

Wählt ein item in der TreeView aus.
Das Item wird am Typ des COM Objekts identifiziert (IEvtCfgSource [▶ 33], IEvtCfgEvent [▶ 42] or IEvtCfgDocLink [▶ 51]).
Unteritems werden mit dem enum EVTCFGEDITOR_ITEMTYPES [▶ 67] und der language Id identifiziert.

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

Parameter

pItem

[in] IDispatch Pointer der einen Pointer zu einen der oben angegebenen Interfaces enthält.

subItem

[in] Das Subitem adas ausgewählt werden soll. Wenn TYPE_NULL wird das Item Selbst ausgewählt. Hier [▶ 67] gibt es mehr Details.
Wenn das Unteritem TYPE_SRCNAME, TYPE_EVTMSG oder TYPE_URL ist dann muss eine language id angegeben werden.

langId

[in] Die language id für das Item dass ausgewählt werden soll.

Rückgabe Werte

S_OK

Das Item wurde ausgewählt

E_FAIL

Das Item konnte nicht ausgewählt werden, zB weil es nicht den übergeben Itemtyp entspricht oder es konnte kein Item für die angegebene Sprache gefunden werden.

E_POINTER

pItem ist kein gültiger Pointer.

6.7.9.10 Enum EVTCFGEDITOR_ITEMTYPES

Stellt fest welches Item ausgewählt ist.

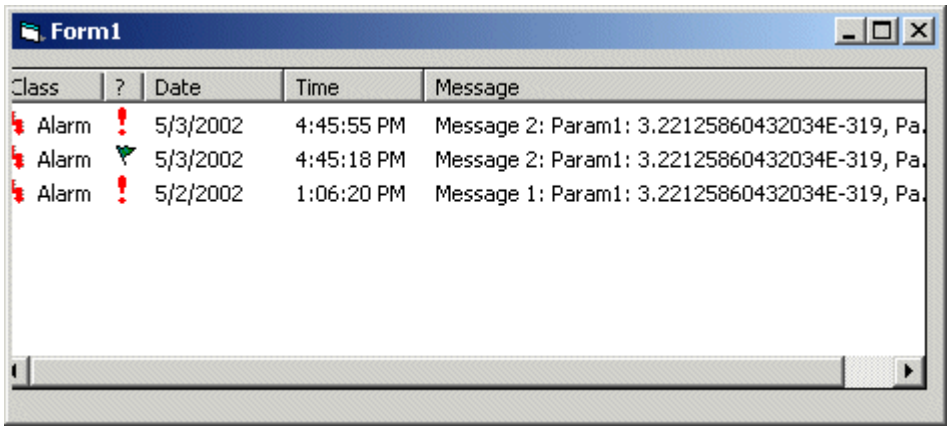
```
typedef enum
{
    TYPE_NULL = 0,
    TYPE_SRC,
    TYPE_SRCNAMETAG,
    TYPE_SRCNAME,
    TYPE_EVTTAG,
    TYPE_EVT,
    TYPE_EVTMSG,
    TYPE_EVTMSGTAG,
    TYPE_LNKTAG,
    TYPE_LNK,
    TYPE_URL
}EVTCFGEDITOR_ITEMTYPES;
```

Wert	Beschreibung	Editor Symbol
TYPE_NULL	Undefiniert, ein Zeichen für einen Fehler	
TYPE_SRC	Ein Source Item	
TYPE_SRCNAMETAG	Namen Tag unter TYPE_SRC	
TYPE_SRCNAME	Ein Sourcename unter TYPE_SRCNAMETAG	
TYPE_EVTTAG	Ein Event Tag unter TYPE_SRC	
TYPE_EVT	Ein Event	
TYPE_EVTMSGTAG	Ein Message Tag unter TYPE_EVT	
TYPE_EVTMSG	Ein Message String unter TYPE_EVT	
TYPE_LNKTAG	Ein DocLink Tag unter TYPE_EVT	
TYPE_LNK	Ein Doc Link Item.	
TYPE_URL	Eine URL unter TYPE_LNK	

7 Event HMI

Die TcEventViewer.dll stellt den Beckhoff TcEventViewer zur Verfügung. Der EventViewer ist ein ActiveX Control, das sich im HMI Baustein des TcEvent Loggers befindet. Durch das Eintragen in Form1und der Konfiguration einiger Eigenschaften kann es verwendet werden, um in weniger als 5 Minuten den sichtbaren Teil des TcEvent Loggers zu implementieren.

Der nächste Screenshot zeigt ein Visual Basic Programm, das den TcEventLogger zur Anzeige der logged Alarms verwendet.



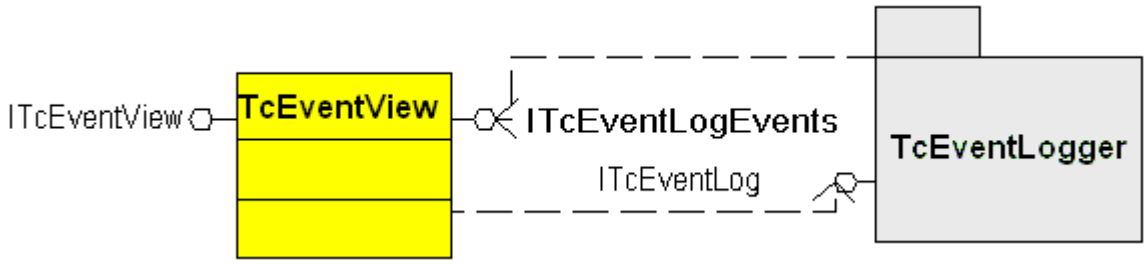
Der TcEventViewer kann Aktive und logged Alarme anzeigen. Die Spalten des TcEventViewers können aus dem Code ausgeblendet und in der Größe verändert werden.
Alle Eventstrings werden in der ausgesuchten Sprache dargestellt, da die Sprachumschaltung durch den Formatter erfolgt.

Ab Version 2.9.0.56 ist der EventViewer in zwei Komponenten aufgeteilt: TcEventview und TcEventviewLight. Die Light Version des Viewers ist auf Performance optiemiert, während der TcEventviewer eine vielzahl von Methoden zur Steuerung der Darstellung bietet.

Interfaces

Der TcEventViewer kommuniziert über seine Schnittstelle mit anderen Komponenten.

Die nächste Abbildung zeigt eine Übersicht der Hauptschnittstelle des TcEventViewers:



Schnittstelle	Beschreibung
ITcEventView [▶ 70]	Hauptschnittstelle des TcEventViewer zu COM Clients wie das HMI. Hauptfeatures: Das Auftreten des ActiveX ist zu kontrollieren.
ITcEventViewLight [▶ 113]	Auf Performance optimierte Version des EventViewers.
ITcEventLogEvents [▶ 146]	Die Event Schnittstelle wird vom TcEventLogger zur Verfügung gestellt. Wenn das TcEvent gestartet ist, bekommt es eine Liste der logged und aktiven Events durch einen Aufruf der Methoden ITcEventLog [▶ 130]::EnumActiveEvents [▶ 133] oder ITcEventLog [▶ 130]::EnumLoggedEvents [▶ 135] derTcEventLog [▶ 129] Klasse und zeigt die Events an. Danach meldet der TcEventLogger den TcEventViewer an die ITcEventLogEvents Schnittstelle. Diese Meldung wird zur Synchronisierung der Liste der angezeigten Events mit dem Server verwendet.

Klassen

Der TcEventViewer unterstützt nur eine Klasse nach außen: **TcEventView**

Client Server

Der TcEventLogger stellt eine Client Server Architektur zur Verfügung. Der TcEventLogger ist der Server, der alle Informationen über aktive und logged Events enthält, selbst derer, die nicht auf einem Display zu sehen sind. Der TcEventViewer ist ein Client des TcEventLoggers. Wenn der EventViewer gestartet ist, bekommt er zuerst die Liste der aktiven und logged Events, zeigt sie an und synchronisiert die Liste durch das Event, dass durch die implementierte Schnittstelle ITcEventLoggEvents [▶ 146] ausgegeben wird.

Der TcEventViewer kann mit drei verschiedene Client Server Konfigurationen verwendet werden.

1. Ein TcEventViewer zeigt das Event eines TcEventLoggers an. In diesem Fall befindet sich das HMI mit dem TcEvenViewer und dem TcEventLogger auf der gleichen Maschine.
2. Ein TcEventViewer zeigt die Alarme auf verschiedenen TcEventLoggers an, die sich auf unterschiedlichen Maschinen PCs befinden. Die Remoteverbindung ist über *DCOM hergestellt.
3. Mehrere TcEventViewer zeigen die Events auf TcEventLoggern an. Die Remoteverbindung ist über *DCOM hergestellt.

*DCOM DCOM muss mit Hilfe von dcomcnfg.exe richtig auf dem Client und den Maschinen PC Server konfiguriert werden. Weitere Informationen über DCOM finden Sie in der MSDN Library.

Da DCOM sehr lange timeout Zeiten benutzt, sollte es nur in sehr stabilen Netzwerkverbindungen verwendet werden. Ein weiteres Problem ist die Tatsache, dass der EventLogger nicht multithreaded ist, der TcEventViewer blockiert also den TcEventLogger. Wenn die 2. Konfiguration verwendet wird, blockiert die Anzeige der Events eines TcEventLoggers den anderen.

7.1 Klassen

7.1.1 TcEventView

Die Hauptklasse des TcEventViewer stellt Schnittstellen zum COM Client wie das HMI zur Verfügung. Diese Klasse stellt den Alarm in einer Listenansicht dar.

Die Klasse ist durch die folgenden Schnittstellen steuerbar.

Schnittstelle	Beschreibung
ITcEventView [▶ 70]	Hauptschnittstelle des TcEventViewers zu COM Clients wie das HMI. Hauptfeatures: das Auftreten des ActiveX Controls ist zu steuern.
ITcEventLogEvents [▶ 146]	Die Event Schnittstelle wird vom TcEventLogger zur Verfügung gestellt. Wenn TcEvent gestartet wird, bekommt es eine Auflistung der logged oder active Events durch einen Aufruf der Methoden ITcEventLog [▶ 130]::EnumActiveEvents [▶ 133] oder ITcEventLog [▶ 130]::EnumLoggedEvents [▶ 135] der TcEventLog Klasse und zeigt die Events an. Danach meldet der TcEventLogger dem TcEventViewer die ITcEventLogEvents Schnittstelle. Diese Benachrichtigung wird zur Synchronisierung der aufgelisteten und angezeigten Events mit dem Server verwendet.

7.2 Interfaces

7.2.1 ITcEventView

Die Schnittstelle ITcEventView wird zur Kontrolle der Eigenschaften des TcEventViewer ActiveX Control verwendet. Die Schnittstelle leitet sich von **IDispatch** ab, so dass sie von allen Sprachen verwendet werden kann, die die COM Automation Schnittstellen unterstützen. IDispatch selbst leitet sich von IUnknown ab.

Tab. 12: Methoden und Eigenschaften in

IUnknown Methoden	Beschreibung
QueryInterface	Gibt einen Pointer zur angefragten Schnittstelle zurück.
AddRef	Inkrementiert den Referenzzähler.
Release	Dekrementiert den Referenzzähler.

IDispatch Methoden	Beschreibung
GetIDsOfNames	Mappt einen einzelnen Teilnehmernamen und einen Satz optionaler Parameternamen mit einem entsprechenden Satz gesendeter Integer Identifier (DISPIDs). Diese können dann als Folgeabrufe an **IDispatch::Invoke** benutzt werden.
GetTypeInfo	Ruft die Typ Informationen für ein Objekt ab.
GetTypeInfoCount	Ruft die Anzahl der Typ Informations-Schnittstellen ab, die ein Objekt zur Verfügung stellt, entweder 0 oder 1.
Invoke	Stellt den Zugriff auf Eigenschaften und Methoden, ausgestellt durch ein Objekt, zur Verfügung.

ITcEventView Methoden und Eigenschaften	Beschreibung
ShowLoggedEvents [▶ 72]	Zeigt die logged Events an.
ShowActiveEvents [▶ 72]	Zeigt die akiven Events an.
ConfirmEvent [▶ 73]	Bestätigt ein Event.
ResetEvent [▶ 73]	Setzt ein Event zurück.
LangId [▶ 74]	Setzt die Sprache, die zur Anzeige der Event Meldung und des Source Namens verwendet wird, zurück.
DisableMenuItems [▶ 75]	Setzt oder gibt den abgeschalteten Status des Kontextmenüs zurück.
SelectEvent [▶ 77]	Markiert ein Event als ausgewählt.
AddConnection [▶ 78]	Addiert eine Remoteverbindung zu einem EventLogger.
DeleteConnection [▶ 79]	Entfernt eine Remoteverbindung zu einem EventLogger.
DisplayComputerName [▶ 79]	Aktiviert/Deaktiviert die Spalte, die den Computernamen anzeigt oder den Status zurückgibt.
DisplayEventId [▶ 80]	Aktiviert/Deaktiviert die Spalte, die die Event Id anzeigt oder den Status zurückgibt.
AlarmTextColor [▶ 83]	Setzt oder gibt die zur Anzeige der Alarme verwendete Farbe zurück.
WarningTextColor [▶ 85]	Setzt oder gibt die zur Anzeige der Warnungen verwendete Farbe zurück.
InstructionTextColor [▶ 86]	Setzt oder gibt die zur Anzeige der Anweisungen verwendete Farbe zurück.

ITcEventView Methoden und Eigenschaften	Beschreibung
ParamErrorTextColor [▶ 87]	Setzt oder gibt die zur Anzeige der Parameterfehler verwendete Farbe zurück.
StateInfoTextColor [▶ 88]	Setzt oder gibt die zur Anzeige der Statusinformationen verwendete Farbe zurück.
HintTextColor [▶ 89]	Setzt oder gibt die zur Anzeige der Hinweise verwendete Farbe zurück.
MessageTextColor [▶ 90]	Setzt oder gibt die zur Anzeige der Meldungen verwendete Farbe zurück.
MaintenanceTextColor [▶ 91]	Setzt oder gibt die zur Anzeige der Informationen zur Instandhaltung verwendete Farbe zurück.
UndefTextColor [▶ 92]	Setzt oder gibt die zur Anzeige der undefinierten Events verwendete Farbe zurück.
ColWidthClass [▶ 93]	Setzt oder gibt die Weite der Spalte Event Klasse zurück.
ColWidthMustCon [▶ 94]	Setzt oder gibt die Weite der Spalte Bestätigungsstatus (confirm state) zurück.
ColWidthState [▶ 94]	Setzt oder gibt die Weite der Spalte Event Status zurück.
ColWidthSource [▶ 95]	Setzt oder gibt die Weite der Spalte Event Source zurück.
ColWidthDate [▶ 96]	Setzt oder gibt die Weite der Spalte Event Datum zurück.
ColWidthTime [▶ 97]	Setzt oder gibt die Weite der Spalte Event Zeit zurück.
ColWidthEventId [▶ 98]	Setzt oder gibt die Weite der Spalte Event Id zurück.
ColWidthComputer [▶ 99]	Setzt oder gibt die Weite der Spalte Computername zurück.
ColWidthMsg [▶ 99]	Setzt oder gibt die Weite der Spalte Event Meldung zurück.
DisplayClass [▶ 100]	Aktiviert/Deaktiviert die Spalte, die die Event Klasse anzeigt oder gibt den Status zurück.
DisplayMustCon [▶ 101]	Aktiviert/Deaktiviert die Spalte, die den Bestätigungsstatus (confirm State) anzeigt, oder gibt den Status zurück.
DisplayState [▶ 102]	Aktiviert/Deaktiviert die Spalte, die den Eventstatus anzeigt, oder gibt den Status zurück.
DisplaySource [▶ 103]	Aktiviert/Deaktiviert die Spalte, die den Sourcenamen anzeigt, oder gibt den Status zurück.
DisplayDate [▶ 104]	Aktiviert/Deaktiviert die Spalte, die das Eventdatum anzeigt, oder gibt den Status zurück.
DisplayTime [▶ 105]	Aktiviert/Deaktiviert die Spalte, die die Eventzeit anzeigt, oder gibt den Status zurück.
DisplayMsg [▶ 106]	Aktiviert/deaktiviert die Spalte, die die Eventmeldung anzeigt, oder gibt den Status zurück.
GetSelectedEvent [▶ 107]	Gibt den Index des Events zurück, dass vom Benutzer ausgewählt wurde.
Mode [▶ 108]	Setzt oder gibt den Anzeigemodus des TcEventViewers zurück.
MaxEvents [▶ 109]	Setzt oder gibt die maximale Anzahl der angezeigten Events zurück.
DisplayTitlebar [▶ 81]	Aktiviert/Deaktiviert den Columnheader.
DisplayUserComment [▶ 82]	Aktiviert/Deaktiviert die Spalte, die den Benutzer Kommentar anzeigt, oder gibt den Status zurück.
DisableContextMenu [▶ 76]	Aktiviert/Deaktiviert das Kontext Menu.
DisableScrollbars [▶ 77]	Aktiviert/Deaktiviert die Scrollbars.
SortMode [▶ 111]	Setzt oder gibt das Sortierverfahren zurück.
putFont [▶ 110]	Setzt die Schriftart für das control.
getExtendedInfo [▶ 107]	Gibt erweiterte Informationen für ein Event zurück.
UseExternalLanguageResource [▶ 112]	Aktiviert/Deaktiviert die Nutzung einer externen resourcen Datei.

ITcEventView Methoden und Eigenschaften	Beschreibung
BackgroundColor [▶ 84]	Setzt oder gibt die Hintergrund Farbe zurück.
ModifyLVStyle [▶ 85]	Leitet die Parameter an die CWindow::ModifyStyle Methode weiter.
SetFilter [▶ 110]	Ermöglicht das Setzen eines Anzeigefilters.

7.2.1.1 ShowLoggedEvents

ITcEventView [▶ 70]

Diese Methode zeigt alle logged Events aller EventLogger mit denen TcEventView verbunden ist. Bevor sie die Events anzeigt, werden die alten Events gelöscht.

```
HRESULT ShowLoggedEvents();
```

Parameter

Rückgabe Werte

S_OK

Funktion wurde erfolgreich aufgerufen.

Visual Basic Beispielcode

```
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 [▶ 70]

Diese Methode zeigt alle aktiven Events aller EventLogger mit denen TcEventView verbunden ist. Bevor sie die Events anzeigt, werden die alten Events gelöscht.

```
HRESULT ShowActivedEvents();
```

Parameter

Rückgabe Werte

S_OK

Funktion wurde erfolgreich aufgerufen.

Visual Basic Beispielcode

```
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 [▶ 70]

Diese Methode bestätigt (confirms) ein angezeigtes Event durch den vorgegebenen Index in der Liste Ansicht und Bestätigungscode.

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

Parameter

nIndex

[in] Index des Events, das bestätigt werden soll. Der Index in der Liste Ansicht startet oben mit dem Index 0.

nCode

[in] Bestätigungscode, der durch das Enum TcEventConCodes beschrieben wird.

Rückgabe Werte

S_OK

Funktion wurde erfolgreich aufgerufen.

Visual Basic Beispielcode

```
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 [▶ 70]

Diese Methode setzt ein angezeigtes Event durch den gegebenen Index in der Listenansicht zurück.

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

Parameter

nIndex

[in] Index des Events, das zurückgesetzt werden soll. Der Index in der Listenansicht startet oben mit dem Index 0.

Rückgabe Werte

S_OK

Funktion wurde erfolgreich aufgerufen.

Visual Basic Beispielcode

```
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 [▶ 70]

Diese Eigenschaft setzt und gibt die Sprache zurück, in welcher der TcEventViewer, Meldungstexte und Quellen angezeigt werden.
Es kann entweder nur die primäre Sprach Id (z.B. ENG = 9; GER = 7) oder eine komplette LCID (z.B. ENG = 1033; GER = 1031) angegeben werden. Der TcEventViewer wird die Parameter entsprechend interpretieren. Das Setzen der Sprache hat zwei Effekte: zum einen werden die intern benutzten String Tables geändert, zum anderen wird der übergebene Parameter an den TcEventFormatter weitergeleitet. Es ist wichtig, dass der Parameter nur weitergeleitet wird. Das heißt, wenn 7 für Deutsch als Sprache übergeben wird, sucht der Formatter in seinen Ressourcen nach dem String '7'. Ist die deutsche Meldung in der Ressource als '1031' definiert, wird sie nicht gefunden.

Property get

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

Parameter

pVal

[out, retval] Pointer auf eine long Variable, die die aktuell angewählte Sprach-Id erhält.

Rückgabe Werte

S_OK

Funktion wurde erfolgreich aufgerufen.

E_POINTER

pVal war kein gültiger Pointer.

Property let

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

Parameter

newVal

[in] Die Sprach-Id der angefragten Sprache. Die gewünschte Sprache sollte mit *LCIDs markiert werden. Die nächste Tabelle zeigt ein Beispiel für einige Sprach-Ids. In der Konfiguration des Event Formatters werden die Sprachen mit der gleichen Sprach-Id gekennzeichnet.

LCID	Beschreibung
1031	Deutsch
1033	US Englisch
1034	Spanisch
1036	Französisch

Rückgabe Werte

S_OK

Funktion wurde erfolgreich aufgerufen.

Visual Basic Beispielcode

```
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
```

Anmerkungen

Wenn eine neue Sprache zugeordnet wird, werden alle angezeigten Events erneut in der neu ausgewählten Sprache aufgerufen.

7.2.1.6 DisableMenuItems

ITcEventView [▶ 70]

Diese Eigenschaft setzt oder gibt den abgeschalteten Status des Kontextmenüs zurück.

Property get

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

Parameter

pVal

[out, retval] Pointer auf eine Long Variable, die den aktuell abgeschalteten Status des Kontextmenüs erhält. Der abgeschaltete Status wird durch die Verknüpfung des Enum Typs TCEVENTVIEW_CONTEXTMENUFLAGS [▶ 114] dargestellt.

Rückgabe Werte

S_OK

Funktion wurde erfolgreich aufgerufen.

E_POINTER

pVal war kein gültiger Pointer.

Property let

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

Parameter

newVal

[in] Eine long Variable, die den neuen abgeschalteten Status des Kontektmenüs enthält. Der abgeschaltete Status wird durch die Verknüpfung des Enum Typs TCEVENTVIEW_CONTEXTMENUFLAGS dargestellt.

Rückgabe Werte

S_OK

Funktion wurde erfolgreich aufgerufen.

Visual Basic Beispielcode

```
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

ITcEventView [▶ 70]

Diese Eigenschaft aktiviert/deaktiviert das Context Menü des TcEventView.

Property get

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

Parameter

pVal

[out, retval] Pointer auf eine Variable, die den aktuellen Status erhält.

Rückgabe Werte

S_OK
Funktion wurde erfolgreich aufgerufen.
E_POINTER
pVal war kein gültiger Pointer.

Property set

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

Parameter

newVal

[in] Eine Variable, die den neuen Status enthält.

Rückgabe Werte

S_OK
Funktion wurde erfolgreich aufgerufen.

Visual Basic Beispielcode

```
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 [▶ 70]

Diese Eigenschaft aktiviert/deaktiviert die Scrollbars für das Control.

Property get

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

Parameter

pVal

[out, retval] Pointer auf eine Variable, die den aktuellen Status erhält.

Rückgabe Werte

S_OK
Funktion wurde erfolgreich aufgerufen.
E_POINTER
pVal war kein gültiger Pointer.

Property set

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

Parameters

newVal

[in] Eine Variable, die den neuen Status enthält.

Rückgabe Werte

S_OK
Funktion wurde erfolgreich aufgerufen.

Visual Basic Beispielcode

```
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 [▶ 70]

Diese Methode markiert ein Event in der Listenansicht durch den vorgegebenen Index als ausgewählt. Das ausgewählte Event wird hervorgehoben dargestellt.

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

Parameter

nIndex

[in] Index des Events, das ausgewählt werden soll. Der Index in der Listenansicht startet oben mit dem Index 0.

Return Values

S_OK

Funktion wurde erfolgreich aufgerufen.

Visual Basic Beispielcode

```
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
```

Anmerkungen

Diese Methode hat das gleiche Verhalten wie ein Alarm, der durch einen Klick mit der linken Maustaste auf den Eventeintrag in der Listenansicht ausgewählt wird.

7.2.1.10 AddConnection

ITcEventView [▶ 70]

Diese Methode fügt eine neue Verbindung zu einem PC im Netzwerk hinzu.

Standardmäßig wird die Remoteverbindung via *DCOM hergestellt, durch das Präfix *ADS://* wird die Kommunikationsstrecke per **ADS überbrückt.

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

Parameter

strComputerName

[in] BSTR Name des remote PC mit dem der TcEventViewer eine Verbindung herstellen soll.

Rückgabe Werte

S_OK

Funktion wurde erfolgreich aufgerufen.

```
TCEVTVIEWERR_ADVISESINK = 0xF1000000
```

Eine Verbindung mit dem remote EventLogger konnte hergestellt werden aber das Anmelden der EventSink ist fehlgeschlagen. In diesem Fall müssen Meldungen durch Aufrufe der ShowEvents Methode gepollt werden. Das high order Byte beschreibt den Fehlerfall, die low order Bytes stellen den ursprünglichen Win32 Fehler Code dar, der vom DispEventAdvise Aufruf zurückgegeben wurde.

```
TCEVENTERR_ADS = 0x98200000
```

Bei Verwendung der ADS Schnittstelle werden ADS Fehler gegebenenfalls durchgereicht. Das Lowword enthält dann den Zugrundeliegenden Ads Fehlercode.

Visual Basic Beispielcode

```
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
```

Anmerkungen

*DCOM: DCOM sollte nur in stabilen Netzwerken verwendet werden. DCOM muss korrekt auf dem PC eingerichtet sein, der TcEvenView verwendet, und auf dem PC, der den TcEventLogger durch die Verwendung von dcomcnfg.exe bereitstellt. Weitere Informationen finden Sie in der MSDN Library.

**ADS: Um ADS zur Verbindung mit einem Remotesystem zu nutzen, muss die Verbindung zunächst eingerichtet werden. Beide Systeme müssen gewisse Mindestanforderungen erfüllen. Weitere Informationen hierzu finden sie hier [▶ 17]

7.2.1.11 DeleteConnection

ITcEventView [▶ 70]

Diese Methode entfernt die Verbindung eines EventLoggers zu einem remote TcEventLogger auf einem anderen Rechner, die durch AddConnection [▶ 78] hergestellt wurde.

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

Parameter

strComputerName

[in] BSTR String, der den Namen des remote PC enthält, von dem EventView getrennt werden soll.

Rückgabe Werte

S_OK

Funktion wurde erfolgreich aufgerufen.

Visual Basic Beispielcode

```
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
```

Anmerkungen

7.2.1.12 DisplayComputerName

ITcEventView [▶ 70]

Diese Eigenschaft aktiviert/deaktiviert die Spalte, die den Computernamen anzeigt oder den Status zurückgibt.

Property get

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

Parameter

pVal

[out, retval] Pointer auf eine VARIANT_BOOL Variable, die den sichtbaren Status der Spalte Computername in der Listenansicht erhält. Der Status ist sichtbar, wenn die Variable VARIANT_TRUE (TRUE) ist, und unsichtbar, wenn die Variable VARIANT_FALSE (FALSE) ist.

Rückgabe Werte

S_OK

Funktion wurde erfolgreich aufgerufen.

E_POINTER

pVal war kein gültiger Pointer.

Property let

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

Parameter

newVal

[in] Variable, die den sichtbaren Status der Spalte Computername in der Listenansicht setzt. Der Status ist sichtbar, wenn die Variable VARIANT_TRUE (TRUE) ist, und unsichtbar, wenn die Variable VARIANT_FALSE (FALSE) ist.

Rückgabe Werte

S_OK

Funktion wurde erfolgreich aufgerufen.

Visual Basic Beispielcode

```
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 [▶ 70]

Aktiviert/Deaktiviert die Spalte, die die Event Id anzeigt oder den Status zurückgibt.

Property get

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

Parameters

pVal

[out, retval] Pointer auf eine VARIANT_BOOL Variable, die den sichtbaren Status der Spalte Event Id in der Listenansicht erhält. Der Status ist sichtbar, wenn die Variable VARIANT_TRUE (TRUE) ist, und unsichtbar, wenn die Variable VARIANT_FALSE (FALSE) ist.

Rückgabe Werte

S_OK

Funktion wurde erfolgreich aufgerufen.

E_POINTER

pVal war kein gültiger Pointer.

Property let

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

Parameter

newVal

[in] Variable, die den sichtbaren Status der Spalte Event Id in der Listenansicht setzt. Der Status ist sichtbar, wenn die Variable VARIANT_TRUE (TRUE) ist, und unsichtbar, wenn die Variable VARIANT_FALSE (FALSE) ist.

Rückgabe Werte

S_OK

Funktion wurde erfolgreich aufgerufen.

Visual Basic Beispielcode

```
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 [▶ 70]

Diese Eigenschaft zeigt oder versteckt den 'Columnheader' des Controls.

Property get

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

Parameter

pVal

[out, retval] Pointer auf eine Variable, die den aktuellen Status erhält.

Rückgabe Werte

S_OK
Funktion wurde erfolgreich aufgerufen.
E_POINTER
pVal war kein gültiger Pointer.

Property set

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

Parameter

newVal

[in] Eine Variable, die den aktuellen Status setzt.

Rückgabe Werte

S_OK
Funktion wurde erfolgreich aufgerufen.

Visual Basic Beispielcode

```
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 [▶ 70]

Diese Eigenschaft aktiviert/deaktiviert die Spalte 'UserComment'.

Property get

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

Parameter

pVal

[out, retval] Pointer zu einer Variablen, die den aktuellen Wert erhält.

Rückgabe Werte

S_OK
Funktion wurde erfolgreich aufgerufen.
E_POINTER
pVal war kein gültiger Pointer.

Property set

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

Parameter

newVal

[in] Eine Variable, die den neuen Status setzt.

Rückgabe Werte

S_OK

Funktion wurde erfolgreich aufgerufen.

Visual Basic Beispielcode

```
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 [▶ 70]::AlarmTextColor

Diese Eigenschaft setzt oder gibt die zur Anzeige der Alarme verwendeten Farbe zurück. Der Alarm wurde vom SPS Funktionsblock ADSLOGEVENT ausgegeben, indem ein Merker von FALSE auf TRUE gesetzt wurde (oder ein anderes ADS Gerät), oder durch den Aufruf einer Report Event Funktion, wie ITcEventLogC [▶ 139]::ReportEvent [▶ 131],ITcEventC3 [▶ 142]::ReportEventEx [▶ 144] oder ITcEventLog [▶ 130]::ReportEvent [▶ 131] (dabei ist die TcEventClass [▶ 181] auf TCEVENTCLASS_ALARM gesetzt).

Property get

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

Parameter

pVal

[out, retval] Pointer auf eine OLE_COLOR Variable, welche die Textfarbe erhält, die zur Anzeige der Alarme verwendet wird.

Rückgabe Werte

S_OK

Funktion wurde erfolgreich aufgerufen.

E_POINTER

pVal war kein gültiger Pointer.

Property let

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

Parameter

newVal

[in] Variable, welche die Textfarbe setzt, die zur Anzeige der Alarme verwendet wird.

Rückgabe Werte

S_OK

Funktion wurde erfolgreich aufgerufen.

Visual Basic Beispielcode

```
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
```

Sehen Sie dazu auch

ReportEvent [▶ 139]

7.2.1.17 BackgroundColor

ITcEventView [▶ 70]

Diese Eigenschaft stellt die Hintergrundfarbe des TcEventViewers ein.

Property get

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

Parameters

pVal

[out, retval] Pointer auf eine Variable, die den aktuellen Status erhält.

Rückgabe Werte

S_OK
Funktion wurde erfolgreich aufgerufen.
E_POINTER
pVal war kein gültiger Pointer.

Property set

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

Parameters

newVal [in] Eine Variable, die den neuen Status enthält.

Rückgabe Werte

S_OK

Funktion wurde erfolgreich aufgerufen.

Visual Basic Beispielcode

```
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 [▶ 70]

Diese Methode leitet ihre Parameter an die Cwindow::ModifyStyle des ListView Controls weiter. Weitere Informatinen finden Sie in der MSDN Library.

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

Parameters

dwRemove

[in] Window flags, die entfernt werden sollen.

dwAdd

[in] Window flags, die hinzugefügt werden sollen.

Rückgabe Werte

S_OK

Funktion wurde erfolgreich aufgerufen

7.2.1.19 WarningTextColor

ITcEventView [▶ 70]::WarningTextColor

Diese Eigenschaft setzt oder gibt die zur Anzeige der Warnungen verwendeten Farbe zurück. Die Warnung wurde vom SPS Funktionsblock ADSLOGEVENT ausgegeben, indem ein Merker von FALSE auf TRUE gesetzt wurde (oder ein anderes ADS Gerät) oder durch den Aufruf einer Report Event Funktion, wie ITcEventLogC::ReportEvent [▶ 139], ITcEventC3::ReportEventEx [▶ 144] oder ITcEventLog::ReportEvent [▶ 131] (dabei ist TcEventClass [▶ 181] auf TCEVENTCLASS_WARNING gesetzt).

Property get

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

Parameter

pVal

[out, retval] Pointer auf eine OLE_COLOR Variable, welche die Textfarbe erhält, die zur Anzeige der Warnungen verwendet wird.

Rückgabe Werte

S_OK

Funktion wurde erfolgreich aufgerufen.

E_POINTER

pVal war kein gültiger Pointer.

Property let

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

Parameter

newVal

[in] Variable, welche die Textfarbe setzt, die zur Anzeige der Warnungen verwendet wird.

Rückgabe Werte

S_OK

Funktion wurde erfolgreich aufgerufen.

Visual Basic Beispielcode

```
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 [▶ 70]::InstructionTextColor

Diese Eigenschaft setzt oder gibt die zur Anzeige der Anweisungen verwendeten Farbe zurück. Die Anweisung wurde vom SPS Funktionsblock ADSLOGEVENT ausgegeben, indem ein Merker von FALSE auf TRUE gesetzt wurde (oder ein anderes ADS Gerät) oder durch den Aufruf einer Report Event Funktion, wie ITcEventLogC::ReportEvent [▶ 139], ITcEventC3::ReportEventEx [▶ 144] oder ITcEventLog::ReportEvent [▶ 131] (dabei ist TcEventClass [▶ 181] auf TCEVENTCLASS_INSTRUCTION gesetzt).

Property get

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

Parameter

pVal

[out, retval] Pointer auf eine OLE_COLOR Variable, welche die Textfarbe erhält, die zur Anzeige der Anweisungen verwendet wird.

Rückgabe Werte

S_OK

Funktion wurde erfolgreich aufgerufen.

E_POINTER

pVal war kein gültiger Pointer.

Property let

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

Parameter

newVal

[in] Variable, welche die Textfarbe setzt, die zur Anzeige der Anweisungen verwendet wird.

Rückgabe Werte

S_OK

Funktion wurde erfolgreich aufgerufen.

Visual Basic Beispielcode

```
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 [▶ 70]::ParamErrorTextColor

Diese Eigenschaft setzt oder gibt die zur Anzeige der Parameterfehler verwendeten Farbe zurück. Der Parameterfehler wurde vom SPS Funktionsblock ADSLOGEVENT ausgegeben, indem ein Merker von FALSE auf TRUE gesetzt wurde (oder ein anderes ADS Gerät) oder durch den Aufruf einer Report Event Funktion, wie ITcEventLogC::ReportEvent [▶ 139], ITcEventC3::ReportEventEx [▶ 144] oder ITcEventLog::ReportEvent [▶ 131] (dabei ist TcEventClass [▶ 181] auf TCEVENTCLASS_PARAMERROR gesetzt).

Property get

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

Parameter

pVal

[out, retval] Pointer auf eine OLE_COLOR Variable, welche die Textfarbe erhält, die zur Anzeige der Parameterfehler verwendet wird.

Rückgabe Werte

S_OK

Funktion wurde erfolgreich aufgerufen.

E_POINTER

pVal war kein gültiger Pointer.

Property let

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

Parameter

newVal

[in] Variable, welche die Textfarbe setzt, die zur Anzeige der Parameterfehler verwendet wird.

Rückgabe Werte

S_OK

Funktion wurde erfolgreich aufgerufen.

Visual Basic Beispielcode

```
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

ITcEventView [▶ 70]::StateInfoTextColor

Diese Eigenschaft setzt oder gibt die zur Anzeige der Statusinformation verwendeten Farbe zurück. Die Statusinformation wurde vom SPS Funktionsblock ADSLOGEVENT ausgegeben, indem ein Merker von FALSE auf TRUE gesetzt wurde (oder ein anderes ADS Gerät) oder durch den Aufruf einer Report Event Funktion, wie ITcEventLogC::ReportEvent [▶ 139], ITcEventC3::ReportEventEx [▶ 144] oder ITcEventLog::ReportEvent [▶ 131] (dabei ist TcEventClass [▶ 181] auf TCEVENTCLASS_STATEINFO gesetzt).

Property get

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

Parameter

pVal

[out, retval] Pointer auf eine OLE_COLOR Variable, welche die Textfarbe erhält, die zur Anzeige der Statusinformationen verwendet wird.

Rückgabe Werte

S_OK

Funktion wurde erfolgreich aufgerufen.

E_POINTER

pVal war kein gültiger Pointer.

Property let

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

Parameter

newVal

[in] Variable, welche die Textfarbe setzt, die zur Anzeige der Statusinformationen verwendet wird.

Rückgabe Werte

S_OK

Funktion wurde erfolgreich aufgerufen.

Visual Basic Beispielcode

```
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 [▶ 70]::HintTextColor

Diese Eigenschaft setzt oder gibt die zur Anzeige der Hinweise verwendeten Farbe zurück. Der Hinweis wurde vom SPS Funktionsblock ADSLOGEVENT ausgegeben, indem ein Merker von FALSE auf TRUE gesetzt wurde (oder ein anderes ADS Gerät) oder durch den Aufruf einer Report Event Funktion, wie ITcEventLogC::ReportEvent [▶ 139], ITcEventC3::ReportEventEx [▶ 144] oder ITcEventLog::ReportEvent [▶ 131] (dabei ist TcEventClass [▶ 181] auf TCEVENTCLASS_HINT gesetzt).

Property get

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

Parameter

pVal

[out, retval] Pointer auf eine OLE_COLOR Variable, welche die Textfarbe erhält, die zur Anzeige der Hinweise verwendet wird.

Rückgabe Werte

S_OK

Funktion wurde erfolgreich aufgerufen.

E_POINTER

pVal war kein gültiger Pointer.

Property let

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

Parameter

newVal

[in] Variable, welche die Textfarbe setzt, die zur Anzeige der Hinweise verwendet wird.

Rückgabe Werte

S_OK

Funktion wurde erfolgreich aufgerufen.

Visual Basic Beispielcode

```
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 [▶ 70]::MessageTextColor

Diese Eigenschaft setzt oder gibt die zur Anzeige der Meldungen verwendeten Farbe zurück. Die Meldung wurde vom SPS Funktionsblock ADSLOGEVENT ausgegeben, indem ein Merker von FALSE auf TRUE gesetzt wurde (oder ein anderes ADS Gerät) oder durch den Aufruf einer Report Event Funktion, wie ITcEventLogC::ReportEvent [▶ 139],ITcEventC3::ReportEventEx [▶ 144] oder ITcEventLog::ReportEvent [▶ 131] (dabei ist TcEventClass [▶ 181] auf TCEVENTCLASS_MESSAGE gesetzt).

Property get

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

Parameter

pVal

[out, retval] Pointer auf eine OLE_COLOR Variable, welche die Textfarbe erhält, die zur Anzeige der Meldungen verwendet wird.

Rückgabe Werte

S_OK

Funktion wurde erfolgreich aufgerufen.

E_POINTER

pVal war kein gültiger Pointer.

Property let

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

Parameter

newVal

[in] Variable, welche die Textfarbe setzt, die zur Anzeige der Meldungen verwendet wird.

Rückgabe Werte

S_OK

Funktion wurde erfolgreich aufgerufen.

Visual Basic Beispielcode

```
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

Diese Eigenschaft setzt oder gibt die zur Anzeige der Instandhaltungsmeldung verwendeten Farbe zurück. Die Instandhaltungsmeldung wurde vom SPS Funktionsblock ADSLOGEVENT ausgegeben, indem ein Merker von FALSE auf TRUE gesetzt wurde (oder ein anderes ADS Gerät) oder durch den Aufruf einer Report Event Funktion, wie ITcEventLogC::ReportEvent [▶ 139],ITcEventC3::ReportEventEx [▶ 144] oder ITcEventLog::ReportEvent [▶ 131] (dabei ist TcEventClass [▶ 181] auf TCEVENTCLASS_MAINTENANCE gesetzt).

Property get

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

Parameter

pVal

[out, retval] Pointer auf eine OLE_COLOR Variable, welche die Textfarbe erhält, die zur Anzeige der Instandhaltungsmeldungen verwendet wird.

Rückgabe Werte

S_OK

Funktion wurde erfolgreich aufgerufen.

E_POINTER

pVal war kein gültiger Pointer.

Property let

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

Parameter

newVal	[in] Variable, welche die Textfarbe setzt, die zur Anzeige der Instandhaltungsmeldungen verwendet wird.

Rückgabe Werte

S_OK

Funktion wurde erfolgreich aufgerufen.

Visual Basic Beispielcode

```
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
```

Sehen Sie dazu auch

ITcEventView [▶ 70]

7.2.1.26 UndefTextColor

ITcEventView [▶ 70]::UndefTextColor

Diese Eigenschaft setzt oder gibt die zur Anzeige der undefinierten Meldungen verwendeten Farbe zurück. Die undefinierte Meldung wurde vom SPS Funktionsblock ADSLOGEVENT ausgegeben, indem ein Merker von FALSE auf TRUE gesetzt wurde (oder ein anderes ADS Gerät) oder durch den Aufruf einer Report Event Funktion, wie ITcEventLogC::ReportEvent [▶ 139],ITcEventC3::ReportEventEx [▶ 144] oder ITcEventLog::ReportEvent [▶ 131] (dabei ist TcEventClass [▶ 181] auf einen nicht definierten Wert gesetzt).

Property get

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

Parameter

pVal

[out, retval] Pointer auf eine OLE_COLOR Variable, welche die Textfarbe erhält, die zur Anzeige der undefinierten Meldungen verwendet wird.

Rückgabe Werte

S_OK

Funktion wurde erfolgreich aufgerufen.

E_POINTER

pVal war kein gültiger Pointer.

Property let

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

Parameter

newVal

[in] Variable, welche die Textfarbe setzt, die zur Anzeige der undefinierten Meldungen verwendet wird.

Rückgabe Werte

S_OK

Funktion wurde erfolgreich aufgerufen.

Visual Basic Beispielcode

```
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 [▶ 70]

Diese Eigenschaft setzt oder gibt die Weite der Spalte Event Klasse zurück.

Property get

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

Parameter

pVal

[out, retval] Pointer auf eine Variable, welche die Weite der Spalte Event Klasse in der Listenansicht erhält.

Rückgabe Werte

S_OK
Funktion wurde erfolgreich aufgerufen.
E_POINTER
pVal war kein gültiger Pointer.

Property let

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

Parameter

newVal	[in] Variable, welche die Weite der Spalte Event Klasse in der Listenansicht setzt.

Rückgabe Werte

S_OK
Funktion wurde erfolgreich aufgerufen.

Visual Basic Beispielcode

```
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 [▶ 70]

Diese Eigenschaft setzt oder gibt die Weite der Spalte Bestätigungsstatus (confirm state) zurück.

Property get

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

Parameter

pVal

[out, retval] Pointer auf eine Variable, welche die Weite der Spalte Bestätigungsstatus in der Listenansicht erhält.

Rückgabe Werte

S_OK

Funktion wurde erfolgreich aufgerufen.

E_POINTER

pVal war kein gültiger Pointer.

Property let

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

Parameter

newVal	[in] Variable, welche die Weite der Spalte Bestätigungsstatus in der Listenansicht setzt.

Rückgabe Werte

S_OK

Funktion wurde erfolgreich aufgerufen.

Visual Basic Beispielcode

```
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 [▶ 70]

Diese Eigenschaft setzt oder gibt die Weite der Spalte Event Status zurück.

Property get

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

Parameter

pVal

[out, retval] Pointer auf eine Variable, welche die Weite der Spalte Event Status in der Listenansicht erhält.

Rückgabe Werte

S_OK
Funktion wurde erfolgreich aufgerufen.
E_POINTER
pVal war kein gültiger Pointer.

Property let

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

Parameter

newVal	[in] Variable, welche die Weite der Spalte Event Status in der Listenansicht setzt.

Rückgabe Werte

S_OK
Funktion wurde erfolgreich aufgerufen.

Visual Basic Beispielcode

```
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 [▶ 70]

Diese Eigenschaft setzt oder gibt die Weite der Spalte Event Source zurück.

Property get

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

Parameter

pVal

[out, retval] Pointer auf eine Variable, welche die Weite der Spalte Event Source in der Listenansicht erhält.

Rückgabe Werte

S_OK

Funktion wurde erfolgreich aufgerufen.
E_POINTER
pVal war kein gültiger Pointer.

Property let

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

Parameter

newVal	[in] Variable, welche die Weite der Spalte Event Source in der Listenansicht setzt.

Rückgabe Werte

S_OK
Funktion wurde erfolgreich aufgerufen.

Visual Basic Beispielcode

```
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

ITcEventView [▶ 70]

Diese Eigenschaft setzt oder gibt die Weite der Spalte Event Datum zurück.

Property get

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

Parameter

pVal

[out, retval] Pointer auf eine Variable, welche die Weite der Spalte Event Datum in der Listenansicht erhält.

Rückgabe Werte

S_OK
Funktion wurde erfolgreich aufgerufen.
E_POINTER
pVal war kein gültiger Pointer.

Property let

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

Parameter

newVal	[in] Variable, welche die Weite der Spalte Event Datum in der Listenansicht setzt.

Rückgabe Werte

S_OK
Funktion wurde erfolgreich aufgerufen.

Visual Basic Beispielcode

```
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 [▶ 70]

Diese Eigenschaft setzt oder gibt die Weite der Spalte Event Zeit zurück.

Property get

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

Parameter

pVal

[out, retval] Pointer auf eine Variable, welche die Weite der Spalte Event Zeit in der Listenansicht erhält.

Rückgabe Werte

S_OK
Funktion wurde erfolgreich aufgerufen.
E_POINTER
pVal war kein gültiger Pointer.

Property let

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

Parameter

newVal	[in] Variable, welche die Weite der Spalte Event Zeit in der Listenansicht setzt.

Rückgabe Werte

S_OK
Funktion wurde erfolgreich aufgerufen.

Visual Basic Beispielcode

```
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 [▶ 70]

Diese Eigenschaft setzt oder gibt die Weite der Spalte Event Id zurück.

Property get

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

Parameter

pVal

[out, retval] Pointer auf eine Variable, welche die Weite der Spalte Event Id in der Listenansicht erhält.

Rückgabe Werte

S_OK
Funktion wurde erfolgreich aufgerufen.
E_POINTER
pVal war kein gültiger Pointer.

Property let

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

Parameter

newVal	[in] Variable, welche die Weite der Spalte Event Id in der Listenansicht setzt.

Rückgabe Werte

S_OK
Funktion wurde erfolgreich aufgerufen.

Visual Basic Beispielcode

```
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

ITcEventView [▶ 70]

Diese Eigenschaft setzt oder gibt die Weite der Spalte Computername zurück.

Property get

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

Parameter

pVal

[out, retval] Pointer auf eine Variable, welche die Weite der Spalte Computername in der Listenansicht erhält.

Rückgabe Werte

S_OK
Funktion wurde erfolgreich aufgerufen.
E_POINTER
pVal war kein gültiger Pointer.

Property let

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

Parameter

newVal	[in] Variable, welche die Weite der Spalte Computername in der Listenansicht setzt.

Rückgabe Werte

S_OK
Funktion wurde erfolgreich aufgerufen.

Visual Basic Beispielcode

```
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

ITcEventView [▶ 70]::ColWidthMsg

Diese Eigenschaft setzt oder gibt die Weite der Spalte Event Meldung zurück.

Property get

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

Parameter

pVal

[out, retval] Pointer auf eine Variable, welche die Weite der Spalte Event Meldung in der Listenansicht erhält.

Rückgabe Werte

S_OK
Funktion wurde erfolgreich aufgerufen.
E_POINTER
pVal war kein gültiger Pointer.

Property let

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

Parameter

newVal	[in] Variable, welche die Weite der Spalte Event Meldung in der Listenansicht setzt.

Rückgabe Werte

S_OK
Funktion wurde erfolgreich aufgerufen.

Visual Basic Beispielcode

```
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 [▶ 70]

Diese Eigenschaft aktiviert/deaktiviert die Spalte, die die Event Klasse anzeigt oder gibt den Status zurück.

Property get

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

Parameter

pVal

[out, retval] Pointer auf eine VARIANT_BOOL Variable, die den sichtbaren Status der Spalte Event Klasse in der Listenansicht erhält. Der Status ist sichtbar, wenn die Variable VARIANT_TRUE (TRUE) ist, und unsichtbar, wenn die Variable VARIANT_FALSE (FALSE) ist.

Rückgabe Werte

S_OK
Funktion wurde erfolgreich aufgerufen.
E_POINTER

pVal war kein gültiger Pointer.

Property let

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

Parameter

newVal

[in] Variable, die den sichtbaren Status der Spalte Event Klasse in der Listenansicht setzt. Der Status ist sichtbar, wenn die Variable VARIANT_TRUE (TRUE) ist, und unsichtbar, wenn die Variable VARIANT_FALSE (FALSE) ist.

Rückgabe Werte

S_OK

Funktion wurde erfolgreich aufgerufen.

Visual Basic Beispielcode

```
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 [▶ 70]

Diese Eigenschaft aktiviert/deaktiviert die Spalte, die den Bestätigungsstatus anzeigt oder gibt den Status zurück.

Property get

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

Parameter

pVal

[out, retval] Pointer auf eine VARIANT_BOOL Variable, die den sichtbaren Status der Spalte Bestätigungsstatus in der Listenansicht erhält. Der Status ist sichtbar, wenn die Variable VARIANT_TRUE (TRUE) ist, und unsichtbar, wenn die Variable VARIANT_FALSE (FALSE) ist.

Rückgabe Werte

S_OK

Funktion wurde erfolgreich aufgerufen.

E_POINTER

pVal war kein gültiger Pointer.

Property let

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

Parameter

newVal

[in] Variable, die den sichtbaren Status der Spalte Bestätigungsstatus in der Listenansicht setzt. Der Status ist sichtbar, wenn die Variable VARIANT_TRUE (TRUE) ist, und unsichtbar, wenn die Variable VARIANT_FALSE (FALSE) ist.

Rückgabe Werte

S_OK

Funktion wurde erfolgreich aufgerufen.

Visual Basic Beispielcode

```
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 [▶ 70]

Diese Eigenschaft aktiviert/deaktiviert die Spalte, die den Event Status anzeigt oder gibt den Status zurück.

Property get

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

Parameter

pVal

[out, retval] Pointer auf eine VARIANT_BOOL Variable, die den sichtbaren Status der Spalte Event Status in der Listenansicht erhält. Der Status ist sichtbar, wenn die Variable VARIANT_TRUE (TRUE) ist, und unsichtbar, wenn die Variable VARIANT_FALSE (FALSE) ist.

Rückgabe Werte

S_OK

Funktion wurde erfolgreich aufgerufen.

E_POINTER

pVal war kein gültiger Pointer.

Property let

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

Parameter

newVal

[in] Variable, die den sichtbaren Status der Spalte Event Status in der Listenansicht setzt. Der Status ist sichtbar, wenn die Variable VARIANT_TRUE (TRUE) ist, und unsichtbar, wenn die Variable VARIANT_FALSE (FALSE) ist.

Rückgabe Werte

S_OK

Funktion wurde erfolgreich aufgerufen.

Visual Basic Beispielcode

```
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

ITcEventView [▶ 70]

Diese Eigenschaft aktiviert/deaktiviert die Spalte, die die Event Source anzeigt oder gibt den Status zurück.

Property get

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

Parameter

pVal

[out, retval] Pointer auf eine VARIANT_BOOL Variable, die den sichtbaren Status der Spalte Event Source in der Listenansicht erhält. Der Status ist sichtbar, wenn die Variable VARIANT_TRUE (TRUE) ist, und unsichtbar, wenn die Variable VARIANT_FALSE (FALSE) ist.

Rückgabe Werte

S_OK

Funktion wurde erfolgreich aufgerufen.

E_POINTER

pVal war kein gültiger Pointer.

Property let

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

Parameter

newVal

[in] Variable, die den sichtbaren Status der Spalte Event Source in der Listenansicht setzt. Der Status ist sichtbar, wenn die Variable VARIANT_TRUE (TRUE) ist, und unsichtbar, wenn die Variable VARIANT_FALSE (FALSE) ist.

Rückgabe Werte

S_OK

Funktion wurde erfolgreich aufgerufen.

Visual Basic Beispielcode

```
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 [▶ 70]

Diese Eigenschaft aktiviert/deaktiviert die Spalte, die das Event Datum anzeigt oder gibt den Status zurück.

Property get

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

Parameter

pVal

[out, retval] Pointer auf eine VARIANT_BOOL Variable, die den sichtbaren Status der Spalte Event Datum in der Listenansicht erhält. Der Status ist sichtbar, wenn die Variable VARIANT_TRUE (TRUE) ist, und unsichtbar, wenn die Variable VARIANT_FALSE (FALSE) ist.

Rückgabe Werte

S_OK

Funktion wurde erfolgreich aufgerufen.

E_POINTER

pVal war kein gültiger Pointer.

Property let

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

Parameter

newVal

[in] Variable, die den sichtbaren Status der Spalte Event Datum in der Listenansicht setzt. Der Status ist sichtbar, wenn die Variable VARIANT_TRUE (TRUE) ist, und unsichtbar, wenn die Variable VARIANT_FALSE (FALSE) ist.

Rückgabe Werte

S_OK

Funktion wurde erfolgreich aufgerufen.

Visual Basic Beispielcode

```
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 [▶ 70]

Diese Eigenschaft aktiviert/deaktiviert die Spalte, die die Event Zeit anzeigt oder gibt den Status zurück.

Property get

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

Parameter

pVal

[out, retval] Pointer auf eine VARIANT_BOOL Variable, die den sichtbaren Status der Spalte Event Zeit in der Listenansicht erhält. Der Status ist sichtbar, wenn die Variable VARIANT_TRUE (TRUE) ist, und unsichtbar, wenn die Variable VARIANT_FALSE (FALSE) ist.

Rückgabe Werte

S_OK

Funktion wurde erfolgreich aufgerufen.

E_POINTER

pVal war kein gültiger Pointer.

Property let

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

Parameter

newVal

[in] Variable, die den sichtbaren Status der Spalte Event Zeit in der Listenansicht setzt. Der Status ist sichtbar, wenn die Variable VARIANT_TRUE (TRUE) ist, und unsichtbar, wenn die Variable VARIANT_FALSE (FALSE) ist.

Rückgabe Werte

S_OK

Funktion wurde erfolgreich aufgerufen.

Visual Basic Beispielcode

```
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 [▶ 70]

Diese Eigenschaft aktiviert/deaktiviert die Spalte, die den Event Meldungstext anzeigt oder gibt den Status zurück.

Property get

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

Parameter

pVal

[out, retval] Pointer auf eine VARIANT_BOOL Variable, die den sichtbaren Status der Spalte Event Meldungstext in der Listenansicht erhält. Der Status ist sichtbar, wenn die Variable VARIANT_TRUE (TRUE) ist, und unsichtbar, wenn die Variable VARIANT_FALSE (FALSE) ist.

Rückgabe Werte

S_OK

Funktion wurde erfolgreich aufgerufen.

E_POINTER

pVal war kein gültiger Pointer.

Property let

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

Parameter

newVal

[in] Variable, die den sichtbaren Status der Spalte Event Meldungstext in der Listenansicht setzt. Der Status ist sichtbar, wenn die Variable VARIANT_TRUE (TRUE) ist, und unsichtbar, wenn die Variable VARIANT_FALSE (FALSE) ist.

Rückgabe Werte

S_OK

Funktion wurde erfolgreich aufgerufen.

Visual Basic Beispielcode

```
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 [▶ 70]

Diese Methode gibt den Index des ausgewählten Events zurück.. Die Auswahl wird durch einen Klick mit der linken Maustaste auf den Eventeintrag in der Listenansicht getroffen, oder durch die Methode SelectEvent [▶ 77].

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

Parameter

pSelEvent

[out, retval] Pointer auf eine Variable, die den Index des ausgewählten Events erhält. Der Index in der Listenansicht startet oben mit dem Index 0.

Rückgabe Werte

S_OK

Funktion wurde erfolgreich aufgerufen.

E_POINTER

pSelEvent war kein gültiger Pointer.

Visual Basic Beispielcode

```
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
```

7.2.1.44 getExtendedInfo

ITcEventView [▶ 70]

Diese Methode liefert erweiterte Informationen zürück, die mit dem gewählten Event assoziert sind.

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

Parameter

item

[in] Index des Event Items in dem ListView Control.

message

[out, retval] Zeiger auf eine Variable, die die erweiterten Informationen enthält, die mit diesem Event verknüpft sind.

Return Values

S_OK

Funktion wurde erfolgreich aufgerufen.

S_FALSE

Funktion wurde erfolgreich aufgerufen, aber keine erweiterte Information für das gewählte Event gefunden.

E_POINTER

Message war kein gültiger Pointer.

Visual Basic Beispielcode

```
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 [▶ 70]::Mode

Diese Eigenschaft setzt oder gibt den Anzeigemodus des TcEventViewers zurück. Die Anzeigemodi sind durch das Enum TCEVENTVIEW_DISPLAYMODE [▶ 115] definiert.

Property get

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

Parameters

pVal

[out, retval] Pointer auf eine Variable, die den Anzeigemodus vom Typ TCEVENTVIEW_DISPLAYMODE erhält.

Rückgabe Werte

S_OK

Funktion wurde erfolgreich aufgerufen.

E_POINTER

pVal war kein gültiger Pointer.

Property let

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

Parameter

newVal

[in] Variable, die den Anzeigemodus des Typs TCEVENTVIEW_DISPLAYMODE setzt.

Rückgabe Werte

S_OK

Funktion wurde erfolgreich aufgerufen.

Visual Basic Beispielcode

```
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 [▶ 70]

Diese Eigenschaft setzt oder gibt die maximale Anzahl der angezeigten Events im TcEventViewer zurück.

Property get

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

Parameter

pVal

[out, retval] Pointer auf eine Variable, die die maximale Anzahle der Events erhält.

Rückgabe Werte

S_OK
Funktion wurde erfolgreich aufgerufen.
E_POINTER
pVal war kein gültiger Pointer.

Property let

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

Parameters

newVal

[in] Variable, die die maximale Anzahl der Events setzt.

Rückgabe Werte

S_OK
Funktion wurde erfolgreich aufgerufen.

Visual Basic Beispielcode

```
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
```

Anmerkungen

Wenn **MaxEvent** auf 0 gesetzt ist, werden alle Events angezeigt. Die Reduzierung der maximalen Eventanzahl kann die Leistung des TcEventView erhöhen.

7.2.1.47 putFont

ITcEventView [▶ 70]

Methode zum Ändern der Schriftart für die angezeigten Events.

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

Parameter

faceName

[in] Name der Schriftart.

size

[in] Grösse der Schrift.

flags

[in] Verknüpfung von einem oder mehreren der folgenden Werte:

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

Rückgabe Werte

S_OK
Funktion wurde erfolgreich aufgerufen.

Remarks

Die putFont Methode entspricht nicht den Standard COM Interfaces. Unter Sprachen wie Visual Basic ist sie nicht benutzbar.

7.2.1.48 SetFilter

ITcEventView [▶ 70]

Mit dieser Methode kann ein Display-Filter eingeschaltet werden. Der TcEventViewer wird dann nur noch die ausgewählten Events anzeigen - (z.B. nur Alarms oder Events von einer bestimmten Quelle)

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

Parameters

Filter

[in] Art des Filters. Dieser Wert ist durch das **TCEVENTVIEW_FILTER** Enum definiert

Value

[in] Wert, welcher gefiltert wird. Die folgende Tabelle zeigt, welcher Vartype für welchen Filter erwartet wird.

Filter	Vartype
TCEVENTVIEW_FILTER_NONE	Wird Ignoriert.
TCEVENTVIEW_FILTER_CLASS **TCEVENTVIEW_FILTER_SOURCEID** **TCEVENTVIEW_FILTER_ID**	Werte, die sich zu Long (VT_I4) konvertieren lassen.
TCEVENTVIEW_FILTER_SOURCENAME	String (VT_BSTR).

Return Values

S_OK

Funktion erfolgreich ausgeführt.

E_INVALIDARG

Entweder ist der Filter Typ ungültig, oder Value lässt sich nicht in einen entsprechenden Wert konvertieren.

7.2.1.49 SortMode

ITcEventView [▶ 70]

Diese Eigenschaft stellt das Sortierverhalten ein. Das geschieht auch beim Klicken in den Columnheader. Wenn der Columnheader deaktiviert ist, oder das TcEventView mit einem bestimmten Sortierverhalten initialisiert werden soll, kann diese Methode benutzt werden.

Property get

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

Parameter

pVal

[out, retval] Pointer auf eine Variable, die das aktuelle Sortierverhalten enthält. Sortiermodi sind in dem Enum TC_SORTMODE [▶ 115] definiert.

S_OK

Funktion wurde erfolgreich aufgerufen.

E_POINTER

pVal war kein gültiger Pointer.

Property set

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

Parameters

newVal

[in] Eine Variable, die den neuen Sortiermodus enthält. Sortiermodi sind in dem Enum TC_SORTMODE [▶ 115] definiert.

Rückgabe Werte

S_OK

Funktion wurde erfolgreich aufgerufen.

Visual Basic Beispielcode

```
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
```

Sehen Sie dazu auch

TC_SORTMODE [▶ 115]

7.2.1.50 UseExternalLanguageResource

ITcEventView [▶ 70]

Diese Eigenschaft setzt, ob extern definierte oder die standard String Tables genutzt werden. Der TcEventViewer wählt die Sprach Ressourcen entsprechend der aktuellen LCID des Threads aus. Es werden zur Zeit die Sprachen Deutsch, English(U.S.), Französisch, Italienisch und Spanisch unterstützt. Wenn das HMI in einer anderen Sprache angezeigt werden soll, setzen Sie diese Eigenschaft auf true. Der EventViewer wird dann in dem Verzeichnis '%TwinCat/Resource' nach der Datei 'TcEventViewLangRes.xml' suchen. In dieser Datei können beliebig andere Sprachen definiert werden. Diese Beispiel Konfiguration zeigt, wie die XML Ressource aussehen muss.

Property get

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

Parameter

pVal

[out, retval] Pointer auf eine Variable, die den aktuellen Status erhält.

Rückgabe Werte

S_OK

Funktion wurde erfolgreich aufgerufen.

E_POINTER

pVal war kein gültiger Pointer.

Property set

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

Parameters

newVal

[in] Eine Variable, die den neuen Status enthält.

Rückgabe Werte

S_OK

Funktion wurde erfolgreich aufgerufen.

Visual Basic Beispielcode

```
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

Die Schnittstelle ITcEventViewLight wird zur Kontrolle der Eigenschaften der Light Version des EventViewers verwendet. Sie leitet sich von **IDispatch** ab, so dass sie von allen Sprachen verwendet werden kann, die die COM Automation Schnittstellen unterstützen. IDispatch selbst leitet sich von IUnknown ab.

Tab. 13: Methoden und Eigenschaften in

IUnknown Methoden	Beschreibung
QueryInterface	Gibt einen Pointer zur angefragten Schnittstelle zurück.
AddRef	Inkrementiert den Referenzzähler.
Release	Dekrementiert den Referenzzähler.

IDispatch Methoden	Beschreibung
GetIDsOfNames	Mappt einen einzelnen Teilnehmernamen und einen Satz optionaler Parameternamen mit einem entsprechenden Satz gesendeter Integer Identifier (DISPIDs). Diese können dann als Folgeabrufe an **IDispatch::Invoke** benutzt werden.
GetTypeInfo	Ruft die Typ Informationen für ein Objekt ab.
GetTypeInfoCount	Ruft die Anzahl der Typ Informations-Schnittstellen ab, die ein Objekt zur Verfügung stellt, entweder 0 oder 1.
Invoke	Stellt den Zugriff auf Eigenschaften und Methoden, ausgestellt durch ein Objekt, zur Verfügung.

ITcEventView Methoden und Eigenschaften	Beschreibung
ShowLoggedEvents [▶ 72]	Zeigt die logged Events an.
ShowActiveEvents [▶ 72]	Zeigt die aktiven Events an.
ConfirmEvent [▶ 73]	Bestätigt ein Event.
ResetEvent [▶ 73]	Setzt ein Event zurück.
LangId [▶ 74]	Setzt die Sprache, die zur Anzeige der Event Meldung und des Source Namens verwendet wird, zurück.
DisableMenuItems [▶ 75]	Setzt oder gibt den abgeschalteten Status des Kontextmenüs zurück.
SelectEvent [▶ 77]	Markiert ein Event als ausgewählt.
AddConnection [▶ 78]	Addiert eine Remoteverbindung zu einem EventLogger.
DeleteConnection [▶ 79]	Entfernt eine Remoteverbindung zu einem EventLogger.
DisplayComputerName [▶ 79]	Aktiviert/Deaktiviert die Spalte, die den Computernamen anzeigt oder den Status zurückgibt.

ITcEventView Methoden und Eigenschaften	Beschreibung
DisplayEventId [▶ 80]	Aktiviert/Deaktiviert die Spalte, die die Event Id anzeigt oder den Status zurückgibt.
ColWidthClass [▶ 93]	Setzt oder gibt die Weite der Spalte Event Klasse zurück.
ColWidthMustCon [▶ 94]	Setzt oder gibt die Weite der Spalte Bestätigungsstatus (confirm state) zurück.
ColWidthState [▶ 94]	Setzt oder gibt die Weite der Spalte Event Status zurück.
ColWidthSource [▶ 95]	Setzt oder gibt die Weite der Spalte Event Source zurück.
ColWidthDate [▶ 96]	Setzt oder gibt die Weite der Spalte Event Datum zurück.
ColWidthTime [▶ 97]	Setzt oder gibt die Weite der Spalte Event Zeit zurück.
ColWidthEventId [▶ 98]	Setzt oder gibt die Weite der Spalte Event Id zurück.
ColWidthComputer [▶ 99]	Setzt oder gibt die Weite der Spalte Computername zurück.
ColWidthMsg [▶ 99]	Setzt oder gibt die Weite der Spalte Event Meldung zurück.
DisplayClass [▶ 100]	Aktiviert/Deaktiviert die Spalte, die die Event Klasse anzeigt, oder gibt den Status zurück.
DisplayMustCon [▶ 101]	Aktiviert/Deaktiviert die Spalte, die den Bestätigungsstatus (confirm State) anzeigt, oder gibt den Status zurück.
DisplayState [▶ 102]	Aktiviert/Deaktiviert die Spalte, die den Eventstatus anzeigt, oder gibt den Status zurück.
DisplaySource [▶ 103]	Aktiviert/Deaktiviert die Spalte, die den Sourcenamen anzeigt, oder gibt den Status zurück.
DisplayDate [▶ 104]	Aktiviert/Deaktiviert die Spalte, die das Eventdatum anzeigt, oder gibt den Status zurück.
DisplayTime [▶ 105]	Aktiviert/Deaktiviert die Spalte, die die Eventzeit anzeigt, oder gibt den Status zurück.
DisplayMsg [▶ 106]	Aktiviert/Deaktiviert die Spalte, die die Eventmeldung anzeigt, oder gibt den Status zurück.
GetSelectedEvent [▶ 107]	Gibt den Index des Events zurück, dass vom Benutzer ausgewählt wurde.
Mode [▶ 108]	Setzt oder gibt den Anzeigemodus des TcEventViewers zurück.
MaxEvents [▶ 109]	Setzt oder gibt die maximale Anzahl der angezeigten Events zurück.
DisplayTitlebar [▶ 81]	Aktiviert/Deaktiviert den Columnheader.
DisableContextMenu [▶ 76]	Aktiviert/Deaktiviert das Kontext Menu.
DisableScrollbars [▶ 77]	Aktiviert/Deaktiviert die Scrollbars.

7.3 Enums

7.3.1 TCEVENTVIEW_CONTEXTMENUFLAGS

Dieses Enum wird zur Abschaltung von Funktionen im Kontextmenü verwendet.

```
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_DISABLEALL =0xFFFF
}
```

Parameter

Eintrag	Beschreibung
TCEVENTVIEW_NOTHINGDISABLED	Alle Funktionen des Kontextmenüs sind aktiviert.
TCEVENTVIEW_DISABLECONFIRM	Entfernt bestätigte Funktionen aus dem Kontextmenü.
TCEVENTVIEW_DISABLERESET	Entfernt Reset-Funktioinen aus dem Kontextmenü.
TCEVENTVIEW_DISABLEDETAILS	Entfernt Detail-Funktionen aus dem Kontextmenü.
TCEVENTVIEW_DISABLEDOCLINKS	Entfernt verlinkte Dokument-Funktionen aus dem Kontextmenü.
TCEVENTVIEW_DISABLECLEARACTIVEALL	Entfernt alle aktiven Funktionen aus dem Kontektmenü.
TCEVENTVIEW_DISABLECLEARLOGGEDALL	Entfernt alle gelöschten und geloggten Funktionen aus dem Kontextmenü.
TCEVENTVIEW_DISABLEALL	Deaktiviert alle Funktionen des Kontextmenüs.

7.3.2 TCEVENTVIEW_DISPLAYMODE

Dieses Enum wird zu Einstellung des Anzeigemodus des TcEventView verwendet.

```
enum TCEVENTVIEW_DISPLAYMODE
{
    TCEVENTVIEW_MODE_ACTVIVE =0x00000000,
    TCEVENTVIEW_MODE_LOGGEGED =0x00000001,
    TCEVENTVIEW_MODE_SNAPSHOT =0x00000002,
    TCEVENTVIEW_MODE_TRACE =0x00000003
}
```

Parameter

Eintrag	Beschreibung
TCEVENTVIEW_MODE_ACTVIVE	Anzeige aktiver Events. Die TcEventView wird synchron zur Liste der aktiven Events des angeschlossenen TcEventLog Servers gehalten.
TCEVENTVIEW_MODE_LOGGEGED	Anzeige aktiver Events. Die TcEventView wird synchron zur Liste der logged Events des angeschlossenen TcEventLog Servers gehalten.
TCEVENTVIEW_MODE_SNAPSHOT	Gibt eine Momentaufnahme der logged oder aktiven Events zurück. Die TcEventView wird nicht synchron zur Liste der aktiven Events des angeschlossenen TcEventLog Servers gehalten. Eine neue Momentaufnahme der aktiven Events wird durch einen Aufruf des ITcEventView::ShowActiveEvents [▶ 72] angezeigt. Eine neue Momentaufnahme der logged Events wird durch einen Aufruf des ITcEventView::ShowLoggedEvents [▶ 72] angezeigt.
TCEVENTVIEW_MODE_TRACE	Neu auftretende Events werden zum TcEventView addiert; kein Event wird gelöscht.

7.3.3 TC_SORTMODE

Dieses Enum wird benutzt, um die Reihenfolge einzustellen, in welcher angezeigte Events sortiert werden.

```
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

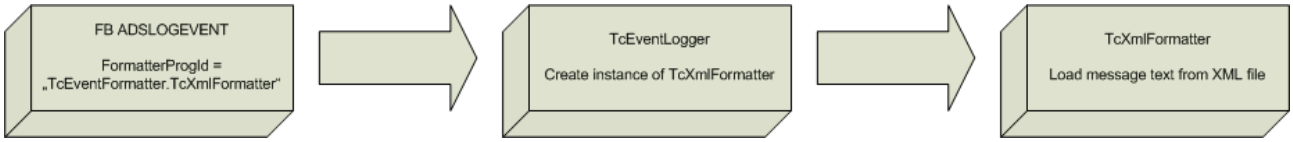
Item	Beschreibung
SORT_BY_CLASS_ASCENDING	Events werden aufsteigend nach ihrer Klasse sortiert.
SORT_BY_CLASS_DESCENDING	Events werden absteigend nach ihrer Klasse sortiert.
SORT_BY_SOURCE_ASCENDING	Events werden aufsteigend nach ihrer Quelle sortiert.
SORT_BY_SOURCE_DESCENDING	Events werden absteigend nach ihrer Quelle sortiert.
SORT_BY_DATE_ASCENDING	Events werden aufsteigend nach Datum sortiert.
SORT_BY_DATE_DESCENDING	Events werden absteigend nach Datum sortiert.
SORT_BY_TIME_ASCENDING	Events werden aufsteigend nach Zeit sortiert.
SORT_BY_TIME_DESCENDING	Events werden absteigend nach Zeit sortiert.
SORT_BY_EVT_ID_ASCENDING	Events werden aufsteigend nach Id sortiert.
SORT_BY_EVT_ID_DESCENDING	Events werden absteigend nach Id sortiert.
SORT_BY_COMPUTER_ASCENDING	Events werden aufsteigend nach Computer sortiert.
SORT_BY_COMPUTER_DESCENDING	Events werden absteigend nach Computer sortiert.

8 Event Formatierung

Der EventFormatter wird benutzt, um Meldungs- und sprachspezifisch Texte aus einer Datenbank zu laden.

Beckhoff bietet standard Formatter Implementierungen. Die EventFormatter API ist dokumentiert, so dass kundenspezifische Formatter implementiert werden können um proprietäre Formate zu unterstützen.

Wen ein Event in der SPS ausgelöst wird muss eine Formatter ProgId angegeben werden. Der TcEvenlogger versucht nun eine Instanz des COM Objectes mit dieser ProgId zu erzeugen. Wenn dieses Objekt existiert und das ITcLogFormatterC [▶ 117] Interface implementiert wird der Meldungstest nun an diesem Objekt abgefragt.



TwinCAT standard formatters:

Voraussetzungen

Formatter	Description	Supported Platforms
TcEventFormatter	Lädt Meldungstexte aus dem TwinCAT Project Storage (*.tps).	NT/2000/XP/Vista.
TcXMLFormatter [▶ 123]	Lädt Meldungstexte aus XML Dateien.	NT/2000/XP/Vista/CE.

8.1 Interfaces

8.1.1 ITcLogFormatterC

Tab. 14: 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

ITcLogFormatterC Methods	Description
GetFormatString [▶ 117]	Gibt den Formatstring zurück
GetCompleteString [▶ 118]	Gibt einen formatierten Meldungstext in einer bestimmten Sprache zurück
SetFormatString [▶ 120]	Setzt den Formatstring für ein bestimmtes Event
GetClassPrio [▶ 120]	Gibt die Event Klasse und Priorität zurück
EnumDocLinks [▶ 121]	Gibt einen Enumerator zurück
GetSourceName [▶ 122]	Gibt den Sourcename in einer bestimmten Sprache zurück.

8.1.1.1 GetFormatString

Diese Methode gibt den Formatstring zurück wie er von **TcEvent.DataFormatStrAddress** zugeordnet wurde, als das Event vom SPS Funktionsblock **ADSLOGEVENT** ausgegeben wurde.

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

Parameter

nEventId

[in] Variable, die die Event Id darstellt.

nSrcId

[in] Variable, die die Source Id darstellt.

langId

[in] Die Sprach-Id der angefragten Sprache. Die gewünschte Sprache sollte mit *LCIDs markiert werden. Die nächste Tabelle zeigt ein Beispiel für einige Sprach-Ids. In der Konfiguration des Event Formatters werden die Sprachen mit der gleichen Sprach-Id gekennzeichnet.

LCID	Beschreibung
1031	Deutsch
1033	US Englisch
1034	Spanisch
1036	Französisch

szFormat

[out, retval] Pointer auf einen BSTR String, der das Format zurückgibt.

Rückgabe Werte

S_OK

Funktion wurde erfolgreich aufgerufen.

E_POINTER

szFormat war kein gültiger Pointer.

E_NOTIMPL

Die Methode ist nicht implementiert.

Anmerkungen

Der TcXmlFormatter unterstützt diese Methode nicht.

*LCID: weitere Informationen finden Sie in der MSDN Library

8.1.1.2 GetCompleteString

Die Methode GetCompleteString gibt den formatierten Meldungsstring der angefragten Sprache zurück. Diese Methode wird von der TcEvent [▶ 129] Objekt Methode ITcEvent [▶ 159]::GetMsgString [▶ 160] aufgerufen.

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

Parameter

nEventId

[in] Variable, die die Event Id darstellt.

nSrcId

[in] Variable, die die Source Id darstellt.

langId

[in] Die Sprach-Id der angefragten Sprache. Die gewünschte Sprache sollte mit *LCIDs markiert werden. Die nächste Tabelle zeigt ein Beispiel für einige Sprach-Ids. In der Konfiguration des Event Formatters werden die Sprachen mit der gleichen Sprach-Id gekennzeichnet.

LCID	Beschreibung
1031	Deutsch
1033	US Englisch
1034	Spanisch
1036	Französisch

pEventHead

[in] Pointer auf ein Objekt vom Typ TcEventHeader [▶ 184]. Das Objekt stellt die Eventkonfiguration dar.

eventData

[in] Pointer auf ein Safearray, das die Event Argumente darstellt. In der Konfiguration des Event Formatters ist definiert, wie die Event Argumente in der angezeigten Event Meldung platziert werden.

Die folgenden Datentypen werden von Standard Formatter unterstützt:

int (16bit), long (32bit), float(32bit), double(64bit), string(BSTR). Die Eventdaten werden zugeordnet, wenn das Event ausgegeben wird.

msg

[out, retval] PPointer auf einen BSTR String der den formatierten String für die angefragte Sprache zurückgibt

Rückgabe Werte

S_OK

Funktion wurde erfolgreich aufgerufen.

E_POINTER

szFormat war kein gültiger Pointer.

E_NOTIMPL

Die Methode ist nicht implementiert.

Anmerkungen

Die meisten Standard Formatter, wie der XML basierende Formatter (TcEventFormatter.TcXmlFormatter), versuchen, einen String für die Default-Sprache zurückzugeben, falls die angefragte Sprache nicht in der Konfiguration des Formatters existiert.

*LCID: weitere Informationen finden Sie in der MSDN Library.

8.1.1.3 SetFormatString

ITcLogFormatterC [▶ 117]

Diese Methode ordnet einen neuen Formatstring an, wie er von **TcEvent.DataFormatStrAddressz**ugeordnet wurde, als das Event vom SPS Funktionsblock **ADSLOGEVENT** ausgegeben wurde.

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

Parameter

nEventId

[in] Variable, die die Event Id darstellt.

nSrcId

[in] Variable, die die Source Id darstellt.

langId

[in] Die Sprach-Id der angefragten Sprache. Die gewünschte Sprache sollte mit *LCIDs markiert werden. Die nächste Tabelle zeigt ein Beispiel für einige Sprach-Ids. In der Konfiguration des Event Formatters werden die Sprachen mit der gleichen Sprach-Id gekennzeichnet.

LCID	Beschreibung
1031	Deutsch
1033	US Englisch
1034	Spanisch
1036	Französisch

msg

[in] Ein BSTR String, der den Formatstring enthält.

Rückgabe Werte

S_OK

Funktion wurde erfolgreich aufgerufen.

E_NOTIMPL

Die Methode ist nicht implementiert.

Anmerkungen

Der TcXmlFormatter unterstützt diese Methode nicht.

*LCID: weitere Informationen finden Sie in der MSDN Library

8.1.1.4 GetClassPrio

ITcLogFormatterC

Diese Methode gibt die Event Klasse und Priorität zurück. Dies wird von Alarmen benutzt, die vom SPS Funktionsblock ADSLOGEVENT ausgegeben werden, indem das Flag von FALSE auf TRUE gesetzt wird (oder eines anderen ADS Geräts), oder durch den Aufruf einer Report Funktion wie IT [▶ 139]cEventLogC [▶ 139]::ReportEvent [▶ 139],ITcEventC3 [▶ 142]::ReportEventEx [▶ 144] oder ITcEventLog [▶ 130]::ReportEvent [▶ 131] mit den TcEventFlags [▶ 182] auf TCEVENTFLAG_PRIOCLASS gesetzt.

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

pClass

[in] Pointer auf einen TcEventClass [▶ 181] Wert, der die Eventklasse erhält.

pPriority

[out, retval] Pointer auf eine long Variable, die die Event Priorität erhält. Die Event Priority wird durch den Typ TcEventPriority [▶ 183] definiert.

Rückgabe Werte

S_OK

Funktion wurde erfolgreich aufgerufen.

E_POINTER

pClass oder Priorität sind keine gültigen Pointer.

8.1.1.5 EnumDocLinks

Die Methode gibt ein Aufzählungsobjekt zurück, das zur Wiederholung aufgelisteter Events verwendet wird. Dokumentverknüpfungen sind der Pfad und die Dateinamen der Dokumente ,wie HTML Seiten und Bilder, die eine genauere Beschreibung über einen einzelnen Alarm für die angewählte Sprache ausgeben. Diese Methode wird von der TcEvent [▶ 129] Objekt Methode ITcEvent [▶ 159]::EnumDocLinks [▶ 176] aufgerufen.

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

Parameter

nEventId

[in] Variable, die die Event Id darstellt.

nSrcId

[in] Variable, die die Source Id darstellt.

langId

[in] Die Sprach-Id der angefragten Sprache. Die gewünschte Sprache sollte mit *LCIDs markiert werden. Die nächste Tabelle zeigt ein Beispiel für einige Sprach-Ids. In der Konfiguration des Event Formatters werden die Sprachen mit der gleichen Sprach-Id gekennzeichnet.

LCID	Beschreibung
1031	Deutsch
1033	US Englisch
1034	Spanisch
1036	Französisch

ppEnum

[out, retval] Pointer auf den I ITcEnumEventDocLink [▶ 178] Schnittstellen-Pointer, der die Aufzählungsobjekte der Dokumentverknüpfungen erhält.

Rückgabe Werte

S_OK

Funktion wurde erfolgreich aufgerufen.

E_POINTER

ppEnum war kein gültiger Pointer.

Anmerkungen

*LCID: weitere Informationen finden Sie in der MSDN Library.

Sehen Sie dazu auch

ITcEnumEventDocLink [▶ 178]

8.1.1.6 GetSourceName

ITcLogFormatterC

Die Methode GetSourceName gibt einen formatierten Sourcenamen für die angefragte Sprache zurück. Diese Methode wird von der TcEvent [▶ 129] Objekt Methode ITcEvent [▶ 159]::SourceName [▶ 165] aufgerufen.

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

Parameter

nSrcId

[in] Variable, die die Event Source Id darstellt.

langId

[in] Die Sprach-Id der angefragten Sprache. Die gewünschte Sprache sollte mit *LCIDs markiert werden. Die nächste Tabelle zeigt ein Beispiel für einige Sprach-Ids. In der Konfiguration des Event Formatters werden die Sprachen mit der gleichen Sprach-Id gekennzeichnet.

LCID	Beschreibung
1031	Deutsch
1033	US Englisch
1034	Spanisch
1036	Französisch

szName

[out, retval]

Pointer auf einen BSTR String, der den formatierten String für den Event Sourcenamen zurückgibt.

Rückgabe Werte

S_OK

Funktion wurde erfolgreich aufgerufen.

E_POINTER

szName war kein gültiger Pointer.

Anmerkungen

Die meisten Standard Formatter, wie der XML basierende Formatter (TcEventFormatter.TcXmlFormatter), versuchen, einen String für die Default Sprache zurückzugeben, wenn die angefragte Sprache in der Konfiguration des Formatters nicht existiert.

*LCID: weitere Informationen finden Sie in der MSDN Library

8.2 TcXmlFormatter

Der TcXmlFormatter stellt die Textformatierungsfunktionen für TcEventLogger Events zur Verfügung. Er kann mehrere Sprachen bedienen und optional Event Argumente wie Integer, Floating Point Numbers und Strings in den ausgegebenen String einfügen. Die ausgegebenen Strings werden in einer XML-Datei eingerichtet. Der TcEventFormatter befindet sich in der TcEventFormatter.dll.

Datenbank

Der TcXmlFormatter benutzt XML files als Datenbank. Die Konfiguration erfolgt mit zwei verschiedenen XML Dateien:

1. In der Datei TcEventSourceLocation.xml [▶ 123] werden die Meldungs-Quellen definiert. Für jede Quelle wird hier eine Source ID und der Name der Meldungs-Konfiguration Datei angegeben.
2. Die Meldungen werden durch eine XmlEventConfguration [▶ 125] beschrieben.

Wenn Veränderungen an einer der XML Dateien durchgeführt wurden, muss der TcEventLogger neu gestartet (oder Die ITcServer Methode Reset() am Eventlogger aufgerufen) werden.

Um den Eventlogger neu zu starten

1) Stoppen Sie alle Clients, die den TcEventLogger verwenden, wie z.B. HMI, TcEventbar.

2) Stoppen Sie das TwinCAT System.

3) Starten Sie den Task Manager und stellen Sie sicher, dass das Programm TcEventLogger.exe nicht in der Liste der Prozesse existiert.

4) Starten Sie das TwinCAT System erneut. Die neue Eventkonfiguration ist jetzt gültig.

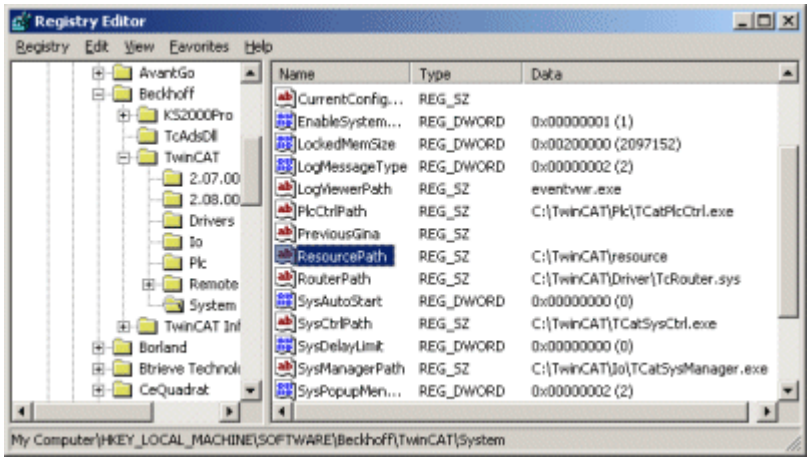
Falls der TcEventLogger nicht wie unter 3) beschrieben gestoppt werden kann, ist evtl. ein Neustart des Betriebssystems erforderlich.

8.2.1 TcEventSourceLocation

Die Datei **TcEventSourceLocation.xml** definiert den Standort der Event Konfiguration eines jeden Event Source für den TcXmlFormatter [▶ 123]**.** Die Datei befindet sich im TwinCAT Resource Verzeichnis.

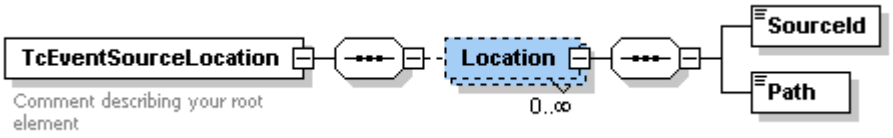
Der aktuelle Pfad des Resource Verzeichnis kann aus der Registry ausgelesen werden. Unter dem Key-Wert **HKEY_LOCAL_MACHINE\SOFTWARE\Beckhoff\TwinCAT\System\[ResourcePath]** befindet sich der gesuchte Pfad.

Der nächste Screenshot zeigt den Wert im Registry Editor **regedit.exe**.



Schema

Die Abbildung zeigt das XML Schema, das die Struktur der **TcEventSourceLocation** XML Datei beschreibt.



Die Tabelle beschreibt die Knoten des XML Dokuments:

Schnittstelle	Beschreibung
TcEventSourceLocation	Das einzige Rootelement. Es enthält eine Liste der Location Knoten von 0 bis unendlich.
Location	Location stellt den Ort der XML-Dateien mit Source Ids dar, die die XmlEventConfguration [▶ 125] enthält. Der Knoten hat zwei erforderliche Unterknoten: SourceId und Path
SourceId	Integer, der die Event Source Id darstellt.
Path	String, der den relativen Pfad oder Dateinamen der XmlEventConfguration [▶ 125] für eine Source Id enthält.

Beispiel

Der nächste Textauszug zeigt eine Beispielkonfiguration der **TcEventSourceLocation.xml** Datei.

```
<?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>
```

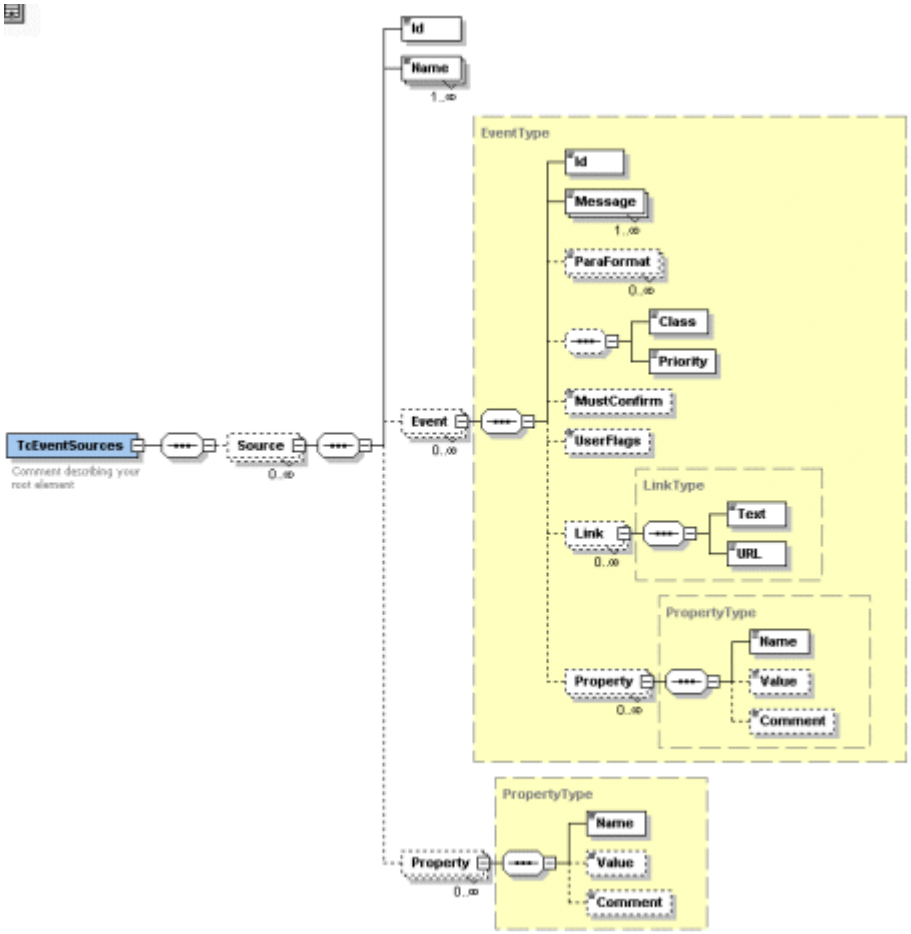
Das Beispiel zeigt den Ort der I/O Meldungen, NC Meldungen und benutzerdefinierter Meldungen mit der Source Id 1 in der Datei user.xml, die sich im gleichen Verzeichnis befindet.

8.2.2 XmlEventConfguration

Die Datei **XmlEventConfguration** definiert die Event Konfiguration einer Sourde Id für verschiedene Sprachen. Die **XmlEventConfguration** Source id und der Standortpfad müssen zur TcEventSourceLocation [▶ 123] als ein **Location** node (Knoten) eingetragen werden.

Schema

Der folgende Screenshot zeigt das XML Schema, das die Struktur des **XmlEventConfguration xml file** beschreibt.



Die folgende Tabelle beschreibt die Nodes des XML Dokuments

Schnittstelle	Beschreibung
TcEventSource	Das einzige Rootelement. Es enthält eine Liste der **Source** Knoten von 0 bis unendlich.
Source	Unterelement des **TcEventSource** Knoten. Der Knoten stellt die Konfiguration eines Event Source dar.. Die Unterelemente sind die **ids** des Sources.1 bis unendlich für die Source **Name** der verschiedenen Sprachen und jeweils 0 bis unendlich für die **Events** und **Properties**.
id	Unterelement des **Source** Knoten. Ein Integer, der die Event Source darstellt.
Name	Unterelement des **Source** Knoten. Der Event Source **Name** für die verschiedenen Sprachen. Die Sprache wird durch das Attribut ***LcId** definiert. Wenn kein ***LcId** Attribut existiert, wird die ***LcId** 1033 (US-Englisch) verwendet.
Event	Unterelement des **Source** Knoten. Stellt die Event Konfiguration für ein Event dar. Die Unterelemente sind Event **id**, ein **Message** String für die verschiedenen Sprachen von 1 bis unendlich, **ParaFormat** der Formatstring für die Eventdaten, das **MustConfirm** Flag, die **UserFlags**, Dokumentverknüpfungen von 0 bis unendlich und **Property** von 0 bis unendlich**.**
Property	Unterelement des **Source** Knoten. Die Property wird durch Name, Value and Comment dargestellt.

Schnittstelle	Beschreibung
id	Unterelement des **Event** Knoten. Ein Integer, der die Event Id darstellt.
Message	Unterelement des **Event** Knoten. Die Event Meldung für die verschiedenen Sprachen. Die Sprache wird durch das Attribut ***LcId** definiert. Wenn kein ***LcId** Attribut existiert, wird die ***LcId** 1033 (US-Englisch) verwendet. Das eventData wurde zugeordnet, als das Event vom SPS Funktionsblock **ADSLOGEVENT** durch Setzen des Flags von FALSE auf TRUE (oder eines anderen ADS Geräts) ausgegeben wurde, oder vom Aufruf einer Report Event Funktion wie ITcEventLogC::ReportEvent [▶ 139], ITcEventC3 ::ReportEventEx [▶ 144] oder ITcEventLog:: ReportEvent [▶ 131]. Es kann der Meldung durch **%1, %2, %,3 ...** hinzugefügt werden, wobei die Ziffer den Index des **eventData** Parameter darstellt.
ParaFormat	Unterelement des **Event** Knoten. Definiert eine printf Formatbeschreibung, die für einen Eintrag im **eventData** verwendet wird. Das Attribut **ParaNo** repräsentiert den Index des **eventData** für den der Formatstring verwendet wird.
UserFlags	Unterelement des **Event** Knoten. Ein Integer, der die benutzerdefinierten Flags definiert.
Link	Unterelement des **Event** Knoten. Eine Dokumentverknüpfung, die den Link auf ein Dokument darstellt, das weitere Informationen über ein Event für die verschiedenen Sprachen bereitstellt. Der Zugriff auf die DocLink Eigenschaft erfolgt über das TcEvent [▶ 129] Objekt ITcEvent [▶ 159]:: EnumDocLink [▶ 176] und ist im Kontextmenü des TcEventViewers [▶ 69] verfügbar. Die mit dem Dokument verknüpfte Sprache wird durch das Attribut ***LcId** definiert. Die Dokumentverknüpfung hat ein Unterelement Text, das den angezeigten Text für die ausgewählte ***LcId** und den URL Knoten darstellt. Der URL Knoten enthält die Url zum Dokument für die ausgewählte ***LcId,** die angezeigt werden soll.
Property	Unterelement des **Event** Knoten. Die Property wird durch Name, Value and Comment dargestellt.

Beispiel

Der Ausschnitt zeigt eine Beispielkonfiguration einer **XmlEventConfguration** Datei.

```
<?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 seconf 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>
```

Das Beispiel zeigt die Eventkonfiguration für zwei Events für die unterstützten Sprachen Deutsch und Englisch.

* Die folgende Tabelle zeigt einige häufig verwendete *LCIDs:

LCID	Beschreibung
1031	Deutsch
1033	US English
1034	Spanisch
1036	Französisch

*LCID: weitere Informationen finden Sie in der MSDN Library.

9 API

Der TcEventlogger ist ein Mechanismus zum Verwalten von Meldungen (z.B. aus der SPS). Er ist als TcCOM-Server realisiert und wird vom TwinCAT System gesteuert. Er bietet umfassende COM Interfaces zum Abfragen und Manipulieren von Meldungen:

Voraussetzungen

Schnittstelle	Beschreibung
ITcServer	Intern vom TwinCAT System verwendet, um diesen Server zu regeln.
ITcAdsAsync	Wird intern benutzt, um die ADS Funktionalität im TcEventLogger zu implementieren. Der TcEvent Logger bekommt die ADS Befehle über den ADS Port 110 durch seine Schnittstelle.
ITcEventLog [▶ 130]	Dies ist die Haupt-Schnittstelle des TcEventLoggers für COM Clients wie das HMI. Hauptfeatures um aktive und logged Alarme zu regeln und auch um neue Alarme hinzuzufügen und Bestehende zu löschen.
ITcEventLogC [▶ 139]	Diese Schnittstelle ist die Basis-Schnittstelle von ITcEventLogC3 und ITcEventLogC2.
ITcEventLogC2 [▶ 140]	Die Schnittstelle leitet sich von ITcEventLogC ab und ist die Basis-Schnittstelle von ITcEventLogC3
ITcEventLogC3 [▶ 142]	Dies ist eine anwendungsspezifische Schnittstelle des TcEventLoggers für COM Clients wie HMI, die anwendungsspezifische Schnittstellen unterstützen. Die Schnittstelle leitet sich von ITcEventLogC2 ab.
ITcEventLogEvents [▶ 146]	Callback Funktionen für Rückrufe an COM-Clients des TcEventloggers. Hierdurch können sich die Clients z.B. über das auftreten und zurücksetzen von Meldungen informieren lassen.
ITcEvent [▶ 159]	Meldungsobjekt. Mithilfe dieses Objektes können die Eigenschaften einer Meldung (z.B. Event- und Source ID) oder der Meldungstext abgefragt werden. Mithilfe dieses Objektes können Meldungen z.B. auch programmatisch zurückgesetzt werden.
ITcEnumEvents [▶ 152]	Mithilfe dieser Schnittstelle lassen sich die aktiven oder geloggten Meldungen enumerieren.
ITcEnumEventDocLink [▶ 178]	Mithilfe dieser Schnittstelle lassen sich die Doc Links einer Meldung enumerieren.

9.1 TcEventLogger Return Codes

Code (hex)	Description
0x00000000	Kein Fehler
0x98210001	Event Objekt ist signalisiert
0x98210002	Event Objekt ist quittiert
0x98210003	Event Objekt ist zurückgesetzt
0x98210004	Event Objekt ist quittiert aber nicht zurückgesetzt

Code (hex)	Description
0x98210005	Event braucht keine Quittierung
0x98210006	Reset oder Quittierung für ein unbekanntes Event
0x98210007	Event wird aus dem Storage geladen
0x98210008	Event Konfiguration konnte im Storage nicht gefunden werden
0x98210009	es konnte nicht auf den Storage zugegriffen werden

9.2 Klassen

9.2.1 TcEventLog

Die Hauptklasse des TcEvent Loggers. Stellt die Schnittstelle vom COM Client zur ADS Welt zur Verfügung. Diese Klasse implementiert die State Machine zum Handling des Verhaltens von Alarmen und dem Signal Transfer zwischen SPS und den COM Clients. Intern regelt sie die TcEventKeyToEventMap. Diese listet die aktiven Alarme und ein Objekt vom Typ IEnumSTATSTGPtr auf, welches die Liste der Streams enthält, die die Logged Alarme präsentieren. Bei Anfrage eines Methodenaufrufs an die ITEventLog Schnittstelle wird ein Aufzählungsobjekt für aktive Events und eines für Logged Events zurückgegeben.

Voraussetzungen

Schnittstelle	Beschreibung
ITcServer	Intern vom TwinCAT System verwendet, um diesen Server zu regeln.
ITcAdsAsync	Wird intern benutzt, um die ADS Funktionalität im TcEventLogger zu implementieren. Der TcEvent Logger bekommt die ADS Befehle über den ADS Port 110 durch seine Schnittstelle.
ITcEventLog [▶ 130]	Dies ist die Haupt-Schnittstelle des TcEventLoggers für COM Clients wie das HMI. Hauptfeatures, um aktive und logged Alarme zu regeln und auch um neue Alarme hinzuzufügen und Bestehende zu löschen.
ITcEventLogC3 [▶ 142]	Dies ist eine anwendungsspezifische Schnittstelle des TcEventLoggers für COM Clients wie HMI, die anwendungsspezifischen Schnittstellen unterstützen. Die Schnittstelle leitet sich von ITcEventLogC2 ab.
ITcEventLogC2 [▶ 140]	Die Schnittstelle leitet sich von ITcEventLogC ab und ist die Basis-Schnittstelle von ITcEventLogC3
ITcEventLogC [▶ 139]	Diese Schnittstelle ist die Basis-Schnittstelle von ITcEventLogC3 und ITcEventLogC2.
ITcEventLogBaseC	unbenutzt

Zusätzlich bietet diese Klasse die Event Schnittstelle _ITcEventLogEvents [▶ 146] die durch den Client implementiert werden kann.

9.2.2 TcEvent

Diese Klasse stellt ein einzelnes Event dar. Die Klasse bietet Schnittstellen zu den COM Clients. Objekte dieser Klasse werden vom TcEventLog erstellt, der Hauptklasse des TcEventLoggers. Die Klasse bietet über seine Schnittstelle alle Parameter des Alarms und benutzt intern das zugewiesene Formatter-Objekt, um die Sprache Event-Meldung unabhängig zurückzugeben.

Schnittstelle	Beschreibung
ITcEvent [▶ 159]	Die Hauptschnittstelle stellt alle Parameter eines Event Objekts zur Verfügung.

Schnittstelle	Beschreibung
ITcEventC [▶ 177]	Diese kundenspezifische Schnittstelle stellt weitere Alarmparameter zur Verfügung.

9.3 Interfaces

9.3.1 ITcEventLog

Der ITcEventLog ist eine Standard-Schnittstelle der Hauptklasse des TcEventLoggers [▶ 128]. Die Schnittstelle bietet die Hauptfeatures um aktive und logged Alarme zu regeln und auch um neue Alarme hinzuzufügen und Bestehende zu löschen. Die Schnittstelle leitet sich von **IDispatch** ab, so dass es von allen Sprachen verwendet werden kann, die COM Automation Schnittstellen unterstützen. IDispatch selbst leitet sich von IUnknown ab.

Tab. 15: Methoden und Eigenschaften

IUnknown Methoden	Beschreibung
QueryInterface	Gibt einen Pointer zur angefragten Schnittstelle zurück.
AddRef	Inkrementiert den Referenzzähler.
Release	Dekrementiert den Referenzzähler.

IDispatch Methoden	Beschreibung
GetIDsOfNames	Mappt einen einzelnen Teilnehmernamen und einen Satz optionaler Parameternamen mit einem entsprechenden Satz gesendeter Integer Identifier (DISPIDs). Diese können dann als Folgeabrufe an **IDispatch::Invoke** benutzt werden.
GetTypeInfo	Ruft die Typ Informationen für ein Objekt ab.
GetTypeInfoCount	Ruft die Anzahl der Typ Informations-Schnittstellen ab, die ein Objekt zur Verfügung stellt, entweder 0 oder 1.
Invoke	Stellt den Zugriff auf Eigenschaften und Methoden, ausgestellt durch ein Objekt, zur Verfügung.

ITcEventLog Methoden und Eigenschaften	Beschreibung
ReportEvent [▶ 131]	Diese Methode wird verwendet, um vom Client ein neues Event auszugeben.
ClearActiveEvents [▶ 132]	Diese Methode setzt alle aktiven Events zurück.
GetLastEvent [▶ 133]	Diese Methode gibt die letzten Events zurück.
EnumActiveEvents [▶ 133]	Gibt ein Aufzählungsobjekt zurück, welches zur Wiederholung der aufgelisteten aktiven Events benutzt wird.
EnumActiveEventsEx [▶ 133]	Liefert einen Standard Enumerator zum Aufzählen der aktiven Meldungen.
EnumLoggedEvents [▶ 135]	Gibt ein Aufzählungsobjekt zurück, welches zur Wiederholung der aufgelisteten logged Events benutzt wird.
EnumLoggedEventsEx [▶ 133]	Liefert einen Standard Enumerator zum Aufzählen der geloggten Meldungen.
Export [▶ 137]	Kopiert die Liste der Logged Events in einen strukturierten Storage und gibt den Storage zurück.
ExportText	Nicht implementiert
ActiveEvents [▶ 137]	Setzt die Anzahl der aktiven Events zurück.
LoggedEvents [▶ 138]	Setzt die Anzahl der Logged Events zurück.

ITcEventLog Methoden und Eigenschaften	Beschreibung
ClearLoggedEvents [▶ 138]	Diese Methode setzt alle Logged Events zurück.

Sehen Sie dazu auch

- EnumActiveEventsEx [▶ 134]
- EnumLoggedEventsEx [▶ 136]

9.3.1.1 ReportEvent

ITcEventLog [▶ 130]::ReportEvent

Diese Methode wird verwendet, um von einem COM Client (z.B. ein HMI Programm) ein neues Event auszugeben (an den TcEventLogger).

```
HRESULT ReportEvent([in] SAFEARRAY(VARIANT)* eventHead,
                    [in] SAFEARRAY(VARIANT)* eventData)
);
```

Parameter

eventHead

[in] Pointer auf ein Safearray mit den Grenzen 0 bis 9, dass den Event Header darstellt. Das Array besitzt die gleiche Information wie der TcEventHeader. Die Punkte des Arrays sind in der folgenden Tabelle erklärt.

Index	TcEventHeader [▶ 184] Punkt	Datentyp	Beschreibung
0	**nClass**	TcEventClass [▶ 181]	Die Klasse des Events, wie Alarm, Warnung, Hinweis.
1	**nPriority**	TcEventPriority [▶ 183] **(long)**	Die Eventpriorität. Ab jetzt ist die Priorität immer: TcEventPriority.TCEVENTPRIO_IMPLICIT=0
2	**dwFlags**	TcEventFlags [▶ 182] **(long)**	Die Merker regeln das Verhalten des Alarms. Dies kann mit einer OR Verknüpfung kombiniert werden.
3	**dwUserData**	**long**	Reservevariable, die vom Benutzer verwendet werden kann.
4	**nId**	**long**	Die Event Id, die eindeutig den Eventtyp für diesen einzelnen Event Source darstellt.
5	**nInvokeId**	**long**	
6	**fDate**	**VARIANT-DATE**	Datum und Zeit des Fehlerauftritts.
7	**nMs**	**long**	Der Millisekunden Anteil des Event Datums.
8	**varSource**	**variant**	Source id und Source Name. Die Event Source wird zu Kennzeichnung verschiedener Geräte verwendet (z. B. I/O Event, oder verschiedene Teile des SPS Programms.)
9	**szFmtProgId**	**BSTR string**	Die PrgId des Formatters. Wenn der Standard XML Formatter verwendet werden soll, muss der String auf "TcEventFormatter.TcXmlFormatter" gesetzt werden.

eventData

[in] Pointer auf ein Safearray, das die Event Argumente darstellt. In der Konfiguration des Event Formatters ist definiert, wie die Event Argumente in der angezeigten Event Meldung darzustellen sind. Die folgenden Datentypen werden durch den Standard Formatter unterstützt: int (16bit), long (32bit), float(32bit), double(64bit), string(BSTR).

Rückgabe Werte

S_OK

Funktion wurde erfolgreich aufgerufen.

TCEVENTERR_ISSIGNALED

Das Event mit dieser Source Id und Event Id wurde bereits angezeigt.

Visual Basic Beispielcode

```
' 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 soruce 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)
```

Anmerkungen

Nach diesem Methodenaufruf wird der Event Logger das Event OnNewEvent [▶ 146] auf allen Clients, die die Event-Schnittstelle _ITcEventLogEvents [▶ 146] (VB: Dim WithEvents) implementieren, erhöhen.

Für C++ und Visual Basic Programmierer ist es bequemer, die Methode ITcEventLogC [▶ 139]::ReportEvent [▶ 139] oder ITcEventC3 [▶ 142]::ReportEventEx [▶ 144] zu benutzen.

9.3.1.2 ClearActiveEvents

ITcEventLog [▶ 130]::ClearActiveEvents

ITcEventLog [▶ 130]

Diese Methode setzt alle aktiven Events zurück.

```
HRESULT ClearActiveEvents();
```

Parameter

Rückgabe Werte

S_OK

Funktion wurde erfolgreich aufgerufen.

Visual Basic Beispielcode

```
' get the one and only event logger
Dim evtLogger As TCEVENTLOGGERLib.TcEventLog
Set evtLogger = New TCEVENTLOGGERLib.TcEventLog

' clear all active alarms
evtLogger.ClearActiveEvents
```

Anmerkungen

Nach diesem Methodenaufruf wird der Event Logger das Event OnActiveEventsCleared [▶ 149] auf allen Clients, die die Event-Schnittstelle _ITcEventLogEvents [▶ 146] (VB: Dim WithEvents) implementieren, erhöhen.

9.3.1.3 GetLastEvent

ITcEventLog [▶ 130]::GetLastEvent

Diese Methode gibt die letzten aktiven Events zurück.

```
HRESULT GetLastEvent([out,retval] IDispatch** lastEvent);
```

Parameter

lastEvent

[out, retval] Pointer auf ein Objekt des Schnittstellentypen IDispatch, welches das letzte aktive Event zurückgibt.

Rückgabe Werte

S_OK
Funktion wurde erfolgreich aufgerufen.

E_POINTER

Die Pointerzuweisung an lastEvent war nicht gültig.

Visual Basic Beispielcode

```
' get the one and only event logger
Dim evtLogger As TCEVENTLOGGERLib.TcEventLog
Set evtLogger = New TCEVENTLOGGERLib.TcEventLog

' get the most resent logged event
Dim resentEvt As TcEvent
Set resentEvt = evtLogger.GetLastEvent
```

9.3.1.4 EnumActiveEvents

ITcEventLog [▶ 130]::EnumActiveEvents

Gibt ein Aufzählungsobjekt zurück, welches zum Iterieren über die aufgelisteten aktiven Events benutzt wird.

```
HRESULT EnumActiveEvents([out,retval] IUnknown** ppEnum);
```

Parameter

ppEnum

[out, retval] Pointer auf den ITcEnumEvents [▶ 152] Schnittstellen-Pointer, der die Aufzählungsobjekte der aktiven Events erhält.

Rückgabe Werte

S_OK

Funktion wurde erfolgreich aufgerufen.

E_POINTER

ppEnum war kein gültiger Pointer.

Visual Basic Beispielcode

```
' get the one and only event logger
Dim evtLogger As TCEVENTLOGGERLib.TcEventLog
Set evtLogger = New TCEVENTLOGGERLib.TcEventLog

' get the active event enumaration 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)
```

Anmerkungen

Die importierten TcEventLogger-Typblibliotheken verhalten sich unter CodeGear C++Builder 2009 anders als unter Visual Studio C++. Beide Entwicklungsumgebungen generieren dabei einen unterschiedlichen Code. In C++Builder wird das zurückgelieferte Auzählungsobjekt nicht automatisch freigegeben wenn es den Sichtbarkeitsbereich verlässt ("out of scope" geht). Z.B. nach der Zuweisung:

```
ITcEnumEventsPtr enumEvt = evtLogger->EnumActiveEvents();
```

muss in C++Builder das nicht mehr benötigte Objekt explizit freigegeben werden:

```
enumEvt->Release();
```

Hier gibt es einen Unterschied zu Microsoft Visual Studio C++ Implementierung. Dort ist der Release-Aufruf nicht notwendig. Aus diesem Grund verwenden Sie bitte in C++Builder die **EnumActiveEventsEx**()-Methode. Bei dieser Methode wird das zurückgelieferte Objekt auch unter C++Builder automatisch freigegeben.

Sehen Sie dazu auch

- ClearActiveEvents [▶ 132]

9.3.1.5 EnumActiveEventsEx

ITcEventLog [▶ 130]

Liefert ein enumerator mit dessen Hilfe durch die Liste der aktiven Events iteriert werden kann.

Dieses Interface wird für eine bessere Kompatibilität mit Programmiersprachen aus der .NET Familie bereitgestellt. Für die Programmierung in C++ kann in einigen fällen die ursprüngliche ITcEnumEvents [▶ 152] Schnitstelle besser geignet sein

```
HRESULT EnumActiveEventsEx([out,retval] ITcEnumEventEx**
ppEnum);
```

Parameters

ppEnum

[out, retval] Zeiger auf einen Zeiger zur ITcEnumEventsEx [▶ 157] Schnittstelle.

Return Values

S_OK

Der Aufruf ist erfolgreich abgeschlossen.

E_POINTER

ppEnum ist kein gültiger Zeiger.

Visual Studio 2005 C# Sample Code:

```
foreach(TcEventevtinnewTcEventLog().EnumActiveEventsEx())

  evt.Reset();
```

Sehen Sie dazu auch

EnumActiveEvents [▶ 133]

9.3.1.6 EnumLoggedEvents

ITcEventLog [▶ 130]::EnumLoggedEvents

Gibt ein Aufzählungsobjekt zurück, welches zum Iterieren über die aufgelisteten geloggten Events benutzt wird.

```
HRESULT EnumLoggedEvents([out,retval] IUnknown** ppEnum);
```

Parameter

ppEnum

[out, retval] Pointer auf den ITcEnumEvents [▶ 152] Schnittstellen-Pointer, der die Aufzählungsobjekte der logged Events erhält.

Rückgabe Werte

S_OK

Funktion wurde erfolgreich aufgerufen.

E_POINTER

ppEnum war kein gültiger Pointer.

Visual Basic Beispielcode

```
' get the one and only event logger
Dim evtLogger As TCEVENTLOGGERLib.TcEventLog
Set evtLogger = New TCEVENTLOGGERLib.TcEventLog

' get the active event enumaration object
Dim enumEvt As ITcEnumEvents
Set enumEvt = evtLogger.EnumLoggedEvents

' 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)
```

Anmerkungen

Die importierten TcEventLogger-Typblibliotheken verhalten sich unter CodeGear C++Builder 2009 anders als unter Visual Studio C++. Beide Entwicklungsumgebungen generieren dabei einen unterschiedlichen Code. In C++Builder wird das zurückgelieferte Auzählungsobjekt nicht automatisch freigegeben wenn es den Sichtbarkeitsbereich verlässt ("out of scope" geht). Z.B. nach der Zuweisung:

```
ITcEnumEventsPtr enumEvt = evtLogger->EnumLoggedEvents();
```

muss in C++Builder das nicht mehr benötigte Objekt explizit freigegeben werden:

```
enumEvt->Release();
```

Hier gibt es einen Unterschied zu Microsoft Visual Studio C++ Implementierung. Dort ist der Release-Aufruf nicht notwendig. Aus diesem Grund verwenden Sie bitte in C++Builder die **EnumLoggedEventsEx**()-Methode. Bei dieser Methode wird das zurückgelieferte Objekt auch unter C++Builder automatisch freigegeben.

Sehen Sie dazu auch

- EnumLoggedEventsEx [▶ 136]

9.3.1.7 EnumLoggedEventsEx

ITcEventLog [▶ 130]

Liefert ein enumerator mit dessen Hilfe durch die Liste der geloggten Events iteriert werden kann.

Dieses Interface wird für eine bessere Kompatibilität mit Programmiersprachen aus der .NET Familie bereitgestellt. Für die Programmierung in C++ kann in einigen fällen die ursprüngliche ITcEnumEvents [▶ 152] Schnitstelle besser geignet sein

```
HRESULT EnumLoggedEventsEx([out,retval] ITcEnumEventEx**
ppEnum);
```

Parameters

ppEnum

[out, retval] Zeiger auf einen Zeiger zur ITcEnumEventsEx [▶ 157] Schnittstelle.

Return Values

S_OK

Der Aufruf ist erfolgreich abgeschlossen.

E_POINTER

ppEnum ist kein gültiger Zeiger.

Visual Studio 2005 C# Sample Code:

```
foreach(TcEventevtinnewTcEventLog().EnumLoggedEventsEx())

  evt.Reset();
```

Sehen Sie dazu auch

EnumLoggedEvents [▶ 135]

9.3.1.8 Export

ITcEventLog [▶ 130]::Export

Kopiert die Liste der Logged Events in einen strukturierten Storage und gibt den Storage zurück.

```
HRESULT Export([in] IUnknown
*pExport);
```

Parameter

ppEnum

[in] IUnknown Pointer auf eine IStorage Schnittstelle, die die Kopie des strukturierten Storage erhält, der die Liste der logged Events in Streams enthält.

Rückgabe Werte

S_OK

Funktion wurde erfolgreich aufgerufen.

E_POINTER

pExport war kein gültiger Pointer.

Anmerkungen

Da der strukturierte Strorage das systemspezifische Datenbankformat ist und der Strukturentwurf nicht dokumentiert ist, sollte die Exportfunktion nicht von Benutzerprogrammen verwendet werden.

9.3.1.9 ActiveEvents

ITcEventLog [▶ 130]::ActiveEvents

Diese Eigenschaft setzt die Anzahl der aktiven Events zurück.

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

Parameter

pVal

[out, retval] Pointer auf einen langen Wert, der die aktuelle Anzahl von aktiven Events zurückgibt.

Rückgabe Werte

S_OK

Funktion wurde erfolgreich aufgerufen.

E_POINTER

pVal war kein gültiger Pointer.

Visual Basic Beispielcode

```
' 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 [▶ 130]::LoggedEvents

Diese Eigenschaft setzt die Anzahl der Logged Events zurück.

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

Parameter

pVal

[out, retval] Pointer auf einen langen Wert, der die aktuelle Anzahl von logged Events zurückgibt.

Rückgabe Werte

S_OK
Funktion wurde erfolgreich aufgerufen.

E_POINTER

pVal war kein gültiger Pointer.

Visual Basic Beispielcode

```
' 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 [▶ 130]::ClearLoggedEvents

Diese Methode setzt alle logged Events zurück.

```
HRESULT ClearLoggedEvents();
```

Parameter

Rückgabe Werte

S_OK

Funktion wurde erfolgreich aufgerufen.

Visual Basic Beispielcode

```
' get the one and only event logger
Dim evtLogger As TCEVENTLOGGERLib.TcEventLog
Set evtLogger = New TCEVENTLOGGERLib.TcEventLog
```

```
' clear all logged alarms
evtLogger.ClearLoggedEvents
```

Anmerkungen

Nach diesem Methodenaufruf wird der Event Logger das Event OnLoggedEventsCleared [▶ 150] auf allen Clients, die die Event-Schnittstelle _ITcEventLogEvents [▶ 146] (VB: Dim WithEvents) implementieren, erhöhen.

9.3.2 ITcEventLogC

Das ITcEventLogC ist eine Schnittstelle der Hauptklasse des TcEventLoggers [▶ 128] . Die Schnittstelle stellt die Features zur Verfügung und fügt neue Alarme hinzu. Die Schnittstelle leitet sich ab von **IUnknown** , so dass sie von allen Sprachen verwendet werden kann, die kundenspezifische COM Schnittstellen unterstützen.

Tab. 16: Methoden

IUnknown Methoden	Beschreibung
QueryInterface	Gibt einen Pointer zur angefragten Schnittstelle zurück.
AddRef	Inkrementiert den Referenzzähler.
Release	Dekrementiert den Referenzzähler.

ITcEventLogC Methoden	Beschreibung
ReportEvent [▶ 131]	Diese Methode wird verwendet, um vom Client ein neues Event auszugeben.
EnumEventSources	Abgelehnt. Wurde im abgelehnten strukturierten Storage basierenden Formatter verwendet, um die Liste der Event Source Storages zu durchschleifen.
OpenEventSource	Abgelehnt. Wurde im abgelehnten strukturierten Storage basierenden Formatter verwendet, um ein einzelnes Event Source Storage zurückzugeben.

9.3.2.1 ReportEvent

ITcEventLogC [▶ 139]::ReportEvent

Diese Methode wird verwendet, um von einem COM Client (z.B. ein HMI Programm) ein neues Event auszugeben (an den TcEventLogger).

```
HRESULT ReportEvent([in] TcEventHeader* pEventHead, [in] SAFEARRAY(VARIANT)* pEventData);
```

Parameter

pEventHead

[in] Pointer auf ein Objekt des Typs TcEventHeader [▶ 184]. Das Objekt stellt die Alarm Konfiguration dar.

eventData

[in] Pointer auf ein Safearray, dass die Event Argumente darstellt. In der Konfiguration des Event Formatters ist definiert, wie die Event Argumente in der Event Meldung zu platzieren sind.

Die folgenden Datentypen werden vom Standard Formatter unterstützt: int (16bit), long (32bit), float(32bit), double(64bit), string(BSTR).

Rückgabe Werte

S_OK

Funktion wurde erfolgreich aufgerufen.

TCEVENTERR_ISSIGNALED

Das Event mit dieser Source id und dieser Event id wurde bereits angezeigt.

E_POINTER

pEventHead oder pEventData waren keine gültigen Pointer.

Visual Basic Beispielcode

```
' 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 soruce id
header.szFmtProgId = "TcEventFormatter.TcXmlFormatter" ' event prog id

' creade 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)
```

Anmerkungen

Nach diesem Methodenaufruf wird der Event Logger das Event OnNewEvent [▶ 146] auf allen Clients, die die Event-Schnittstelle _ITcEventLogEvents [▶ 146] (VB: Dim WithEvents) implementieren, erhöhen.

Die Methode kann gerade verwendet werden, um ein Alarmsignal auszugeben. Ein zweiter Aufruf zur Rücksetzung des Alarms ist mit dieser Methode nicht möglich. Um den Status eines bereits ausgelösten Alarms zu ändern, ist das Event Objekt selber erforderlich.

Komfortabler arbeitet die Methode ITcEventLogC3 [▶ 142]::ReportEventEx [▶ 144] . Sie gibt das neu erstellte Event Objekt direkt zurück.

9.3.3 ITcEventLogC2

Das ITcEventLogC2 ist eine Schnittstelle der Hauptklasse des TcEventLoggers [▶ 128]. Die Schnittstelle stellt die abgelehnten Features zur Verfügung, um die abgelehnten strukturierten Storage basierenden Formatter zu kontrollieren.

Die Schnittstelle leitet sich von ITcEventLogC [▶ 139] und **IUnknown** ab, so dass sie von allen Sprachen verwendet werden kann, die kundenspezifische COM Schnittstellen unterstützen.

Tab. 17: Methoden

IUnknown Methoden	Beschreibung
QueryInterface	Gibt einen Pointer zur angefragten Schnittstelle zurück.
AddRef	Inkrementiert den Referenzzähler.
Release	Dekrementiert den Referenzzähler.

ITcEventLogC Methoden	Beschreibung
ReportEvent [▶ 131]	Diese Methode wird verwendet, um vom Client ein neues Event auszugeben.
EnumEventSources	Abgelehnt. Wurde im abgelehnten strukurierten Storage basierenden Formatter verwendet, um die Liste der Event Source Storages zu durchschleifen.
OpenEventSource	Abgelehnt. Wurde im abgelehnten strukturierten Storage basierenden Formatter verwendet, um ein einzelnes Event Source Storage zurückzugeben.

ITcEventLogC2 Methoden	Beschreibung
CreateEventSource	Abgelehnt. Wurde im abgelehnten strukurierten Storage basierenden Formatter verwendet, um ein neues Event Source Stroage zu bilden.
RemoveEventSource	Abgelehnt. Wurde im abgelehnten strukurierten Storage basierenden Formatter verwendet, um eine bestehende Event Source Storage zu entfernen.
RenameEventSource	Abgelehnt. Wurde im abgelehnten strukurierten Storage basierenden Formatter verwendet, um eine bestehende Event Source Storage umzubenennen.

9.3.3.1 ReportEvent

ITcEventLogC [▶ 139]::ReportEvent

Diese Methode wird verwendet, um von einem COM Client (z.B. ein HMI Programm) ein neues Event auszugeben (an den TcEventLogger).

```
HRESULT ReportEvent([in] TcEventHeader* pEventHead, [in] SAFEARRAY(VARIANT)* pEventData);
```

Parameter

pEventHead

[in] Pointer auf ein Objekt des Typs TcEventHeader [▶ 184]. Das Objekt stellt die Alarm Konfiguration dar.

eventData

[in] Pointer auf ein Safearray, dass die Event Argumente darstellt. In der Konfiguration des Event Formatters ist definiert, wie die Event Argumente in der Event Meldung zu platzieren sind.

Die folgenden Datentypen werden vom Standard Formatter unterstützt: int (16bit), long (32bit), float(32bit), double(64bit), string(BSTR).

Rückgabe Werte

S_OK

Funktion wurde erfolgreich aufgerufen.

TCEVENTERR_ISSIGNALED

Das Event mit dieser Source id und dieser Event id wurde bereits angezeigt.

E_POINTER

pEventHead oder pEventData waren keine gültigen Pointer.

Visual Basic Beispielcode

```
' 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 soruce id
header.szFmtProgId = "TcEventFormatter.TcXmlFormatter" ' event prog id

' creade 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)
```

Anmerkungen

Nach diesem Methodenaufruf wird der Event Logger das Event OnNewEvent [▶ 146] auf allen Clients, die die Event-Schnittstelle _ITcEventLogEvents [▶ 146] (VB: Dim WithEvents) implementieren, erhöhen.

Die Methode kann gerade verwendet werden, um ein Alarmsignal auszugeben. Ein zweiter Aufruf zur Rücksetzung des Alarms ist mit dieser Methode nicht möglich. Um den Status eines bereits ausgelösten Alarms zu ändern, ist das Event Objekt selber erforderlich.

Komfortabler arbeitet die Methode ITcEventLogC3 [▶ 142]**::**ReportEventEx [▶ 144] . Sie gibt das neu erstellte Event Objekt direkt zurück.

9.3.4 ITcEventLogC3

Das ITcEventLogC3 ist eine Schnittstelle der Hauptklasse des TcEventLoggers [▶ 128]. Die Schnittstelle stellt die Features zur Verfügung und fügt neue Alarme hinzu. Die Schnittstelle leitet sich ab von ITcEventLogC2 [▶ 140], ITcEventLogC [▶ 139] und **IUnknown,** so dass sie von allen Sprachen verwendet werden kann, die kundenspezifische COM Schnittstellen unterstützen.

Tab. 18: Methoden

IUnknown Methoden	Beschreibung
QueryInterface	Gibt einen Pointer zur angefragten Schnittstelle zurück.
AddRef	Inkrementiert den Referenzzähler.
Release	Dekrementiert den Referenzzähler.

ITcEventLogC [▶ 139] Methoden	Beschreibung
ReportEvent [▶ 131]	Diese Methode wird verwendet, um vom Client ein neues Event auszugeben.
EnumEventSources	Abgelehnt. Wurde im abgelehnten strukturierten Storage basierenden Formatter verwendet, um die Liste der Event Source Storages zu durchschleifen.
OpenEventSource	Abgelehnt. Wurde im abgelehnten strukturierten Storage basierenden Formatter verwendet, um ein einzelnes Event Source Storage zurückzugeben.

ITcEventLogC2 [▶ 140] Methoden	Beschreibung
CreateEventSource	Abgelehnt. Wurde im abgelehnten strukturierten Storage basierenden Formatter verwendet, um ein neues Event Source Storage zu bilden.
RemoveEventSource	Abgelehnt. Wurde im abgelehnten strukturierten Storage basierenden Formatter verwendet, um eine bestehende Event Source Storage zu entfernen.
RenameEventSource	Abgelehnt. Wurde im abgelehnten strukturierten Storage basierenden Formatter verwendet, um eine bestehende Event Source Storage umzubenennen.

ITcEventLogC3 Methoden	Beschreibung
ReportEventEx [▶ 144]	Diese Methode wird verwendet, um vom Client ein neues Event auszugeben. Die Funktion gibt den neu ausgelösten Alarm zur weiteren Regelung zurück.

9.3.4.1 ReportEvent

ITcEventLogC [▶ 139]::ReportEvent

Diese Methode wird verwendet, um von einem COM Client (z.B. ein HMI Programm) ein neues Event auszugeben (an den TcEventLogger).

```
HRESULT ReportEvent([in] TcEventHeader*
pEventHead, [in] SAFEARRAY(VARIANT)* pEventData);
```

Parameter

pEventHead

[in] Pointer auf ein Objekt des Typs TcEventHeader. Das Objekt stellt die Alarm Konfiguration dar.

eventData

[in] Pointer auf ein Safearray, dass die Event Argumente darstellt. In der Konfiguration des Event Formatters ist definiert, wie die Event Argumente in der Event Meldung zu platzieren sind.

Die folgenden Datentypen werden vom Standard Formatter unterstützt: int (16bit), long (32bit), float(32bit), double(64bit), string(BSTR).

Rückgabe Werte

S_OK

Funktion wurde erfolgreich aufgerufen.

TCEVENTERR_ISSIGNALED

Das Event mit dieser Source id und dieser Event id wurde bereits angezeigt.

E_POINTER

pEventHead oder pEventData waren keine gültigen Pointer.

Visual Basic Beispielcode

```
' 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 soruce id
header.szFmtProgId = "TcEventFormatter.TcXmlFormatter" ' event prog id

' creade 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)
```

Anmerkungen

Nach diesem Methodenaufruf wird der Event Logger das Event OnNewEvent [▶ 146] auf allen Clients, die die Event-Schnittstelle _ITcEventLogEvents [▶ 146] (VB: Dim WithEvents) implementieren, erhöhen.

Die Methode kann gerade verwendet werden, um ein Alarmsignal auszugeben. Ein zweiter Aufruf zur Rücksetzung des Alarms ist mit dieser Methode nicht möglich. Um den Status eines bereits ausgelösten Alarms zu ändern, ist das Event Objekt selber erforderlich.

Komfortabler arbeitet die Methode ITcEventLogC3 [▶ 142]::ReportEventEx [▶ 144] . Sie gibt das neu erstellte Event Objekt direkt zurück.

Sehen Sie dazu auch

- TcEventHeader [▶ 184]

9.3.4.2 ReportEventEx

ITcEventLogC3 [▶ 142]::ReportEventEx

Diese Methode wird verwendet, um von einem COM Client (z.B. ein HMI Programm) ein neues Event auszugeben (an den TcEventLogger).

```
HRESULTReportEventEx([in]TcEventHeader* pEventHead,
                     [in] SAFEARRAY(VARIANT)* pEventData
                     [out, retval] [out,retval] ITcEvent** pEvent);
```

Parameter

pEventHead

[in] Pointer auf ein Objekt vom Typ TcEventHeader. Das Objket stellt die Alarm Konfiguration dar.

eventData

[in] Pointer auf ein Safearray , das die Event Argumente darstellt. In der Konfiguration des Event Formatters ist definiert, wie die Event Argumente in der angezeigten Event Meldung darzustellen sind. Die folgenden Datentypen werden vom Standardformatter unterstützt: int (16bit), long (32bit), float(32bit), double(64bit), string(BSTR).

pEvent

[out, retval] Pointer auf ein ITcEvent Pointer, der das neue logged Event Objekt erhält.

Return Values

S_OK

Funktion wurde erfolgreich aufgerufen.

TCEVENTERR_ISSIGNALED

Das Event mit dieser Source Id und Event Id wurde bereits angezeigt.

E_POINTER

pEventHead oder pEventData oder pEvent waren keine gültigen Pointer.

Visual Basic Beispielcode

```
' 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 soruce id
header.szFmtProgId = "TcEventFormatter.TcXmlFormatter" ' event prog id

' creade 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
```

Anmerkungen

Nach diesem Methodenaufruf wird der Event Logger das Event OnNewEvent [▶ 146] auf allen Clients, die die Event-Schnittstelle _ITcEventLogEvents [▶ 146] (VB: Dim WithEvents) implementieren, erhöhen.

Sehen Sie dazu auch

- TcEventHeader [▶ 184]

9.3.5 ITcEventLogEvents

Das _ITcEventLogEvent ist die Standard Event Schnittstelle der Hauptklasse des TcEventLoggers [▶ 128]. Die Schnittstelle stellt eine Callback Methode zu Verfügung mit der ein Client mit dem Ist-Status des TcEvent Loggers synchronisiert werden kann. Die Schnittstelle leitet sich von **IDispatch** ab, so dass sie von allen Sprachen verwendet werden kann, die kundenspezifische COM Schnittstellen unterstützen. IDispatch leitet sich wiederum von IUnknown ab.

Tab. 19: Methoden in

IUnknown Methoden	Beschreibung
QueryInterface	Gibt einen Pointer zur angefragten Schnittstelle zurück.
AddRef	Inkrementiert den Referenzzähler.
Release	Dekrementiert den Referenzzähler.

IDispatch Methoden	Beschreibung
GetIDsOfNames	Mappt einen einzelnen Teilnehmernamen und einen Satz optionaler Parameternamen mit einem entsprechenden Satz gesendeter Integer Identifier (DISPIDs). Diese können dann als Folgeabrufe an **IDispatch::Invoke** benutzt werden.
GetTypeInfo	Ruft die Typ Informationen für ein Objekt ab.
GetTypeInfoCount	Ruft die Anzahl der Typ Informations-Schnittstellen ab, die ein Objekt zur Verfügung stellt, entweder 0 oder 1.
Invoke	Stellt den Zugriff auf Eigenschaften und Methoden, ausgestellt durch ein Objekt, zur Verfügung.

ITcEventLog Methoden	Beschreibung
OnNewEvent [▶ 146]	Diese Methode wird aufgerufen wenn ein neues Event ausgegeben wird.
OnConfirmEvent [▶ 173]	Diese Methode wird aufgerufen wenn ein Event bestätigt wird.
OnResetEvent [▶ 148]	Diese Methode wird aufgerufen wenn ein Event zurückgesetzt wird.
OnSignalEvent [▶ 177]	Diese Methode wird aufgerufen wenn ein Event signalisiert wird.
OnActiveEventsCleared [▶ 149]	Diese Methode wird aufgerufen wenn alle aktiven Events gelöscht werden.
OnLoggedEventsCleared [▶ 150]	Diese Methode wird aufgerufen wenn alle logged Events gelöscht werden.
OnClearEvent [▶ 151]	Diese Methode wird aufgerufen wenn ein Event gelöscht wird.
OnShutdown [▶ 151]	Diese Methode wird aufgerufen wenn das Betriebssystem heruntergefahren wird (shut down).
OnNewEventConfiguration [▶ 152]	Diese Methode wird aufgerufen wenn ein eine neue Konfiguration geladen wurde.

9.3.5.1 OnNewEvent

_ITcEventLogEvents [▶ 146]

Diese Event Methode wird auf allen Clients aufgerufen, die diese Event Schnittstelle implementieren und verbinden, wenn ein neues Event ausgegeben wird. Das Event wurde vom SPS Funktionsblock ADSLOGEVENT ausgegeben, indem ein Merker von FALSE auf TRUE gesetzt wurde (oder ein anderes

ADS Gerät) oder durch den Aufruf einer Report Event Funktion, wie IT [▶ 139]cEventLogC [▶ 139]:: ReportEvent [▶ 139],ITcEventC3 [▶ 142]:: ReportEventEx [▶ 144] oder ITcEventLog [▶ 130]::ReportEvent [▶ 131].

```
HRESULT OnNewEvent([in] IDispatch* evtObj);
```

Parameter

evtObj

[in] **IDispatch** Pointer auf ein neues Eventobjekt der Klasse TvEvent [▶ 129]. Das Eventobjekt stellt den vollen Zugriff auf die Neuausgabe des Events zur Verfügung. Jeder Client erhält einen Hinweis auf das Eventobjekt und eine Kopie.

Rückgabe Werte

S_OK

Funktion wurde erfolgreich aufgerufen.

E_POINTER

evtObj war kein gültiger Pointer.

Visual Basic Beispielcode

```
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 [▶ 146]

Diese Event Methode wird auf allen Clients aufgerufen, die diese Event Schnittstelle implementieren und verbinden, wenn ein Event bestätigt (confirmed) wird. Das Event wurde durch den SPS Funktionsblock bestätigt, indem **EventQuit=TRUE** (oder eines anderen ADS Gerätes) gesetzt wurde, oder durch einen Aufruf von ITcEvent [▶ 159]::.Confirm [▶ 169]

```
HRESULT OnConfirmEvent([in] IDispatch* evtObj);
```

Parameter

evtObj

[in] **IDispatch** Pointer auf ein Eventobjekt der Klasse TvEvent [▶ 129]. Das Eventobjekt bietet den vollen Zugriff auf das bestätigte Event. Jeder Client erhält einen Hinweis auf das Eventobjekt und eine Kopie.

Rückgabe Werte

S_OK

Funktion wurde erfolgreich aufgerufen.

E_POINTER

evtObj war kein gültiger Pointer.

Visual Basic Beispielcode

```
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
```

9.3.5.3 OnResetEvent

ITcEventLogEvents [▶ 146]

Diese Event Methode wird auf allen Clients aufgerufen, die diese Event Schnittstelle implementieren und verbinden, wenn ein Event zurückgesetzt wird. Das Event wurde vom SPS Funktionsblock ADSLOGEVENT durch Setzen des **Events** von **TRUE** auf **FALSE** (oder eines anderes ADS Gerätes) oder durch ITcEvent [▶ 159]::Reset [▶ 170] aufgerufen.

```
HRESULT OnResetEvent([in] IDispatch* evtObj);
```

Parameter

evtObj

[in] **IDispatch** Pointer auf ein Eventobjekt der Klasse TvEvent [▶ 129]. Das Eventobjekt bietet den vollen Zugriff auf das zurückgesetzte Event. Jeder Client erhält einen Hinweis auf das Eventobjekt und eine Kopie.

Rückgabe Werte

S_OK

Funktion wurde erfolgreich aufgerufen.

E_POINTER

evtObj war kein gültiger Pointer.

Visual Basic Beispielcode

```
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 [▶ 146]

Diese Event Methode wird auf allen Clients aufgerufen, die diese Event Schnittstelle implementieren und verbinden, wenn ein Event signalisiert wird. Nur Events, die confirmed werden müssen, können signalisiert werden. Ein signalisiertes Event wartet auf seine confirmation, und wurde vorher zurückgesetzt. Das Event wurde vom SPS Funktionsblock ADSLOGEVENT durch Setzen des **Events** von **TRUE** auf **FALSE** (oder eines anderes ADS Gerätes) oder durch ITcEvent [▶ 159]**::**Signal [▶ 177] aufgerufen.

```
HRESULT OnSignalEvent([in] IDispatch* evtObj);
```

Parameter

evtObj

[in] **IDispatch** Pointer auf ein Eventobjekt der Klasse TvEvent [▶ 129]. Das Eventobjekt bietet den vollen Zugriff auf das signalisierte Event. Jeder Client erhält einen Hinweis auf das Eventobjekt und eine Kopie.

Rückgabe Werte

S_OK

Funktion wurde erfolgreich aufgerufen.

E_POINTER

evtObj war kein gültiger Pointer.

Visual Basic Beispielcode

```
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 [▶ 146]

Diese Event Methode wird auf allen Clients aufgerufen, die diese Event Schnittstelle implementieren und verbinden, wenn alle aktiven Events gelöscht sind.

Vorausgesetzt:

· Das TwinCAT System wird heruntergefahren.

· Die Methode wird vom SPS Funktionsblock ADSCLEAREVENTS ausgeführt. Dabei muss der iMode auf TCEEVENTLOGIOFFS_CLEARACTIVE bzw. TCEVENTLOGIOFFS_CLEARALL gesetzt sein.

· Die Methode wird durchITcEventLog [▶ 130]::ClearActiveEvents [▶ 132]aufgerufen.

```
HRESULT OnActiveEventsCleared();
```

Parameter

S_OK

Funktion wurde erfolgreich aufgerufen.

Visual Basic Beispielcode

```
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 [▶ 146]

Diese Event Methode wird auf allen Clients aufgerufen, die diese Event Schnittstelle implementieren und verbinden.

Vorausgesetzt:

· Alle logged Events sind gelöscht.

· Die Methode wird vom SPS Funktionsblock ADSCLEAREVENTS ausgeführt. Dabei muss der iMode auf TCEVENTLOGIOFFS_CLEARLOGGED oder TCEVENTLOGIOFFS_CLEARALL gesetzt sein.

· Die Methode wird durch ITcEventLog [▶ 130]::ClearLoggedEvents [▶ 138]aufgerufen.

```
HRESULT OnLoggedEventsCleared();
```

Parameter

S_OK

Funktion wurde erfolgreich aufgerufen.

Visual Basic Beispielcode

```
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 [▶ 146]

Diese Event Methode wird auf allen Clients aufgerufen, die diese Event Schnittstelle implementieren und verbinden, wenn ein Event gelöscht wird. Diese Methode wird auf allen Events aufgerufen, für eine Source ID nach einem Aufruf des SPS Funktionsblocks ADSCLEAREVENTS mit dem gesetzten iMode TCEVENTLOGIOFFS_CLEAREVENTSRCID.

```
HRESULT OnClearEvent(ByVal evtObj As Object);
```

Parameter

evtObj

[in] IDispatch Pointer auf ein neues Eventobjekt der Klasse TvEvent. Das Eventobjekt bietet den vollen Zugriff auf das gelöschte Event. Jeder Client erhält einen Hinweis auf das Eventobjekt und eine Kopie.

S_OK

Funktion wurde erfolgreich aufgerufen.

Visual Basic Beispielcode

```
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 [▶ 146]

Diese Event Methode wird benutzt, wenn das Betriebssystem heruntergefahren wird. Dies kann von DCOM Clients zur Freigabe der Verbindung verwendet werden.

```
HRESULT OnShutdown([in] VARIANT shutdownParm);
```

Parameter

shutdownParm

[in] Eine Variante, die den Grund für den Shutdown beschreibt (hier immer REASON_SYSSHUTDOWN).

Die Einträge des Arrays sind in der folgenden Tabelle erklärt.

Voraussetzungen

Index	TcEventHeader [▶ 184] Eintrag	Beschreibung
0	REASON_INVALID	ungültig

Index	TcEventHeader [▶ 184] Eintrag	Beschreibung
1	REASON_TCSTOP	Das TwinCAT System fährt herunter
2	REASON_SYSSHUTDOWN	Das Betriebssystem fährt herunter

S_OK

Funktion wurde erfolgreich aufgerufen.

Visual Basic Beispielcode

```
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 [▶ 146]

Diese Event Methode wird auf allen Clients aufgerufen, die diese Event Schnittstelle implementieren und verbinden, wenn die ITcEventLog::Reset Methode aufgerufen wurde und ein Eventformatter neu initialisiert wurde. Die Methode wird nicht aufgerufen wenn ITcEventLog::Reset ausgelöst wurde, aber kein Formatter neu initialisiert werden musste.

```
HRESULT OnNewEventConfiguration();
```

Parameter

S_OK

Die Funktion wurde erfolgreich ausgeführt

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

Die Schnittstelle ITcEnumEvents wird zum Abfragen von Events verwendet. Aufzählungen von Events werden in einem Array Typ zurückgegeben. Die Schnittstelle leitet sich von **IUnknown** ab, so dass sie von allen Sprachen verwendet werden kann, die COM Schnittstellen unterstützen.

Tab. 20: Methoden und Eigenschaften in Vtable Reihenfolge

IUnknown Methoden	Beschreibung
QueryInterface	Gibt einen Pointer zur angefragten Schnittstelle zurück.
AddRef	Inkrementiert den Referenzzähler.
Release	Dekrementiert den Referenzzähler.

ITcEnumEvents Methoden	Beschreibung
Next [▶ 153]	Gibt die nächste Nummer des Events in der Aufzählungsreihenfolge zurück.
Skip [▶ 154]	Überspringt die nächste Nummer der Events in der Aufzählungsreihenfolge.
Reset [▶ 155]	Setzt die Aufzählungsreihenfolge auf das erste Element zurück.
Clone [▶ 155]	Bildet das Aufzählungsobjekt nach.
VBNext [▶ 156]	Gibt die nächste Nummer des Events zurück, die in ein Safearray der Aufzählungsreihenfolge übernommen wurden.

9.3.6.1 Next

ITcEnumEvents [▶ 152]::Next

Diese Methode gibt die nächsten Events in der Aufzählungsreihenfolge zurück.

```
HRESULT Next(
    [in] long celt,
    [out,size_is(celt),length_is(*pceltFetched)]IDispatch** ppElements,
    [out, retval] long *pceltFetched);
```

Parameter

celt

[in] Anzahl der angefragten Elemente der Aufzählungsreihenfolge.

ppElements

[out, size_is(celt), length_is(*pceltFetched)] Pointer auf das erste Element eines Arrays aus Events. Gibt die durch pceltFetched vergebene Anzahl der Events zurück.

pceltFetched

Gibt die Anzahl der zurückgegebenen Events zurück.

Rückgabe Werte

S_OK

Funktion wurde erfolgreich aufgerufen.

S_FALSE

Die Anzahl der zurückgegebenen Elemente war kleiner als die der Angefragten.

E_POINTER

Elemente oder pceltFetched waren keine gültigen Pointer.

Visual Basic Beispielcode

```
' get the one and only event logger
Dim evtLogger As TCEVENTLOGGERLib.TcEventLog
Set evtLogger = New TCEVENTLOGGERLib.TcEventLog
```

```
' get the active event enumaration 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)
```

Anmerkungen

Die **Next** Methode funktioniert nur, wenn die aufrufende Sprache eine echte Referenz auf den Array übergibt.
.NET Sprachen tun dies z.B. nicht. Aus solchen Sprachen benutzen Sie die VBNext [▶ 156] Methode.

9.3.6.2 Skip

ITcEnumEvents [▶ 152]::Skip

ITcEnumEvents [▶ 152]

Diese Methode überspringt die nächste Nummer der Events in der Aufzählungsreihenfolge.

```
HRESULT Skip(
    [in] long cSkipElem);
```

Parameter

cSkipElem

[in] Anzahl der zu überspringenden Elemente.

Return Value

S_OK

Die Funktion wurde erfolgreich aufgerufen.

S_FALSE

Die Anzahl der übersprungenen Elemente war nicht identisch mit cSkipElem.

E_NOTIMPL

Die Methode ist nicht implementiert.

Anmerkungen

Die Methode ist nicht implementiert für die Enum Objekte, die durch ITcEventLog [▶ 130]::EnumActiveEvents [▶ 133] und ITcEventLog [▶ 130]::EnumLoggedEvents [▶ 135] der Klasse

TcEventLog [▶ 129] zurückgegeben werden.

9.3.6.3 Reset

ITcEnumEvents [▶ 152]::Reset

Diese Methode setzt die Aufzählungsreihenfolge auf das erste Element zurück.

```
HRESULT Reset(
    void);
```

Parameter

Rückgabe Werte

S_OK

Funktion wurde erfolgreich aufgerufen.

S_FALSE

Die Anzahl der übersprungenen Elemente ist nicht identisch mit cSkipElem.

E_NOTIMPL

Die Methode ist nicht implementiert.

Anmerkungen

Die Methode ist nicht implementiert für die Enum Objekte, die durch ITcEventLog [▶ 130]::EnumActiveEvents [▶ 133] und ITcEventLog [▶ 130]::EnumLoggedEvents [▶ 135] der Klasse

TcEventLog [▶ 129] zurückgegeben werden.

9.3.6.4 Clone

ITcEnumEvents [▶ 152]::Clone

DIese Methode bildet das Aufzählungsobjekt nach.

```
HRESULTClone([out] ITcEnumEvents** ppEnum);
```

Parameter

[out] Pointer vom Typ ITcEnumEvents [▶ 152] auf einen Pointer, der eine Kopie des Aufzählungsobjekts erhält.

Rückgabe Werte

S_OK

Funktion wurde erfolgreich aufgerufen.

E_POINTER

ppEnum war kein gültiger Pointer.

E_NOTIMPL

Die Methode ist nicht implementiert.

Anmerkungen

Die Methode ist nicht implementiert für die Enum Objekte, die durch ITcEventLog [▶ 130]::EnumActiveEvents [▶ 133] und ITcEventLog [▶ 130]::EnumLoggedEvents [▶ 135] der Klasse

TcEventLog [▶ 129] zurückgegeben werden.

9.3.6.5 VbNext

ITcEnumEvents [▶ 152]::VbNext

Diese Methode gibt die nächsten Events in der Aufzählungsreihenfolge zurück.

```
HRESULTNext(
    [in] long celt,
    [in,out]SAFEARRAY(VARIANT)*elements,
    [out, retval] long *pceltFetched);
```

Parameter

celt

[in] Anzahl der angefragten Elemente der Aufzählungsphase.

ppElements

[in] Pointer auf ein safearray, das mit pceltFetched Event Objekten gefüllt wird.

pceltFetched

Gibt die Anzahl der zurückgegebenen Events zurück.

Rückgabe Werte

S_OK

Funktion wurde erfolgreich aufgerufen.

S_FALSE

Die Anzahl der zurückgegebenen Elemente war kleiner als die der Angefragten.

E_POINTER

Elements oder pceltFetched waren keine gültigen Pointer.

Visual Basic Beispielcode

```
' get the one and only event logger
Dim evtLogger As TCEVENTLOGGERLib.TcEventLog
Set evtLogger = New TCEVENTLOGGERLib.TcEventLog

' get the active event enumaration 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

Mit der ITcEnumEventsEx Schnitstelle kann durch eine Liste von Events iteriert werden.

Dieses Interface wird für eine bessere Kompatibilität mit Programmiersprachen aus der .NET Familie bereitgestellt.Es implementiert IEnumVARIANT um einen standard enumerator zu erzeugen. Für die Programmierung in C++ kann in einigen fällen die ursprüngliche ITcEnumEvents [▶ 152] Schnitstelle besser geignet sein

Tab. 21: Methoden und Properties in Vtable Reihenfolge

IUnknown Methode	Beschreibung
QueryInterface	Liefert einen Zeiger auf das angefragte Interface
AddRef	Inkrementiert den Referenz Zähler
Release	Dekrementiert den Referenz Zähler

ITcEnumEventsExMethods	Description
Item [▶ 157]	Liefert ein Event zurück
Add [▶ 158]	Fügt ein Event hinzu
Count [▶ 158]	Liefert die menge der Events zurück
Remove [▶ 158]	Entfern ein Event.
GetEnumerator	Returns a standard IEnumerator object

Visual Studio 2005 C# Sample Code:

```
foreach(TcEventevtinnewTcEventLog().EnumActiveEventsEx())
  evt.Reset();
```

9.3.7.1 Item

ITcEnumEvents [▶ 152]

Diese Methode liefert ein Event aus der Liste.

```
HRESULT Item(
    [in] int Index,
    [out, retval]ITcEvent** pVal );
```

Parameter

Index

[in] Index des angefragten Events

pVal

[out] Element an der angefragten position.

return Values

S_OK

Der Aufruf ist erfolgreich abgeschlossen.

Anmerkungen

Je nach Programmiersprache, die Sie verwenden, möchten Sie vielleicht lieber den Standard-Enumerator des Objekts anstelle seiner Schnittstellenmethoden verwenden.

9.3.7.2 Add

ITcEnumEvents [▶ 152]

Fügt ein Event hinzu.

```
HRESULT Add(
    [out] ITcEvent* pEvent);
```

Parameter

[out] Zeiger zu einem ITcEvent [▶ 159] welches der Liste hinzugefügt werden soll.

Return Value

S_OK

Der Aufruf ist erfolgreich abgeschlossen

Remarks

Die Enumeratoren sollten mittels der Methoden ITcEventLog [▶ 130]::EnumActiveEventsEx [▶ 134] und ITcEventLog [▶ 130]::EnumLoggedEventsEx [▶ 136] erzeugt und keine Events nachträglich hinzugefügt werden

9.3.7.3 Count

ITcEnumEvents [▶ 152]

Diese Methode liefert die Anzahl der Events

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

Parameter

pVal

[out, retval] Zeiger auf eine long Variable welche den Wert erhält.

return Values

S_OK

Der Aufruf ist erfolgreich abgeschlossen.

Anmerkungen

Je nach Programmiersprache, die Sie verwenden, möchten Sie vielleicht lieber den Standard-Enumerator des Objekts anstelle seiner Schnittstellenmethoden verwenden.

9.3.7.4 Remove

ITcEnumEvents [▶ 152]

Entfernt ein Event aus der Liste

```
HRESULT Remove( [in]int Index );
```

Parameter

Index

[in] Index des angefragten Events.

return Values

S_OK

Der Aufruf ist erfolgreich abgeschlossen.

Anmerkungen

Je nach Programmiersprache, die Sie verwenden, möchten Sie vielleicht lieber den Standard-Enumerator des Objekts anstelle seiner Schnittstellenmethoden verwenden.

9.3.8 ITcEvent

Die Schnittstelle ITcEvent ermöglicht den Zugriff auf die Eigenschaften eines einzelnen Alarms. Die Schnittstelle leitet sich von **IDispatch** ab, so dass sie von allen Sprachen verwendet werden kann, die die COM Automation Schnittstelle unterstützen. IDispatch selbst leitet sich von IUnknown ab.

Tab. 22: Methoden und Eigenschaften in

IUnknown Methoden	Beschreibung
QueryInterface	Gibt einen Pointer zur angefragten Schnittstelle zurück
AddRef	Inkrementiert den Referenzzähler
Release	Dekrementiert den Referenzzähler

IDispatch Methoden	Beschreibung
GetIDsOfNames	Mappt einen einzelnen Teilnehmernamen und einen Satz optionaler Parameternamen mit einem entsprechenden Satz gesendeter Integer Identifier (DISPIDs). Diese können dann als Folgeabrufe an **IDispatch::Invoke** benutzt werden.
GetTypeInfo	Ruft die Typ Informationen für ein Objekt ab.
GetTypeInfoCount	Ruft die Anzahl der Typ Informations-Schnittstellen ab, die ein Objekt zur Verfügung stellt, entweder 0 oder 1.
Invoke	Stellt den Zugriff auf Eigenschaften und Methoden, ausgestellt durch ein Objekt, zur Verfügung.

ITcEvent Methoden u. Eigenschaften	Beschreibung
GetMsgString [▶ 160]	Gibt den formatierten Meldestring für die angefragte Sprache zurück.
Id [▶ 161]	Gibt die Event Id zurück.
SrcId [▶ 162]	Gibt die Id der Event source zurück.
InvokeId [▶ 162]	Gibt die invoke id zurück.
Class [▶ 163]	Gibt die Event Klasse zurück.
Priority [▶ 163]	Gibt die Event Priorität zurück.
Flags [▶ 164]	Gibt die Event Flags zurück.
Ms [▶ 165]	Gibt die tausendstel Sekunde des Event Datums zurück.

ITcEvent Methoden u. Eigenschaften	Beschreibung
SourceName [▶ 165]	Gibt den Namen der Events source für die angefragte Sprache zurück.
FmtProgId [▶ 166]	Gibt die ProgId für den Event Formatter zurück.
Date [▶ 167]	Gibt das Event Datum und die Zeit zurück.
GetData [▶ 167]	Gibt die Event Daten Parameter zurück.
GetHeader [▶ 168]	Gibt den Event Header als Safearray zurück.
Confirm [▶ 169]	Bestätigt den Alarm.
Reset [▶ 170]	Setzt den Alarm zurück.
State [▶ 171]	Gibt den Event Status zurück.
MustConfirm [▶ 171]	Gibt den Alram zurück, wenn der Alarm bestätigt werden muss.
DateConfirmed [▶ 172]	Gibt das Datum und die Zeit der Bestätigung zurück.
MsConfirmed [▶ 173]	Gibt die tausendstel Sekunde der Bestätigungszeit zurück.
DateReset [▶ 174]	Gibt das Datum und die Zeit zurück, zu der das Event zurückgesetzt wurde.
MsReset [▶ 174]	Gibt die tausendstel Sekunde der Rücksetzzeit zurück.
UserData [▶ 175]	Gibt die Benutzerdaten zurück.
EnumDocLinks [▶ 176]	Gibt ein Aufzählungsobjekt zurück, das zur Aufzählung aufgelisteter Dokumentverknüpfungen verwendet werden kann.
TcId	nicht implementiert.
Signal [▶ 177]	Signalisiert den Alarm.

9.3.8.1 GetMsgString

ITcEvent [▶ 159]

Die Methode GetMsgString gibt den formatierten Meldungsstring für die angefragte Sprache zurück. Intern ruft sie die Methode TcLogFormatter::GetCompleteString des Formatters auf, der zugeordnet wurde, als das Event von einem COM Client ausgegeben wurde, oder durch einen ADS Aufruf an das Objekt TcEventLog [▶ 129] des TcEventLoggers.

```
HRESULT GetMsgString([in] long langId, [out,retval] BSTR* msg);
```

Parameter

langId

[in] Die Sprach-Id der angefragten Sprache. Die gewünschte Sprache sollte mit *LCIDs markiert werden. Die nächste Tabelle zeigt ein Beispiel für einige Sprach-Ids. In der Konfiguration des Event Formatters werden die Sprachen mit der gleichen Sprach-Id gekennzeichnet.

LCID	Beschreibung
1031	Deutsch
1033	US Englisch
1034	Spanisch
1036	Französisch

msg

[out, retval] Pointer auf ein BSTR String, der den formatierten String für die angefragte Sprache zurückgibt.

Rückgabe Werte

S_OK

Funktion wurde erfolgreich aufgerufen.

E_POINTER

msg war kein gültiger Pointer.

Visual Basic Beispielcode

```
' get the one and only event logger
Dim evtLogger As TCEVENTLOGGERLib.TcEventLog
Set evtLogger = New TCEVENTLOGGERLib.TcEventLog

' get the most resent 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
```

Anmerkungen

Die meisten standardisierten Formatter wie der XML basierende Formatter (TcEventFormatter.TcXmlFormatter) werden versuchen, den String für die Standard-Sprache zurückzugeben, falls die angefragte Sprache nicht in der Konfiguration des Formatters existiert.
*LCID: weitere Informationen finden Sie in der MSDN Library.

9.3.8.2 Id

ITcEvent [▶ 159]

Diese Eigenschaft gibt die Event Id zurück. Zusammen mit der Source Id beschreibt die Event Id ein einzelnes Event.

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

Parameter

pVal

[out, retval] Pointer auf einen long Wert, der die Event id erhält.

Rückgabe Werte

S_OK

Funktion wurde erfolgreich aufgerufen.

E_POINTER

pVal war kein gültiger Pointer.

Visual Basic Beispielcode

```
' get the one and only event logger
Dim evtLogger As TCEVENTLOGGERLib.TcEventLog
Set evtLogger = New TCEVENTLOGGERLib.TcEventLog

' get the most resent 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 [▶ 159]

Diese Eigenschaft gibt die Event Source Id zurück. Zusammen mit der Event Id beschreibt die Source Id ein einzelnes Event.

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

Parameter

pVal

[out, retval] Pointer auf einen long Wert, der die Source id erhält.

Rückgabe Werte

S_OK
Funktion wurde erfolgreich aufgerufen.
E_POINTER
pVal war kein gültiger Pointer.

Visual Basic Beispielcode

```
' get the one and only event logger
Dim evtLogger As TCEVENTLOGGERLib.TcEventLog
Set evtLogger = New TCEVENTLOGGERLib.TcEventLog

' get the most resent 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 InvokeId

ITcEvent [▶ 159]

Diese Eigenschaft gibt die Invoke Id zurück.

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

Parameter

pVal

[out, retval] Pointer auf einen long Wert, der die InvokeId erhält.

Rückgabe Werte

S_OK

Funktion wurde erfolgreich aufgerufen.

E_POINTER

pVal war kein gültiger Pointer.

Visual Basic Beispielcode

```
' get the one and only event logger
Dim evtLogger As TCEVENTLOGGERLib.TcEventLog
Set evtLogger = New TCEVENTLOGGERLib.TcEventLog

' get the most resent 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 [▶ 159]

Diese Eigenschaft gibt die Klasse des Events, wie Alarm, Warnung, Hinweis zurück. Die Alarmklassen werden durch das Enum TcEventClass [▶ 181] beschrieben.

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

Parameter

pVal

[out, retval] Pointer auf einen longWert, der die Event Klasse beschreibt.

Rückgabe Werte

S_OK

Funktion wurde erfolgreich aufgerufen.

E_POINTER

pVal war kein gültiger Pointer.

Visual Basic Beispielcode

```
' get the one and only event logger
Dim evtLogger As TCEVENTLOGGERLib.TcEventLog
Set evtLogger = New TCEVENTLOGGERLib.TcEventLog

' get the most resent 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 [▶ 159]

Diese Eigenschaft gibt die Priorität des Events zurück. Die Event Priorität wird durch das Enum TcEventPriority [▶ 163] beschrieben.

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

Parameter

pVal

[out, retval] Pointer auf einen long Wert, der die Event Priorität erhält.

Rückgabe Werte

S_OK

Funktion wurde erfolgreich aufgerufen.

E_POINTER

pVal war kein gültiger Pointer.

Visual Basic Beispielcode

```
' get the one and only event logger
Dim evtLogger As TCEVENTLOGGERLib.TcEventLog
Set evtLogger = New TCEVENTLOGGERLib.TcEventLog

' get the most resent active event
Dim evt As TcEvent
Set evt = evtLogger.GetLastEvent

' get the event priorty
Dim prio As TcEventPriority
prio = evt.Priority
```

9.3.8.7 Flags

ITcEvent [▶ 159]

Diese Eigenschaft gibt das Event Flag zurück. Die Event Flags werden durch das Enum TcEventFlags [▶ 182] beschrieben.

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

Parameter

pVal

[out, retval] Pointer auf einen long Wert, der die Event Flags erhält.

Rückgabe Werte

S_OK

Funktion wurde erfolgreich aufgerufen.

E_POINTER

pVal war kein gültiger Pointer.

Visual Basic Beispielcode

```
' get the one and only event logger
Dim evtLogger As TCEVENTLOGGERLib.TcEventLog
Set evtLogger = New TCEVENTLOGGERLib.TcEventLog

' get the most resent 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 [▶ 159]

Diese Eigenschaft gibt den Millisekundenanteil des Events "Datum und Zeit" zurück. Das Event "Datum und Zeit" ist die Zeit, zu der das Event ausgegeben wurde.

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

Parameter

pVal

[out, retval] Pointer auf einen long Wert, der den Millisekundenanteil des Events "Datum und Zeit" erhält.

Rückgabe Werte

S_OK

Funktion wurde erfolgreich aufgerufen.

E_POINTER

pVal war kein gültiger Pointer.

Visual Basic Beispielcode

```
' get the one and only event logger
Dim evtLogger As TCEVENTLOGGERLib.TcEventLog
Set evtLogger = New TCEVENTLOGGERLib.TcEventLog

' get the most resent 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 [▶ 159]

Die Eigenschaft SourceName gibt den formatierten Sourcenamen für die angefragte Sprache zurück. Intern ruft sie die Methode TcLogFormatter::GetCompleteString des Formatters auf, der zugeordnet wurde, als das Event von einem COM Client ausgegeben wurde, oder durch einen ADS Aufruf an das Objekt TcEventLog [▶ 129] des TcEventLoggers.

```
HRESULT SourceName([in] long langId, [out,retval] BSTR* szName);
```

Parameter

langId

[in] Die Sprach Id der angefragten Sprache. Die angeforderte Sprache sollte durch *LCIDs markiert werden. Die nächste Tabelle zeigt Beispiele für einige Sprach Ids. In der Konfiguration des Event Formatters sind die Sprachen mit der gleichen Sprach Id gekennzeichnet.

LCID	Beschreibung
1031	Deutsch
1033	US Englisch
1034	Spanisch

LCID	Beschreibung
1036	Französisch

szName

[out, retval] Pointer auf ein BSTR String, der den formatierten String für das Event SourceName zurückgibt.

Rückgabe Werte

S_OK

Funktion wurde erfolgreich aufgerufen.

E_POINTER

szName war kein gültiger Pointer.

Visual Basic Beispielcode

```
' get the one and only event logger
Dim evtLogger As TCEVENTLOGGERLib.TcEventLog
Set evtLogger = New TCEVENTLOGGERLib.TcEventLog

' get the most resent 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
```

Anmerkungen

Die meisten standardisierten Formatter, wie der XML basierende Formatter (TcEventFormatter.TcXmlFormatter), werden versuchen, den String für die Standard-Sprache zurückzugeben, falls die angefragte Sprache nicht in der Konfiguration des Formatters existiert.
*LCID: weitere Informationen finden Sie in der MSDN Library.

9.3.8.10 FmtProgId

ITcEvent [▶ 159]

Die Eigenschaft gibt die Formatter ProgId zurück. Die ProgId ist die Verknüpfung mit dem ausgewählten Formatter. Der Formatter wurde ausgesucht, indem das Event vom SPS Funktionsblock (oder von einem anderen ADS Gerät), oder von einem Aufruf der Report Event Funktion wie IT [▶ 139]cEventLogC [▶ 139]::ReportEvent [▶ 139],ITcEventC3 [▶ 142]::ReportEventEx [▶ 144] oder ITcEventLog [▶ 130]::ReportEvent [▶ 131] ausgegeben wurde.

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

Parameter

pVal

[out, retval] Pointer auf ein BSTR String, der die Formatter ProgId erhält.

Rückgabe Werte

S_OK

Funktion wurde erfolgreich aufgerufen.

E_POINTER

pVal war kein gültiger Pointer.

Visual Basic Beispielcode

```
' get the one and only event logger
Dim evtLogger As TCEVENTLOGGERLib.TcEventLog
Set evtLogger = New TCEVENTLOGGERLib.TcEventLog

' get the most resent 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 [▶ 159]

Die Eigenschaft gibt das Event "Datum und Zeit" zurück. Das Event "Datum und Zeit" ist die Zeit, zu der das Event ausgegeben wurde.

```
HRESULT Date([out, retval] DATE *pVal);
```

Parameter

pVal

[out, retval] Pointer auf einen long Wert, der das Event Datum und Zeit erhält.

Rückgabe Werte

S_OK

Funktion wurde erfolgreich aufgerufen.

E_POINTER

pVal war kein gültiger Pointer.

Visual Basic Beispielcode

```
' get the one and only event logger
Dim evtLogger As TCEVENTLOGGERLib.TcEventLog
Set evtLogger = New TCEVENTLOGGERLib.TcEventLog

' get the most resent 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 [▶ 159]

Die Eigenschaft gibt ein Safearray von Daten zurück, das an das Event angehangen wurde, als es eingeloggt war. Die Daten können durch den Formatter in die Event Meldung eingefügt werden. Die Daten wurden hinzugefügt, indem das Event vom SPS Funktionsblock (oder einem anderen ADS Gerät) oder von einem Aufruf der Report Event Funktion wie IT [▶ 139]cEventLogC [▶ 139]::ReportEvent [▶ 139],ITcEventC3 [▶ 142]::ReportEventEx [▶ 144] oder ITcEventLog [▶ 130]::ReportEvent [▶ 131] ausgegeben wurde.

```
HRESULT GetData([in,out] SAFEARRAY(VARIANT)* data);
```

Parameter

data

[out, retval] Pointer auf ein Safearray von Varianten, das die Event Daten erhält.

Rückgabe Werte

S_OK

Funktion wurde erfolgreich aufgerufen.

E_POINTER

Daten waren kein gültiger Pointer.

Visual Basic Beispielcode

```
' get the one and only event logger
Dim evtLogger As TCEVENTLOGGERLib.TcEventLog
Set evtLogger = New TCEVENTLOGGERLib.TcEventLog

' get the most resent active event
Dim evt As TcEvent
Set evt = evtLogger.GetLastEvent

' get the event data
Dim data() As Variant
Call evt.GetData(data)

' loop though 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 [▶ 159]

Die Eigenschaft gibt ein Safearray zurück, das die gleiche Information wie der TcEventHeader [▶ 184] enthält. Die Header Daten wurden hinzugefügt, indem das Event vom SPS Funktionsblock (oder einem anderen ADS Gerät) oder von einem Aufruf der Report Event Funktion wie IT [▶ 139]cEventLogC [▶ 139]::ReportEvent [▶ 139],ITcEventC3 [▶ 142]::ReportEventEx [▶ 144] oder ITcEventLog [▶ 130]::ReportEvent [▶ 131] ausgegeben wurde.

```
HRESULT GetHeader([in,out] SAFEARRAY(VARIANT)* data);
```

Parameter

data

[in] Pointer auf ein Safearray mit den Grenzen 0 bis 9, das den Event Haeder darstellt. Das Array repräsentiert die gleiche Information wie der TcEventHeader. Die Einträge des Arrays sind in der folgenden Tabelle erklärt.

Index	TcEventHeader [▶ 184] Eintrag	Datentyp	Beschreibung
0	**nClass**	TcEventClass [▶ 181]	Die Klasse des Events, wie Alarm, Warnung, Hinweis.
1	**nPriority**	TcEventPriority [▶ 183] **(long)**	Die Event Priorität. Ab jetzt ist die Priorität immer: TcEventPriority.TCEVENTPRIO_IMPLICIT=0
2	**dwFlags**	TcEventFlags [▶ 182] **(long)**	Die Flags regeln das Verhalten des Alarms. Das kann mit einer OR Verknüpfung kombiniert werden.

Index	TcEventHeader [▶ 184] Eintrag	Datentyp	Beschreibung
3	dwUserData	long	Reservevariable, die vom Nutzer verwendet werden kann.
4	nId	long	Die Event Id, die eindeutig diesen Eventtyp für diesen einzelnen Event Source darstellt.
5	nInvokeId	long	
6	fDate	VARIANT-DATE	Datum und Zeit zu dem dieses Event auftritt.
7	nMs	long	Der Millisekundenanteil des Events Datum.
8	varSource	variant	Die Source Id SourceName. Das Event Source wird verwendet, um verschiedene Geräte zu unterscheiden, wie I/O Event, oder verscheidene Teile des SPS Programms.
9	szFmtProgId	BSTR string	Die PrgId des Formatters. Wenn der Standard XML Formatter benutzt werden soll, muss der String auf "TcEventFormatter.TcXmlFormatter" gesetzt werden.

Rückgabewerte

S_OK

Funktion wurde erfolgreich aufgerufen.

E_POINTER

Daten waren kein gültiger Pointer.

Visual Basic Beispielcode

```
' get the one and only event logger
Dim evtLogger As TCEVENTLOGGERLib.TcEventLog
Set evtLogger = New TCEVENTLOGGERLib.TcEventLog

' get the most resent 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
```

Anmerkungen

Für C++ und Visual Basic Programmierer ist es bequemer, die MethodeITcEventC [▶ 139]**::GetHeader** zu benutzen.

9.3.8.14 Confirm

ITcEvent [▶ 159]

Diese Methode wird zur Bestätigung eines Alarms verwendet. Alarme, die vom SPS Funktionsblock (oder einem anderen ADS Gerät) oder von einem Aufruf des Reports EventFunktion wie IT [▶ 139]cEventLogC [▶ 139]::ReportEvent [▶ 139], ITcEventC3 [▶ 142]::ReportEventEx [▶ 144] oder ITcEventLog [▶ 130]::ReportEvent [▶ 131] mit den TcEventFlags [▶ 182] TCEVENTFLAG_REQMUSTCON ausgegeben worden sind, müssen durch den Aufruf der ConfirmMethode des COM clients oder aus der SPS bestätigt werden (confirm).

```
HRESULT Confirm([in] long code);
```

Parameter

pVal

[in] Bestätigungscode, der durch das Enum TcEventConCodes beschrieben wird.

Rückgabe Werte

S_OK

Funktion wurde erfolgreich aufgerufen.

TCEVENTERR_NOCONFIRM

Der Fehler war kein beweisbarer Fehler (TcEventFlags TCEVENTFLAG_REQMUSTCON war nicht gesetzt).

Visual Basic Beispielcode

```
' get the one and only event logger
Dim evtLogger As TCEVENTLOGGERLib.TcEventLog
Set evtLogger = New TCEVENTLOGGERLib.TcEventLog

' get the most resent active event
Dim evt As TcEvent
Set evt = evtLogger.GetLastEvent

' confirm the alarm
Call evt.Confirm(TcEventConCodes.TCEVENTCON_OK)
```

Anmerkungen

Nach diesem Methodenaufruf wird der Event Logger das Event OnConfirmEvent [▶ 147] auf allen Clients, die die Event-Schnittstelle _ITcEventLogEvents [▶ 146] (VB: Dim WithEvents) implementieren, erhöhen. Falls der Alarm durch die SPS ausgelöst wurde wird das TCEVENTSTATE_CONFIRMED am EventState Ausgang des SPS Funktionsblocks gekennzeichnet.

9.3.8.15 Reset

ITcEvent [▶ 159]

Diese Methode wird zum Rücksetzen eines Events verwendet. Events, die vom SPS Funktionsblock (oder einem anderen ADS Gerät), oder einem Aufruf der Report Event Funktion wie IT [▶ 139]cEventLogC [▶ 139]::ReportEvent [▶ 139],ITcEventC3 [▶ 142]::ReportEventEx [▶ 144] oder ITcEventLog [▶ 130]::ReportEvent [▶ 131] ausgegeben worden sind, könnten durch diesen Methodenaufruf zurückgestetzt werden. Sie sind direkt von dem SPS Funktionsblock, der diesen Alarm ausgegeben hat.

Wenn das Event zurückgesetzt ist, wird den Zeitstempeln **DateReset + MsReset** das aktuelle Datum und die aktuelle Zeit zugewiesen.

```
HRESULT Reset();
```

Parameter

Rückgabe Werte

S_OK

Funktion wurde erfolgreich aufgerufen.

Visual Basic Beispielcode

```
' get the one and only event logger
Dim evtLogger As TCEVENTLOGGERLib.TcEventLog
Set evtLogger = New TCEVENTLOGGERLib.TcEventLog

' get the most resent active event
```

```
Dim evt As TcEvent
Set evt = evtLogger.GetLastEvent

' reset the alarm
Call evt.Reset
```

Anmerkungen

Nach diesem Methodenaufruf wird der Event Logger das Event OnResetEvent [▶ 148] auf allen Clients, die die Event-Schnittstelle _ITcEventLogEvents [▶ 146] (VB: Dim WithEvents) implementieren, erhöhen. Falls der Alarm durch die SPS ausgelöst wurde wird das TCEVENTSTATE_RESET am EventState Ausgang des SPS Funktionsblocks gekennzeichnet.

9.3.8.16 State

ITcEvent [▶ 159]

Diese Eigenschaft gibt den Event Status zurück. Der Event Status wird durch das Enum TcEventStates [▶ 183] beschrieben.

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

Parameter

pVal

[out, retval] Pointer auf einen long Wert, der den Event Status mit den Werten des Enum TcEventStates erhält.

Rückgabe Werte

S_OK

Funktion wurde erfolgreich aufgerufen.

E_POINTER

pVal war kein gültiger Pointer.

Visual Basic Beispielcode

```
' get the one and only event logger
Dim evtLogger As TCEVENTLOGGERLib.TcEventLog
Set evtLogger = New TCEVENTLOGGERLib.TcEventLog

' get the most resent 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 [▶ 159]

Die Eigenschaft gibt den Alarm zurück, wenn der Alarm bestätigt werden muss. Dieses Flag ist TRUE, wenn das Event vom SPS Funktionsblock (oder einem anderen ADS Gerät) oder durch einen Aufruf einer Report Event Funktion wie IT [▶ 139]cEventLogC [▶ 139]::ReportEvent [▶ 139], ITcEventC3 [▶ 142]::ReportEventEx [▶ 144] oder ITcEventLog [▶ 130]::ReportEvent [▶ 131] mit den TcEventFlags [▶ 182] TCEVENTFLAG_REQMUSTCON ausgegeben wurde.

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

Parameter

pVal

[out, retval] Pointer auf einen BOOL Wert, der das Bestätigungs-Flag erhält. Es ist FALSE, wenn der Wert 0 ist, und TRUE, wenn der Wert ungleich 0 ist.

Rückgabe Werte

S_OK

Funktion wurde erfolgreich aufgerufen.

E_POINTER

pVal war kein gültiger Pointer.

Visual Basic Beispielcode

```
' get the one and only event logger
Dim evtLogger As TCEVENTLOGGERLib.TcEventLog
Set evtLogger = New TCEVENTLOGGERLib.TcEventLog

' get the most resent 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
```

Anmerkungen

Der zurückgegebene Wert ist nicht vom Typ VARIANT_BOOL des Visual Basic Typs BOOLEAN. In Visual Basic tritt er als long Wert auf. Es muss überprüft werden, ob der Wert ungleich 0 ist, wenn das Flag gesetzt ist.

9.3.8.18 DateConfirmed

ITcEvent [▶ 159]

Diese Eigenschaft gibt das Datum und die Zeit der Event Bestätigung (confirm) zurück. Die Event Bestätigung Datum und Zeit ist die Zeit, wenn das Event vom SPS Funktionsblock durch das Setzen des **EventQuit=TRUE** (oder eines anderen ADS Geräts) oder durch den Aufruf von ITcEvent [▶ 159]**::**Confirm [▶ 169] bestätigt wird.

```
HRESULT DateConfirmed([out, retval]
DATE *pVal);
```

Parameter

pVal

[out, retval] Pointer auf einen long Wert, der die Event Bestätigung Datum und Zeit erhält.

Rückgabe Werte

S_OK
Funktion wurde erfolgreich aufgerufen
E_POINTER
pVal war kein gültiger Pointer

Visual Basic Beispielcode

```
' get the one and only event logger
Dim evtLogger As TCEVENTLOGGERLib.TcEventLog
Set evtLogger = New TCEVENTLOGGERLib.TcEventLog

' get the most resent 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
```

Anmerkungen

Die Event Bestätigung Datum und Zeit wird nur bei den Events gesetzt, die bestätigt werden müssen, siehe ITcEvent [▶ 159]::.MustConfirm [▶ 171]

9.3.8.19 MsConfirmed

ITcEvent [▶ 159]

Diese Eigenschaft gibt die Millisekunde der Event Bestätigung Datum und Zeit an. Die Event Bestätigung Datum und Zeit ist die Zeit, zu der das Event vom SPS Funktionsblock durch Setzen des **EventQuit=TRUE** (oder eines anderen ADS Geräts) oder von einem Aufruf ITcEvent [▶ 159]::Confirm [▶ 169] bestätigt wird.

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

Parameter

pVal

[out, retval] Pointer auf einen long Wert, der die tausendstel Sekunde der Event Bestätigung Datum und Zeit erhält.

Rückgabe Werte

S_OK
Funktion wurde erfolgreich aufgerufen
E_POINTER
pVal war kein gültiger Pointer.

Visual Basic Beispielcode

```
' get the one and only event logger
Dim evtLogger As TCEVENTLOGGERLib.TcEventLog
Set evtLogger = New TCEVENTLOGGERLib.TcEventLog

' get the most resent 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
```

Anmerkungen

Die tausendstel Sekunde der Event Bestätigung Datum und Zeit ist genau auf diese Events gesetzt, die bestätigt werden müssen, siehe ITcEvent [▶ 159]::.MustConfirm [▶ 171]

9.3.8.20 DateReset

ITcEvent [▶ 159]

Diese Eigenschaft gibt das Event Reset Datum und Zeit zurück. Das Event Reset Datum und Zeit ist der Zeitpunkt, an dem das Event vom SPS Funktionsblock durch Setzen des **Events** von **TRUE** auf **FALSE** (oder von einen anderen ADS Gerät) oder von einem Aufruf von ITcEvent [▶ 159]::Reset [▶ 170] zurückgesetzt wird.

```
HRESULT DateReset([out, retval] DATE *pVal);
```

Parameter

pVal

[out, retval] Pointer auf einen Long Wert, der das Event Reset Datum und Zeit erhält.

Rückgabe Werte

S_OK
Funktion wurde erfolgreich aufgerufen.
E_POINTER
pVal war kein gültiger Pointer.

Visual Basic Beispielcode

```
' get the one and only event logger
Dim evtLogger As TCEVENTLOGGERLib.TcEventLog
Set evtLogger = New TCEVENTLOGGERLib.TcEventLog

' get the most resent 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 [▶ 159]

Die Eigenschaft gibt den Millisekundenteil des Events Reset Datum und Zeit. Das Event Reset Datum und Zeit ist die Zeit, wenn das Event vom SPS Funktionsblock zurückgesetzt wird, durch Setzen des **Event** von **TRUE** auf **FALSE** (oder eines anderen ADS Geräts) oder durch den Aufruf von ITcEvent [▶ 159]::.Reset [▶ 170]

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

Parameter

pVal

[out, retval] Pointer auf eine long Wert, der den Millisekundenteil des Event Datum und Zeit erhält.

Rückgabe Werte

S_OK

Funktion wurde erfolgreich aufgerufen.

E_POINTER

pVal war kein gültiger Pointer.

Visual Basic Beispielcode

```
' get the one and only event logger
Dim evtLogger As TCEVENTLOGGERLib.TcEventLog
Set evtLogger = New TCEVENTLOGGERLib.TcEventLog

' get the most resent 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 [▶ 159]

Die Eigenschaft gibt die benutzerdefinierten Daten des Events zurück. Die Benutzerdaten wurden zugeordnet, als das Event vom SPS Funktionsblock (oder eines anderen ADS Geräts) oder durch den Aufruf der Report EventFunction wie IT [▶ 139]cEventLogC [▶ 139]::ReportEvent [▶ 139], ITcEventC3 [▶ 142]::ReportEventEx [▶ 144] oder ITcEventLog [▶ 130]::ReportEvent [▶ 131] mit den TcEventFlags [▶ 182] ausgegeben wurde.

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

Parameter

pVal

[out, retval] Pointer auf einen long Wert, der die Benutzerdaten enthält.

Rückgabe Werte

S_OK

Funktion wurde erfolgreich aufgerufen.

E_POINTER

pVal war kein gültiger Pointer.

Visual Basic Beispiel-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
```

Anmerkungen

Der Rückgabewert ist nicht vom Typ VARIANT_BOOL sondern vom Visual Basic Typ BOOLEAN. In Visual Basic kommt er als Long-Wert vor. Wir müssen prüfen, ob er ungleich 0 ist, wenn das Flag gesetzt ist.

9.3.8.23 EnumDocLinks

ITcEvent [▶ 159]

Gibt ein Aufzählungsobjekt zurück, das zur Wiederholung aufgelisteter Dokumentverknüpfungen verwendet wird. Dokumentverknüpfungen sind der Pfad und die Dateinamen der Dokumente ,wie HTML Seiten und Bilder, die eine genauere Beschreibung über einen einzelnen Alarm für die angewählte Sprache ausgeben. Intern beziehen sich die EnumDocLinks auf die Formatter Methode ITcLogFormatter::EnumDocLinks. Die Dokumentverknüpfungen müssen in der Formatter Konfiguration des Events eingerichtet sein.

```
HRESULTEnumDocLinks([in] long
langId, [out,retval] IUnknown** ppEnum);
```

Parameter

langId

[in]

Die Sprach-Id der angefragten Sprache. Die gewünschte Sprache sollte mit *LCIDs markiert werden. Die nächste Tabelle zeigt ein Beispiel für einige Sprach-Ids. In der Konfiguration des Event Formatters werden die Sprachen mit der gleichen Sprach-Id gekennzeichnet.

LCID	Beschreibung
1031	Deutsch
1033	US Englisch
1034	Spanisch
1036	Französisch

ppEnum

[out, retval] Pointer auf den ITcEnumEventDocLink Schnittstellen.Pionter, der die Aufzählungsobjekte der Doumentverknüpfungen erhält.

Rückgabe Werte

S_OK

Funktion wurde erfolgreich aufgerufen.

E_POINTER

ppEnum war kein gültiger Pointer.

Anmerkungen

Diese Methode funktioniert nicht mit Visual Basic.

*LCID: weitere Informationen finden Sie in der MSDN Library.

9.3.8.24 Signal

ITcEvent [▶ 159]

Diese Methode wird verwendet, um ein rückgesetztes Event erneut anzuzeigen. Diese funktioniert nur bei einem Alarm, der eine Bestätigung benötigt, siehe ITcEvent [▶ 159]**::.**MustConfirm [▶ 171]

```
HRESULT Signal();
```

Parameter

Rückgabe Werte

S_OK

Funktion wurde erfolgreich aufgerufen.

Visual Basic Beispielcode

```
' get the one and only event logger
Dim evtLogger As TCEVENTLOGGERLib.TcEventLog
Set evtLogger = New TCEVENTLOGGERLib.TcEventLog

' get the most resent active event
Dim evt As TcEvent
Set evt = evtLogger.GetLastEvent

' signal the alarm
Call evt.Signal
```

Anmerkungen

Nach diesem Methodenaufruf wird der Event Logger das Event OnSignalEvent [▶ 149] auf allen Clients, die die Event-Schnittstelle_ITcEventLogEvents [▶ 146] (VB: Dim WithEvents) implementieren, erhöhen.

9.3.9 ITcEventC

Die Schnittstelle ITcEvent gibt einen Zugriff auf die Eigenschaften eines einzelnen Alarms. Die Schnittstelle leitet sich von **IUnknown ab,** so dass sie von allen Sprachen verwendet werden kann, die die COM Custom Interfaces unterstützen.

Tab. 23: Methoden und Eigenschaften

IUnknown Methoden	Beschreibung
QueryInterface	Gibt einen Pointer zur angefragten Schnittstelle zurück.
AddRef	Inkrementiert den Referenzzähler.
Release	Dekrementiert den Referenzzähler.

ITcEventC Methoden	Beschreibung
GetHeader [▶ 178]	Gibt den Event Header zurück, der die meisten der Eventparameter in einem Bündel erfasst.

9.3.9.1 GetHeader

ITcEventC [▶ 177]::GetHeader

Diese Methode gibt den Event Header zurück, der die meisten Eventparameter in einem Bündel erfasst.

```
HRESULT GetHeader([out]TcEventHeader* pHead);
```

Parameter

pHead

[out] Pointer auf ein Objekt des Typs TcEventHeader [▶ 184]. Das Objekt stellt die Alarm Konfiguration dar.

Rückgabe Werte

S_OK

Funktion wurde erfolgreich aufgerufen.

E_POINTER

pVal war kein gültiger Pointer.

Visual Basic Beispielcode

```
' get the one and only event logger
Dim evtLogger As TCEVENTLOGGERLib.TcEventLog
Set evtLogger = New TCEVENTLOGGERLib.TcEventLog

' get the most resent active event
Dim evt As ITcEventC
Set evt = evtLogger.GetLastEvent

' get the event header
Dim head As TcEventHeader
Call evt.GetHeader(head)
```

9.3.10 ITcEnumEventDocLink

Die Schnittstelle ITcEnumEventDocLink wird zur Aufzählung der Dokumentverbindungen für ein einzelnes Event für die gewählte Sprache verwendet. Die Schnittstelle leitet sich von **IUnknown** ab, so dass sie von allen Sprachen verwendet werden kann, die COM Custom Schnittstellen unterstützen.

Tab. 24: Methoden in

IUnknown Methoden	Beschreibung
QueryInterface	Gibt einen Pointer zur angefragten Schnittstelle zurück
AddRef	Inkrementiert den Referenzzähler
Release	Dekrementiert den Referenzzähler

ITcEnumEvent-DocLinkMethoden	Beschreibung
Next [▶ 179]	Gibt die nächste Nummer desTcEventDocLink [▶ 183] Objekts in der Aufzählungsreihenfolge zurück.
Skip [▶ 179]	Überspringt in der Aufzählungsreihenfolge eine Anzahl vonTcEventDocLink [▶ 183] Objekten.
Reset [▶ 180]	Setzt die Aufzählungsreihenfolge auf das erste Element zurück.
Clone [▶ 180]	Bildet das Aufzählungsobjekt nach.

9.3.10.1 Next

Gibt die nächste Nummer desTcEventDocLink [▶ 183] Objekts in der Aufzählungsreihenfolge zurück.

```
HRESULT Next(
    [in] long celt,
    [out, size_is(celt), length_is(*pceltFetched)] TcEventDocLink *ppElements,
    [out, retval] long *pceltFetched);
```

Parameter

celt

[in] Anzahl der angefragten Elemente in der Aufzählungsreihenfolge.

ppElements

[out, size_is(celt), length_is(*pceltFetched)] Pointer auf das erste Element des Arrays mit TcEventDocLink Objekten. Gibt die Anzahl der Dokumentverknüpfungen durch pceltFetched zurück.

pceltFetched

Gibt die Anzahl der zurückgegebenen Dokumentverknüpfungen zurück.

Return Values

S_OK

Funktion wurde erfolgreich aufgerufen.

S_FALSE

Die Anzahl der zurückgegebenen Elemente ist geringer als die der Angefragten.

E_POINTER

Elemente oder pceltFetched sind keine gültigen Pointer.

Anmerkungen

Die Methode kann nicht aus Visual Basic aufgerufen werden.

Sehen Sie dazu auch

ITcEnumEventDocLink [▶ 178]

9.3.10.2 Skip

Diese Methode überspringt in der Aufzählungsreihenfolge eine Anzahl vonTcEventDocLink [▶ 183] Objekten.

```
HRESULT Skip(
    [in] long cSkipElem);
```

Parameter

cSkipElem

[in] Anzahl der zu überspringenden Elemente.

Rückgabe Werte

S_OK

Die Methode wurde erfolgreich aufgerufen.

S_FALSE

Die Anzahl der übersprungenen Elemente ist ungleich cSkipElem.

E_NOTIMPL

Die Methode ist nicht implementiert.

Anmerkungen

Sehen Sie dazu auch

ITcEnumEventDocLink [▶ 178]

9.3.10.3 Reset

Diese Methode setzt die Aufzählungsreihenfolge auf das erste Element zurück.

```
HRESULT Reset(
    void);
```

Parameter

Rückgabe Werte

S_OK

Funktion wurde erfolgreich aufgerufen.

S_FALSE

Die Anzahl der übersprungenen Elemente ist nicht identisch mit cSkipElem.

E_NOTIMPL

Die Methode ist nicht implementiert.

Sehen Sie dazu auch

ITcEnumEventDocLink [▶ 178]

9.3.10.4 Clone

Diese Methode bildet das Aufzählungsobjekt nach.

```
HRESULT Clone(
    [out] ITcEnumEventDocLink** ppEnum);
```

Parameter

[out] Pointer vom Typ ITcEnumEventDocLink [▶ 178] auf einen Pointer, der eine Kopie des Aufzählungsobjektes erhält.

Rückgabe Werte

S_OK

Methode wurde erfolgreich aufgerufen.

E_POINTER

ppEnum war kein gültiger Pointer.

E_NOTIMPL

Die Methode ist nicht implementiert.

Sehen Sie dazu auch

ITcEnumEventDocLink [▶ 178]

9.4 Enums

9.4.1 TcEventClass

Dieses Enum beschreibt die Event Klasse. Die Event Klasse kann beim Ausgeben eines Alarmsignals gesetzt werden und kann von den Clients verwendet werden, um zwischen verschiedenen Alarmsignalen zu unterscheiden. Der Formatter, zum Beispiel, zeigt für die verschiedenen Alarmklassen verschiedene Icons an.

```
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
};
```

Tab. 25: Parameter

Eintrag	Beschreibung
TCEVENTCLASS_NONE	Keine bestimmte Event Klasse
TCEVENTCLASS_MAINTENANCE	Das Event ist als Wartungsalarm klassifiziert.
TCEVENTCLASS_MESSAGE	Das Event ist als Meldung klassifiziert.
TCEVENTCLASS_HINT	Das Event ist als Hinweis klassifiziert.
TCEVENTCLASS_STATEINFO	Das Event ist als Statusinformation klassifiziert.
TCEVENTCLASS_INSTRUCTION	Das Event ist als Anleitung klassifiziert.
TCEVENTCLASS_WARNING	Das Event ist als Warnung klassifiziert.
TCEVENTCLASS_PARAMERROR	Das Event ist als Parameterfehler klassifiziert.
TCEVENTCLASS_MAX	Die Event Klasse Max wird nicht als Classifier benutzt, sondern beschreibt den maximalen Wert des Enums.

9.4.2 TcEventConCodes

Dieses Enum beschreibt die Event Bestätigungscodes (confirm). Der Bestätigungscode wird zur Beschreibung der Event Bestätigung verwendet.

```
enum TcEventConCodes
{
    TCEVENTCON_OK =0,
    TCEVENTCON_ABORT =1,
    TCEVENTCON_IGNORE =2,
    TCEVENTCON_RETRY =3
};
```

Tab. 26: Parameter

Eintrag	Beschreibung
TCEVENTCON_OK	Die Bestätigung war in Ordnung
TCEVENTCON_ABORT	Die Bestätigung wurde abgebrochen
TCEVENTCON_IGNORE	Die Bestätigung wurde ignoriert
TCEVENTCON_RETRY	Die Bestätigung wurde wiederholt

9.4.3 TcEventFlags

Dieses Enum beschreibt die Event Klasse. Die Event Klasse kann beim Ausgeben eines Alarmsignals gesetzt werden und kann von den Clients verwendet werden, um zwischen verschiedenen Alarmsignalen zu unterscheiden. Der Formatter, zum Beispiel, zeigt für die verschiedenen Alarmklassen verschiedene Icons an.

```
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,
};
```

Tab. 27: Parameter

Eintrag	Beschreibung
TCEVENTFLAG_REQ	Setzt den Alarm, der nicht beweisbar ist.
TCEVENTFLAG_REQMUSTCON	Setzt den Alarm, der beweisbar ist.
TCEVENTFLAG_CON	Setzt die Bestätigung
TCEVENTFLAG_RESET	Setzt das Event zurück
TCEVENTFLAG_PRIOCLASS	Die Event Priorität und Klasse wird von der Formatter Konfiguration gelesen.
TCEVENTFLAG_FMTSELF	noch unbenutzt
TCEVENTFLAG_LOG	Schreibt das Event in die Liste der logged Events
TCEVENTFLAG_MSGBOX	noch unbenutzt
TCEVENTFLAG_SRCID	Es wird eine Source Id statt eines Source Namens verwendet.
TCEVENTFLAG_SELFRESET	Das Event setzt sich selber direkt zurück. Wenn TCEVENTFLAG_LOG gesetzt ist, ist das Event in der Liste der logged Alarme sichtbar.
CEVENTFLAG_TCID	unbenutzt
TCEVENTFLAG_SIGNAL	Signalisiert erneut einen Alarm
TCEVENTFLAG_ADS	Zeigt an, dass das Event via ADS erzeugt wurde.

9.4.4 TcEventPriority

Dieses Enum beschreibt die Event Priorität. Nur die implizite Priorität ist verfügbar.

```
enum TcEventPriority
{
    TCEVENTPRIO_IMPLICIT=0,
};
```

Tab. 28: Parameter

Eintrag	Beschreibung
TCEVENTPRIO_IMPLICIT	Die Standard Priorität

Anmerkungen

Die einzige Priorität, die durch ein Enum definiert wird, ist TCEVENTPRIO_IMPLICIT. Für höhere Prioritäten verwendet der Benutzer einen Integer > TCEVENTPRIO_IMPLICIT. In jeder Methode, in der eine Priorität verwendet wird, wird der Typ Long verwendet, nicht der Typ TcEventPriority.

9.4.5 TcEventStates

Dieses Enum beschreibt den Event Status.

```
enum TcEventStates
{
    CEVENTSTATE_INVALID =0x00,
    TCEVENTSTATE_SIGNALED =0x01,
    TCEVENTSTATE_RESET =0x02,
    TCEVENTSTATE_CONFIRMED =0x10,
    TCEVENTSTATE_RESETCON =0x12
}
```

Tab. 29: Parameter

Eintrag	Beschreibung
CEVENTSTATE_INVALID	Ungültiger Status, Fehler sind aufgetreten
TCEVENTSTATE_SIGNALED	Das Event ist signalisiert.
TCEVENTSTATE_RESET	Das Event ist zurückgesetzt
TCEVENTSTATE_CONFIRMED	Das Event ist bestätigt (confirmed)
TCEVENTSTATE_RESETCON	Das Event ist zurückgesetzt und bestätigt.

9.5 Strukturen

9.5.1 TcEventDocLink

Diese Struktur beschreibt eine Dokumentverknüpfung.

```
struct TcEventDocLink
{
  BSTR strDocType;
  BSTR strDocLink;
}
```

Tab. 30: Parameter

TcEventHeader Eintrag	Beschreibung
strDocType	Ein BSTR String, der den Typ der Dokumentverknüpfung beschreibt.
strDocLink	Ein BSTR String, der den Dateipfad zum Dokument enthält.

9.5.2 TcEventHeader

Diese Struktur beschreibt den Header eines Event Objekts. Mit dem Header kann der Parameter des Events konfiguriert werden.

```
struct TcEventHeader
{
    TcEventClass nClass;
    TcEventPriority nPriority;
    long dwFlags;
    long dwUserData;
    long nId;
    long nInvokeId;
    DATE fDate;
    long nMs;
    VARIANT varSource;
    BSTR szFmtProgId;
};
```

Tab. 31: Parameter

TcEventHeader Eintrag	DataType	Beschreibung
nClass	TcEventClass [▶ 181]	Die Klasse des Alarms, wie Alarm, Warnung, Hinweis.
nPriority	TcEventPriority [▶ 183] **(long)**	Die Event Priorität. Ab jetzt ist die Priorität immer: TcEventPriority.TCEVENTPRIO_IMPLICIT=0
dwFlags	TcEventFlags [▶ 182] **(long)**	Die Flags steuern das Alarmverhalten. Dies kann mit einer OR Verknüpfung kombiniert werden.
dwUserData	**long**	Eine Reserve Variable, die vom Benutzer verwendet werden kann.
nId	**long**	Die Event Id, die allein diesen Typ des Events für diesen einzelnen Event Source repräsentiert.
nInvokeId	**long**	
fDate	**VARIANT-DATE**	Datum und Zeit, wenn dieses Event auftritt.
nMs	**long**	Der Millisekundenteil des Events Datum.
varSource	**variant**	Die Source Id Source Name. Das Event Source wird zur Unterscheidung zwischen verschiedenen Geräten verwendet, wie I/O Events, oder verschiedene Teile des SPS Programms.
szFmtProgId	**BSTR string**	Die PrgId des Formatters. Wenn der Standard XML Formatter benutzt werden soll, muss der String auf "TcEventFormatter.TcXmlFormatter" gesetzt werden.

10 Beispiele

Eventkonfiguration

Beschreibung
Formatierung der Meldungstexte mit dem TcEventConfigurator [▶ 186]

TwinCAT SPS-Schnittstelle (Strukturierter Text)

Beschreibung	Quellcode
Auslösen einer einfachen Meldung [▶ 189]	https://infosys.beckhoff.com/content/1031/TcEventLogger/Resources/12333057163.zip
Auslösen einer komplexen Meldung [▶ 190]	https://infosys.beckhoff.com/content/1031/TcEventLogger/Resources/12333058571.zip
Übergabe von Parametern an die Meldung [▶ 192]	https://infosys.beckhoff.com/content/1031/TcEventLogger/Resources/12333059979.zip

Microsoft Visual Studio C++

Beschreibung	Quellcode
Einbinde von TcEventViewer-ActiveX-Control [▶ 193]	Sample01.zip (in Vorbereitung)
Konsolenanwendung - Geloggte Events via DCOM-Schnittstelle lesen [▶ 193]	https://infosys.beckhoff.com/content/1031/TcEventLogger/Resources/12333058571.zip
Connection Points (EventSink) [▶ 194]	https://infosys.beckhoff.com/content/1031/TcEventLogger/Resources/12333059979.zip
TcEventConfigurator ActiveX [▶ 194]	Sample04.zip (in Vorbereitung)

Microsoft Visual Studio C#

Beschreibung	Quellcode
C# Visualisierung - Geloggte Meldungen [▶ 195]	https://infosys.beckhoff.com/content/1031/TcEventLogger/Resources/12333057163.zip
C# Visualisierung - Aktive Meldungen [▶ 198]	https://infosys.beckhoff.com/content/1031/TcEventLogger/Resources/12333058571.zip

Microsoft Visual Basic

Beschreibung	Quellcode
Einbinden von TcEventViewer-ActiveX-Control [▶ 199]	Sample01.zip (in Vorbereitung)

Beschreibung	Quellcode
Aktive Alarme in einer benutzerdefinierten Listenansicht [▶ 199]	https://infosys.beckhoff.com/content/1031/TcEventLogger/Resources/12333058571.zip

CodeGear C++Builder 2009

Beschreibung	Quellcode
Einbinden in CodeGear C++Builder 2009 [▶ 201]	-
Konsolenanwendung - Geloggte Meldungen via DCOM-Schnittstelle [▶ 208]	https://infosys.beckhoff.com/content/1031/TcEventLogger/Resources/12333057163.zip
Konsolenanwendung - Geloggte Meldungen via ADS-Proxy-Schnittstelle [▶ 210]	https://infosys.beckhoff.com/content/1031/TcEventLogger/Resources/12333058571.zip
Aktive Alarme in einer benutzerdefinierten Listenansicht [▶ 213]	https://infosys.beckhoff.com/content/1031/TcEventLogger/Resources/12333059979.zip
Einbinden von TcEventViewer-ActiveX-Control [▶ 216]	https://infosys.beckhoff.com/content/1031/TcEventLogger/Resources/12333061387.zip

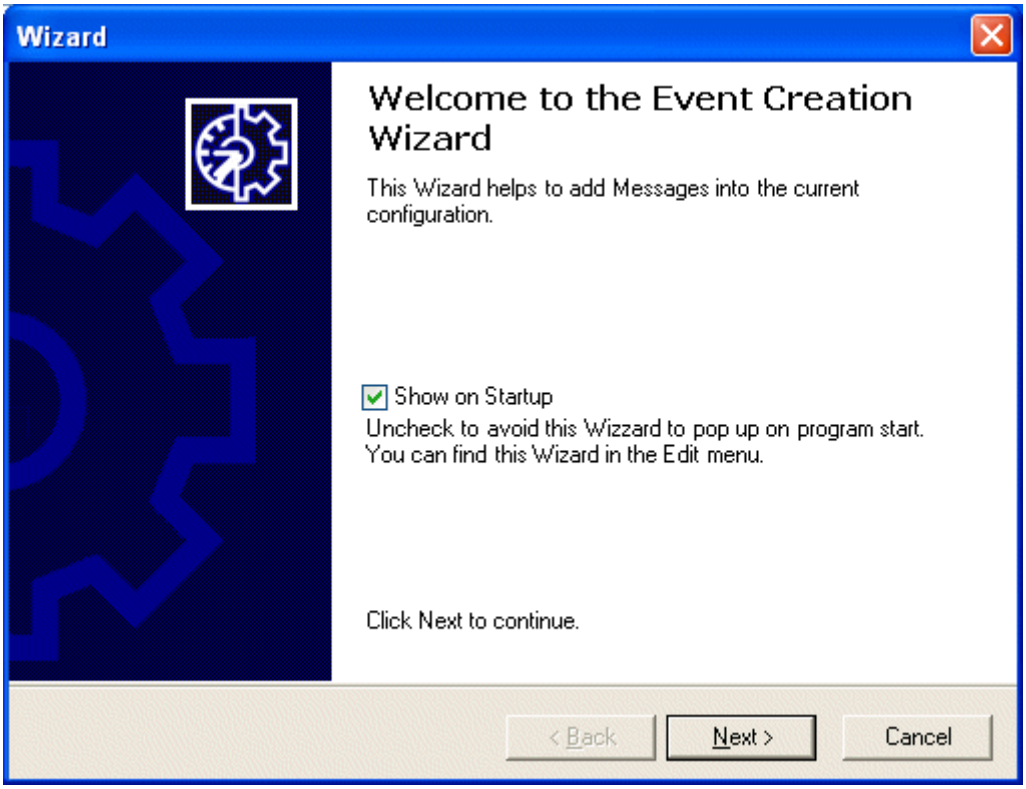
10.1 Konfigurieren von Meldungen

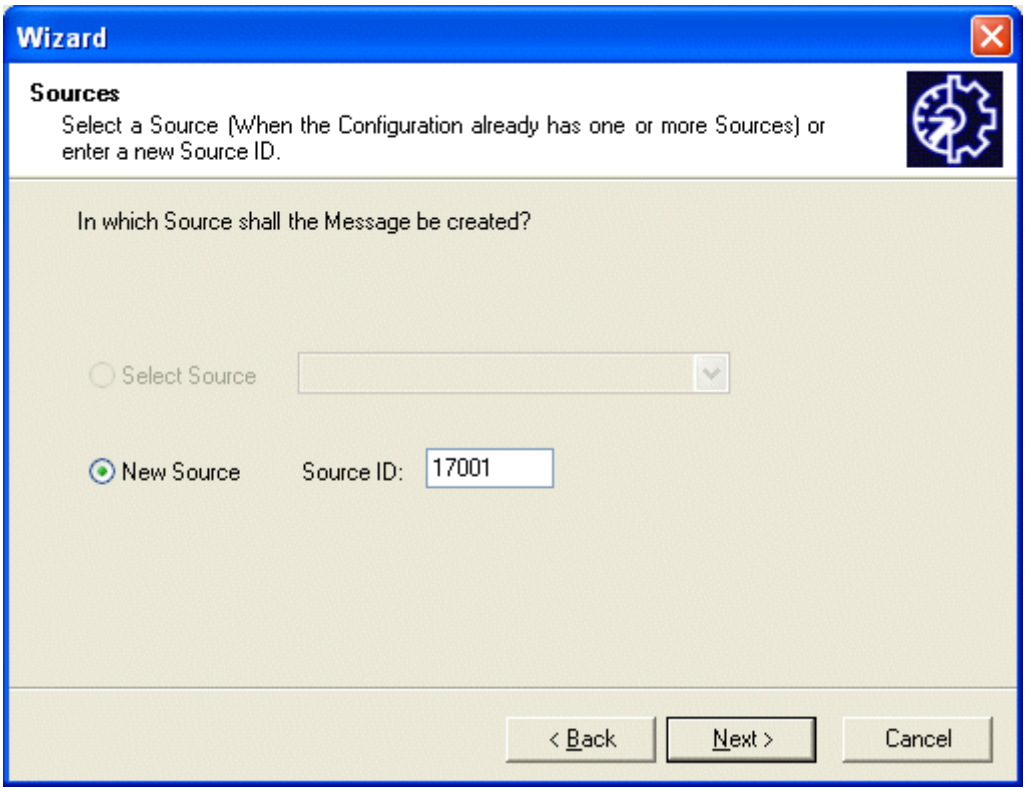
In diesem Sample wird die Verwendung des TcEventkonfigurators beschrieben.

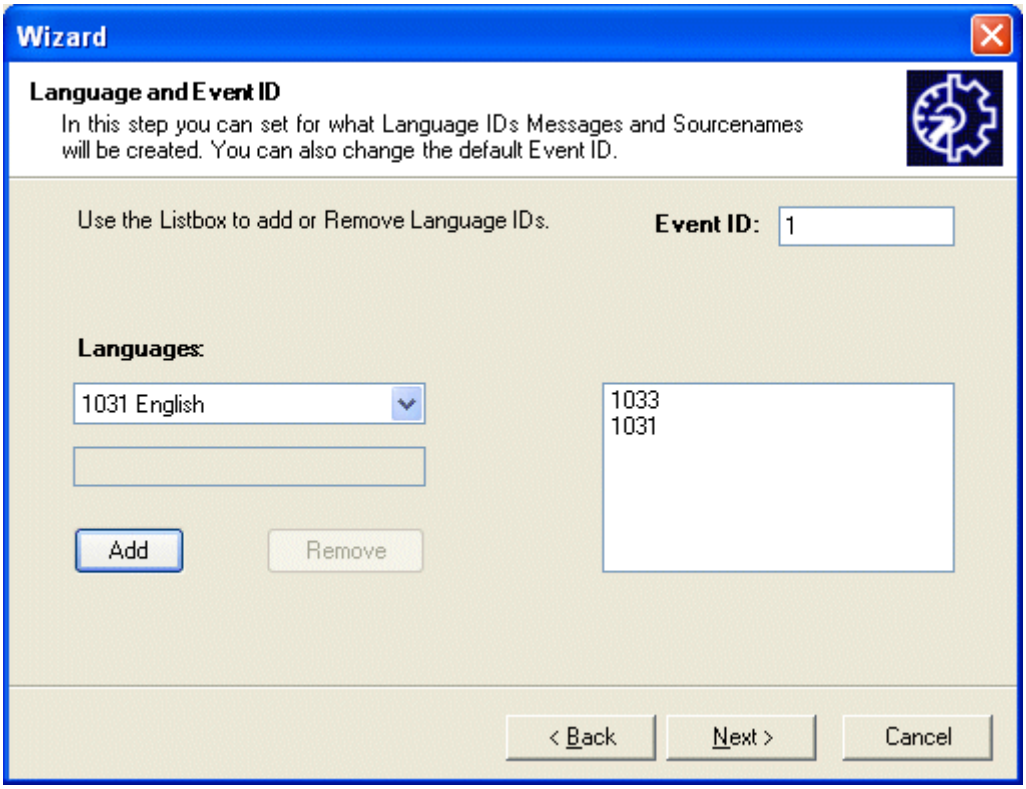
Mithilfe des TcEventkonfigurators können Sie Ihr Meldungssystem konfigurieren und mehrsprachige Meldungstexte erzeugen. Es ist nicht nötig, Konfigurationsdateien manuell zu editieren, wodurch viel Zeit gespart und eine Fehlerquelle eliminiert werden kann.

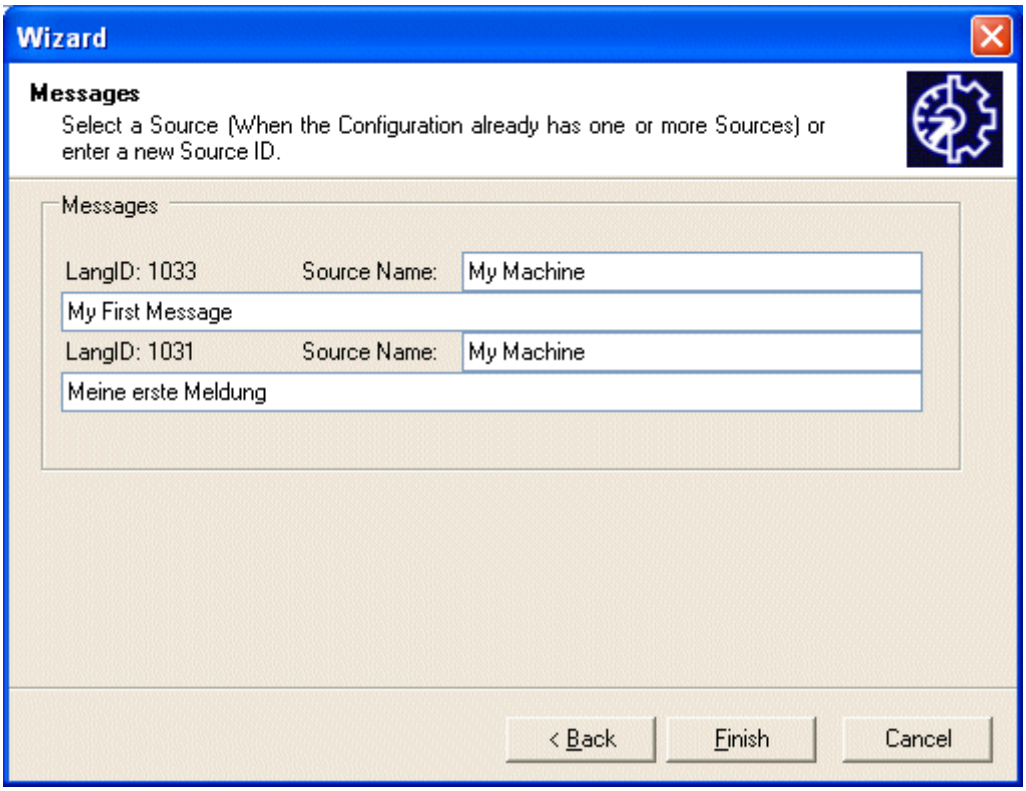
Nach erfolgreicher Konfiguration wird ein SPS Programm erzeugt und Meldungen mithilfe der TcEventbar angezeigt.

1. Installieren und starten Sie den TcEventkonfigurator .
2. Der Konfigurator startet nun mit einem einfachen Wizard zum Erzeugen von Meldungen. Füllen Sie alle Felder wie in den Screenshots dargestellt aus.


Wizard
Welcome to the Event Creation Wizard
This Wizard helps to add Messages into the current configuration.
Show on Startup
Uncheck to avoid this Wizard to pop up on program start.
You can find this Wizard in the Edit menu.
Click Next to continue.
< Back
Next >
Cancel


Wizard
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:
17001
< Back
Next >
Cancel





3. Nach Fertigstellung des Wizards haben Sie Ihre erste Meldung erzeugt. Die Meldung steht in deutscher und englischer Sprache zur Verfügung.
4. Der TcEventkonfigurator bietet eine übersichtliche Oberfläche zum Bearbeiten und Erzeugen von Meldungen. Mit minimalem Aufwand können Sie nun weitere Meldungen und Sprachen erzeugen.
5. Speichern Sie Ihre Konfiguration. Als Dateityp wählen Sie den TcXmlFormatter aus und bestätigen die folgende Message Box mit OK. - Ihre Konfiguration ist nun aktiv.
6. Mit einem Rechtsklick auf eine beliebige Meldung können Sie sich nun den SPS Code zum Auslösen dieser Meldung generieren lassen.

   Fügen Sie diesen Code in ihr SPS Programm ein und lösen Sie die Meldung aus.

Starten Sie die TcEventbar. Die Meldung wird nun vollständig angezeigt.

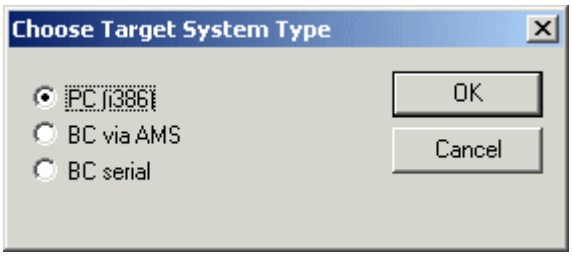
10.2 SPS-Schnittstelle

10.2.1 Auslösen von einfachen Meldungen

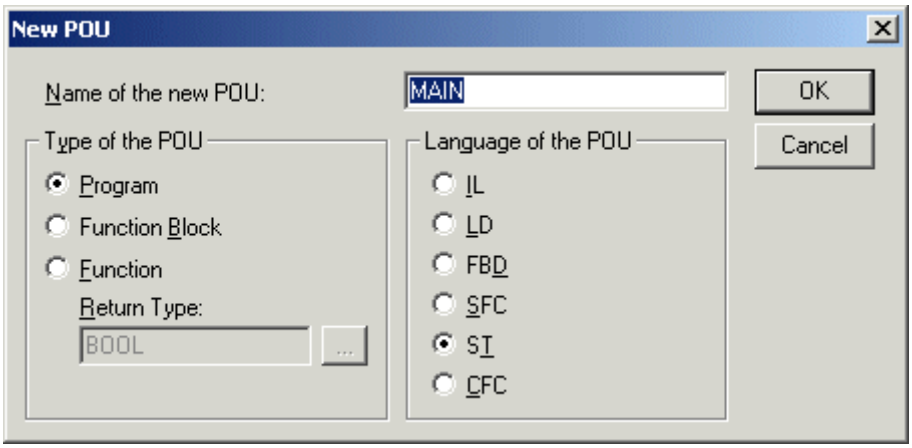
Meldungen können auf zwei wegen ausgelöst werden : einfach und komplex.

In diesem Sample wird das Auslösen einer einfachen Meldung aus der SPS erklärt. Einfache Meldungen lassen sich mit deutlich weniger Code auslösen, bieten jedoch kaum Freiraum zur Konfiguration.

1. Erstellen Sie ein neues SPS-Projekt.
   - Als Zielsystem wählen Sie PC (i386)



   - Behalten Sie die standard Einstellungen der Taskkonfiguration bei.
   - Erzeugen Sie ein neues Programm. Als Sprache wählen sie Strukturierten Text.



   - Öffnen Sie über den Menüpunkt *Fenster->Bibliotheksverwaltung* den Bibliotheksverwalter und fügen die TcSystem.lib zum Projekt hinzu.

     Nun steht Ihnen der ADSLOGEVENT Baustein zur Verfügung.
2. Benutzen sie den TcEventKonfigurator [▶ 21] um eine quittierungspflichtige Meldung mit der SourceID 17001 und EventID 1 zu erzeugen.
3. Fügen Sie nun folgenden Code in Ihr Projekt ein

```
(*Variablendeklaration*)
VAR
    event    : FB_SimpleAdsLogEvent;
    bEvent   : BOOL;
    bQuitEvt : BOOL;
END_VAR

(*SPS Code*)
```

```
Event( SourceID  := 17001,
       EventID   := 1,
       bSetEvent := bEvent,
       bQuit     := bQuitEvt);
```

4. Starten Sie nun das PLC Programm.
5. Starten Sie die TcEventbar (im 'TwinCAT/Eventlogger' Verzeichnis).
6. Durch Toggeln der Variable bSetEvent1 wird nun die entsprechende Meldung ausgelöst. Mit steigender Flanke kommt die Meldung, mit fallender wird sie zurückgesetzt.
7. Beim Toggeln von bSetEvent wird die Meldung nicht zurückgesetzt, da es sich um eine quittierungspflichtige Meldung handelt. Sie wird erst gelöscht, wenn bSetEvent zurückgesetzt wird **und** bQuitEvent gesetzt wird.

Sprache / IDE	Beispielprogram auspacken
TwinCAT PLC (Structured Text)	https://infosys.beckhoff.com/content/1031/TcEventLogger/Resources/12333057163.zip

10.2.2 Auslösen von komplexen Meldungen

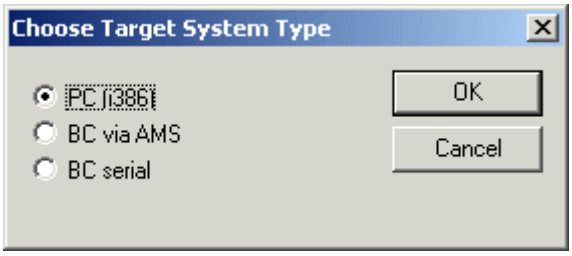
Meldungen können auf zwei wegen ausgelöst werden : einfach und komplex.

In diesem Sample soll gezeigt werden, wie eine komplexe Meldung aus der SPS ausgelöst werden kann. Der Code Aufwand für komplexe Meldungen ist höher als der für einfache Meldungen - es bleibt jedoch deutlich mehr Freiraum zur Konfiguration.

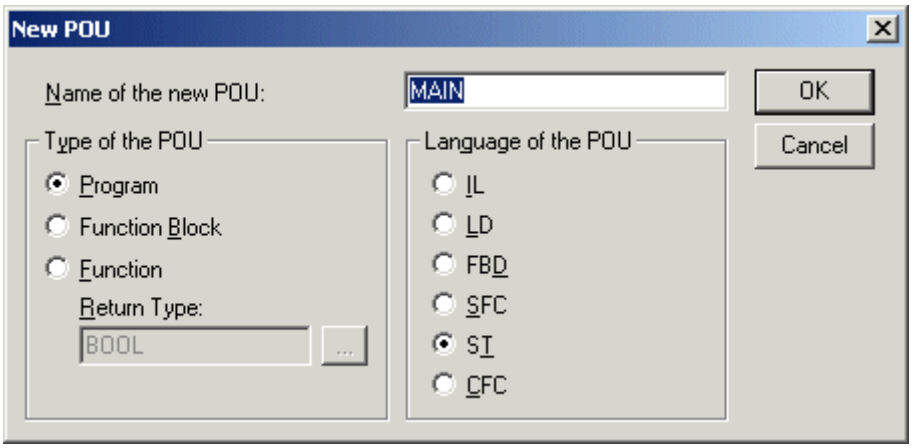
Es werden 2 Meldungen erzeugt - eine einfache Meldung und eine quittierungspflichtige.

Die Meldungen werden dann mithilfe der TcEventbar angezeigt.

1. Erstellen Sie ein neues SPS-Projekt.
   - Als Zielsystem wählen Sie PC (i386)



   - Behalten Sie die standard Einstellungen der Taskkonfiguration bei.
   - Erzeugen Sie ein neues Programm. Als Sprache wählen sie Strukturierten Text.



- Öffnen Sie über den Menüpunkt *Fenster->Bibliotheksverwaltung* den Bibliotheksverwalter und fügen die TcSystem.lib zum Projekt hinzu.

  Nun steht Ihnen der ADSLOGEVENT Baustein zur Verfügung.

2. Fügen Sie nun folgenden Code in Ihr Projekt ein

```
(*Variablendeklaration*)
VAR
    bSetEvent1 : BOOL;
    bSetEvent2 : BOOL;
    bQuitEvent2 : BOOL;

    event1: ADSLOGEVENT;
    event2: ADSLOGEVENT;

    CfgEvent1 : TcEvent;
    CfgEvent2 : TcEvent;
END_VAR

(*SPS Code*)
(* Einstellung des Parameters für Ereignis 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.SourceId := 17001; (* 17001: Achsregelung *)
CfgEvent1.Class := TCEVENTCLASS_MESSAGE;

(* Einstellung des Parameters für Ereignis 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.SourceId := 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. Starten Sie nun das PLC Programm.
4. Starten Sie die TcEventbar (im 'TwinCAT/Eventlogger' Verzeichniss).
5. Durch toggeln der Variable bSetEvent1 wird nun die entsprechende Meldung ausgelöst. Mit steigender Flanke kommt die Meldung, mit fallender wird sie zurückgesetzt. Da noch keine Meldungstexte eingerichtet sind wird in den Feldern Quelle und Meldung ein Fehler gemeldet.

6. beim Toggeln von bSetEvent2 wird die Meldung nicht zurückgesetzt, da es sich um eine quittierungspflichtige Meldung handelt. Sie wird erst gelöscht, wenn bSetEvent2 zurückgesetzt wird **und** bQuitEvent2 gesetzt wird.
7. Fahren Sie nun mit dem Einrichten der Meldungstexte fort. [▶ 186]

Sprache / IDE	Beispielprogram auspacken
TwinCAT PLC (Structured Text)	https://infosys.beckhoff.com/content/1031/TcEventLogger/Resources/12333058571.zip

10.2.3 Übertragen von Variablen mit der Meldung

Es ist möglich mit einer Meldung eine oder mehrere Variablen zu übertragen. In diesem Beispiel werden die dafür nötigen Schritte beschrieben.

1. Generieren Sie ein PLC Projekt wie im vorherigen Beispiel [▶ 190] beschrieben.
2. Fügen Sie die folgenden Variablen zur Variablenkonfiguration hinzu.

   Der Wert 'value' soll übertragen werden; Der String 'format' beschreibt das Datenformat zum Übertragen der Werte.

   ```
   value      : DWORD    := 0;
   format     : STRING   := '%d';
   ```

3. Nun ändern Sie die Konfiguration für das Event, so dass es folgendermaßen aussieht:

   ```
   (* Einstellung des Parameters für Ereignis 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.SourceId := 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. Lösen Sie das Event aus.

Es können auch mehrere Variablen übertragen werden, indem Sie eine Struktur übergeben. Der Formatstring muss dann die Anordnung der Variablen in der Struktur wiedergeben.

Wird ein zusätzlicher Text als Parameter übertragen, muss dieser als letztes Element in der Struktur deklariert werden. Es kann nur ein String übertragen werden.

```
(*Datentypen*)
TYPE ST_EventData1 :
STRUCT
    axis     : UDINT;
    pos      : LREAL;
```

```
    message  : STRING;
END_STRUCT
END_TYPE
```

```
(*Variablendeklaration*)
...
format     : STRING := '%d%f%s'
Eventdata1 : ST_EventData1;
...
```

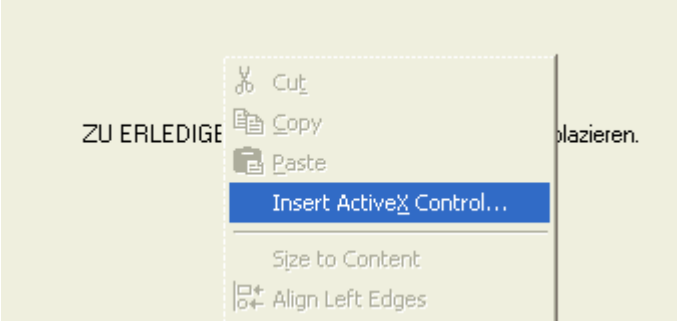
```
(*SPS Code*)
...
EventDataAddress    := ADR(Eventdata1),
EventDataLength     :=SIZEOF(Eventdata1),
...
```

Sprache / IDE	Beispielprogram auspacken
TwinCAT PLC (Structured Text)	https://infosys.beckhoff.com/content/1031/TcEventLogger/Resources/12333059979.zip

10.3 Visual Studio C++

10.3.1 Einbinden von TcEventViewer-ActiveX-Control

1. Erstellen Sie ein neues MFC exe Projekt.
2. Als Applikations Typ wählen Sie Dialogbasiert aus.
3. Rechtsklicken Sie auf den Dialog in der Layoutansicht und wählen 'Insert ActiveX Control'



4. Wählen Sie 'BECKHOFF TcEventView Class' aus

Sprache / IDE	Beispielprogram auspacken
Visual Studio 6 C++	Sample01.zip (in Vorbereitung)

10.3.2 Konsolenanwendung - Geloggte Events via DCOM-Schnittstelle lesen

Das folgende Beispiel zeigt, wie in C++ eine Liste von geloggten Meldungen ausgegeben werden kann.

1. Erstellen Sie eine standard Konsolen Applikation.
2. Fügen Sie folgenden Sourcecode hinzu.

```
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;
```

Das Programm benutzt das ITcEnumEvents Interface um die geloggten Meldungen zu enumerieren.

Für alle geloggten Meldungen wird im Konsolenfenster der Meldungstext ausgegeben.

Sprache / IDE	Beispielprogram auspacken
Visual Studio 6 C++	https://infosys.beckhoff.com/content/1031/TcEventLogger/Resources/12333058571.zip

10.3.3 Connection Points

Dieses Sample Programm enthält eine komplette Implementierung eines ConnectionPoint zum TcEventLogger.

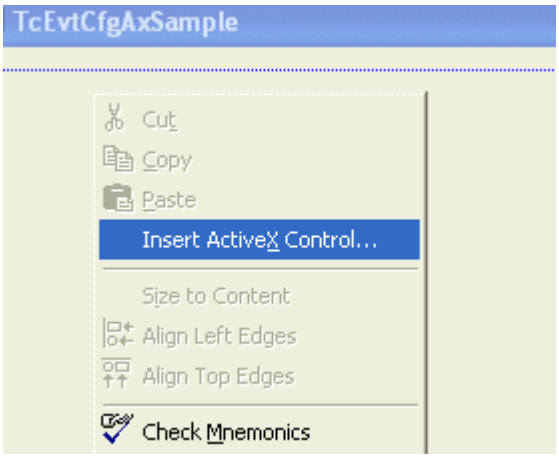
Sprache / IDE	Beispielprogram auspacken
Visual Studio 6 C++	https://infosys.beckhoff.com/content/1031/TcEventLogger/Resources/12333059979.zip

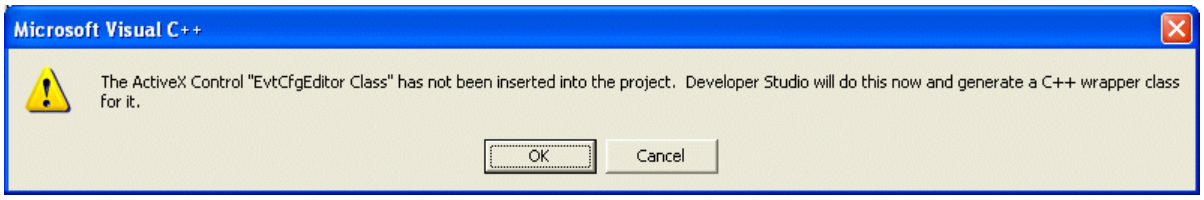
10.3.4 Hinzufügen des ActiveX TcEvtCfgEditor zu einem Dialog

Um den TcEvtCfgEditor zu einem Dialog, der auf einer MFC Applikation basiert, hinzuzufügen folgen Sie einfach diesem Tutorial.

1. Start Sie das MSVisual Studio 6.
2. Erstellen Sie einen neuen auf einer MFC Applikation basierten Dialog.

3. Klicken Sie im Design Modus mit der rechen Maustaste auf die Dialog Ressource und wählen Sie 'Add ActiveX Control'.



4. Wähen Sie den Beckhoff EvtCfgEditor und platzieren Sie ihn auf dem Dialog.
5. Öffnen Sie den Class Wizzard wählen Sie den 'Member Variables' Reiter
6. Wählen Sie IDC_EVTCFGEDITOR1 und klicken Sie auf on 'AddVariable'
7. Im folgenden Dialog wählen Sie OK um eine automatisch generierte Wrapper-Klasse zu bekommen.
8. 

9. Im folgenden Dialog lassen Sie alle Items ausgewählt und klicken OK.
10. Nun müssen Ssie den Namen der Member-Variable eingeben.
11. Das ActiveX Control ist fertig und kann benutzt werden.
    Rufen Sie GetConfiguration [▶ 65] auf um Zugriff auf das darunterliegende Konfigurationsobjekt [▶ 27] zu bekommen

Sprache / IDE	Beispielprogram auspacken
Visual Studio 6 C++	Sample04.zip (in Vorbereitung)

10.4 Visual Studio C#

10.4.1 Anzeige von Logged Events in C#

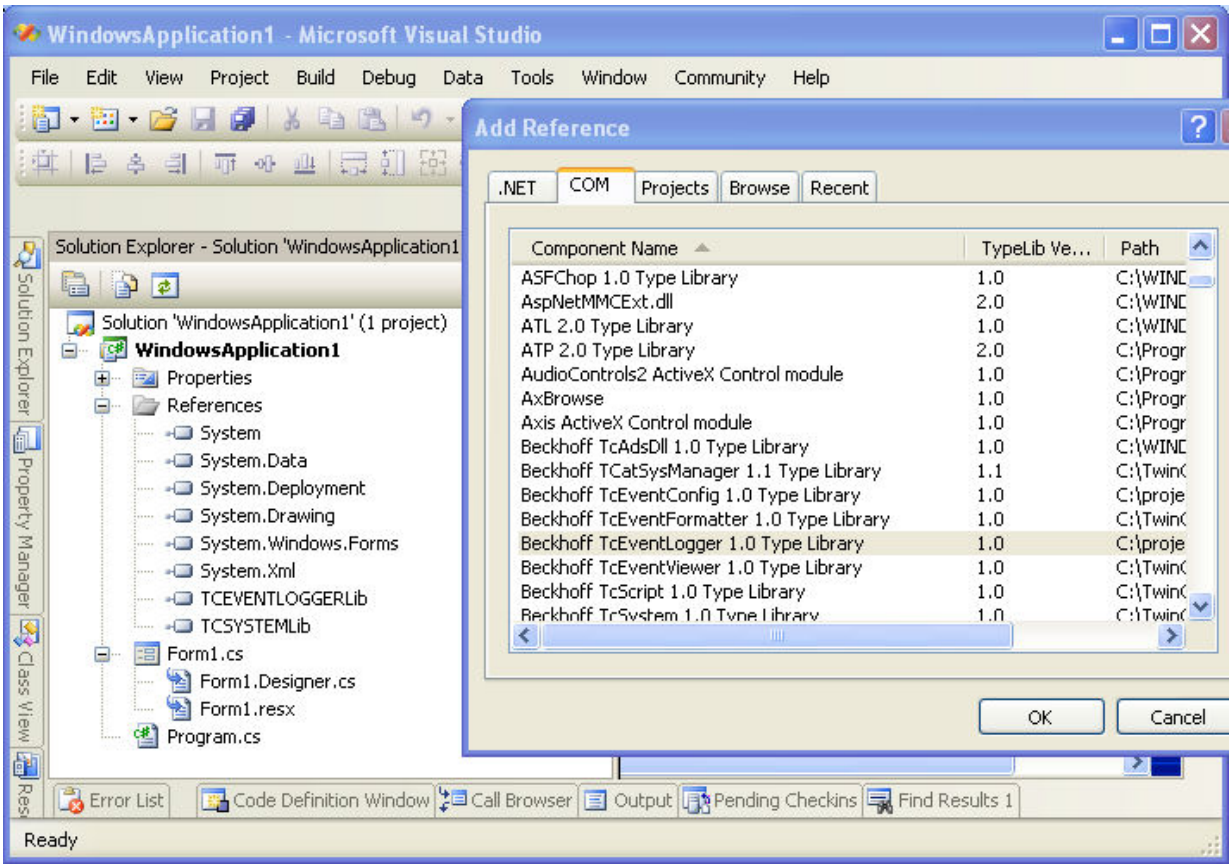
Diese Beispiel bezieht sich auf

- Visual Studio 2005
- TcEventLogger 2.9.0.90 und höher (TwinCAT 2.10. Build1244)

Bevor Sie mit dem Beispiel starten, beachten Sie, dass der TcEventViewer ActiveX keine komplette Implementation einer Event HMI bietet, die einfach an ihre Applikation angefügt werden kann. Wenn Sie spezielle Anforderungen haben, die nicht mit dem standard TcEventViewer realisiert werden können, fahren Sie mit dem Beispiel zur Anzeige geloggter Events fort.

1. Erzeugen Sie eine Win32 C# Applikation.

2. Fügen Sie dem Beckhoff TcEventLogger eine Referenz hinzu:



3. Über Form1.cs ergänzen Sie:
```
using TCEVENTLOGGERLib;
```
4. Erzeugen Sie eine globale Instanz im TcEventlogger.
```
TcEventLogtcEventLogger =newTcEventLog();
```
5. Fügen Sie eine ListView in ihre Main Form ein und benennen es eventView1
6. Intialisieren Sie in Form1_Load die 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. Fügen Sie Code zur Anzeige von Events in die ListView control hinzu:
```
///<summary>
///Display an event on top of 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)
```

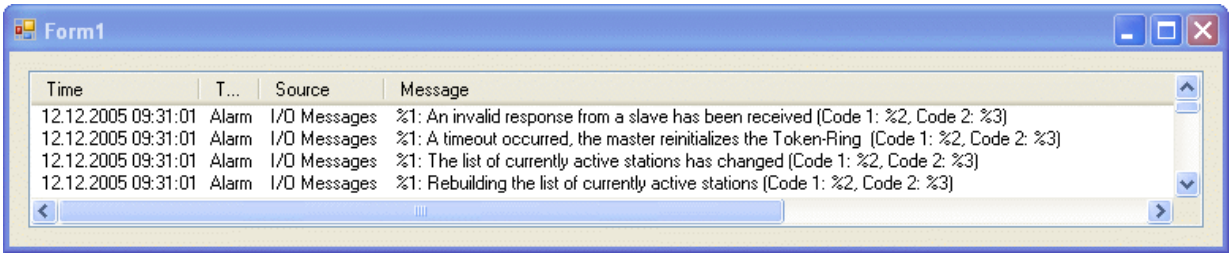
```
        { }
        try
        { AddListViewSubItem(lvi,"Source", evt.get_SourceName(langId)); }
        catch(Exception)
        { }

        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. Zum Schluss lesen Sie die Liste der aktiven Alarme des TcEventLogger in Form1_Load.

```
foreach(TcEvent evt in tcEventLogger.EnumLoggedEventsEx() )
    AddEvent(evt);
```

9. Starten Sie ein SPS Programm um ein paar Events auszugeben.



Sprache / IDE	Beispielprogram auspacken
Visual Studio 2005	https://infosys.beckhoff.com/content/1031/TcEventLogger/Resources/12333057163.zip

10.4.2 Displaying Active Events in C#

Diese Beispiel bezieht sich auf

- Visual Studio 2005
- TcEventLogger 2.9.0.90 und höher (TwinCAT 2.10. Build1244)

Bevor Sie mit dem Beispiel starten beachten Sie, dass der TcEventViewer ActiveX keine komplette Implementation einer Event HMI bietet, die einfach an ihre Applikation angefügt werden kann. Wenn Sie spezielle Anforderungen haben, die nicht mit dem standard TcEventViewer realisiert werden können, fahren Sie mit dem Beispiel zur Anzeige geloggter Events fort.

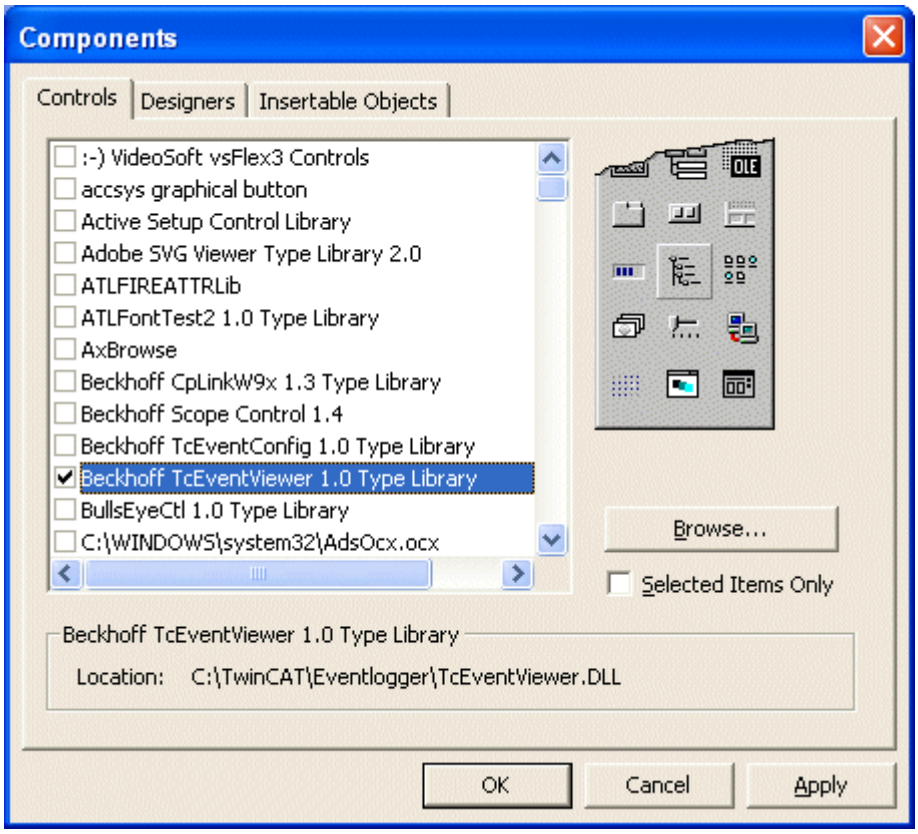
- Diese Beispiel baut auf das Beispiel 'Displaying Logged Events in C# [▶ 195]' auf.
- To implement a program capable of displaying active events we will need to extend the previous sample with the folowing features:
  - Add events dynamically as they occur
  - Remove inactive events from the listview.
  - 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.

Sprache / IDE	Beispielprogram auspacken
Visual Studio 2005	https://infosys.beckhoff.com/content/1031/TcEventLogger/Resources/12333058571.zip

10.5 Visual Basic

10.5.1 Einbinden von TcEventViewer-ActiveX-Control

1. Erstellen Sie ein neues standard exe projekt.
2. Rechtsklicken Sie in die Toolbox und wählen den Menüpunkt Components...
3. Fügen Sie den TcEventViewer hinzu



4. Der TcEventviewer befindet sich nun in der Toolbox. Fügen Sie ihn in die Form ein

Sprache / IDE	Beispielprogram auspacken
Visual Basic 6.0	Sample01.zip (in Vorbereitung)

10.5.2 Aktive Alarme in einer benutzerdefinierten Listenansicht

Das folgende Beispiel zeigt, wie man eine benutzerdefinierte Listenansicht, die aktive Alarme in Visual Basic anzeigt, für den TcEventLogger erstellt.

Die folgenden Schritte müssen ausgeführt werden, um das Beispiel zu erstellen.

1. Erstellen Sie ein Standard EXE Projekt mit einem Formular.
2. Fügen Sie die Microsoft Standardsteuerelemente 6 in die Liste der Steuerelemente ein.
3. Fügen Sie die Beckhoff TcEventLogger Bibliothek zu der Referenzliste hinzu.
4. Fügen Sie ein Listenansicht ActiveX Steuerelement in das Formular ein.
5. Benennen Sie die Listenansicht in lvwAlarm um.
6. Kopieren Sie den folgenden Code in das Formular.

```
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 = lvwAlarm.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 lvwAlarm.ListItems.Remove(newKey)
End Sub

'-------------------------------------------------------
' initalize the list view
'
Sub initListView()
  lvwAlarm.View = lvwReport
  lvwAlarm.GridLines = True
  'init the Header
  Dim objColHdr As ColumnHeaders
  Set objColHdr = lvwAlarm.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
```

Beim Aufruf des Formulars wird eine Referenz zu dem einzigen EventLogger erzeugt, selbst dann wenn wir einen neuen Operator verwenden. Dann wird die Listenansicht initialisiert. Die Listenansicht enthält eine Spalte in der die Ereignismeldung in Englisch angezeigt wird. Die Methode DisplayActiveAlarms holt die Liste der aktiven Alarme vom EventLogger und zeigt sie anschließend an. Wenn ein neuer Alarm ausgegeben wird oder bestehende Alarme zurückgesetzt werden, fügt die Ereignisfunktion OnNewEvent Ereignisse in die Listenansicht ein und die Ereignisfunktion OnResetEvent entfernt Ereignisse aus der Listenansicht. Die Methode GetKeyFromIDs erzeugt einen einmaligen Schlüssel von der Ereignisgruppe und der Ereignis-ID.

Die benutzerdefinierte Ereignisansicht unterstützt nur Ereignisse, die nicht quittiert werden müssen. Um die benutzerdefinierte Ereignisansicht für quittierungspflichtige Alarme verwenden zu können, müssen die Ereignisse ITcEventLogEvents [▶ 146]::OnSignalEvent [▶ 149] und ITcEventLogEvents [▶ 146]::OnConfirmEvent [▶ 147] angelegt werden.

Sprache / IDE	Beispielprogram auspacken
Visual Basic 6.0	https://infosys.beckhoff.com/content/1031/TcEventLogger/Resources/12333058571.zip

10.6 C++Builder 2009

10.6.1 Einbinden in CodeGear C++Builder 2009

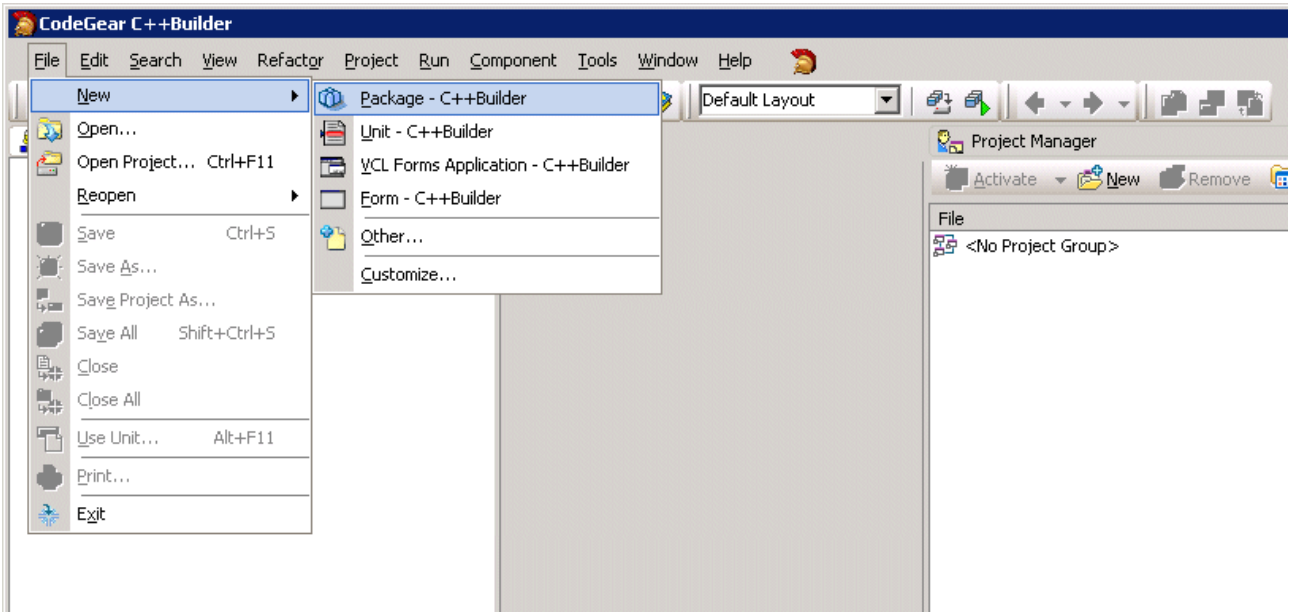
Die Komponenten der EventLogger-Typbibliotheken werden importiert und einem neuen Package hinzugefügt. Das Package wird anschliessend in der ActiveX-Komponentenpalette installiert.

Systemvoraussetzungen

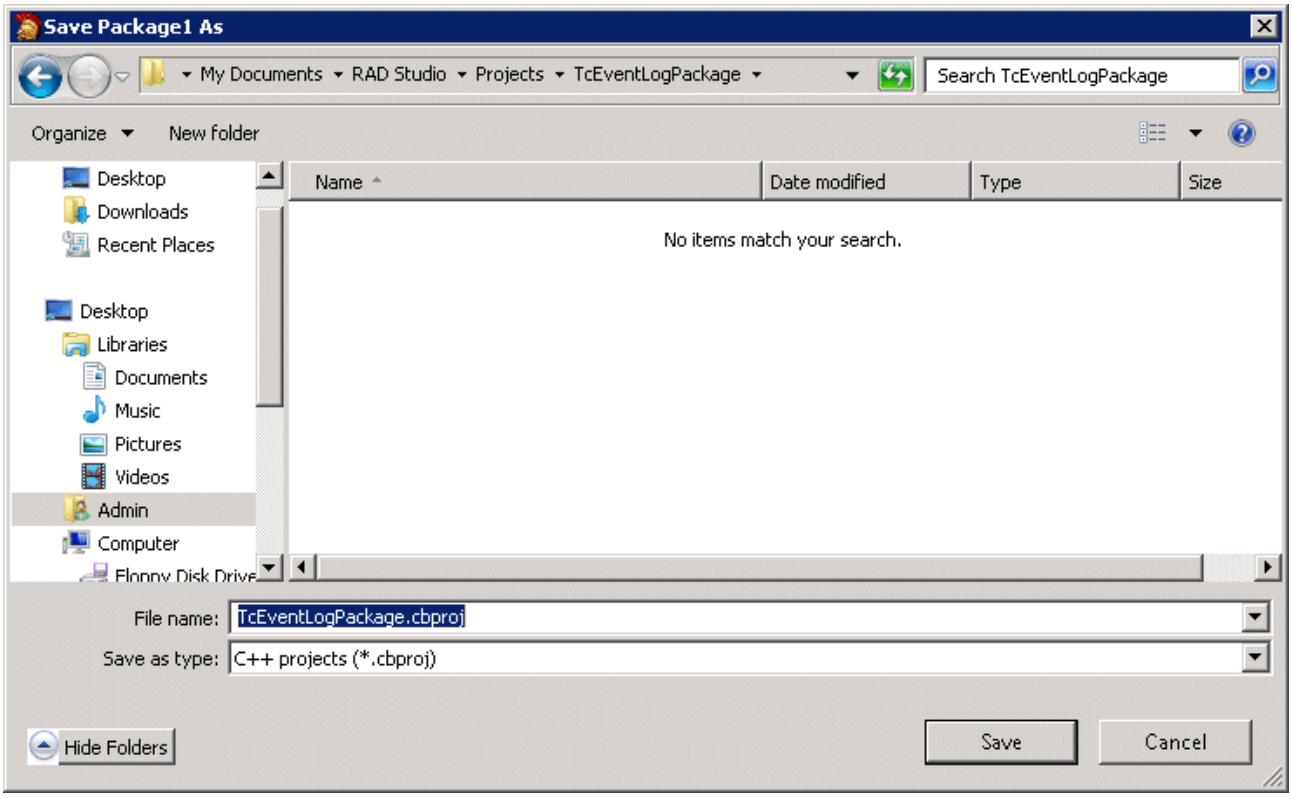
- CodeGear C++Builder 2009;
- Delphi and C++Builder 2009 Update 3;
- Delphi and C++Builder 2009 Update 4 (Database Pack Update);
- C++Builder 2009 Boost Update;
- RAD Studio 2009 Hotfix 2 (dieser Hotfix behebt u.a. folgende Probleme beim Debuggen unter Windows 7: Assertion :... "(!SetThreadContext fehlgeschlagen"));
- TwinCAT v2.11 oder höher;

Neues Package-Projekt erstellen

Erstellen Sie in CodeGear C++Builder 2009 ein neues Projekt vom Typ: "Package - C++Builder".

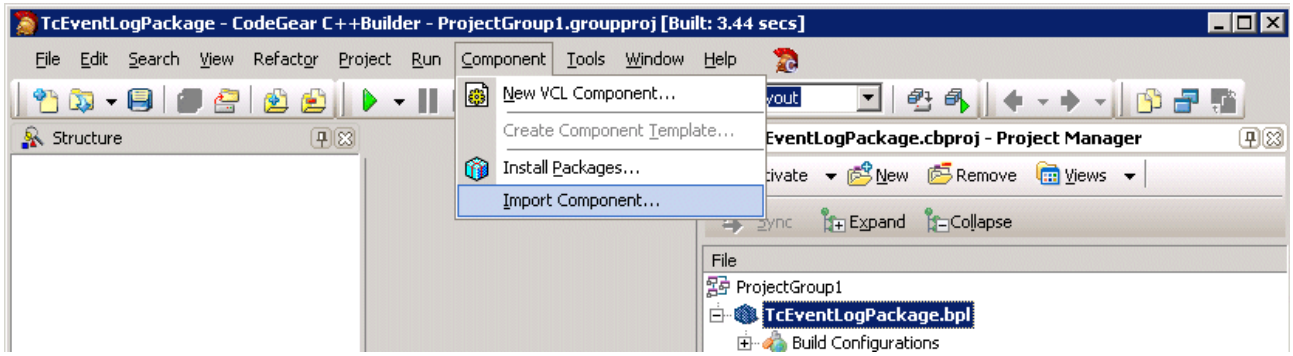


Speichern Sie das neue Projekt mit dem Menübefehl: *Datei->Projekt speichern unter...* unter einem neuen Namen z.B. *TcEventLogPackage.cbproj*.

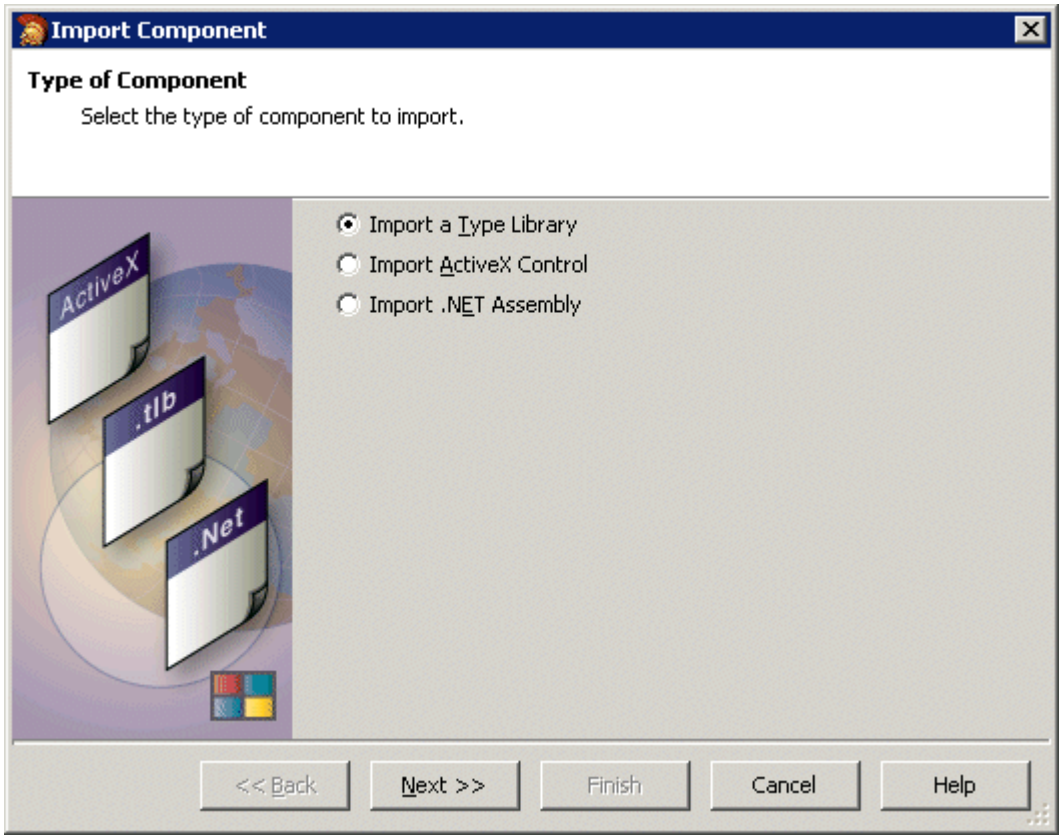


Typbibliotheken importieren

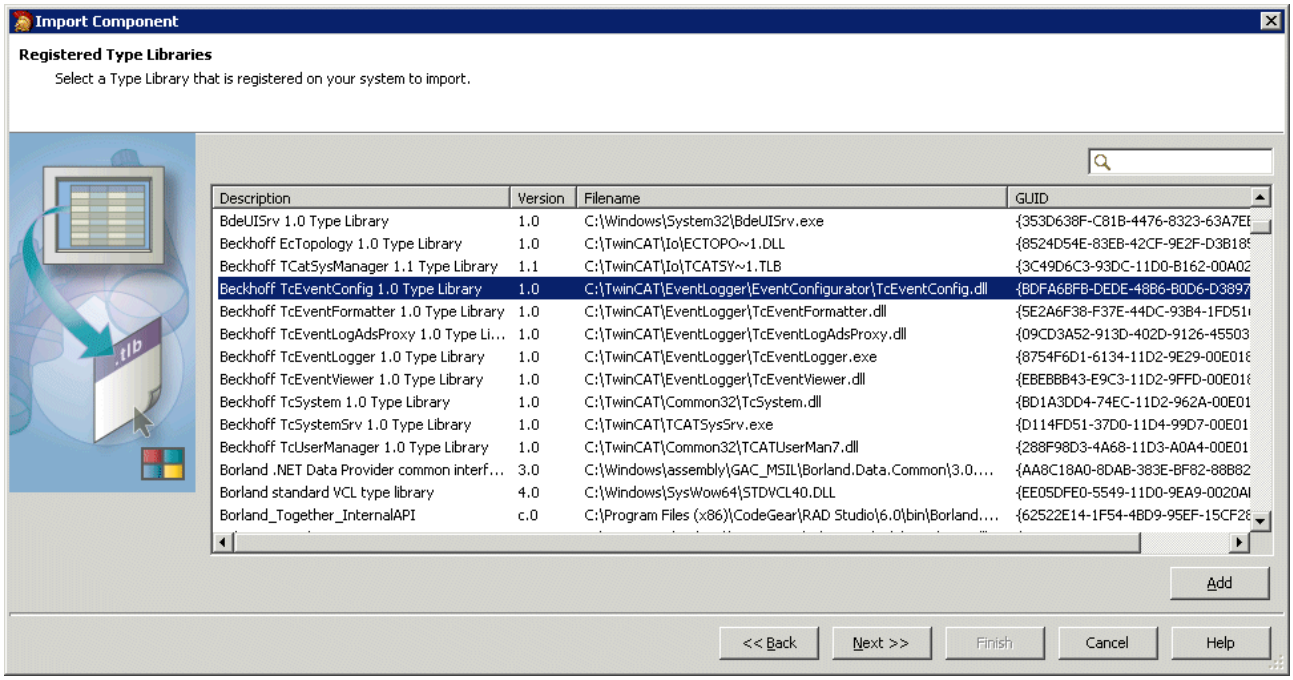
Schritt 1: Im Menue rufen Sie den Befehl: *Komponente -> Komponente importieren...* auf.



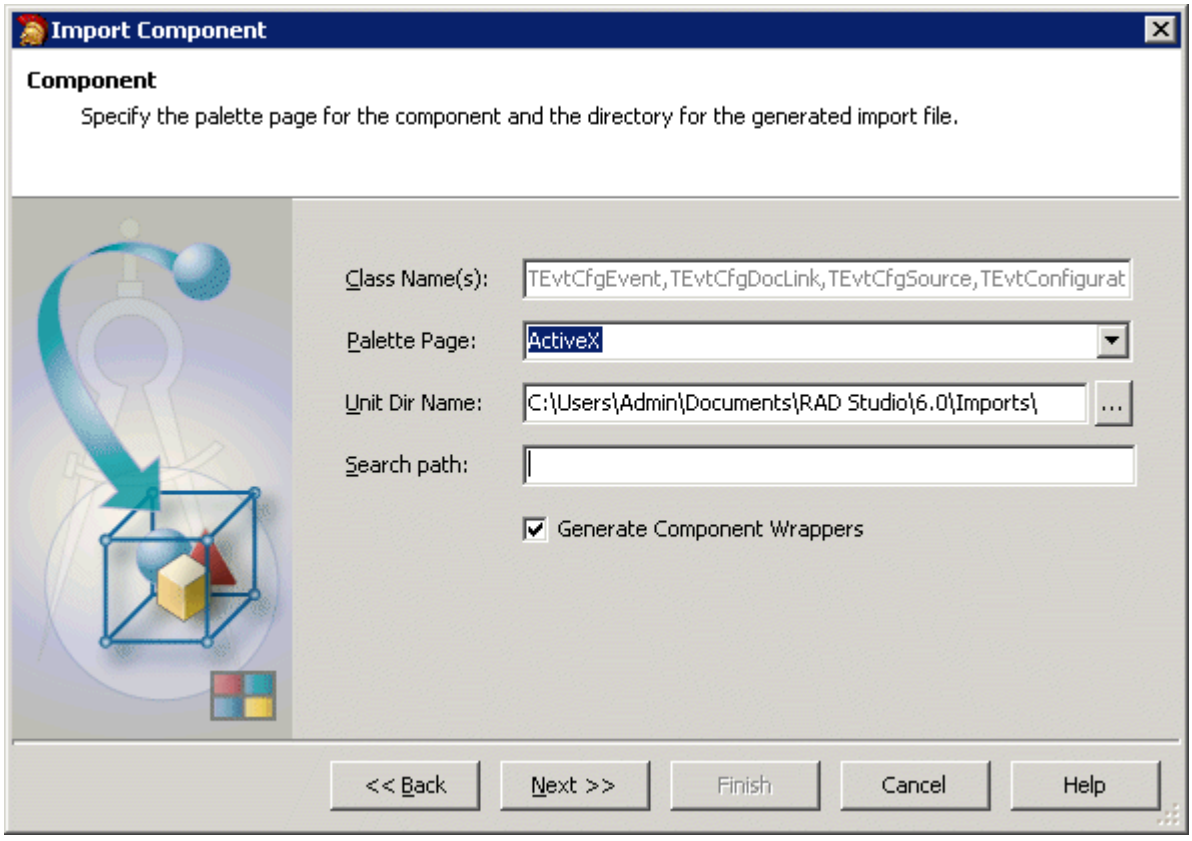
Schritt 2: Wählen Sie die Option: *Typbibliothek importieren* aus und bestätigen Sie mit *Weiter*.



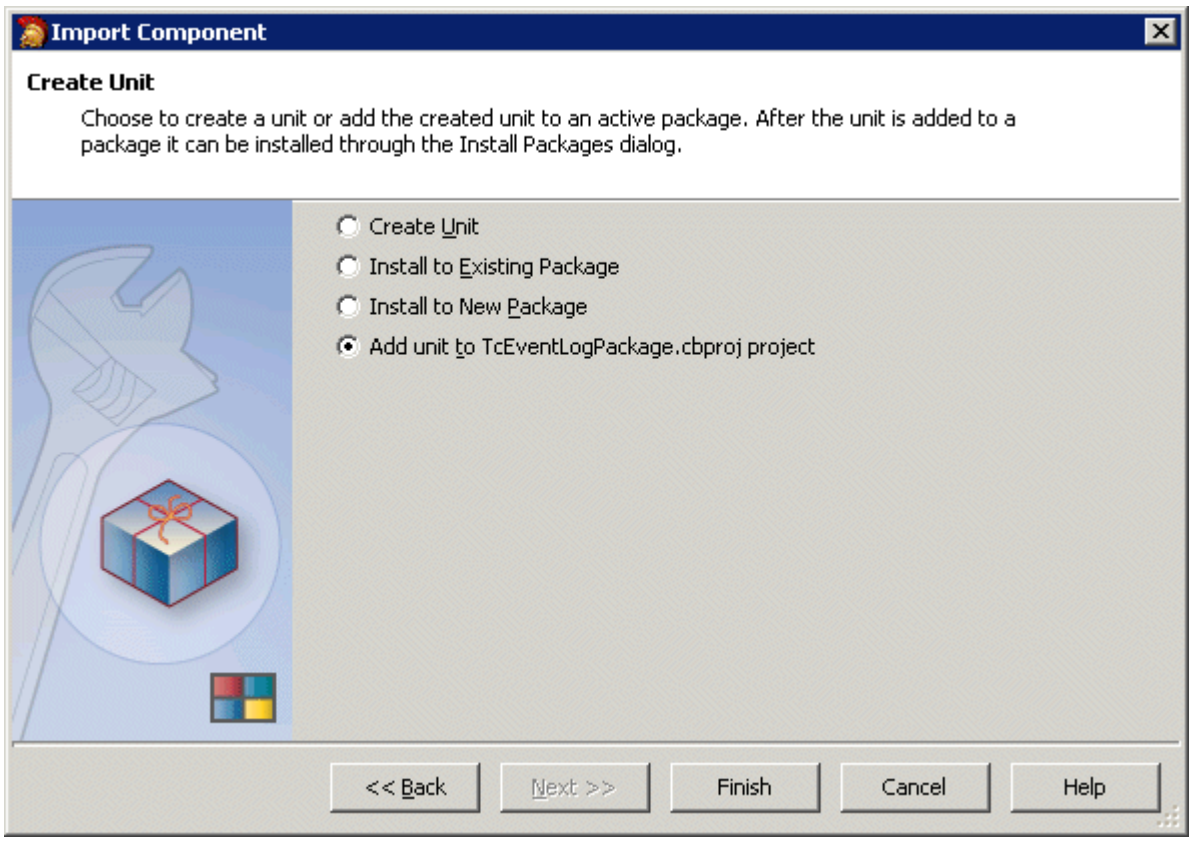
Schritt 3: In der Komponentenliste wählen Sie die erste EventLogger Komponente: *Beckhoff TcEventConfig 1.0 Type Library* aus und bestätigen Sie mit *Weiter*.



Schritt 4: Konfigurieren Sie als Palettenseite: *ActiveX* aus und bestätigen Sie mit *Weiter*.

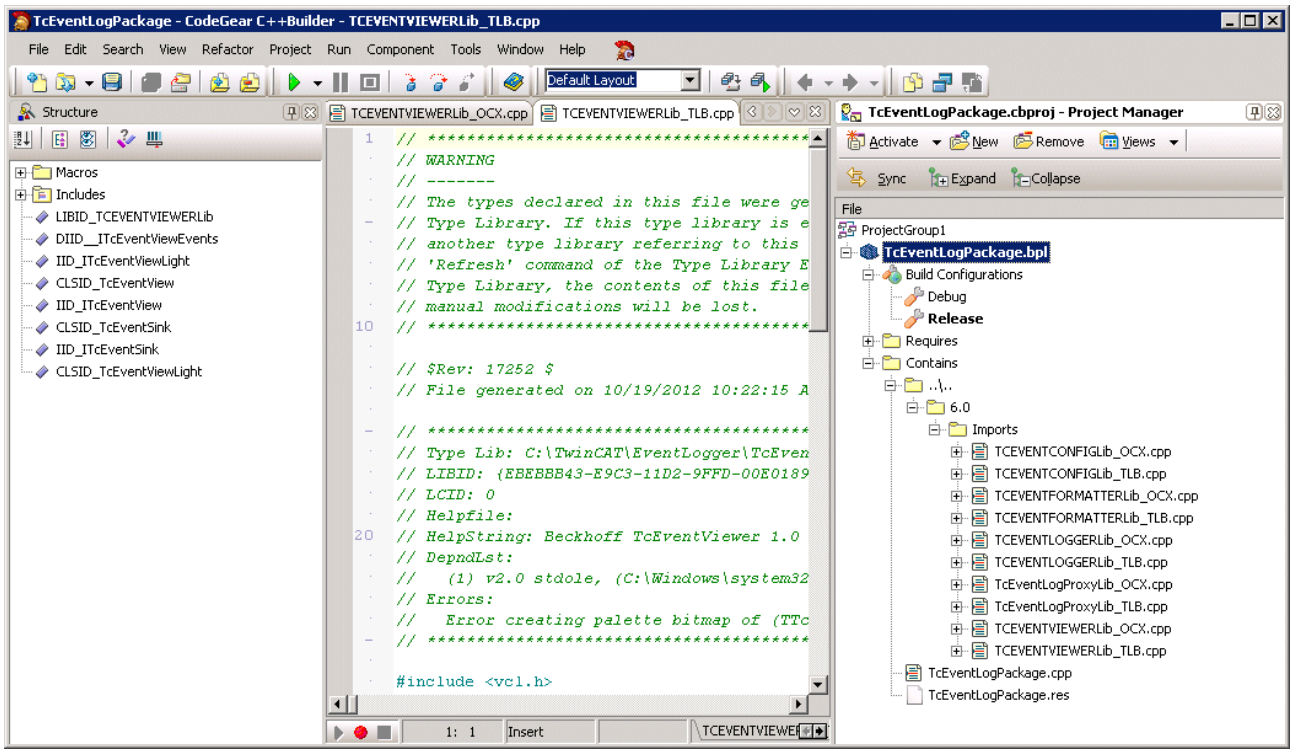


Schritt 5: Im nächsten Dialog wählen Sie die Option: *Unit dem Projekt TcEventLogPackage.cbproj hinzufügen* und bestätigen Sie mit *Fertig stellen*.



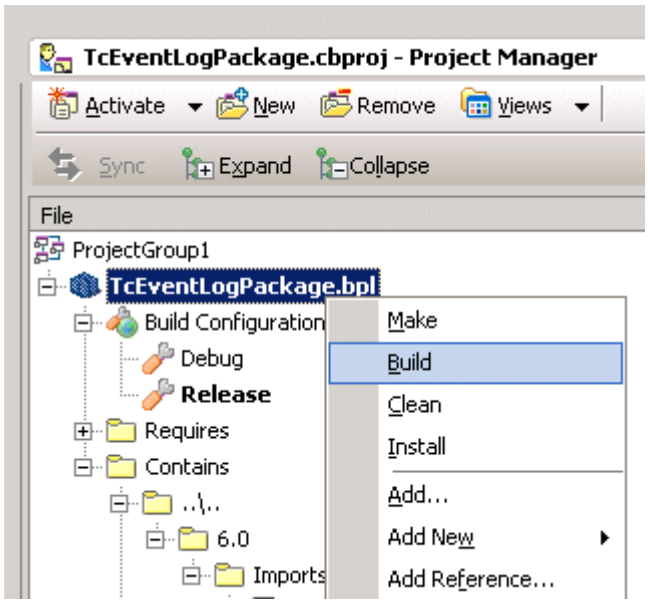
Schritt 6: Wiederholen Sie die Schritte 1 bis 5 für alle anderen EventLogger-Komponenten:

- Beckhoff TcEventFormatter 1.0 Type Library;
- Beckhoff TcEventLogAdsProxy 1.0 Type Library;
- Beckhoff TcEventLogger 1.0 Type Library;
- Beckhoff TcEventViewer 1.0 Type Library;

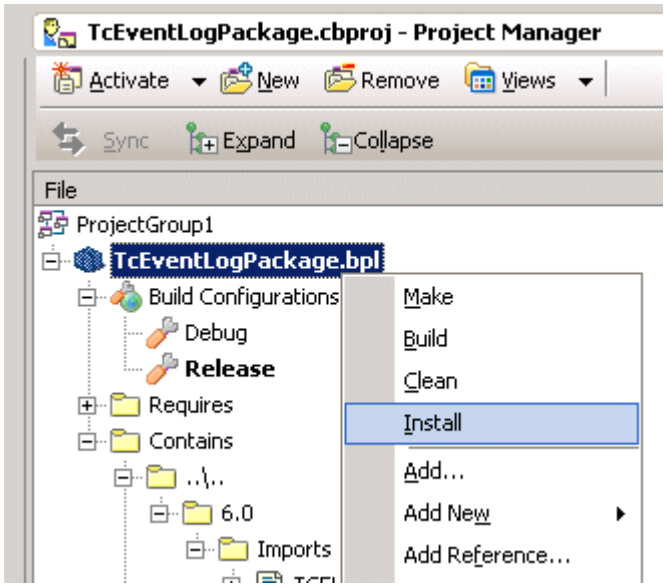


Package installieren

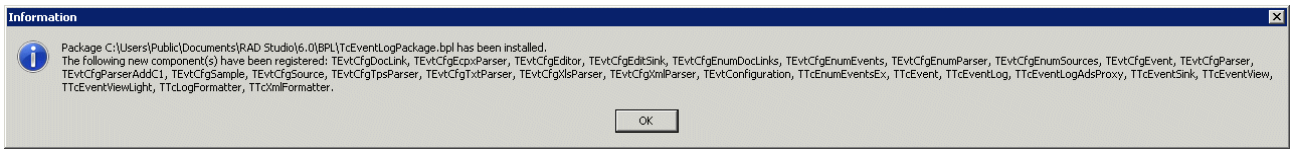
Wählen Sie aus dem Kontextmenu den Befehl: *Erstellen* aus und übersetzen Sie das Package. Das Package muss fehlerfrei übersetzbar sein.



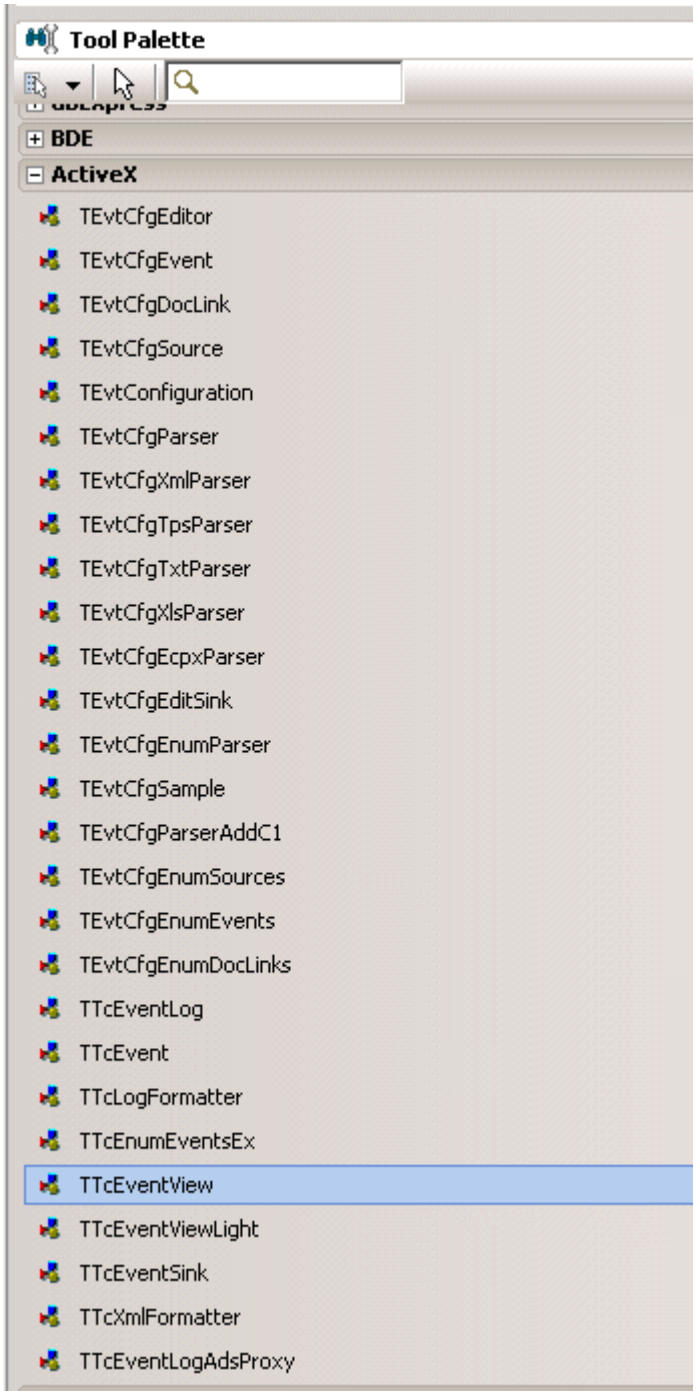
Als nächstes wählen Sie aus dem Kontextmenu den Befehl: *Installieren* aus und installieren Sie die Komponenten in der Komponentenpalettenseite (in unserem Fall unter ActiveX).



Die erfolgreiche Installation der Komponenten wird im folgenden Dialog bestätigt. Schliessen Sie den Dialog mit *OK*.

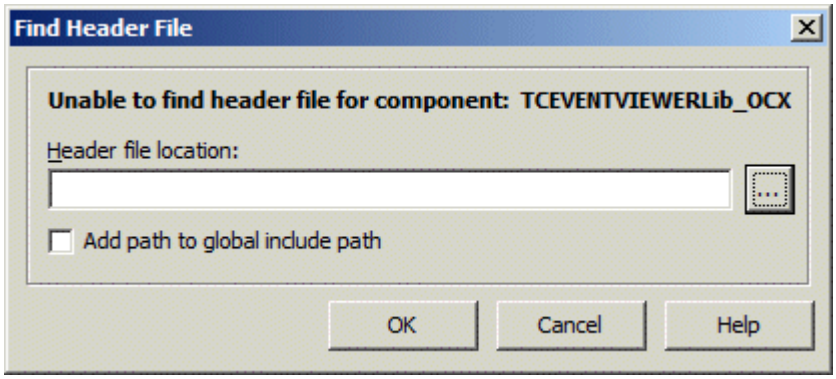


Speichern Sie die letzten Projektänderungen mit dem Befehl: *Datei->Alles speichern*. Schliessen Sie das Projekt mit dem Befehl: *Datei->Alle schliessen*. Die EventLogger-Komponenten können ab jetzt von der Komponentenpalette: ActiveX auf die Form gezogen werden.

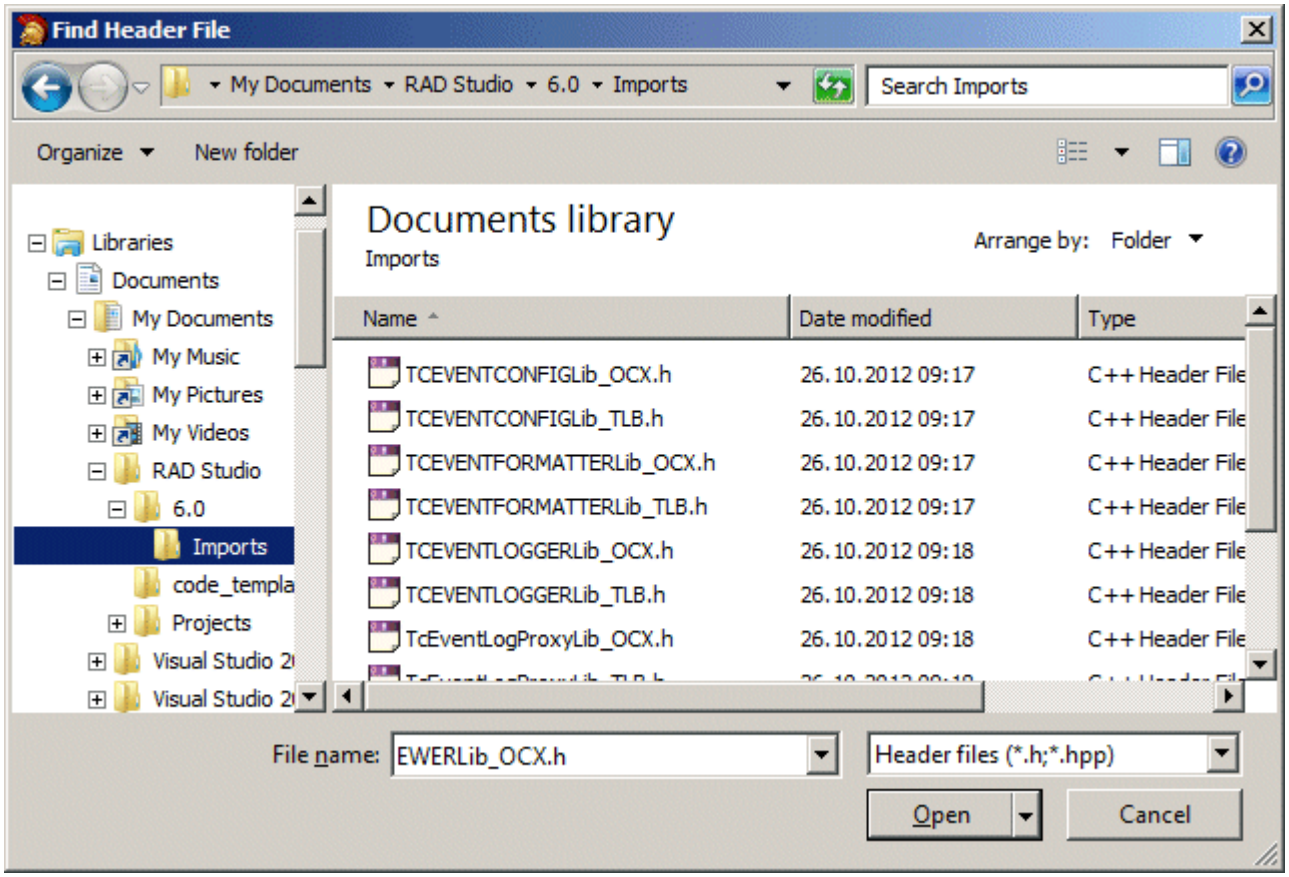


Imports-Pfad zum globalen Include-Pfad hinzufügen

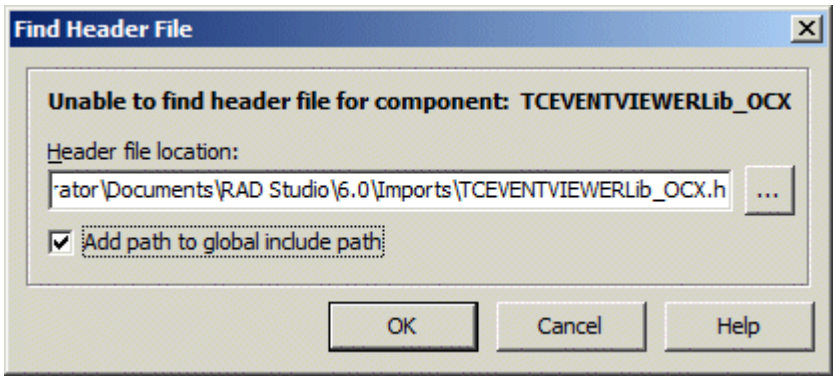
Fügen Sie den Imports-Pfad zum globalen Include-Pfad hinzu wenn beim Zugriff auf die neuen Komponenten die Headerdateien nicht gefunden werden können.



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*.

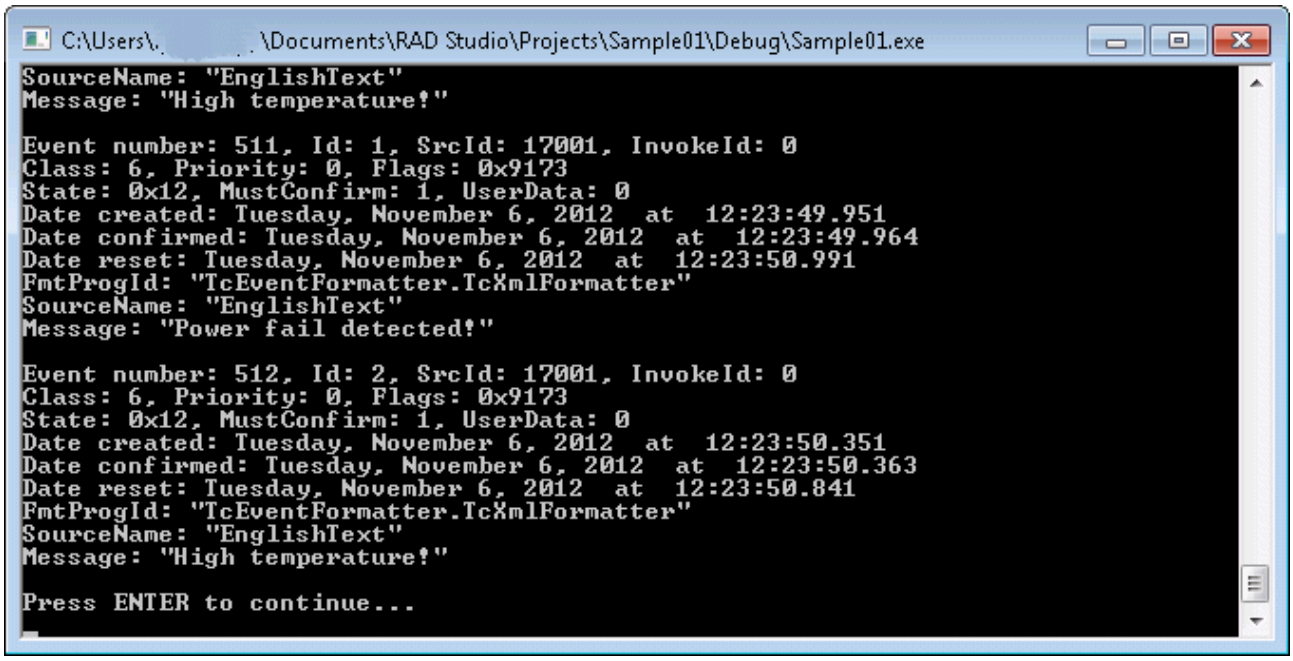


10.6.2 Konsolenanwendung - Geloggte Meldungen via DCOM-Schnittstelle

Systemvoraussetzungen:

- CodeGear C++Builder 2009;
- TwinCAT v2.11 B2228 oder höher;
- Die Typbibiothek der TcEventLogger.exe muss importiert werden (TCEVENTLOGGERLib_TLB.h);

Das Beispiel baut eine DCOM-Verbindung zu einem Remote-EventLogger auf und zeigt in der Konsole alle dort geloggten EventLogger Meldungen.




```
#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 );

        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, 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;
}
```

Sprache / IDE	Beispielprogram auspacken
CodeGear C++Builder 2009	https://infosys.beckhoff.com/content/1031/TcEventLogger/Resources/12333057163.zip

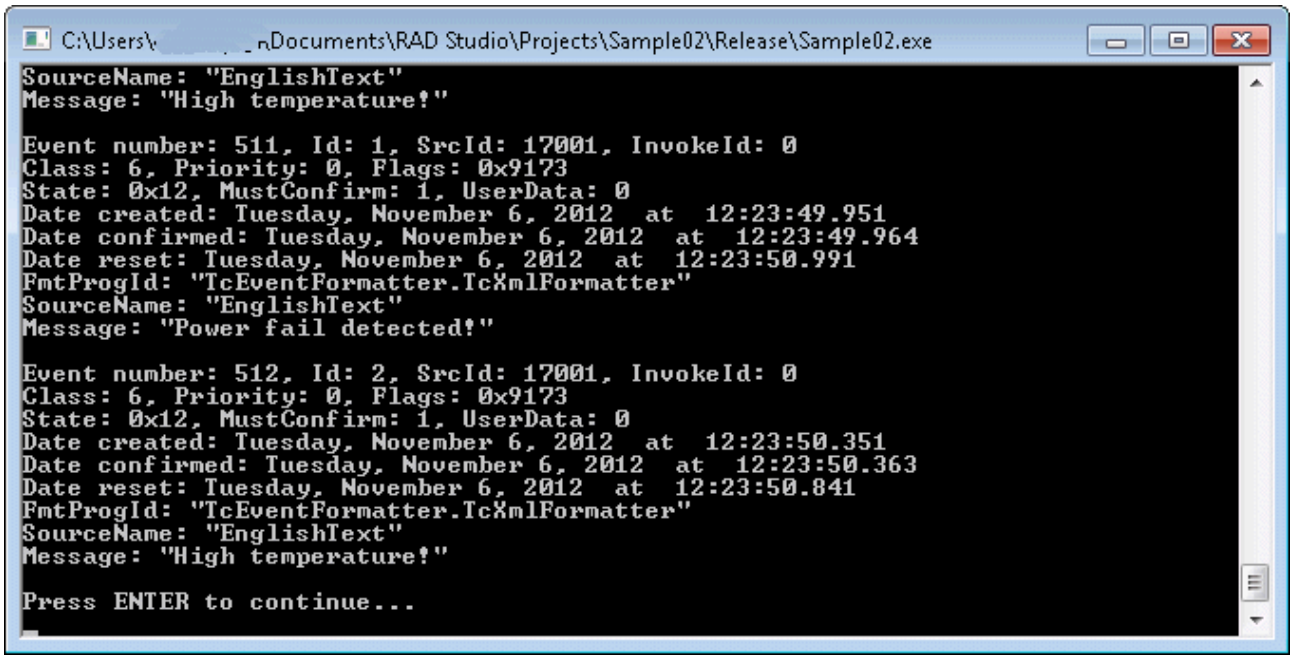
10.6.3 Konsolenanwendung - Geloggte Meldungen via ADS-Proxy-Schnittstelle

Systemvoraussetzungen:

- CodeGear C++Builder 2009;
- TwinCAT v2.11 B2228 oder höher;

- Die Typbibiothek der TcEventLogger.exe muss importiert werden (TCEVENTLOGGERLib_TLB.h);
- Die Typbibliothek der TcEventLogAdsProxy.dll muss importiert werden (TCEVENTLOGPROXYLib_TLB.h):

Das Beispiel baut eine ADS-Proxy-Verbindung zu einem Remote-EventLogger auf und zeigt in der Konsole alle dort geloggten EventLogger Meldungen.




```
#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 t
o 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))
                {
```

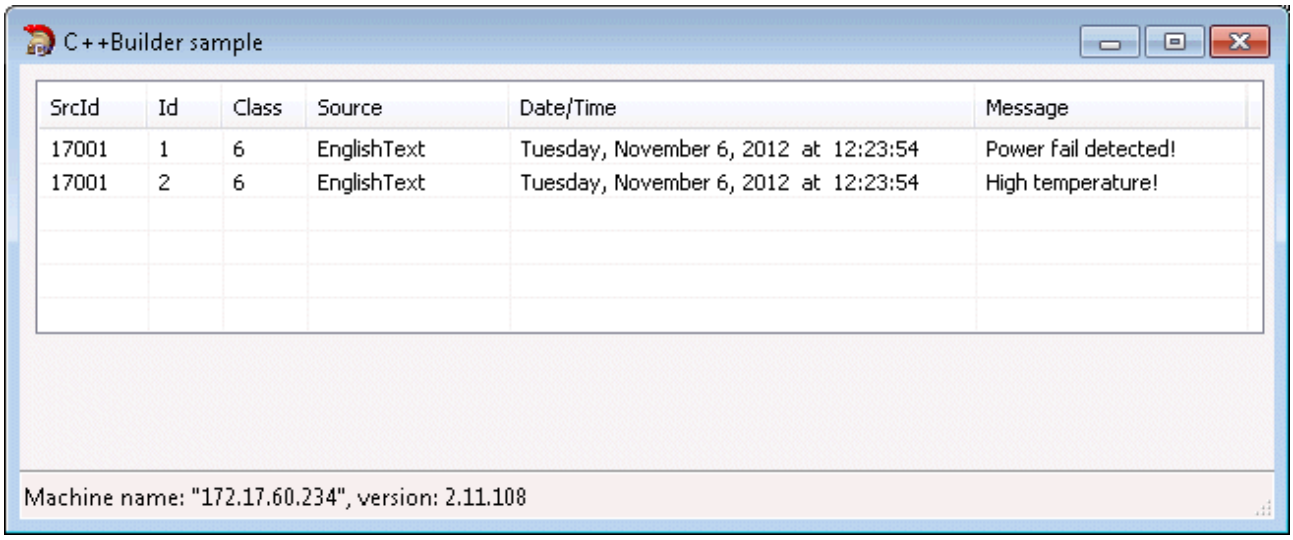
```
                    if (spEnumEvts->Count > 0)
                    {
                        for (long i = 0; i < spEnumEvts->Count; i++)
                        {
                            ITcEventPtr spEvent;
                            hr = spEnumEvts->Item(i, &spEvent); // get event
                         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;
}
```

Sprache / IDE	Beispielprogram auspacken
CodeGear C++Builder 2009	https://infosys.beckhoff.com/content/1031/TcEventLogger/Resources/12333058571.zip

10.6.4 Aktive Alarme in einer benutzerdefinierten Listenansicht

Systemvoraussetzungen:

- CodeGear C++Builder 2009;
- TwinCAT v2.11 B2228 oder höher;
- Die Typbibiothek der TcEventLogger.exe muss importiert werden (TCEVENTLOGGERLib_OCX.h);



Das Beispiel zeigt die aktiven Alarme in einer benutzerdefinierten C++Builder Listenansicht (TListView-Control).

Folgende Schritte müssen ausgeführt werden, um das Beispiel zu erstellen:

1. Erstellen Sie eine neue VCL-Formularanwendung.
2. Ziehen Sie von der Komponentenpalette die TTcEventLog und TStatusBar-ActiveX-Komponenten auf die Form.
3. Implementieren Sie die Methoden: FormCreate(), FormClose(). In FormCreate() wird die Verbindung zum EventLogger konfiguriert/ hergestellt und in FormClose() getrennt;
4. Implementieren Sie die Ereignisroutinen: OnNewEvent, OnResetEvent. Das Beispielprojekt unterstützt dadurch Alarme die nicht quittiert werden müssen. Um Ereignisse für quittierungspflichtige Alarme empfangen zu können müssen Sie weitere Ereignisroutinen implementieren: OnSignalEvent() und On-ConfirmEvent();
5. Implementieren Sie die Ereignisroutine OnShutdown(). Diese Routine wird aufgerufen wenn die Instanz des Eventloggers (z.B. beim Herunterfahren des Systems) komplett entladen wird;
6. Implementieren Sie die Hilfsmethoden: AddNewEvent() und DisplayActiveAlarms(). Diese Methoden werden benutzt um die aktiven Meldungen zu lesen und in der Liste anzuzeigen;
7. Implementieren Sie die Hilfsmethode: GetEventPosByID(). Diese Methode sucht in der Listenansicht nach einer bestimmten Meldung anhand der SourceId und ID und liefert beim Erfolg die Zeilennummer (itemIndex). Mit Hilfe der Zeilennummer kann die Meldung aus der Liste entfernt werden;

```
//---------------------------------------------------------------------------
#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
```

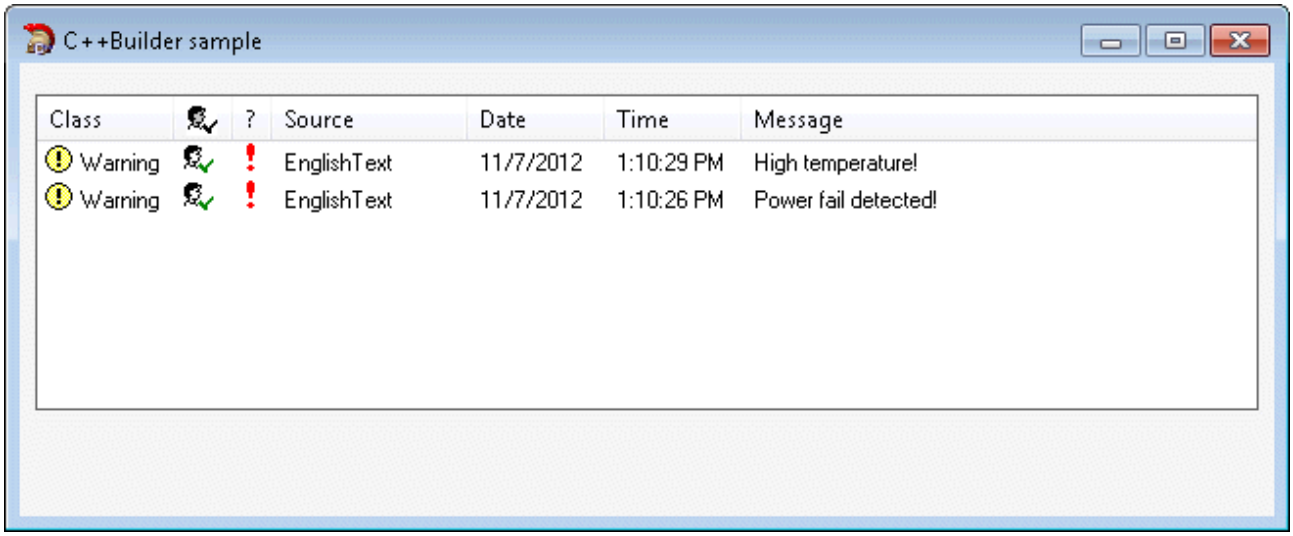
```
     pItem->SubItems->Add( spEvent->Id );

      // 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();
}
```

Sprache / IDE	Beispielprogram auspacken
CodeGear C++Builder 2009	https://infosys.beckhoff.com/content/1031/TcEventLogger/Resources/12333059979.zip

10.6.5 Einbinden von TcEventViewer-ActiveX-Control

Systemvoraussetzungen:

- CodeGear C++Builder 2009;
- TwinCAT v2.11 B2228 oder höher;
- Die Typbibiothek der TcEventViewer.dll muss importiert werden (TCEVENTVIEWERLib_OCX);



Im folgenden Beispiel wird das TcEventViewer-ActiveX-Control benutzt um aktive Alarme auf einem TwinCAT System anzuzeigen.

Folgende Schritte müssen ausgeführt werden, um das Beispiel zu erstellen:

1. Erstellen Sie eine neue VCL-Formularanwendung;
2. Ziehen Sie von der Komponentenpalette die TTcEventView auf die Form;
3. Implementieren Sie die Methoden: FormCreate(), FormClose();

In FormCreate() wird die Verbindung zum EventLogger konfiguriert/ hergestellt und in FormClose() getrennt. Falls Sie auf einen Remote-PC zugreifen wollen dann muss die Netzwerkadresse des TwinCAT Systems (AmsNetId) passend konfiguriert werden.

```
#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;
        }
    }
}
```

Sprache / IDE	Beispielprogram auspacken
CodeGear C++Builder 2009	https://infosys.beckhoff.com/content/1031/TcEventLogger/Resources/12333061387.zip

Mehr Informationen:
www.beckhoff.de/automation

Beckhoff Automation GmbH & Co. KG
Hülshorstweg 20
33415 Verl
Deutschland
Telefon: +49 5246 9630
info@beckhoff.de
www.beckhoff.de