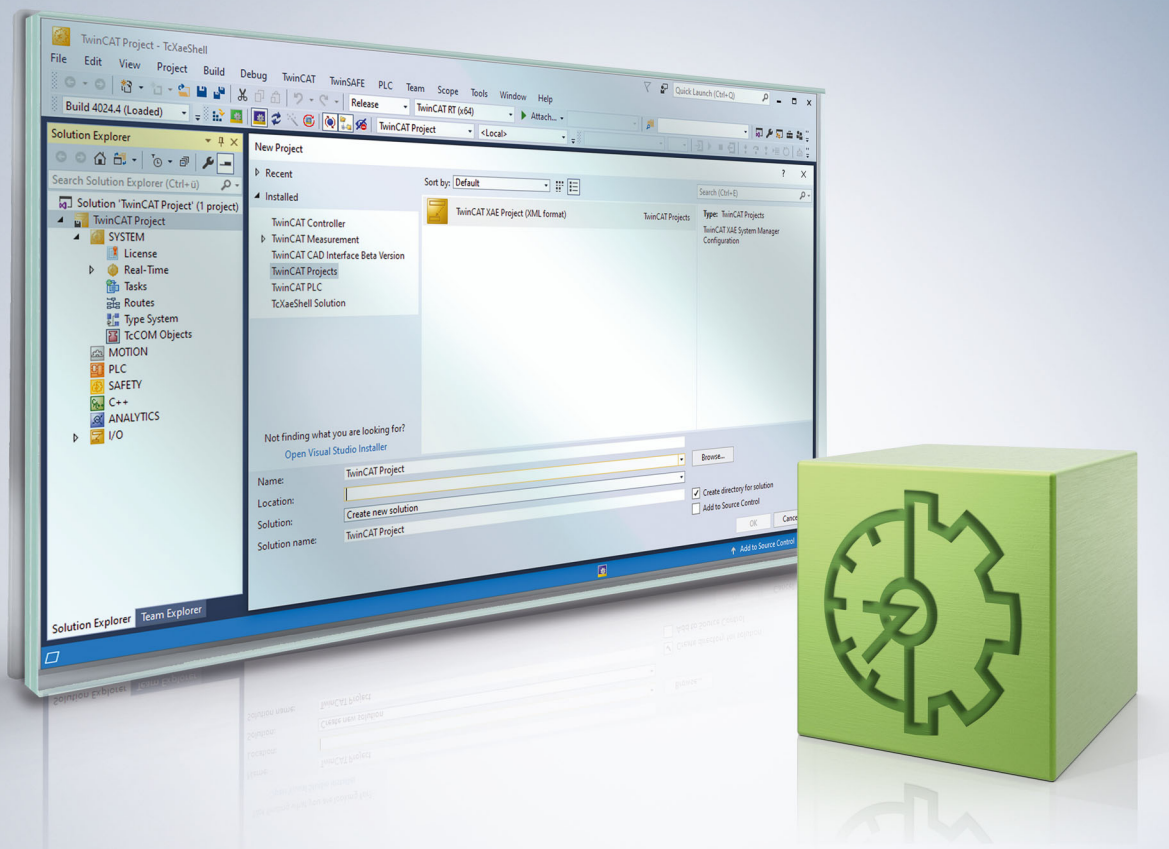
BECKHOFF New Automation Technology

Handbuch | DE

TE1000

TwinCAT 3 | EventLogger



01.08.2025 | Version: 1.11.3

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, stets die aktuell gültige 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 weiterentwickelt.
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®, ATRO®, EtherCAT®, EtherCAT G®, EtherCAT G10®, EtherCAT P®, MX-System®, Safety over EtherCAT®, TC/BSD®, TwinCAT®, TwinCAT/BSD®, TwinSAFE®, XFC®, XPlanar® und XTS® 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 Kennzeichnungen führen.

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

Copyright

Fremdmarken

In dieser Dokumentation können Marken Dritter verwendet werden. Die zugehörigen Markenvermerke finden Sie unter: https://www.beckhoff.com/trademarks.

1.2 Zu Ihrer Sicherheit

Sicherheitsbestimmungen

Lesen Sie die folgenden Erklärungen zu Ihrer Sicherheit.
Beachten und befolgen Sie stets produktspezifische Sicherheitshinweise, die Sie gegebenenfalls an den entsprechenden Stellen in diesem Dokument vorfinden.

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.

Signalwörter

Im Folgenden werden die Signalwörter eingeordnet, die in der Dokumentation verwendet werden. Um Personen- und Sachschäden zu vermeiden, lesen und befolgen Sie die Sicherheits- und Warnhinweise.

Warnungen vor Personenschäden

⚠ GEFAHR

Es besteht eine Gefährdung mit hohem Risikograd, die den Tod oder eine schwere Verletzung zur Folge hat.

⚠ WARNUNG

Es besteht eine Gefährdung mit mittlerem Risikograd, die den Tod oder eine schwere Verletzung zur Folge haben kann.

⚠ VORSICHT

Es besteht eine Gefährdung mit geringem Risikograd, die eine mittelschwere oder leichte Verletzung zur Folge haben kann.

Warnung vor Umwelt- oder Sachschäden

HINWEIS

Es besteht eine mögliche Schädigung für Umwelt, Geräte oder Daten.

Information zum Umgang mit dem Produkt

Diese Information beinhaltet z. B.:
Handlungsempfehlungen, Hilfestellungen oder weiterführende Informationen zum Produkt.

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.

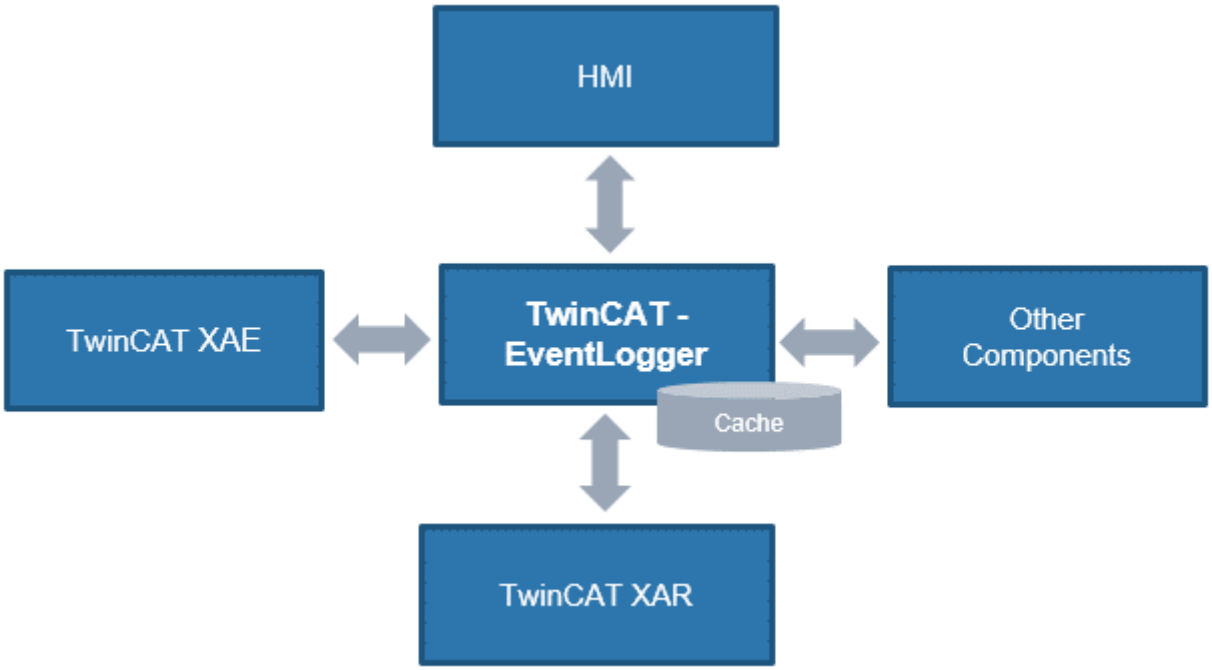
Zusätzlich sollten die Empfehlungen von Beckhoff zu entsprechenden Schutzmaßnahmen beachtet werden. Weiterführende Informationen über Informationssicherheit und Industrial Security finden Sie in unserem https://www.beckhoff.de/secguide.

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 Übersicht

Der TwinCAT 3 EventLogger stellt eine Schnittstelle zum Austausch von Nachrichten zwischen verschiedenen TwinCAT- und Nicht-TwinCAT-Komponenten bereit.



Diese Dokumentation richtet sich an die Anwender des TwinCAT 3 EventLoggers. Informationen dazu, wie der TwinCAT 3 EventLogger von unterschiedlichen TwinCAT-3-Komponenten genutzt wird, finden Sie in den jeweiligen Produktdokumentationen.

Produktkomponenten

Alle Komponenten des TwinCAT 3 EventLogger sind in der jeweiligen Basisinstallation vorhanden.

Zur Programmierung sind im TwinCAT 3 Engineering verschiedene Oberflächen enthalten. Die nötigen Module und Bibliotheken sind Bestandteil der entsprechenden Laufzeit.

Die Verwendung des TwinCAT 3 EventLogger ist lizenzkostenfrei.

3 Systemvoraussetzungen

Technische Daten	Voraussetzung
Betriebssystem	Windows 10, Windows CE 7, TwinCAT/BSD
Zielplattform	PC-Architektur (x86, x64 und Arm®)
Minimale TwinCAT-Version	TwinCAT 3.1 Build 4022.20 und höher
Erforderliche TwinCAT-Lizenz	Beliebige Laufzeitlizenz (PLC, C++)
Visual-Studio-Version	Teile des TwinCAT 3 EventLoggers sind ab Visual Studio 2013 verwendbar.

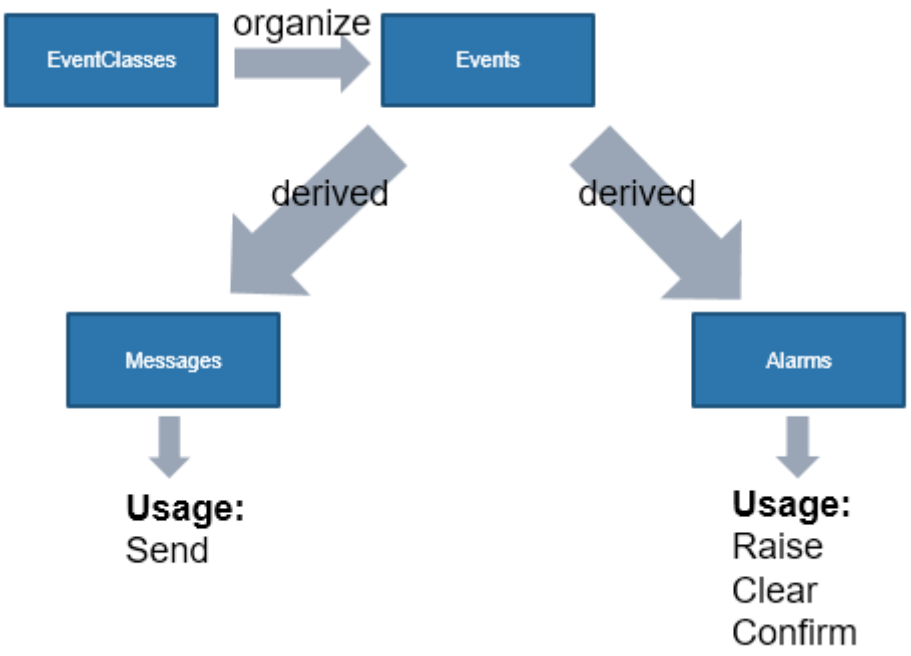
4 Limitierungen

- Ereignisse haben bei der Übertragung eine maximale Größe von 8 kb. Achten Sie bei der Nutzung des TwinCAT 3 EventLoggers darauf, dass diese Grenze eingehalten wird. Sie bezieht sich auf alle Elemente, die übertragen werden und in dieser Dokumentation beschrieben sind, einschließlich der dynamischen Elemente (Attribute, SourceName, JSONAttribute).
- Die Schnittstelle zum Empfangen von Ereignissen in der Echtzeit speichert max. 1024 Ereignisse zwischen, bis diese abgeholt sein müssen.
  Sollten sie nicht rechtzeitig abgeholt worden, gehen Ereignisse verloren.
- Der TwinCAT 3 EventLogger bietet eine Anbindung an das TwinCAT HMI (TF2xxx). Das TwinCAT PLC HMI (TF18xx) hingegen kann die Ereignisse nicht empfangen.

5 Technische Einführung

Der TwinCAT 3 EventLogger überträgt sogenannte Ereignisse (engl. Events). Ein Ereignis ist hier eine Nachricht oder ein Alarm.

Diese technische Einführung fokussiert die Daten, die als Inhalt eines Ereignisses übertragen werden, denn diese sind für das Verständnis des Ablaufs nötig. In den API-Beschreibungen für SPS und C++ sowie in den Beispielen wird das Senden und Empfangen eines Ereignisses detailliert erläutert.



Ereignisse

Ein Ereignis selbst wird nicht direkt verwendet, sondern die abgeleiteten Typen „Nachrichten“ oder „Alarme“.

Ein Ereignis stellt die folgenden gemeinsamen Elemente von Nachrichten und Alarmen bereit:

EventClass (GUID)	Ereignisklassen sind eine Gruppierung von Events (möglicherweise zu einem Thema).
Event-ID (UDINT)	Innerhalb einer Eventklasse identifiziert eine Event-ID das Ereignis eindeutig.
Text (String)	Beschreibung des Ereignisses. Die Beschreibung ist für Menschen gedacht und kann somit auch internationalisiert werden. Zur Laufzeit können Argumente eingefügt werden, um den Text individuell anpassbar zu machen.
Source Info	Quelle des Auftretens eines Ereignisses. Source besteht dabei aus drei Elementen. Diese können beliebig genutzt werden; es gibt aber eine entsprechende Empfehlung, wie sie zu verwenden sind. • **Source-ID (INT)**: TcCOM-Objekt-ID • **Source-Name (STRING)**: Pfad innerhalb des TcCOM-Objektes, z. B. Pfad zu einem Funktionsbaustein in einem SPS-Projekt • **Source-GUID (GUID)**: Kann verwendet werden, um z. B. ein Projekt oder (Teil-) Produkt zu identifizieren.
JSON Attribute (STRING)	Kann beliebig verwendet werden. Während der „Text“ (siehe oben) für Menschen als Empfänger gedacht ist, ist das JSON-Attribut für den programmatischen Empfang vorgesehen. Ein JSON-String kann einfach erzeugt (serialisiert) und auch verarbeitet (deserialisiert) werden. TwinCAT bietet hierfür die JsonXml-Bibliothek in der Echtzeit an (siehe Dokumentation SPS-Bibliothek Tc3_JsonXml).

Neben diesen Elementen besitzen Ereignisse weitere Elemente, die im TMC Editor [▶ 19] beschrieben sind.

Nachrichten

Nachrichten sind zustandslos. Sie werden bei einem Aufruf gesendet und den entsprechend registrierten Komponenten zugestellt.

Identifikation

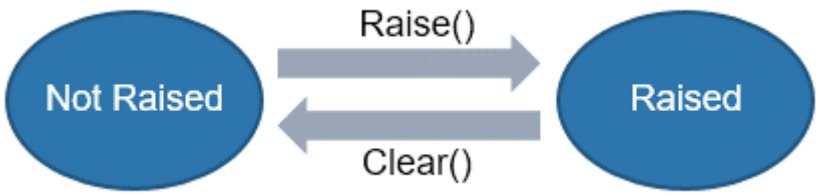
Nachrichten werden anhand der EventClass und Event-ID identifiziert.

Alarm

Ein Alarm ist im Gegensatz zu einer Nachricht nicht zustandslos.

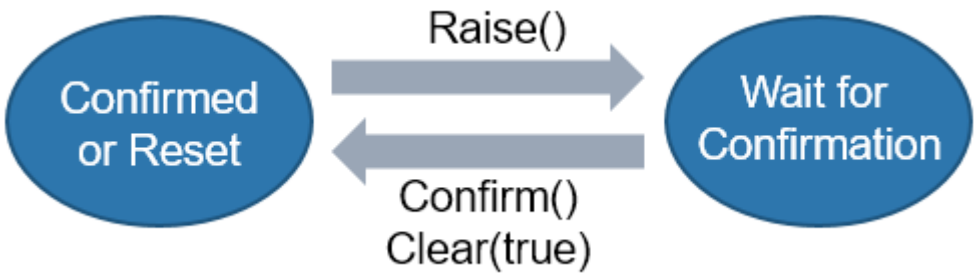
Er besitzt folgende Alarmzustände:

- Not-Raised
- Raised



Zusätzlich kann eine Bestätigung verlangt werden. Folgende Bestätigungszustände werden unterschieden:

- WaitForConfirmation
- Confirmed oder Reset



Ein Aufruf der entsprechenden Methoden versendet ein Ereignis und überführt den Alarm in den entsprechenden Zustand.

- Wird eine Quittierung über Confirm() vorgenommen, wird der Zustand in Confirmed überführt.
- Wird eine Quittierung über Clear(TRUE) vorgenommen, wird der Zustand in Reset überführt.

Ist der Aufruf einer Methode im aktuellen Zustand nicht gültig, wird dies durch einen Rückgabewert angezeigt.

Beim Herunterfahren von TwinCAT (Übergang RUN → CONFIG) wird für alle Alarme im Zustand Raised intern ein Dispose ausgeführt, das einen Clear-Zeitstempel setzt. Eine Bestätigung für diese Alarme entfällt.

Identifikation

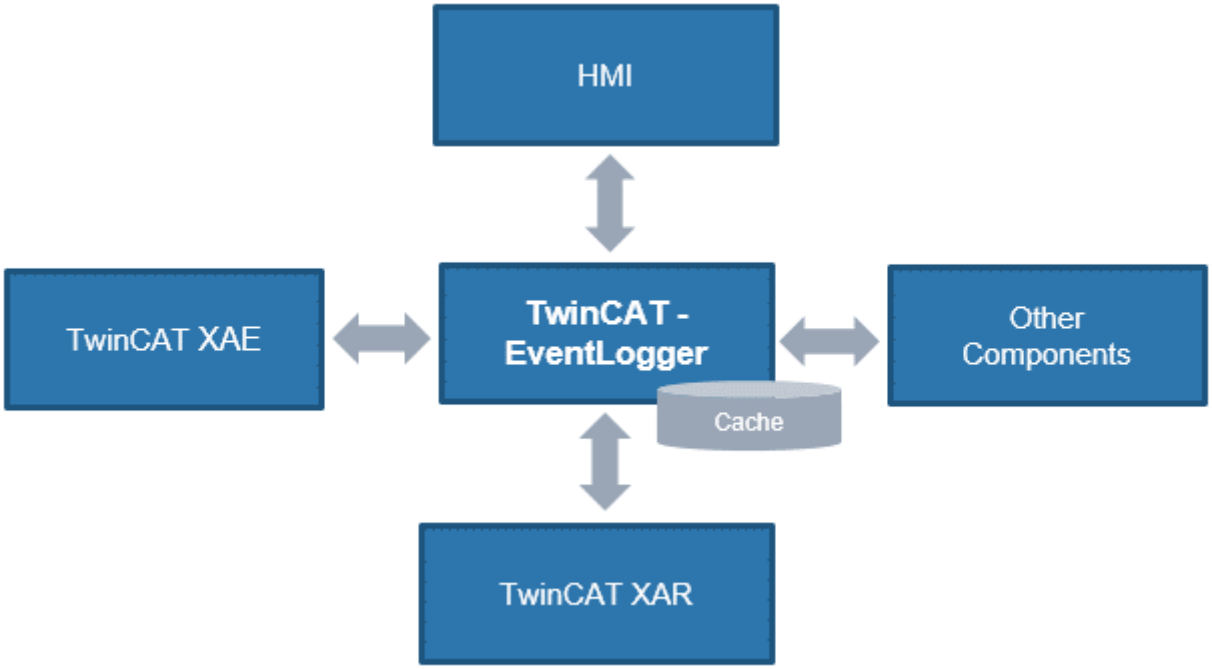
Der TwinCAT 3 EventLogger identifiziert einen Alarm anhand der EventClass, Event-ID und Source Info. Damit kann ein Alarm (Kombination aus EventClass und Event-ID) an unterschiedlichen Stellen im Programm verwendet werden. Beispielsweise kann ein Alarm „Lager leer“ für unterschiedliche Lager verwendet werden, da unterschiedliche „Source Infos“ zur Laufzeit bereitgestellt werden (siehe auch Umgang mit Quellen [▶ 32]).

Architektur

Der TwinCAT 3 EventLogger überträgt Ereignisse zentral zwischen anderen Komponenten. Zu den Komponenten gehören die Echtzeit-Programmierschnittstellen SPS oder C++ als primäre Quelle von Ereignissen.

Während der Entwicklung können die Nachrichten im TwinCAT Engineering (XAE) angezeigt werden.

Ein HMI kann Nachrichten beispielsweise empfangen und entsprechend anzeigen. Weitere Komponenten können durch den Kunden selbst erstellt werden, um Ereignisse zu empfangen.



Der TwinCAT 3 EventLogger hält dabei eine begrenzte Anzahl der letzten Ereignisse in einem Cache. Dieser kann z. B. nach einem Neustart aus dem Engineering heraus abgefragt werden und so eine Diagnose ermöglichen. Der Cache hängt vom gesicherten Herunterfahren des Rechners ab.

Cache unter Windows CE nicht persistent

Ab TwinCAT 3.1 Build 4024.25 wird der Cache der Ereignisse unter Windows CE Systemen aus Performancegründen nicht persistiert.

Workflow

Der allgemeine Workflow zum Aufbau einer asynchronen Kommunikation zwischen Komponenten sieht wie folgt aus:

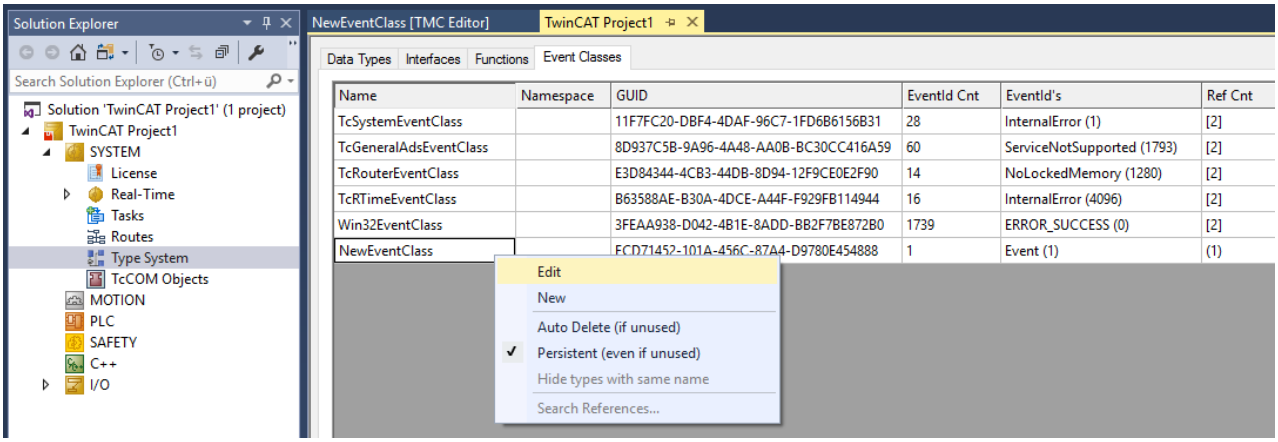
1. Erstellung eines neuen TwinCAT-Projekts.
2. Definition von Eventklassen und Events im TwinCAT-Typsystem.
3. Ausführung der automatischen Code-Generierung, um den Quellcode für die Echtzeit-Programmiersprachen in TwinCAT bereitzustellen.
4. Implementierung der Verwendung und somit des Sendens und Empfangens von Ereignissen.

Siehe auch:

- Dokumentation TwinCAT 3 Typsystem
- API-Beschreibungen SPS [▶ 39] und C++ [▶ 143]
- Beispiele [▶ 201]

5.1 Ereignisklasse

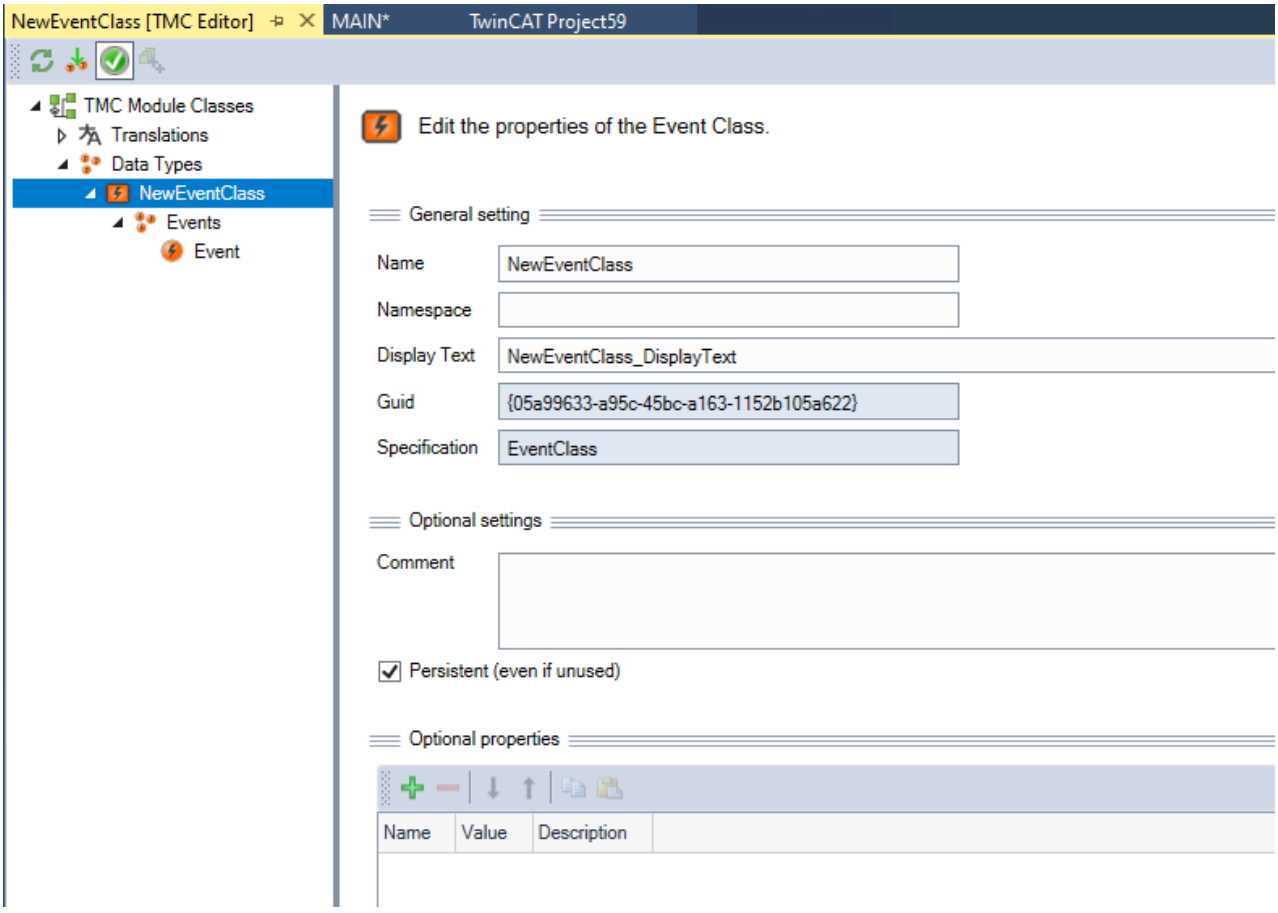
Ereignisklassen sind Gruppierungen von Ereignissen (möglicherweise zu einem Thema) und im Sinne des TwinCAT-Typsystems Datentypen, die in unterschiedlichen Modulen verwendet werden können. Aus diesem Grund werden sie als Datentypen im TwinCAT-Typsystem angelegt (System > Type System > Event Classes).



Im TwinCAT-Typsystem in der Registerkarte **Event Classes** werden alle bekannten Ereignisklassen aufgelistet. Über die Kontextmenübefehle **Edit** und **New** kann der TMC-Editor geöffnet werden, in dem die Ereignisklassen definiert und bearbeitet werden können.

TwinCAT stellt neben den Ereignisklassen des Projekts weitere Ereignisklassen bereit, z. B. ADS-Returncodes und Systemereignisse.

Eigenschaften von Ereignisklassen

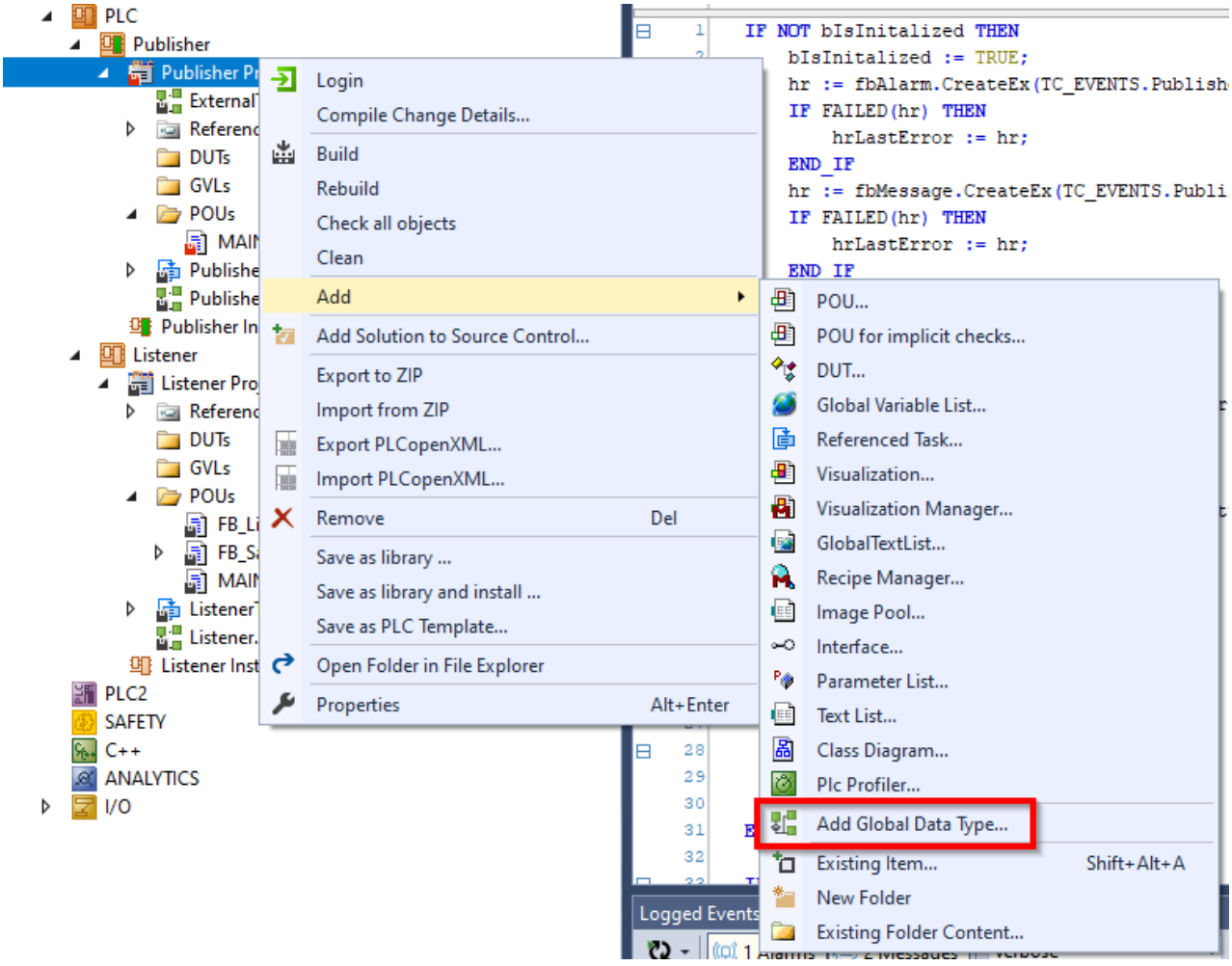


Name	Beschreibung
Name	Dieser Name bezeichnet die Ereignisklasse z. B. in dem generierten Quellcode.
Namespace	Wie alle Datentypen können auch Ereignisklassen einem Namespace angehören.
Display Text	Dieser Text wird für die Ereignisklasse zur Anzeige verwendet. Er ist internationalisierbar (siehe Internationalisierung/Translations [▶ 24]).
Guid	Wie bei allen Datentypen identifiziert sie die aktuell beschriebene Ereignisklasse. Sie wird automatisch berechnet und verändert sich bei Änderungen innerhalb der Ereignisklasse.
Optional properties	Durch das Setzen von Properties kann das Verhalten der Ereignisklassen verändert werden: • Name „plcAttribute_to_string“ auf Value „1“ wird ein Attribut 'to_string' in der PLC erzeugt. • „HideNeedlessArguments“ auf Value „1“ unterdrückt die Ausgabe von überflüssigen Argumenten in den Ereignistexten.

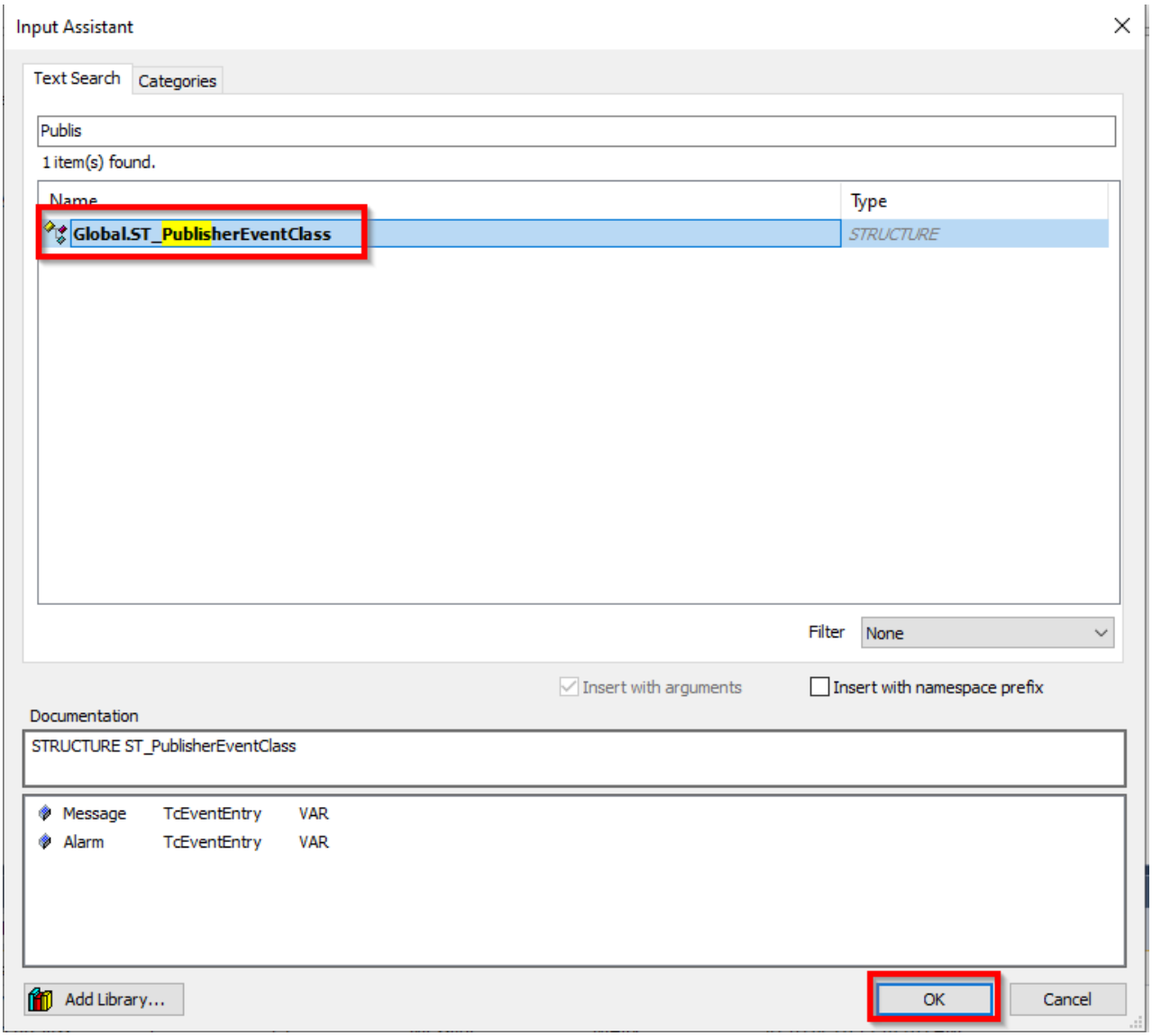
Ereignisklassen in SPS-Bibliotheken

Wenn Ereignisklassen in einer SPS-Bibliothek genutzt werden, sollten diese auch Bestandteil der Bibliothek sein, sodass sichergestellt ist, dass die Datentypen im TwinCAT-Typsystem jeder Applikation, die die SPS-Bibliothek nutzt, enthalten sind.

Hierfür müssen die Ereignisklassen in der SPS-Bibliothek „gepinnt“ werden. Wählen Sie dazu im SPS-Bibliotheksobjekt im Kontextmenü **Add** den Befehl **Add Global Data Type …** (in TwinCAT 3.1 4024: **External Types -> Pin Global Data Type)**.

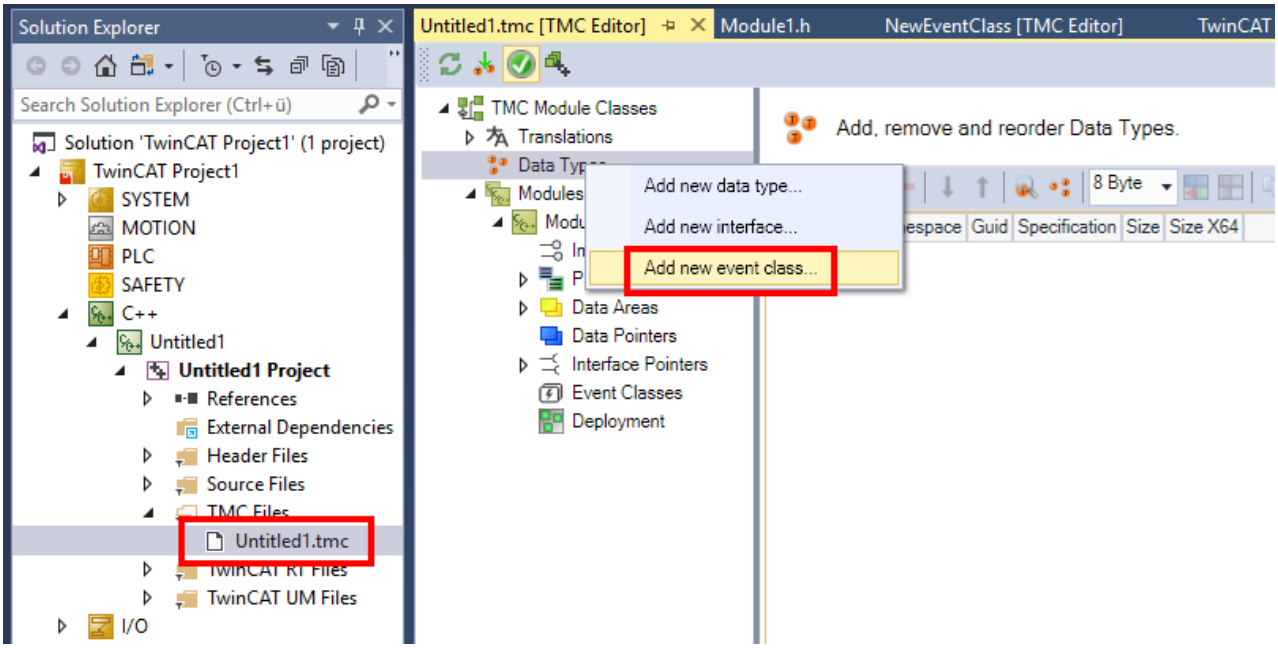


Wählen Sie den Datentyp der Ereignisklasse aus:

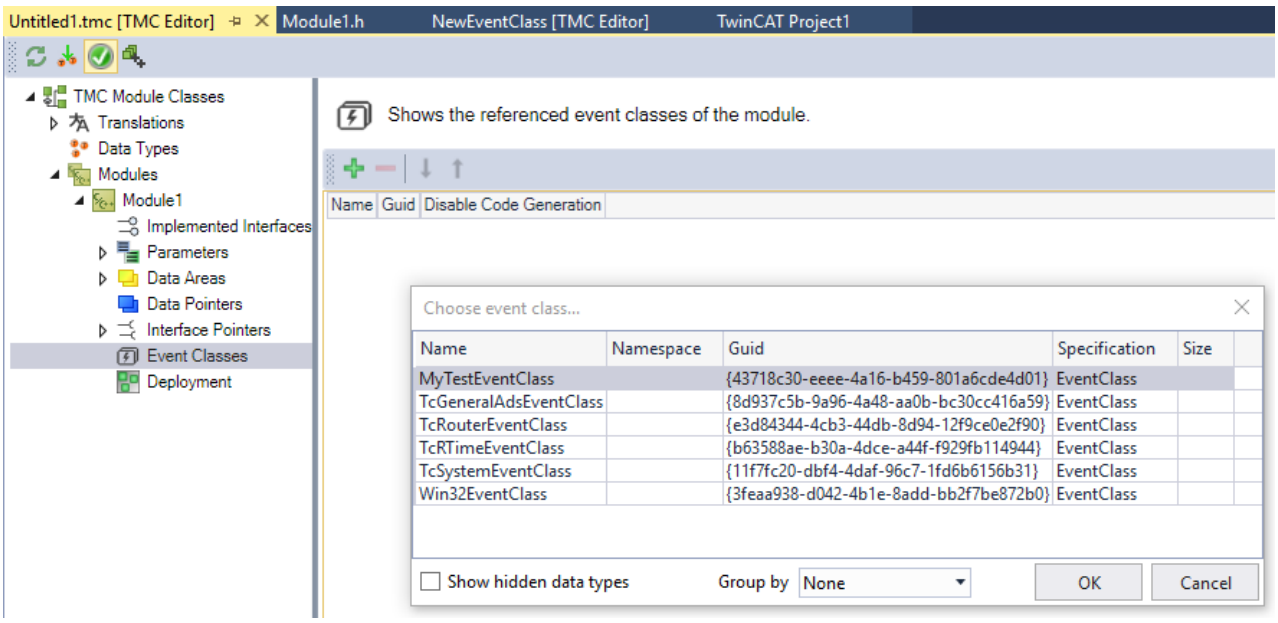


Ereignisklassen in C++-Projekten

Für C++-Projekte gibt es analog zu den anderen TwinCAT-Datentypen die Möglichkeit Ereignisklassen lokal im C++-Projekt zu definieren.

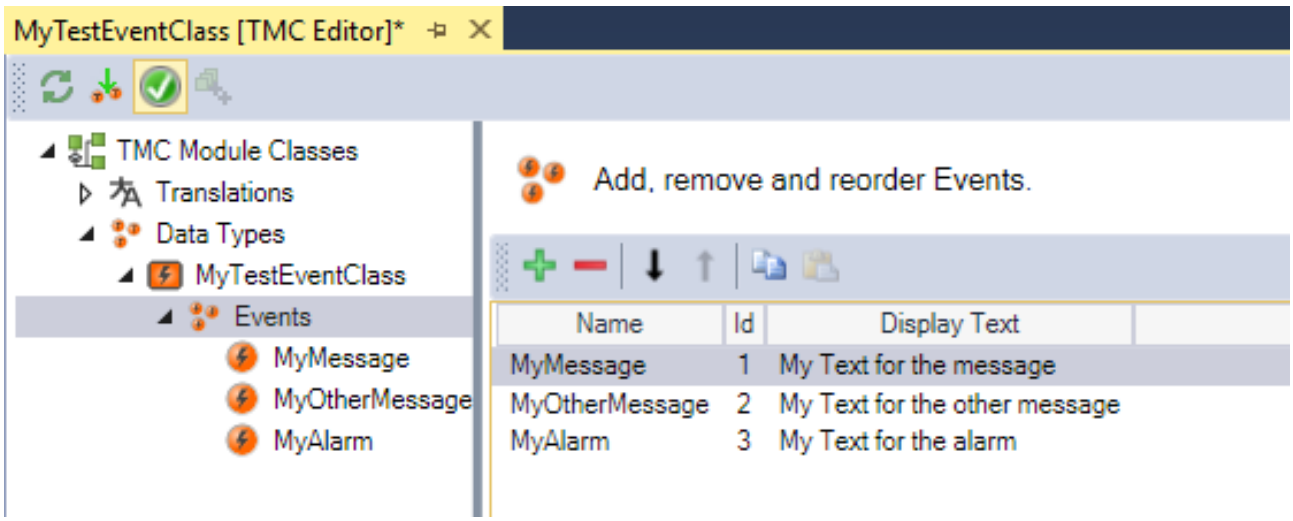


Die Ereignisklassen werden zur Verwendung in den entsprechenden C++-Modulen deklariert (unabhängig davon, wo sie definiert wurden). Wenn in einem C++-Modul eine Ereignisklasse ergänzt wird, wird diese automatisch in die TMC-Datei des Moduls eingebettet.



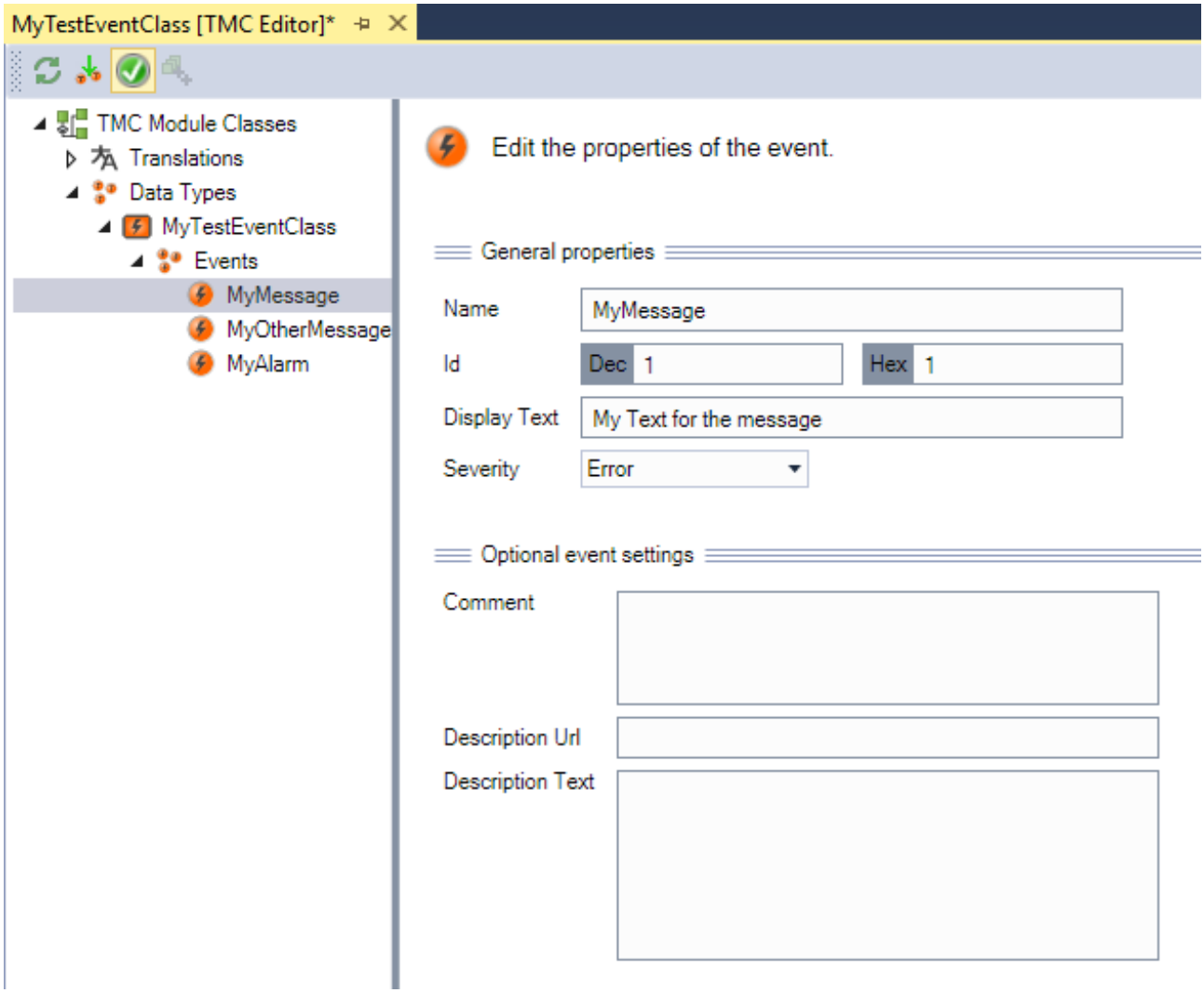
5.2 Ereignis

Ereignisse werden innerhalb der Ereignisklasse beschrieben.



Durch die Unterelemente können die Ereignisse entsprechend konfiguriert werden.

Eigenschaften von Ereignissen



Name	Dieser Name bezeichnet das Ereignis, z. B. in dem generierten Sourcecode.
ID	Identifiziert das Ereignis innerhalb der Ereignisklasse eindeutig.
Display Text	Dieser Text wird für die Ereignisklasse zur Anzeige verwendet. Er ist internationalisierbar (siehe Internationalisierung/Translations [▶ 24]).
Severity	Schweregrad des Ereignisses. Er wird durch den generierten Sourcecode bereitgestellt und stellt somit das Standardverhalten dar, jedoch kann dort auch ein anderer Wert verwendet werden: • Verbose • Info • Warning • Error • Critical

5.3 Ereignisse mit Excel verwalten

TwinCAT Package “TwinCAT.XAE.EventloggerExcelAddIn” wird benötigt

Installieren Sie das Paket „TwinCAT.XAE.EventloggerExcelAddIn“, um die hier beschriebene Funktionalität zu nutzen.

TwinCAT bietet neben dem TMC-Editor als Teil der TwinCAT-Solution auch die Möglichkeit Microsoft Excel zu nutzen, um Ereignisklassen mit Ereignissen und ihren Übersetzungen zu erstellen. Diese können dann aus Excel exportiert und in TwinCAT als „Shared TMC“ eingebunden werden.

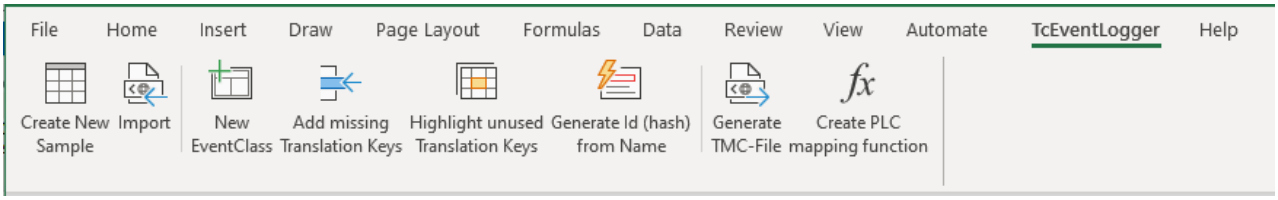
Installation

Nach der Installation des Paketes liegt ein entsprechendes Setup unter *C:\ProgramData\Beckhoff\TwinCAT\EventloggerExcelAddIn* bereit. Das Verzeichnis kann von hier aus auch auf andere Systeme übertragen werden, die kein TwinCAT installiert haben.

Das Setup kann genutzt werden, wenn Microsoft Excel installiert und aktiviert ist. Hierfür einfach die setup.exe ausführen, um die Integration in Excel durchzuführen.

Nutzung

Nach einem Start von Excel steht ein neuer Hauptmenüpunkt in Excel bereit:



- **Create New Sample** erstellt in der aktuellen Projektmappe sowohl die Basisstruktur wie auch die zwei beispielhaften Ereignisklassen „Master“ und „Machine“.
- **Import** wird genutzt, um die Ressource Informationen zu importieren und in Ereignisklassen zu überführen. Vgl. Importieren von Resource Dateien [▶ 37]
- **New EventClass** legt eine neue Ereignisklasse als neues Blatt an.
- **Add missing Translation Keys** legt basierend auf allen vorhandenen Ereignisklassen entsprechende Schlüssel im Blatt „Translations“ an.
- **Highlight unused Translation Keys** markiert die Schlüssel, die aktuell nicht verwendet werden.
- **Generate Id (hash) from Name** erzeugt für eine Ereignisklasse einen Hashwert des Namens als EventId. Vgl. Internationalisierung von Quellen [▶ 33]
- **Generate TMC-File** erzeugt eine TMC-Datei auf Basis der Ereignisklassen. Diese kann als Shared TMC in TwinCAT verwendet werden.

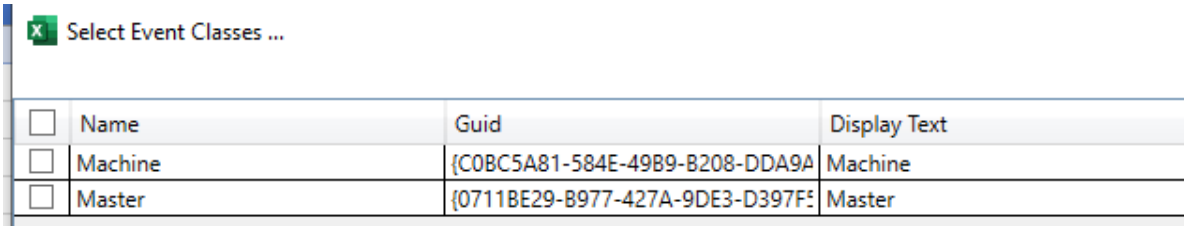
- **Create PLC mapping function** erzeugt eine TcPOU-Datei als PLC Function. Diese stellt ein Mapping dar, um die in Resource Dateien verwendeten SourceId und EventId auf die Ereignisklassen zu mappen. Vgl. Importieren von Resource Dateien [▶ 37]

Arbeitsablauf und Einbindung

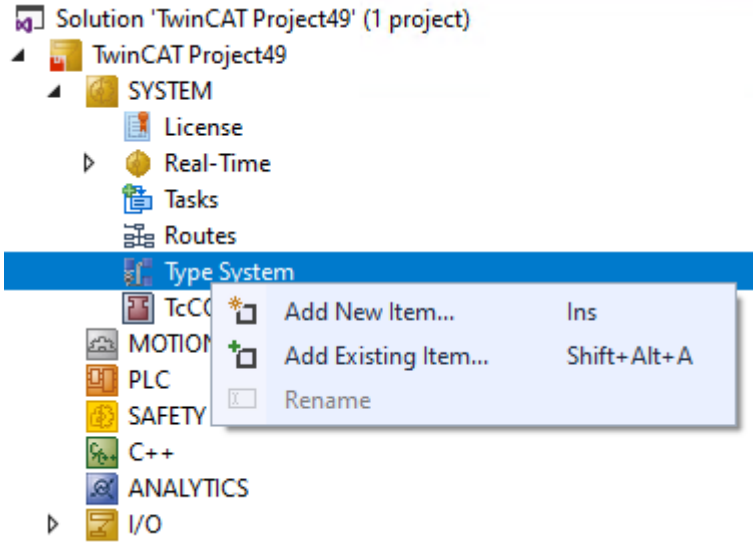
An dieser Stelle wird der Arbeitsablauf zwischen Excel und TwinCAT beschrieben:

✓ Die setup.exe wurde ausgeführt und das Add-in ist im Microsoft Excel registriert.

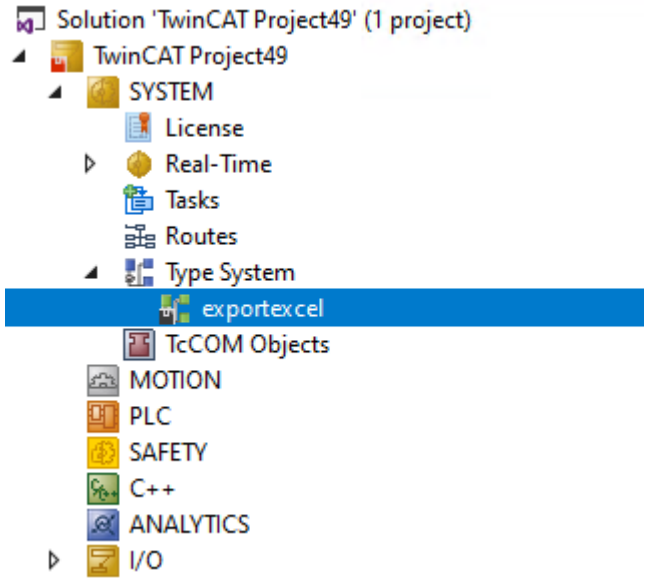
1. Wechseln Sie auf den Menüpunkt TcEventLogger und klicken Sie auf „Create New Sample“.
   ⇨ Die beispielhaften Ereignisklassen werden angelegt.
2. Sie können diese erweitern oder auch Übersetzungen in dem Blatt „Translations“ anlegen.
3. Nutzen Sie für die Erzeugung der Einträge „Add missing Translation Keys“ und auch „Highlight unused Translation Keys“.
   ⇨ Bei „Generate TMC-File“ erscheint ein Auswahldialog, in dem die zu exportierenden Ereignisklassen ausgewählt werden:



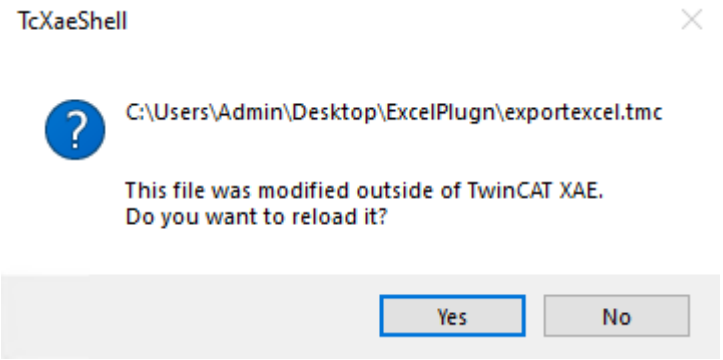
   ⇨ Es wird ein Dialog zum Auswählen des Speicherdialoges angezeigt. Die TMC-Datei wird dabei erzeugt.
4. Binden Sie die TMC-Datei über das Type System -> Rechtsklick -> Add Existing Item… ein:



⇨ Die Datei wird als Referenz in der TwinCAT-Solution aufgenommen. Die enthaltenen Ereignisklassen stehen bereit:



5. Wenn Sie die Datei überschreiben, merkt TwinCAT das und übernimmt entsprechende Updates:



⇨ Die Beschreibung der Ereignisklassen wird in Excel vorgenommen und über die Exportfunktion automatisiert in TwinCAT importiert.

Bei diesem Ablauf ist die Excel-Datei die „führende“ Datei. Es muss organisatorisch verhindert werden, dass Änderungen über den TMC-Editor in der TwinCAT-Solution vorgenommen werden, da diese nicht zurückgespielt werden können.

5.4 Code-Generierung der Ereignisdefinition

Aus der Definition von Ereignisklassen mit Ereignissen wird sowohl in der SPS als auch im C++-Sourcecode generiert.

Bei der Code-Generierung werden die „Namen“ der Ereignisse und Ereignisklassen verwendet. So kann eine Ereignisklasse durch den Namen über unterschiedliche Versionen einer Ereignisklasse hinweg verwendet werden.

Die „Severity“ wird durch die Code-Generierung bereitgestellt. So hat der Programmierer die Möglichkeit, diese individuell beim Anlegen der Events (Create) einzustellen. Die „Severity“, wie sie im TMC-Editor [▶ 19] beschrieben wird, ist damit als Standardverhalten zu betrachten. Im konkreten Verwendungsfall kann die „Severity“ hiervon abweichen.

SPS

In der SPS wird eine GVL TC_EVENTS angelegt, die als Unterelemente die Ereignisklassen hat und nach Änderungen (Speichern/Schließen des TMC-Editors) aktualisiert wird. Diese globalen Konstanten haben wiederum die Ereignisse selbst als Unterelemente zusammen mit den einzelnen Elementen EventId, Severity und der UUID der Ereignisklasse, zu der sie gehören. Zusätzlich wird auch eine GVL TC_EVENT_CLASSES erzeugt, welche die GUIDs der Ereignisklassen direkt zugreifbar macht.

Diese Elemente können per IntelliSense für die Parameter z. B. bei Create()/CreateEx() verwendet werden.

Eingeloggte SPS

Wenn eine Verbindung mit einer SPS besteht (Login), wird der Code nicht aktualisiert. Die Aktualisierung erfolgt dann nach einem Logout.

Per „OnlineChange“ können die Änderungen an einer Ereignisklasse übernommen werden, wenn die Option „Update boot project“ angewählt wird.

C++

TcCOM-Module müssen die Eventklassen verwenden, d. h. im TMC-Editor müssen diese als verwendet eingetragen werden. Eine Code-Generierung erzeugt dann einen Namespace „TcEvents“ als Teil der <DriverName>Services.h-Datei, der per IntelliSense für die Parameter der Ereignisklassen/Ereignisse, z. B. bei CreateMessage()/CreateAlarm(), verwendet werden kann.

Kompatibilität

Die C++-Sourcecode-Generierung setzt Visual Studio 2013 oder neuer voraus.

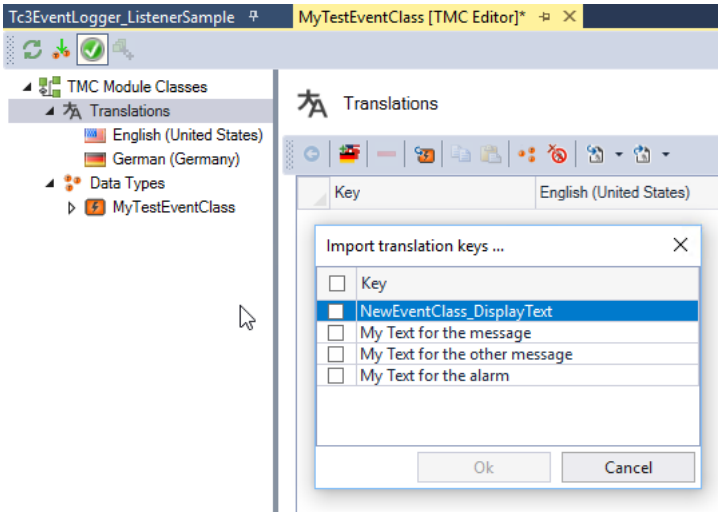
5.5 Internationalisierung/Translations

Die Texte der Ereignisse („Display Text“) für die Anzeige in z. B. HMIs können internationalisiert werden.

Hierfür steht im TMC Editor der Bereich „Translations“ bereit. Der Bereich beschreibt eine Tabelle, die in den Zeilen die Schlüssel (Keys) den Texten in unterschiedlichen Sprachen zuordnet.

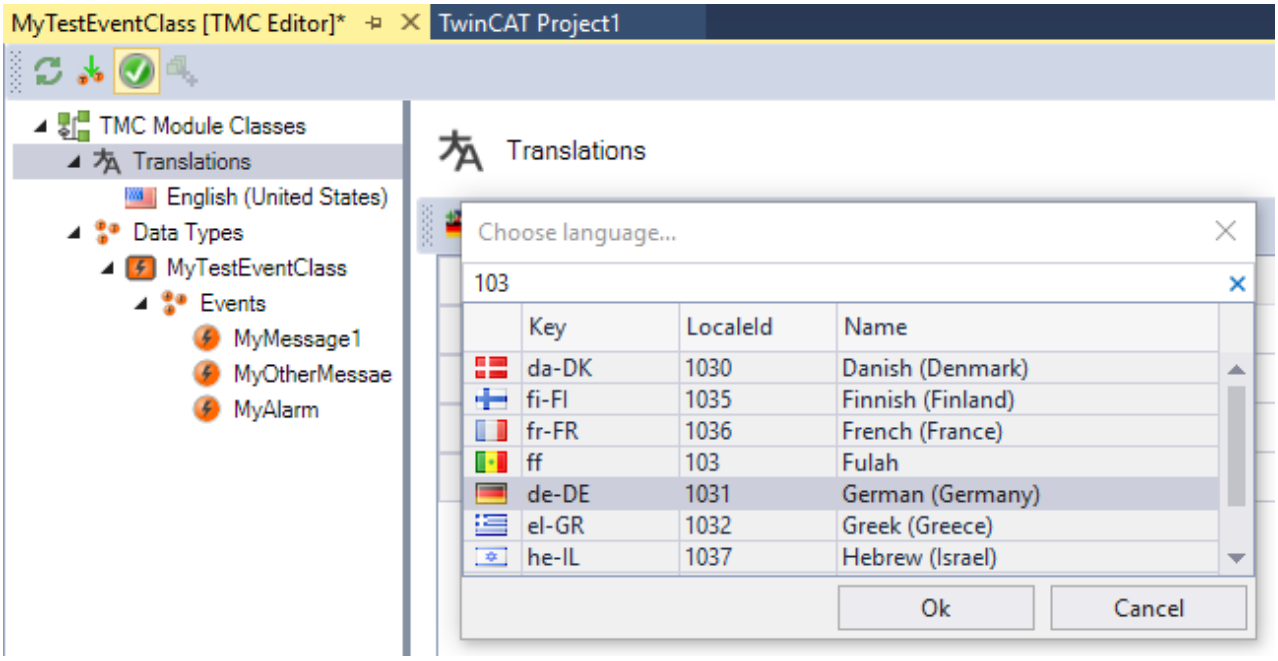
Zu übersetzende Schlüssel (Keys)

Die Keys werden automatisch aus den Texten („Display Text“) der Ereignisse und Ereignisklassen ermittelt. Ob die Keys in die Übersetzung übernommen werden sollen, kann einzeln beschrieben werden:



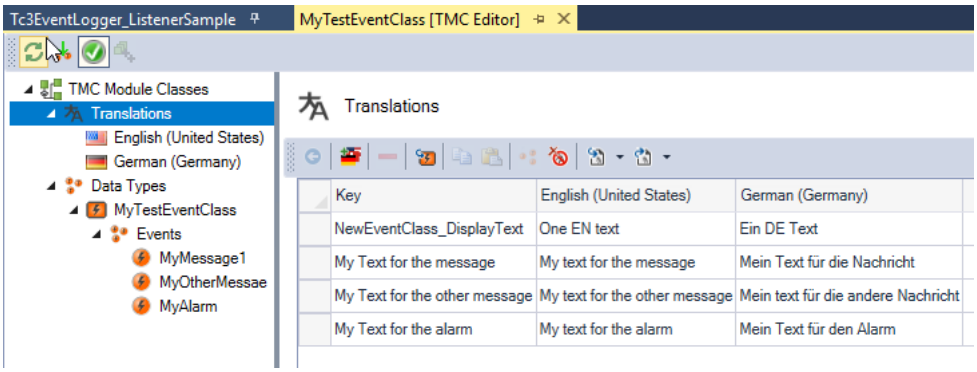
Zu berücksichtigende Sprachen

Sprachen können ausgewählt und hinzugefügt werden. Sollte zur Laufzeit eine Sprache angefragt werden, für die kein Text hinterlegt ist, wird der englische Text genutzt.

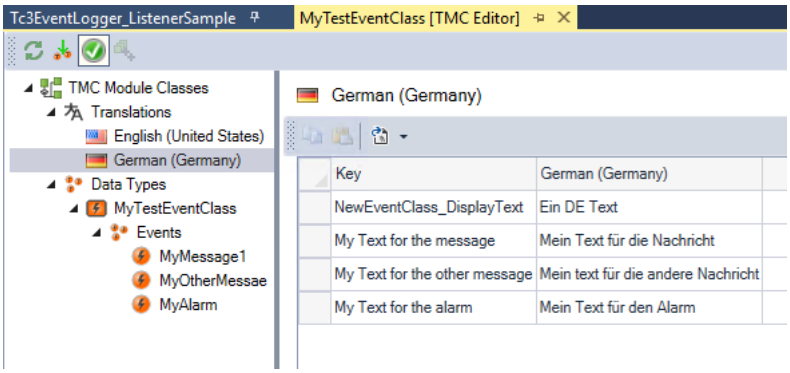


Übersetzungen

Die Übersetzungen können in der Tabelle unterhalb von „Translations“ gesetzt werden.

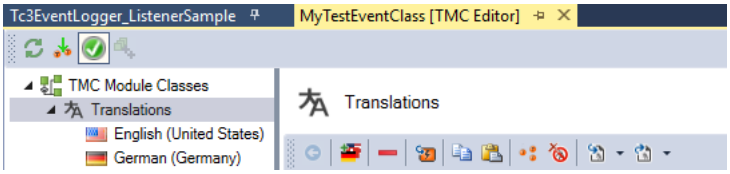


Alternativ können die Sprachen auf den jeweiligen Unterknoten separat bearbeitet werden.



Zusätzliche Funktionen

Es stehen weitergehende Funktionen für die Internationalisierung bereit:



Die folgenden Funktionen stehen bereit:

- Add a language: Fügt eine Sprache hinzu, wie oben beschrieben.
- Remove selected translations: Entfernt einen Schlüssel mit den Übersetzungen.
- Import Translation Keys: Importiert die genutzten Schlüssel aus den Eventklassen, wie oben beschrieben.
- Copy: Kopiert die Übersetzung.
- Paste: Fügt eine Übersetzung ein.
- Show Types: Zeigt die Verwendungen des ausgewählten Schlüssels an.
- Delete unused: Löscht die in keiner Eventclass genutzten Schlüssel aus der Tabelle.
- Import: Importiert die Translations aus einer XML oder CSV Datei.
- Export: Exportiert die Übersetzungen in eine XML oder CSV Datei.

Die Import- und Export-Funktionen beziehen sich auf ein XML / CSV Format, welches beispielhaft durch einen Export erkundet werden kann.

Übersetzungen außerhalb von TwinCAT (XML)

Die Übersetzungsinformationen sind in einem eigenen Bereich der Konfigurationsdateien hinterlegt, der auch außerhalb des TwinCAT XAE bearbeitet werden kann.

5.6 Zielsystem

Auf dem Zielsystem lässt sich der TwinCAT 3 EventLogger durch Registry-Einträge konfigurieren.

Unterhalb von *HKEY_LOCAL_MACHINE\SOFTWARE\[WOW6432Node\]Beckhoff\TwinCAT3* sind die folgenden Schlüssel nutzbar:

Zeitstempel		
\EventLogger\TimestampSource	DWORD	0 = CurPentiumTime 2 = CurSystemTime [default] Die CurPentiumTime stellt einen präzisen Zeitstempel individuell für jedes Event bereit. Dieses ist nur erforderlich, wenn innerhalb einer Task zeitliche Abstände wichtig sein. Es kann je nach Aufkommen der Tasks die benötigte Rechenzeit wesentlich erhöhen.
\EventLogger\TimestampBase	DWORD	0 = SystemTime [default] 1 = ExternalTimeHard 2 = ExternalTimeMedium 3 = ExternalTimeSoft Für 1 bis 3 beachten Sie die Dokumentation Korrigierte Zeitstempel.
Maximale Größe des Nachrichten-Caches		
\EventLogger\MaxDatabaseSize	DWORD	20 [default] in MB Bei Erreichen der Grenze wird die Hälfte der Nachrichten verworfen.
Ort des Nachrichten-Caches		
\EventLogger\DatabaseDir (Ab > TwinCAT 3.1 Build 4024.22)	STRING	:memory: = Arbeitsspeicher (Nachrichten werden mit dem Übergang nach CONFIG und auch Reboot etc. gelöscht.) <Pfad> = Dateisystem-Ordner für die Ablage der Datei LoggedEvents.db Wenn der Schlüssel nicht vorhanden ist, wird die Datenbank in dem Boot-Ordner abgelegt. Unter Windows CE wird dieses genutzt, um die Datenbank nicht-persistent abzulegen.
Nachrichten Speichern in das Windows Error Log		
\EventLogger\WindowsEventLog\TypesSupported	DWORD	0 = None [default] | 0x1 = Messages | 0x2 = Alarms | 0x3 = both
\EventLogger\WindowsEventLog\LogLocaleId	DWORD	1033 [default] 0 = current locale
\EventLogger\WindowsEventLog\MinLogLevel	DWORD	0 = Verbose [default] 1 = Info 2 = Warning 3 = Error
TwinCAT Systemfehler Loggen **Siehe auch** https://infosys.beckhoff.com/content/1031/tceventlogger/12332566027.html		

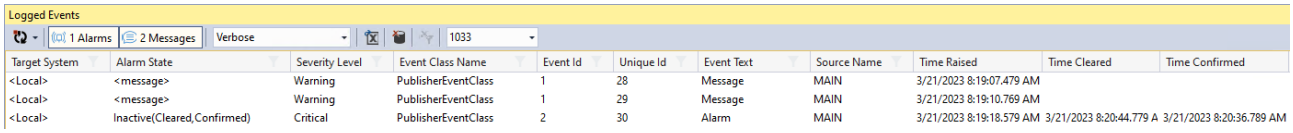
HKEY_LOCAL_MACHINE\SOFTWARE\[WOW6432Node\]Beckhoff\TwinCAT3\System[LogMessageType]	DWORD	3 = Übernahme der Systemfehler in den Tc3 Eventlogger 4 = Übernahme der Systemfehler in den Tc3 Eventlogger und die Error List des XAE.

- Sollten die Schlüssel nicht existieren, müssen diese mit dem angegebenen Typen angelegt werden.
- Nach ändern der RegKeys muss das Gerät neu gebootet werden, damit die Änderungen übernommen werden.

5.7 Engineering

Fenster TwinCAT Logged Events

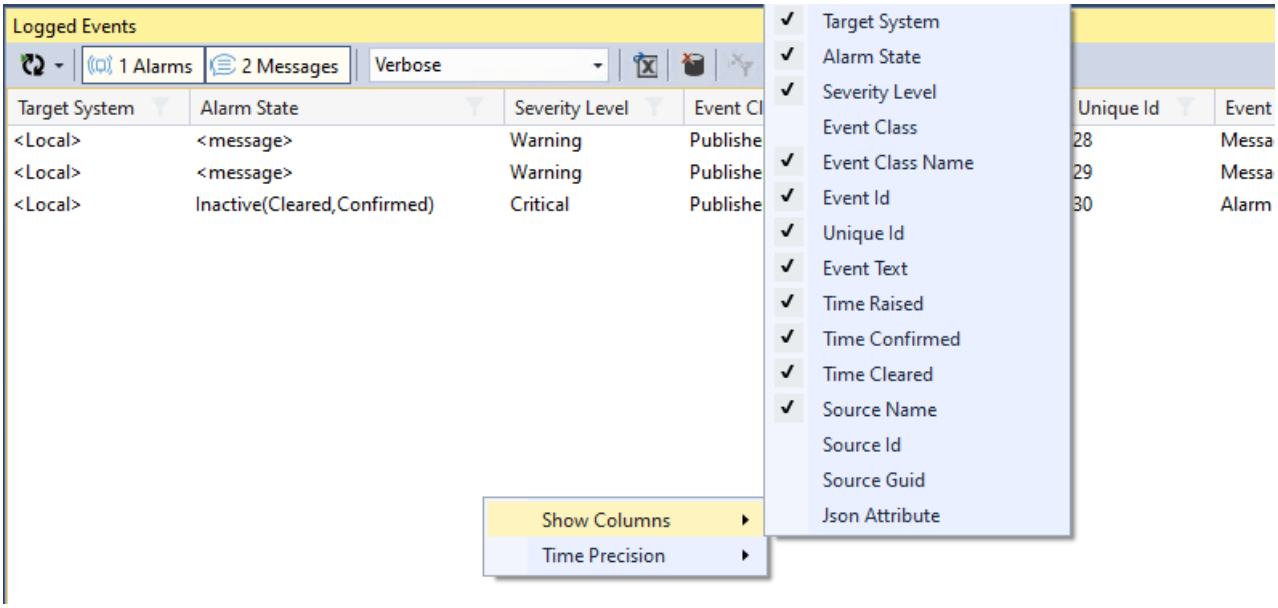
Über das Fenster **Logged Fenster** können die Ereignisse des Zielsystems aus der zuvor beschriebenen Cache-Datenbank geladen und angezeigt werden. Sie öffnen das Fenster im TwinCAT 3 Engineering (XAE) über **View > Other Windows > TwinCAT Logged Events**.



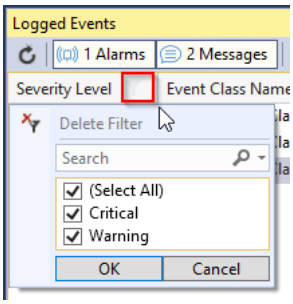
Die Symbolleiste des Fensters bietet die folgenden Funktionen:

	Lädt die Ereignisse vom ausgewählten Zielsystem. Über das Drop-down-Menü können zwei Modi aktiviert werden: • Start/Stop Update: Die Ereignisse werden kontinuierlich aktualisiert. • Read LoggedEvents: Ereignisse werden einmalig gelesen.
	Über die Schaltflächen **Alarms** und **Messages** kann konfiguriert werden, ob Alarme bzw. Nachrichten angezeigt werden sollen.
	Über das Drop-down-Menü kann die „Severity“ ausgewählt werden, ab der die Ereignisse angezeigt werden sollen.
	Stellt einen Export der Daten in ein CVS-Format bereit. Dabei werden die Informationen, die aktuell angezeigt werden, exportiert.
	Löscht (nach Rückfrage) die Cache-Datenbank auf dem Zielsystem.
	Über das Drop-down-Menü kann die Sprache ausgewählt bzw. eingegeben werden.

Die Spalten im Fenster sowie die Zeitauflösung können über die Befehle des Kontextmenüs konfiguriert werden:

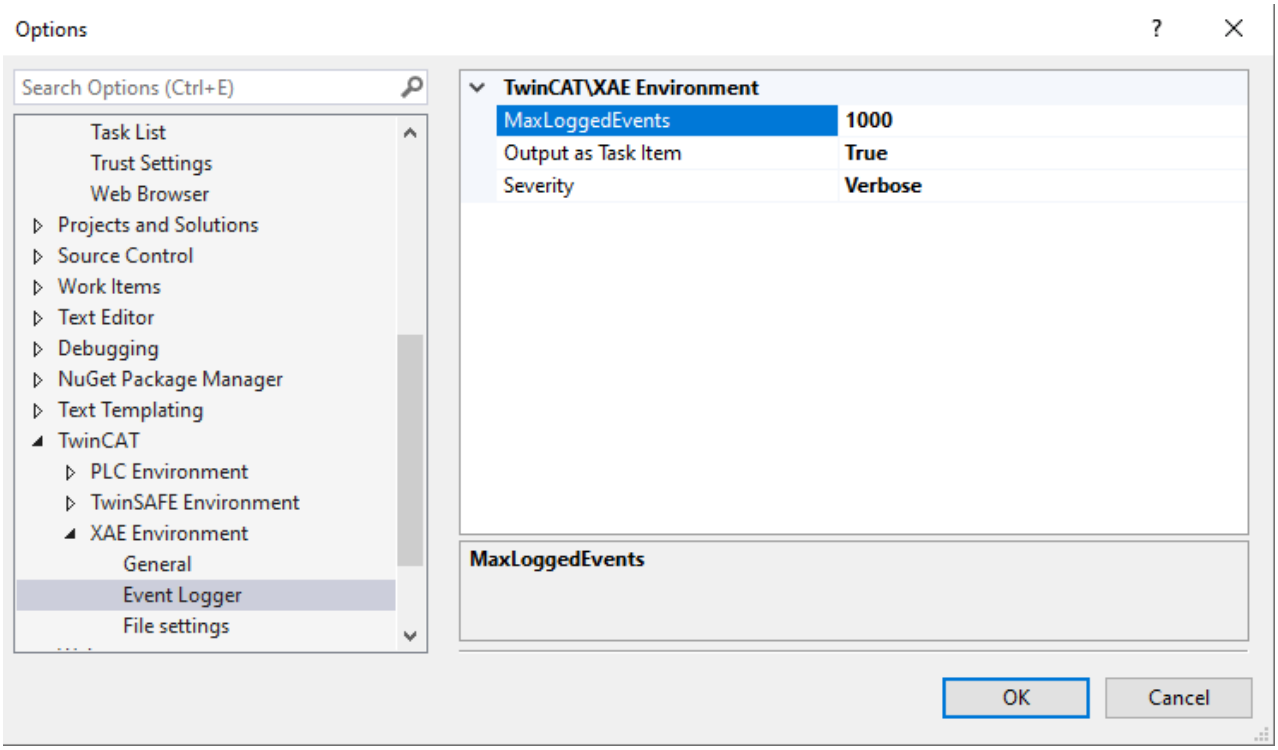


- Die Spalte „Alarm State“ stellt den aktuellen Status eines Alarms dar.
- Über das Menü „Time Precision“ (ms, µs, 100ns) können unterschiedliche Zeitdarstellungen für die Zeitstempel gewählt werden. Dieses betrifft nur die Darstellung im Engineering. Auf dem Zielsystem kann der verwendete Zeitstempel eingestellt werden, wie hier [▶ 26] beschrieben.
- Über die Filterfunktion können Einträge selektiert werden:



TwinCAT-Optionen

Die TwinCAT-Engineering-Einstellungen in den TwinCAT-Optionen (**Tools > Options**) stellen grundsätzliche Einstellungen für den TwinCAT 3 Eventlogger zur Anzeige im Engineering bereit.



MaxLoggedEvents	Anzahl der im Fenster **TwinCAT Logged Events** angezeigten maximalen Nachrichten.
Output as Task Item	Ausgabe der Events in dem Fenster **Error List**. Die Ausgabe erfolgt synchron, sobald das Engineering mit einem Zielsystem verbunden ist. Die Anzeige ist jedoch nicht für eine Vielzahl von Nachrichten ausgelegt, sodass diese Option auch genutzt werden kann, um die Ausgabe ggf. abzuschalten.
Severity	Diese Option kann verwendet werden, um die Ausgabe in der **Error List** einzuschränken.

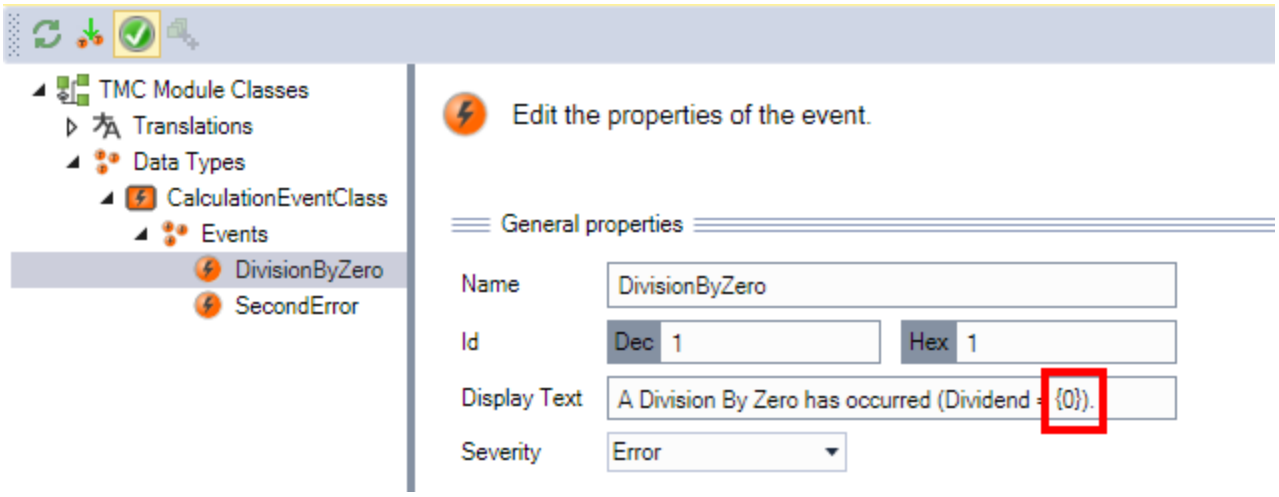
5.8 Argumente

Die Texte der Ereignisse können durch die Programmierung mit „Argumenten“ individualisiert werden.

Hierfür wird während der Beschreibung der Ereignisse im TMC-Editor eine Markierung mit der Notation `{n}` verwendet, wobei `n` eine aufsteigende Zahl von 0 ausgehend ist.

Dabei können bis zu 128 Argumente mit einer maximalen Größe von 1024 Bytes an einem Event verwendet werden.

Im TMC-Editor wird also beispielsweise ein solcher Display-Text für ein Ereignis verwendet:



Im Quellcode kann dieser dann verwendet werden, wie im Folgenden erklärt.

SPS

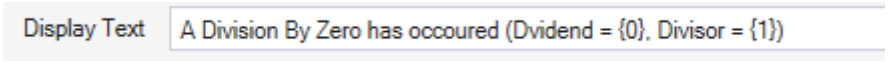
In der SPS kann folgenderweise mit den Argumenten umgegangen werden:

```
fbMsg : FB_TcMessage;
```

```
IF NOT fbMsg.EqualsToEventEntryEx(stOther:=TC_EVENTS.CalculationEventClass.DivisionByZero) THEN
    hr := fbMsg.CreateEx(TC_EVENTS.CalculationEventClass.DivisionByZero, 0 (*fbSource*) );
END_IF

fbMsg.ipArguments.Clear().AddLReal(fDividend); //set Argument
```

Die Argumente müssen dabei nach Create()/CreateEx(), aber vor Send() definiert werden.

Mehrere Argumente können verkettet angegeben werden.




```
fbMsg.ipArguments.Clear().AddLReal(fDividend).AddLReal(fDivisor);
```

Hierbei wird dann `fDividend` an die Stelle von `{0}` gesetzt, sowie `fDivisor` an die Stelle von `{1}`.

C++

In der C++ kann folgenderweise mit den Argumenten umgegangen werden:

```
TcArgs tcArgs(m_spMessage);
tcArgs->Clear();
tcArgs.AddArgument(m_dividend);
```

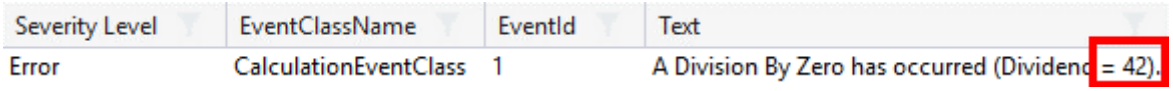
Die Argumente müssen dabei nach CreateMessage()/CreateAlarm(), aber vor Send() definiert werden.

Hierfür muss `TcEventLoggerTemplate.h` im `<ProjectName>Interfaces.h` inkludiert warden.

```
#include"TcRouterInterfaces.h"
#include"TcEventLoggerTemplates.h"
///<AutoGeneratedContent id="Interfaces">
///</AutoGeneratedContent>
```

Ausgabe

Die Ausgabe ist entsprechend:

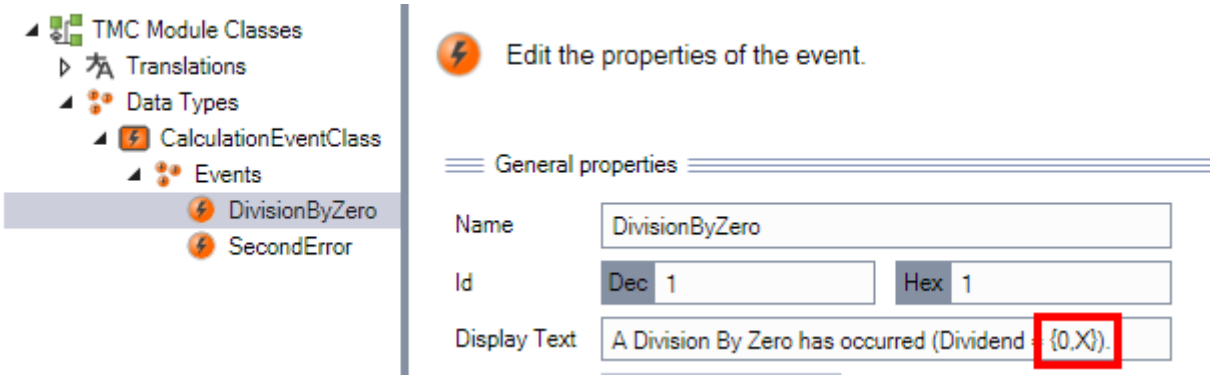


Diese Notation kann auch als Text innerhalb der Übersetzungen verwendet werden.

Werden mehr Argumente abgesendet als durch entsprechende Platzhalter vorgesehen sind, werden diese in Klammern an den Ereignistext angehangen. Dieses Verhalten kann unterdrückt werden indem für die Ereignisklasse die Option „HideNeedlessArguments“ gesetzt wird, wie hier [▶ 16] beschrieben.

Formatierung

Die Ausgabe der Argumente kann auch formatiert werden. Hierfür wird im TMC-Editor entsprechend die Syntax `{n,<Format>}` verwendet:



Folgende Formatierungen stehen bereit:

Typ	Format	Beschreibung
Zahlenwert	d / D	Dezimaldarstellung
	e / E	Exponentialdarstellung
	x / X	Hexadezimaldarstellung
	f / F	Fixed-Point

Beachten Sie, dass z. B. ein REAL nicht als „d“ oder „x“ usw. dargestellt werden kann.

Zusätzliche Hinweise bzgl. Argumente in Meldungstexten

- Zusätzlich stehen die Syntax `{eventID}` (für die Event ID) sowie `{eventclass}` (für die Guid der Ereignisklasse) bereit, um die entsprechenden Informationen als Teil des Textes auszugeben.
- Werden Strings als Argumente `AddString()` verwendet, werden diese als Argument in den Meldungstext eingefügt. Dieser String kann vom TwinCAT 3 Eventlogger nicht übersetzt werden. Soll ein Argument übersetzt werden, kann dafür `.AddEventReferenceEx()` verwendet werden. Hierbei wird der Meldungstext in der entsprechenden Übersetzung eingefügt. Das Beispiel Listener [▶ 204] zeigt die Verwendung.

5.9 Umgang mit Quellen

Gleiche Ereignisse können an unterschiedlichen Stellen eines Programms auftreten. Die Quelle eines Ereignisses wird in der Programmierung durch die „Source Info“ beschrieben und beim Senden mit übertragen.

Die SourceInfo besteht dabei aus drei Teilen (siehe Ereignisse [▶ 13]).

In beiden Programmiersprachen wird die Quelle beim Anlegen des Ereignisses (Create) mit angegeben.

SPS

In der SPS wird hierfür der FB_TcSourceInfo verwendet.

```
VAR
    fbResult : FB_TcMessage;
    fbSource : FB_TcSourceInfo; // optional
```

Dieser wird entsprechend parametriert, bevor Create()/CreateEx() aufgerufen wird:

```
//adapt source name if required (default is ads symbol name)
fbSource.Clear();
fbSource.sName := 'Math Calculation';
fbSource.nId := 12;

IF NOT fbResult.EqualsToEventEntryEx(stOther:=TC_EVENTS.CalculationEventClass.DivisionByZero) THEN
    hr := fbResult.CreateEx(TC_EVENTS.CalculationEventClass.DivisionByZero, fbSource ); //This uses dyn ressources and shouldn't be called cyclically.
```

Alternativ kann beim Create()/CreateEx()-Aufruf dem entsprechenden Parameter eine Null zugewiesen werden, um die interne Standard-Quelleninformation der SPS zu nutzen. Wenn dann **keine** explizite SourceInfo angegeben wird, erfolgt die Ausgabe des Symbolpfades, wo das Event instanziiert wird, als SourceName und die Objekt-ID der SPS-Instanz als SourceID.

Logged Events

1 Alarms | 3 Messages | Info | 1031

Severity Level	EventClassName	EventId	Text	SourceName	SourceId
Error	CalculationEventClass	1	A Division By Zero has occurred (Dividend = 42)	MAIN.fbMath	0x08502000
Error	CalculationEventClass	1	A Division By Zero has occurred	MAIN.fbMath	0x08502000

C++

In C++ wird das TcSourceInfo verwendet, welches beispielsweise auf folgende Art bei CreateMessage()/ CreateAlarm() übergeben werden kann:

```
m_spEventLogger->CreateMessage(TcEvents::MyCppEventClass::MyInit.uuidEventClass,
    TcEvents::MyCppEventClass::MyInit.nEventId,
    TcEvents::MyCppEventClass::MyInit.eSeverity,
    &TcSourceInfo("Math Calculation"),
    &m_spMessageInit);
```

Ausgabe

Diese SourceInfo kann entsprechend im LoggedEvents-Fenster eingeblendet werden:

Logged Events

0 Alarms | 1 Messages | Info | 1031

Severity Level	EventClassName	EventId	Text	SourceName	SourceId
Error	CalculationEventClass	1	A Division By Zero has occurred (Dividend = 42).	Math Calculation	12

5.10 Internationalisierung von Quellen

TwinCAT 3.1 Build 4026 benötigt

Die hier beschriebene Funktionalität setzt die Version TwinCAT 3.1 4026.0 voraus.

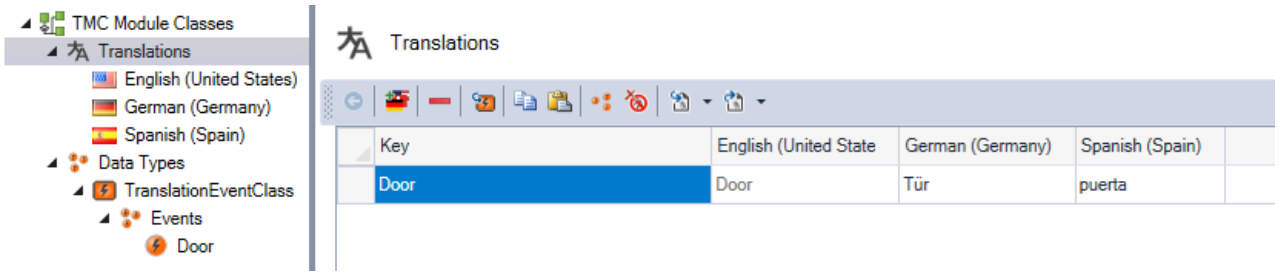
Die **SourceInfo** eines Ereignisses soll den Ort beschreiben, zu dem ein Ereignis gehört. Der **SourceName** kann verwendet werden, um eine Quelle für einen Menschen zu beschreiben. Hieraus ergibt sich, dass der **SourceName** in einigen Fällen auch übersetzt werden soll.

Folgen Sie den folgenden Arbeitsschritten, um eine Internationalisierung des **SourceName** vorzunehmen:

1. Erstellen Sie eine Ereignisklasse, die zum Tragen der Übersetzungsinformationen verwendet wird. Der Name ist beliebig, beispielsweise „TranslationEventClass“.
2. Legen Sie für jeden zu übersetzenden **SourceName** ein Event in der Ereignisklasse an und schreiben Sie den zu übersetzenden **SourceName** als Displaytext in dieses Event.
3. Legen Sie eine Übersetzungstabelle für diese Events mit den unterschiedlichen Sprachen an.

4. Um auf die Übersetzungen zuzugreifen, müssen Sie die erstellte Ereignisklasse als **SourceGuid** an dem jeweiligen Event setzen. Hierdurch wird die Referenz bereitgestellt, so dass die Texte zum einen auf das Zielsystem übertragen werden, sowie für die Übersetzung verwendet werden.
5. Aktivieren Sie die **Solution**.

⇨ Die Übersetzungen stehen auf dem Zielsystem bereit.



Die Übersetzungen sind über unterschiedliche Wege zugänglich:

- TwinCAT/HMI: Bietet in dem Event Grid Control den übersetzten **SourceName** an.
- Das Fenster **LoggedEvents** vom Engineering zeigt die Übersetzung automatisch an, wenn eine Sprache ausgewählt wurde.
- SPS: Über den Baustein FB_RequestTranslation [▶ 55] kann die Übersetzung zu einem **SourceName** abgefragt werden.

5.11 JSON-Attribute

Wie im Einführungsteil des Abschnitts Technische Einführung [▶ 13] beschrieben, gibt es die Möglichkeit ein zusätzliches JSON-Attribut mit einer Nachricht zu übertragen.

Sowohl beim Erzeugen als auch beim Empfangen kann die JsonXml-Bibliothek (SPS-Bibliothek Tc3_JsonXml) verwendet werden, um das JSON zu erzeugen.

SPS

Vor dem Send(), aber nach dem Create()/CreateEx() kann das JSON-Attribut angegeben werden:

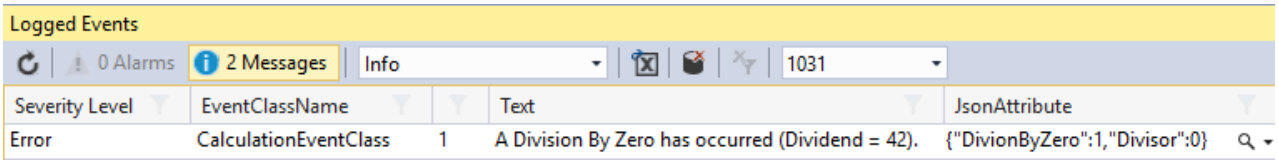
```
    //fbSource.Clear();
    //fbSource.sName := 'Math Calculation';
    IF NOT fbResult.EqualsToEventEntryEx(stOther:=TC_EVENTS.CalculationEventClass.DivisionByZero) THEN
        hr := fbResult.CreateEx(TC_EVENTS.CalculationEventClass.DivisionByZero, 0 ); //This uses dyn ressources and shouldn't be called cyclically.
        IF FAILED(hr) THEN
            hrLastInternalError := hr;
        END_IF

        //set Arguments if required
        //fbResult.ipArguments.Clear();

        fbResult.SetJsonAttribute('{"DivionByZero":1,"Divisor":0}');

        IF fbResult.eSeverity >= eTraceLevel THEN
            hr := fbResult.Send(0);
            IF FAILED(hr) THEN
                hrLastInternalError := hr;
```

C++

Vor dem Send(), aber nach dem CreateMessage()/CreateAlarm() kann das JSON-Attribut angegeben werden:

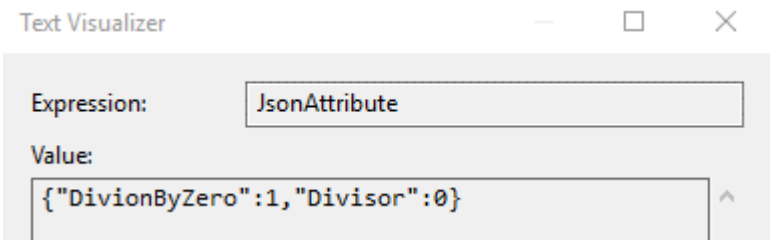
```
237
238         m_spMessage->SetJsonAttribute(m_pjsonAttribute);
239         hr = m_spMessage->Send(0);
240
```

Ausgabe



Das Logged-Events-Fenster stellt zwei Visualisierungen für die JSON-Attribute bereit, die über das Drop-down-Menü innerhalb der Informationsspalte ausgewählt und durch einen Klick auf die Lupe geöffnet werden können:

Text Visualizer



JSON Visualizer

5.12 Filter zur Abfrage

Ab TwinCAT 3.1 Build 4024.17

Die hier beschriebenen Filter stehen ab der Version TwinCAT 3.1 Build 4024.17 zur Verfügung.

Beim Verarbeiten von Nachrichten wie beispielsweise dem Empfangen stellt sich die Frage, welche Nachrichten an der entsprechenden Stelle beachtet werden sollen.

Zur Formulierung der gewünschten Nachrichten wird eine API bereitgestellt. Für alle eintreffenden Nachrichten wird durch die API beschrieben, welche von Relevanz sind, sodass sich ein Filter ergibt.

Diese API steht an unterschiedlichen Stellen zur Nutzung zur Verfügung:

- Empfangen von Nachrichten in der Echtzeit über die Listener-Schnittstelle.
- Empfangen von Nachrichten, welche auf Basis von EtherCAT-Emergency-Nachrichten des IO-Systems auftreten.
- Löschen von Nachrichten aus dem Cache.
- Exportieren von Nachrichten in eine Datei („CSV Export“).

Die konkrete Verwendung der Filter wird in den Samples Beispiel Filter [▶ 205] und Beispiel Listener [▶ 204] gezeigt.

Der Funktionsbaustein FB_TcEventFilter [▶ 69] ist in Bezug auf die Verwendung der Einstiegspunkt.

5.12.1 Rückgabewerte

Bei der Verarbeitung der Filter wird bei der jeweiligen Anwendung die Korrektheit überprüft und ggf. durch entsprechende Rückgabewerte der Fehler angezeigt.

Diese sind hier dokumentiert:

ADS_E_NOTINIT	0x98117018	ADS Verbindung nicht initialisiert
E_POINTER	0x80004003L	Ungültiger Pointer
E_NOTIMPL	0x80004001L	Funktion ist nicht implementiert.
E_OUTOFMEMORY	0x8007000EL	Nicht genügend Speicher
ADS_E_INVALIDPARM	0x9811700B	Validierungsfehler
ADS_E_NOMEMORY	0x9811700A	Nicht genügend Speicher
ADS_E_INVALIDSTATE	0x98117012	Das System ist in einem Zustand, in dem es die Filter nicht verarbeiten kann. Beispielsweise ist ExportLoggedEvents nicht im OP-Zustand.
ADS_E_INVALIDDATA	0x98117006	AddJsonAttributeExpression path ist ungültig.

5.13 Remote Zugriff auf den Eventlogger mittels PLC

Auf den Eventlogger eines Systems kann von einem anderen, entfernten System zugegriffen werden.

Dieses bietet die unter „Usermode API“ beschriebene Schnittstelle an. Hierbei erstellen Nutzer eigene (Usermode-)Programme, die über eine ADS-Route (und damit ggf. auch über Netzwerk) mit dem Eventlogger eines anderen Systems kommunizieren.

Verfügbar mit TwinCAT 3.1 4026

Die folgende Funktionalität steht ab TwinCAT 3.1 Build 4026 bereit.

Die SPS bietet zusätzlich die Möglichkeit des Zugriffs über Netzwerk mittels der Funktionsbausteine in dem Ordner „RemoteEventlogger“ der SPS-Bibliothek. Die bereitstehenden Funktionen werden hier beschrieben.

Verbindungsaufbau

Zuerst wird eine Verbindung zum entfernten System aufgebaut:

```
VAR
  sNetIdTarget : T_AmsNetID; //the AmsNetId of the remote system
  fbRemoteLogger : FB_TcRemoteEventLogger;
END_VAR
hr := fbRemoteLogger.Connect(sNetId:=sNetIdTarget);
```

Dieses kann zyklisch erfolgen, was bei einem Verbindungsabbruch den Wiederaufbau initiiert.

Die Verbindung kann wieder beendet werden:

```
VAR
  fbRemoteLogger : FB_TcRemoteEventLogger;
END_VAR
hr := fbRemoteLogger.Disconnect();
```

Der Zustand kann überwacht werden:

```
hr := fbRemoteLOgger.hrConnect;
```

Um Nachrichten von einem entfernten System zu empfangen, muss ein Funktionsbaustein implementiert werden, der `FB_RemoteListenerBase` erweitert:

```
FUNCTION_BLOCK FB_RemoteListener EXTENDS FB_RemoteListenerBase
```

Neben den aus dem `FB_ListenerBase2` bekannten Callback-Methoden `OnMessageSent` usw. erhält dieser Funktionsbaustein folgende Methoden, die speziell für die Remotefunktionalität zuständig sind:

- `OnEventLoggerError`(): Wird aufgerufen, wenn der entfernte Eventlogger einen Fehlercode als HRESULT liefert.
- `OnConnectionStateChanged`(): Diese Methode wird aufgerufen, wenn sich der Verbindungsstatus ändert. Er stellt die jeweilige Änderung als Enum `TcRemoteConnectionChangeReason` bereit.

Mithilfe eines solchen Bausteins können die Nachrichten empfangen werden, indem eine Anmeldung stattfindet:

```
hr := fbListener.Subscribe(fbRemoteLogger, 0);
```

Entsprechend gibt es auch eine Möglichkeit diese Anmeldung rückgängig zu machen:

```
hrUnsubscribe:= fbListener.Unsubscribe();
```

Der Funktionsbaustein FB_TcRemoteEventLogger stellt ansonsten äquivalente Funktionen zu der lokalen Variante FB_TcEventLogger bereit. Überwachen Sie bei der Nutzung den Verbindungszustand.

5.14 Importieren von Resource Dateien

Resource Dateien (Endung ECPX) liegen typischerweise im Verzeichnis *3.1\Target\Resource*. Der TwinCAT 2 Eventlogger hat diese Dateien über ein Konfigurations-Programm erstellt.
Die Dateien können im neuen Eventlogger über den folgenden Arbeitsablauf verwendet werden:

✓ Das Excel Add-in [▶ 21] wurde installiert.

1. Wähen Sie in Excel über den Menüeintrag **TcEventLogger** > **Import** eine Resource Datei aus.
   ⇨ Es wird pro `SourceId` eine Ereignisklasse angelegt.
   ⇨ Beim Importieren werden einige Abbildungen vorgenommen.
2. Überprüfen Sie diese in der Anwendung inhaltlich und berücksichtigen Sie sie. Die Abbildungen sind unten beschrieben.
3. Erzeugen Sie über **Generate TMC-File** eine TMC-Datei.
   ⇨ Die TMC-Datei wird wie hier [▶ 21] beschrieben in der TwinCAT-Solution eingebunden.
4. Zusätzlich können Sie über Create PLC mapping function eine TcPOU-Datei erzeugen, welche ein Mapping von SourceId/EventId zu dem entsprechenden neuen Event bereitstellt.

⇨ Die weitere Entwicklung kann im Folgenden in Excel oder nach einmaligem Import auch im TMC-Editor erfolgen (dann aber nicht mehr in Excel)

Bei dem Import werden die folgenden Abbildungen vorgenommen:

- Alle Events über Ressourcen haben einen Status `bSetEvent` und werden deswegen als Alarm vorgesehen.
- `SourceId` wird als Ereignisklasse und die `EventId` als Ereignis-ID verwendet und entsprechende Ereignisklassen werden angelegt.
- Die Information `ReqMustCon` wird als Quittierung mittels `clear` importiert und behandelt.
- Die Informationen `nClass` werden auf die `eSeverity` des TwinCAT 3 Eventlogger abgebildet:

New EventLogger eSeverity		Old EventLogger nClass	
0	Verbose	0	None
0	Verbose	1	Maintenance
0	Verbose	2	Message
0	Verbose	3	Hint
1	Info	4	Stateinfo
1	Info	5	Instruction
2	Warning	6	Warning
3	Error	7	Alarm
4	Critical	8	Parameter Error

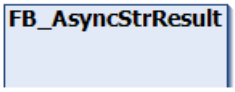
Das Beispiel Resource Import [▶ 206] stellt die nötigen Implementierungen beispielhaft bereit.

6 SPS API

6.1 Funktionen und Funktionsbausteine

6.1.1 Asynchrone Textanfragen

6.1.1.1 FB_AsyncStrResult



Dieser Funktionsbaustein ermöglicht die asynchrone Anfrage eines Textes.

Methoden

Name	Beschreibung
GetString [▶ 39]	Sobald bBusy FALSE ist und wenn kein Fehler aufgetreten ist (bError = FALSE), kann mittels dieser Methode der angefragte Text abgeholt werden.

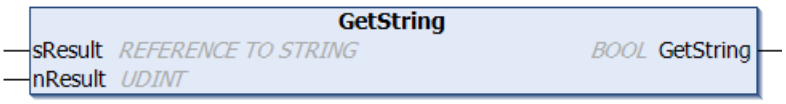
Eigenschaften

Name	Typ	Zugriff	Beschreibung
bBusy	BOOL	Get	TRUE, solange die Abarbeitung noch nicht abgeschlossen ist.
bError	BOOL	Get	TRUE, sobald ein Fehler eintritt.
hrErrorCode	HRESULT	Get	Gibt die Fehlerinformation aus, wenn bError TRUE ist.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.1.4022.20	PC oder CX (x64, x86, Arm®)	Tc3_EventLogger

6.1.1.1.1 GetString



Sobald bBusy = FALSE ist und wenn kein Fehler aufgetreten ist (bError = FALSE), kann mittels dieser Methode der angefragte Text abgeholt werden.

Syntax

```
METHOD GetString : BOOL
VAR_INPUT
    sResult : REFERENCE TO STRING;
    nResult : UDINT;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
sResult	REFERENCE TO STRING	Puffervariable für den angefragten Text
nResult	UDINT	Puffergröße in Bytes

Rückgabewert

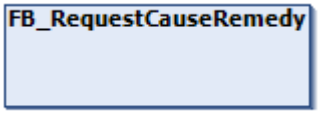
Name	Typ	Beschreibung
GetString	BOOL	Liefert TRUE, wenn der Text zugewiesen werden konnte. Liefert FALSE, wenn der Text nicht vollständig zugewiesen werden konnte, weil die angegebene Puffervariable zu klein ist.

Beispiel

Die Methode darf erst dann aufgerufen werden, wenn mittels bBusy = FALSE und bError = FALSE signalisiert wurde, dass ein Text zur Verfügung steht.

```
IF NOT fb.bBusy AND NOT fb.bError THEN
    bGetStringSuccess := fb.GetString(sText, SIZEOF(sText));
END_IF
```

6.1.1.2 FB_RequestCauseRemedy



Dieser Funktionsbaustein ermöglicht die asynchrone Anfrage von Ursache/Behebung in gewünschter Sprache zu einem Ereignis.

Methoden

Name	Beschreibung
Clear [▶ 40]	Diese Methode löscht das zuletzt abgefragte Ergebnis.
Get [▶ 41]	Sobald bBusy FALSE ist und wenn kein Fehler aufgetreten ist (bError = FALSE), kann mittels dieser Methode ein Ergebnis aus der abgefragten Liste abgeholt werden.
Request [▶ 41]	Der Aufruf dieser Methode triggert die asynchrone Anfrage.
RequestRemote [▶ 42]	Der Aufruf dieser Methode triggert die asynchrone Anfrage auf einem entfernten Zielsystem.

Eigenschaften

Name	Typ	Zugriff	Beschreibung
bBusy	BOOL	GET	TRUE, solange die Abarbeitung noch nicht abgeschlossen ist.
bError	BOOL	GET	TRUE, sobald ein Fehler eintritt.
hrErrorCode	HRESULT	GET	Gibt die Fehlerinformation aus, wenn bError TRUE ist.
nCount	UDINT	GET	Anzahl der gefundenen Ursachen/behobenen Elemente

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.1.4026.0	PC oder CX (x64, x86, Arm®)	Tc3_EventLogger (>=3.3.7.0)

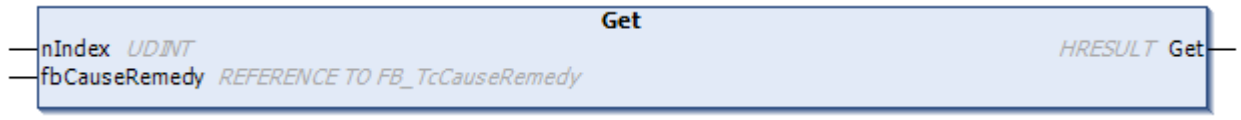
6.1.1.2.1 Clear

Diese Methode löscht das zuletzt abgefragte Ergebnis.

Syntax

```
METHOD Clear
```

6.1.1.2.2 Get



Sobald bBusy FALSE ist und wenn kein Fehler aufgetreten ist (bError = FALSE), kann mittels dieser Methode ein Ergebnis aus der abgefragten Liste abgeholt werden.

Syntax

```
METHOD Get         : HRESULT
VAR_INPUT
    nIndex         : UDINT;
    fbCauseRemedy : REFERENCE TO FB_TcCauseRemedy;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
nIndex	UDINT	Index in der Liste von Ursache/Behebung-Elementen, der abzufragen ist.
fbCauseRemedy	Reference to FB_TcCauseRemedy [▶ 58]	Referenz auf einen FB_TcCauseRemedy, in dem die Texte abgelegt werden.

Ausgänge

Name	Typ	Beschreibung
Get	HRESULT	S_OK bei Erfolg ADS_E_INVALIDSTATE, wenn die Abfrage noch läuft. ADS_E_INVALIDSIZE, wenn ein Index abgefragt wird, der nicht belegt ist.

6.1.1.2.3 Request

Der Aufruf dieser Methode triggert die asynchrone Anfrage.

Syntax

```
METHOD Request : HRESULT
VAR_INPUT
    eventClass : GUID;
    nEventId   : UDINT;
    nLangId    : DINT; // English(US)=1033 ; German(Germay)=1031
END_VAR
```

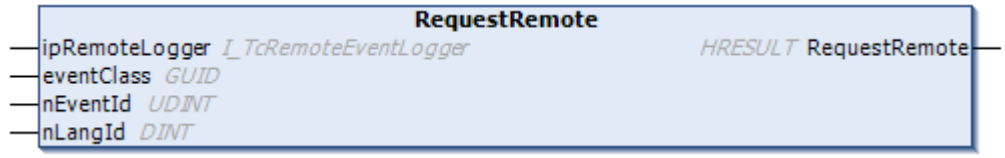
Eingänge

Name	Typ	Beschreibung
eventClass	GUID	Spezifiziert die Ereignisklasse.
nEventId	UDINT	ID von dem Ereignis
nLangId	DINT	Spezifiziert die Sprach-ID: Englisch (en-US) = 1033 Deutsch (de-DE) = 1031 … Der Enum TcLcid kann verwendet werden.

Ausgänge

Name	Typ	Beschreibung
Request	HRESULT	Liefert mögliche Fehlerinformationen. S_OK, wenn die Abfrage erfolgreich ist. ADS_E_INVALIDSTATE, wenn ein illegaler Zustand des Zielsystems vorliegt. ADS_E_NOMEMORY, wenn kein Speicher bereitsteht, um das Ergebnis abzulegen. ADS_E_PENDING, wenn die Abfrage noch bearbeitet wird.

6.1.1.2.4 RequestRemote



Der Aufruf dieser Methode triggert die asynchrone Anfrage auf einem entfernten Zielsystem.

Syntax

```
METHOD RequestRemote : HRESULT
VAR_INPUT
    ipRemoteLogger   : I_TcRemoteEventLogger;
    eventClass       : GUID;
    nEventId         : UDINT;
    nLangId          : DINT; // English(US)=1033 ; German(Germay)=1031
END_VAR
```

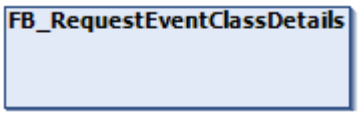
Eingänge

Name	Typ	Beschreibung
ipRemoteLogger	I_TcRemoteEventLogger	Instanz vom I_TcRemoteEventLogger, der das Zielsystem beschreibt.
eventClass	GUID	Spezifiziert die Ereignisklasse.
nEventId	UDINT	ID von dem Ereignis
nLangId	DINT	Spezifiziert die Sprach-ID: Englisch (en-US) = 1033 Deutsch (de-DE) = 1031 … Der Enum TcLcid kann verwendet werden.

Ausgänge

Name	Typ	Beschreibung
RequestRemote	HRESULT	Liefert mögliche Fehlerinformationen. S_OK, wenn die Abfrage erfolgreich ist. ADS_E_INVALIDSTATE, wenn ein illegaler Zustand des Zielsystems vorliegt. ADS_E_NOMEMORY, wenn kein Speicher bereitsteht, um das Ergebnis abzulegen. ADS_E_PENDING, wenn die Abfrage noch bearbeitet wird.

6.1.1.3 FB_RequestEventClassDetails



Mit diesem Funktionsbaustein können die Details einer Ereignisklasse in Form von FB_TcDetail [▶ 60] abgefragt werden.

Jedes „Detail“ (DescriptionText, Descriptionurl, Comment) – vgl. beim Anlegen des Events) hat dabei einen Index und zu jedem Index wird der Key ((DescriptionText, DescriptionUrl, Comment) sowie der Wert geliefert.

Methoden

Name	Beschreibung
Clear [▶ 43]	Diese Methode löscht das zuletzt abgefragte Ergebnis.
Get [▶ 44]	Sobald bBusy FALSE ist und wenn kein Fehler aufgetreten ist (bError = FALSE), kann mittels dieser Methode ein Ergebnis aus der abgefragten Liste abgeholt werden.
Request [▶ 44]	Der Aufruf dieser Methode triggert die asynchrone Anfrage.
RequestRemote [▶ 45]	Der Aufruf dieser Methode triggert die asynchrone Anfrage auf einem entfernten Zielsystem.

Eigenschaften

Name	Typ	Zugriff	Beschreibung
bBusy	BOOL	GET	TRUE, solange die Abarbeitung noch nicht abgeschlossen ist.
bError	BOOL	GET	TRUE, sobald ein Fehler eintritt.
hrErrorCode	HRESULT	GET	Gibt die Fehlerinformation aus, wenn bError TRUE ist.
nCount	UDINT	GET	Anzahl der gefundenen Ursachen/behobenen Elemente

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.1.4026.0	PC oder CX (x64, x86, Arm®)	Tc3_EventLogger (>=3.3.7.0)

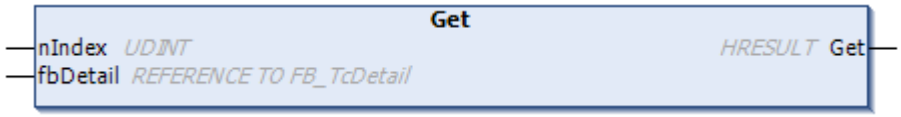
6.1.1.3.1 Clear

Diese Methode löscht das zuletzt abgefragte Ergebnis.

Syntax

```
METHOD Clear
```

6.1.1.3.2 Get



Sobald bBusy FALSE ist und wenn kein Fehler aufgetreten ist (bError = FALSE), kann mittels dieser Methode ein Ergebnis aus der abgefragten Liste abgeholt werden.

Syntax

```
METHOD Get    : HRESULT
VAR_INPUT
    nIndex    : UDINT;
    fbDetail  : REFERENCE TO FB_TcDetail;
END_VAR
```

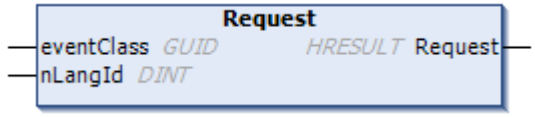
Eingänge

Name	Typ	Beschreibung
nIndex	UDINT	Index in der Liste von Ursache/Behebung-Elementen, der abzufragen ist.
fbDetail	Reference to FB_TcDetail [▶ 60]	Referenz auf einen FB_TcDetail, in dem die Details abgelegt werden.

Ausgänge

Name	Typ	Beschreibung
Get	HRESULT	Liefert S_OK bei Erfolg. ADS_E_INVALIDSTATE, wenn die Abfrage noch läuft. ADS_E_INVALIDSIZE, wenn ein Index abgefragt wird, der nicht belegt ist.

6.1.1.3.3 Request



Der Aufruf dieser Methode triggert die asynchrone Anfrage.

Syntax

```
METHOD Request : HRESULT
VAR_INPUT
    eventClass : GUID;
    nLangId    : DINT; // English(US)=1033 ; German(Germay)=1031
END_VAR
```

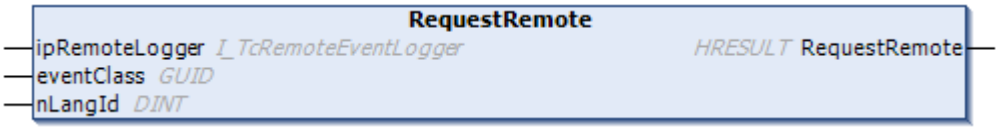
Eingänge

Name	Typ	Beschreibung
eventClass	GUID	Spezifiziert die Ereignisklasse.
nLangId	DINT	Spezifiziert die Sprach-ID: Englisch (en-US) = 1033 Deutsch (de-DE) = 1031 … Der Enum TcLcid kann verwendet werden.

Ausgänge

Name	Typ	Beschreibung
Request	HRESULT	Liefert mögliche Fehlerinformationen. S_OK, wenn die Abfrage erfolgreich ist. ADS_E_INVALIDSTATE, wenn ein illegaler Zustand des Zielsystems vorliegt. ADS_E_NOMEMORY, wenn kein Speicher bereitsteht, um das Ergebnis abzulegen. ADS_E_PENDING, wenn die Abfrage noch bearbeitet wird.

6.1.1.3.4 RequestRemote



Der Aufruf dieser Methode triggert die asynchrone Anfrage auf einem entfernten Zielsystem.

Syntax

```
METHOD RequestRemote : HRESULT
VAR_INPUT
    ipRemoteLogger   : I_TcRemoteEventLogger;
    eventClass       : GUID;
    nLangId          : DINT; // English(US)=1033 ; German(Germay)=1031
END_VAR
```

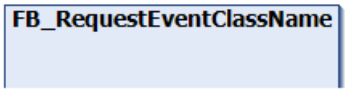
Eingänge

Name	Typ	Beschreibung
ipRemoteLogger	I_TcRemoteEventLogger	Instanz vom I_TcRemoteEventLogger, der das Zielsystem beschreibt.
eventClass	GUID	Spezifiziert die Ereignisklasse.
nLangId	DINT	Spezifiziert die Sprach-ID: Englisch (en-US) = 1033 Deutsch (de-DE) = 1031 … Der Enum TcLcid kann verwendet werden.

Ausgänge

Name	Typ	Beschreibung
RequestRemote	HRESULT	Liefert mögliche Fehlerinformationen. S_OK, wenn die Abfrage erfolgreich ist. ADS_E_INVALIDSTATE, wenn ein illegaler Zustand des Zielsystems vorliegt. ADS_E_NOMEMORY, wenn kein Speicher bereitsteht, um das Ergebnis abzulegen. ADS_E_PENDING, wenn die Abfrage noch bearbeitet wird.

6.1.1.4 FB_RequestEventClassName



Dieser Funktionsbaustein ermöglicht die asynchrone Anfrage des Namens einer Ereignisklasse.

Methoden

Name	Beschreibung
Clear [▶ 46]	Diese Methode löscht das zuletzt abgefragte Ergebnis.
GetString [▶ 47]	Sobald bBusy FALSE ist und wenn kein Fehler aufgetreten ist (bError = FALSE), kann mittels dieser Methode der angefragte Text abgeholt werden.
Request [▶ 47]	Der Aufruf dieser Methode triggert die asynchrone Textanfrage.
RequestRemote [▶ 48]	Der Aufruf dieser Methode triggert die asynchrone Anfrage auf einem entfernten Zielsystem.

Eigenschaften

Name	Typ	Zugriff	Beschreibung
bBusy	BOOL	Get	TRUE, solange die Abarbeitung noch nicht abgeschlossen ist.
bError	BOOL	Get	TRUE, sobald ein Fehler eintritt.
hrErrorCode	HRESULT	Get	Gibt die Fehlerinformation aus, wenn bError TRUE ist.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.1.4022.20	PC oder CX (x64, x86, Arm®)	Tc3_EventLogger

6.1.1.4.1 Clear

Diese Methode löscht das zuletzt abgefragte Ergebnis.

Syntax

```
METHOD Clear
```

6.1.1.4.2 GetString

Sobald bBusy = FALSE ist und wenn kein Fehler aufgetreten ist (bError = FALSE), kann mittels dieser Methode der angefragte Text abgeholt werden.

Syntax

```
METHOD GetString : BOOL
VAR_INPUT
    sResult : REFERENCE TO STRING;
    nResult : UDINT;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
sResult	REFERENCE TO STRING	Puffervariable für den angefragten Text
nResult	UDINT	Puffergröße in Bytes

Rückgabewert

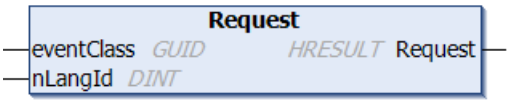
Name	Typ	Beschreibung
GetString	BOOL	Liefert TRUE, wenn der Text zugewiesen werden konnte. Liefert FALSE, wenn der Text nicht vollständig zugewiesen werden konnte, weil die angegebene Puffervariable zu klein ist.

Beispiel

Die Methode darf erst dann aufgerufen werden, wenn mittels bBusy = FALSE und bError = FALSE signalisiert wurde, dass ein Text zur Verfügung steht.

```
IF NOT fb.bBusy AND NOT fb.bError THEN
    bGetStringSuccess := fb.GetString(sText, SIZEOF(sText));
END_IF
```

6.1.1.4.3 Request



Der Aufruf dieser Methode triggert die asynchrone Textanfrage.

Syntax

```
METHOD Request : HRESULT
VAR_INPUT
    eventClass : GUID;
    nLangId    : DINT;
END_VAR
```

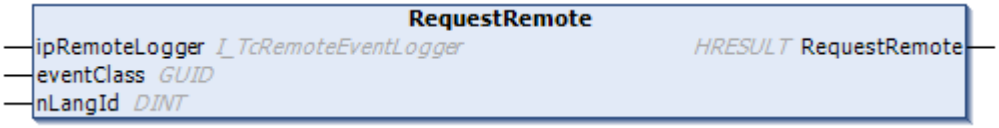
Eingänge

Name	Typ	Beschreibung
eventClass	GUID	GUID der Ereignisklasse.
nLangId	DINT	Spezifiziert die Sprach-ID Englisch (en-US) = 1033 Deutsch (de-DE) = 1031

Rückgabewert

Name	Typ	Beschreibung
Request	HRESULT	Liefert mögliche Fehlerinformationen.

6.1.1.4.4 RequestRemote



Der Aufruf dieser Methode triggert die asynchrone Anfrage auf einem entfernten Zielsystem.

Syntax

```
METHOD RequestRemote : HRESULT
VAR_INPUT
    ipRemoteLogger   : I_TcRemoteEventLogger;
    eventClass       : GUID;
    nLangId          : DINT; // English(US)=1033 ; German(Germay)=1031
END_VAR
```

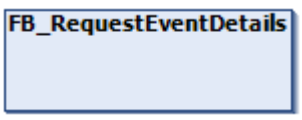
Eingänge

Name	Typ	Beschreibung
ipRemoteLogger	I_TcRemoteEventLogger	Instanz vom I_TcRemoteEventLogger, der das Zielsystem beschreibt.
eventClass	GUID	Spezifiziert die Ereignisklasse.
nLangId	DINT	Spezifiziert die Sprach-ID: Englisch (en-US) = 1033 Deutsch (de-DE) = 1031 … Der Enum TcLcid kann verwendet werden.

Ausgänge

Name	Typ	Beschreibung
RequestRemote	HRESULT	Liefert mögliche Fehlerinformationen. S_OK, wenn die Abfrage erfolgreich ist. ADS_E_INVALIDSTATE, wenn ein illegaler Zustand des Zielsystems vorliegt. ADS_E_NOMEMORY, wenn kein Speicher bereitsteht, um das Ergebnis abzulegen. ADS_E_PENDING, wenn die Abfrage noch bearbeitet wird.

6.1.1.5 FB_RequestEventDetails



Mit diesem Funktionsbaustein können die Details eines Ereignisses in Form von FB_TcDetail [▶ 60] abgefragt werden.

Methoden

Name	Beschreibung
Clear [▶ 49]	Diese Methode löscht das zuletzt abgefragte Ergebnis.
Get [▶ 49]	Sobald bBusy FALSE ist und wenn kein Fehler aufgetreten ist (bError = FALSE), kann mittels dieser Methode ein Ergebnis aus der abgefragten Liste abgeholt werden.
Request [▶ 50]	Der Aufruf dieser Methode triggert die asynchrone Anfrage.
RequestRemote [▶ 51]	Der Aufruf dieser Methode triggert die asynchrone Anfrage auf einem entfernten Zielsystem.

Eigenschaften

Name	Typ	Zugriff	Beschreibung
bBusy	BOOL	GET	TRUE, solange die Abarbeitung noch nicht abgeschlossen ist.
bError	BOOL	GET	TRUE, sobald ein Fehler eintritt.
hrErrorCode	HRESULT	GET	Gibt die Fehlerinformation aus, wenn bError TRUE ist.
nCount	UDINT	GET	Anzahl der gefundenen Ursachen/behobenen Elemente

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.1.4026.0	PC oder CX (x64, x86, Arm®)	Tc3_EventLogger (>=3.3.7.0)

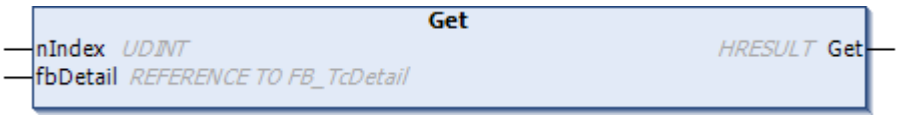
6.1.1.5.1 Clear

Diese Methode löscht das zuletzt abgefragte Ergebnis.

Syntax

```
METHOD Clear
```

6.1.1.5.2 Get



Sobald bBusy FALSE ist und wenn kein Fehler aufgetreten ist (bError = FALSE), kann mittels dieser Methode ein Ergebnis aus der abgefragten Liste abgeholt werden.

Syntax

```
METHOD Get    : HRESULT
VAR_INPUT
    nIndex    : UDINT;
    fbDetail : REFERENCE TO FB_TcDetail;
END_VAR
```

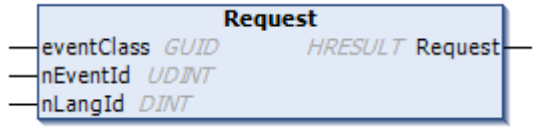
Eingänge

Name	Typ	Beschreibung
nIndex	UDINT	Index in der Liste von Ursachen/behobenen Elementen, der abzufragen ist.
fbDetail	Reference to FB_TcDetail [▶ 60]	Referenz auf einen FB_TcDetail, in dem die Details abgelegt werden.

Ausgänge

Name	Typ	Beschreibung
Get	HRESULT	S_OK bei Erfolg ADS_E_INVALIDSTATE, wenn die Abfrage noch läuft. ADS_E_INVALIDSIZE, wenn ein Index abgefragt wird, der nicht belegt ist.

6.1.1.5.3 Request



Der Aufruf dieser Methode triggert die asynchrone Anfrage.

Syntax

```
METHOD Request : HRESULT
VAR_INPUT
    eventClass : GUID;
    nEventId   : UDINT;
    nLangId    : DINT; // English(US)=1033 ; German(Germay)=1031
END_VAR
```

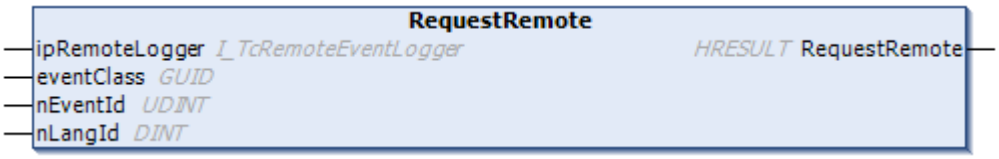
Eingänge

Name	Typ	Beschreibung
eventClass	GUID	Spezifiziert die Ereignisklasse.
nEventId	UDINT	ID von dem Ereignis
nLangId	DINT	Spezifiziert die Sprach-ID: Englisch (en-US) = 1033 Deutsch (de-DE) = 1031 … Der Enum TcLcid kann verwendet werden.

Ausgänge

Name	Typ	Beschreibung
Request	HRESULT	Liefert mögliche Fehlerinformationen. S_OK, wenn die Abfrage erfolgreich ist. ADS_E_INVALIDSTATE, wenn ein illegaler Zustand des Zielsystems vorliegt. ADS_E_NOMEMORY, wenn kein Speicher bereitsteht, um das Ergebnis abzulegen. ADS_E_PENDING, wenn die Abfrage noch bearbeitet wird.

6.1.1.5.4 RequestRemote



Der Aufruf dieser Methode triggert die asynchrone Anfrage auf einem entfernten Zielsystem.

Syntax

```
METHOD RequestRemote : HRESULT
VAR_INPUT
    ipRemoteLogger   : I_TcRemoteEventLogger;
    eventClass       : GUID;
    nEventId         : UDINT;
    nLangId          : DINT; // English(US)=1033 ; German(Germay)=1031
END_VAR
```

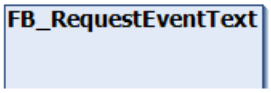
Eingänge

Name	Typ	Beschreibung
ipRemoteLogger	I_TcRemoteEventLogger	Instanz vom I_TcRemoteEventLogger, der das Zielsystem beschreibt.
eventClass	GUID	Spezifiziert die Ereignisklasse.
nEventId	UDINT	ID von dem Ereignis
nLangId	DINT	Spezifiziert die Sprach-ID: Englisch (en-US) = 1033 Deutsch (de-DE) = 1031 … Der Enum TcLcid kann verwendet werden.

Ausgänge

Name	Typ	Beschreibung
RequestRemote	HRESULT	Liefert mögliche Fehlerinformationen. S_OK, wenn die Abfrage erfolgreich ist. ADS_E_INVALIDSTATE, wenn ein illegaler Zustand des Zielsystems vorliegt. ADS_E_NOMEMORY, wenn kein Speicher bereitsteht, um das Ergebnis abzulegen. ADS_E_PENDING, wenn die Abfrage noch bearbeitet wird.

6.1.1.6 FB_RequestEventText



Dieser Funktionsbaustein ermöglicht die asynchrone Anfrage eines Ereignistextes in gewünschter Sprache.

Methoden

Name	Beschreibung
Clear [▶ 54]	Diese Methode löscht das zuletzt abgefragte Ergebnis.
GetString [▶ 52]	Sobald bBusy FALSE ist und wenn kein Fehler aufgetreten ist (bError = FALSE), kann mittels dieser Methode der angefragte Text abgeholt werden.
Request [▶ 53]	Der Aufruf dieser Methode triggert die asynchrone Textanfrage.
RequestRemote [▶ 54]	Der Aufruf dieser Methode triggert die asynchrone Anfrage auf einem entfernten Zielsystem.

Eigenschaften

Name	Typ	Zugriff	Beschreibung
bBusy	BOOL	Get	TRUE, solange die Abarbeitung noch nicht abgeschlossen ist.
bError	BOOL	Get	TRUE, sobald ein Fehler eintritt.
hrErrorCode	HRESULT	Get	Gibt die Fehlerinformation aus, wenn bError TRUE ist.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.1.4022.20	PC oder CX (x64, x86, Arm®)	Tc3_EventLogger

6.1.1.6.1 GetString

Sobald bBusy = FALSE ist und wenn kein Fehler aufgetreten ist (bError = FALSE), kann mittels dieser Methode der angefragte Text abgeholt werden.

Syntax

```
METHOD GetString : BOOL
VAR_INPUT
    sResult : REFERENCE TO STRING;
    nResult : UDINT;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
sResult	REFERENCE TO STRING	Puffervariable für den angefragten Text
nResult	UDINT	Puffergröße in Bytes

Rückgabewert

Name	Typ	Beschreibung
GetString	BOOL	Liefert TRUE, wenn der Text zugewiesen werden konnte. Liefert FALSE, wenn der Text nicht vollständig zugewiesen werden konnte, weil die angegebene Puffervariable zu klein ist.

Beispiel

Die Methode darf erst dann aufgerufen werden, wenn mittels bBusy = FALSE und bError = FALSE signalisiert wurde, dass ein Text zur Verfügung steht.

```
IF NOT fb.bBusy AND NOT fb.bError THEN
    bGetStringSuccess := fb.GetString(sText, SIZEOF(sText));
END_IF
```

6.1.1.6.2 Request

Der Aufruf dieser Methode triggert die asynchrone Textanfrage.

Syntax

```
METHOD Request : BOOL
VAR_INPUT
    eventClass : GUID;
    nEventId   : UDINT;
    nLangId    : DINT;
    ipArgs     : I_TcArguments;
END_VAR
```

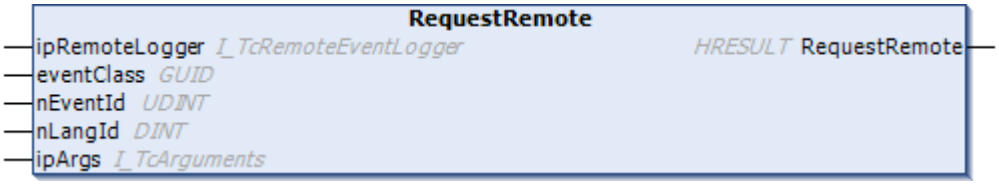
Eingänge

Name	Typ	Beschreibung
eventClass	GUID	Spezifiziert die Ereignisklasse.
nEventId	UDINT	ID von dem Ereignis.
nLangId	DINT	Spezifiziert die Sprach-ID: Englisch (en-US) = 1033 Deutsch (de-DE) = 1031 … Der Enum TcLcid kann verwendet werden.
ipArgs	I_TcArguments [▶ 123]	Optionale Angabe von Argumenten.

Rückgabewert

Name	Typ	Beschreibung
Request	HRESULT	Liefert mögliche Fehlerinformationen.

6.1.1.6.3 RequestRemote



Der Aufruf dieser Methode triggert die asynchrone Anfrage auf einem entfernten Zielsystem.

Syntax

```
METHOD RequestRemote : HRESULT
VAR_INPUT
    ipRemoteLogger   : I_TcRemoteEventLogger;
    eventClass       : GUID;
    nEventId         : UDINT;
    nLangId          : DINT;
    ipArgs           : I_TcArguments;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
ipRemoteLogger	I_TcRemoteEventLogger	Instanz vom I_TcRemoteEventLogger, der das Zielsystem beschreibt.
eventClass	GUID	Spezifiziert die Ereignisklasse.
nEventId	UDINT	ID von dem Ereignis.
nLangId	DINT	Spezifiziert die Sprach-ID: Englisch (en-US) = 1033 Deutsch (de-DE) = 1031 … Der Enum TcLcid kann verwendet werden.
ipArgs	I_TcArguments	Optionale Angabe von Argumenten.

Rückgabewert

Name	Typ	Beschreibung
RequestRemote	HRESULT	

Sehen Sie dazu auch

I_TcArguments [▶ 123]

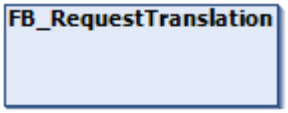
6.1.1.6.4 Clear

Diese Methode löscht das zuletzt abgefragte Ergebnis.

Syntax

```
METHOD Clear
```

6.1.1.7 FB_RequestTranslation



Dieser Funktionsbaustein liefert zu einem Text und einer gewünschten Sprache eine Übersetzung.

Die Übersetzungen werden durch eine Ereignisklasse referenziert, welche bei der Abfrage angegeben wird. Sie wird an dieser Stelle also nicht als Eventklasse verwendet, sondern als Übersetzungstabelle.

Methoden

Name	Beschreibung
Clear [▶ 58]	Diese Methode löscht das zuletzt abgefragte Ergebnis.
GetString [▶ 55]	Sobald bBusy FALSE ist und wenn kein Fehler aufgetreten ist (bError = FALSE), kann mittels dieser Methode der angefragte Text abgeholt werden.
Request [▶ 56]	Der Aufruf dieser Methode triggert die asynchrone Textanfrage.
RequestRemote [▶ 57]	Der Aufruf dieser Methode triggert die asynchrone Anfrage auf einem entfernten Zielsystem.

Eigenschaften

Name	Typ	Zugriff	Beschreibung
bBusy	BOOL	Get	TRUE, solange die Abarbeitung noch nicht abgeschlossen ist.
bError	BOOL	Get	TRUE, sobald ein Fehler eintritt.
hrErrorCode	HRESULT	Get	Gibt die Fehlerinformation aus, wenn bError TRUE ist.

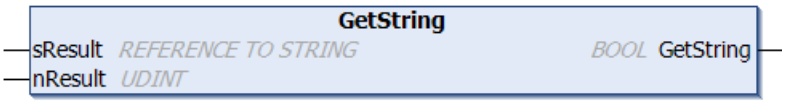
Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.1.4026.0	PC oder CX (x64, x86, Arm®)	Tc3_EventLogger (>=3.3.7.0)

Sehen Sie dazu auch

- Internationalisierung von Quellen [▶ 33]

6.1.1.7.1 GetString



Sobald bBusy = FALSE ist und wenn kein Fehler aufgetreten ist (bError = FALSE), kann mittels dieser Methode der angefragte Text abgeholt werden.

Syntax

```
METHOD GetString : BOOL
VAR_INPUT
    sResult : REFERENCE TO STRING;
    nResult : UDINT;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
sResult	REFERENCE TO STRING	Puffervariable für den angefragten Text
nResult	UDINT	Puffergröße in Bytes

Rückgabewert

Name	Typ	Beschreibung
GetString	BOOL	Liefert TRUE, wenn der Text zugewiesen werden konnte. Liefert FALSE, wenn der Text nicht vollständig zugewiesen werden konnte, weil die angegebene Puffervariable zu klein ist.

Beispiel

Die Methode darf erst dann aufgerufen werden, wenn mittels bBusy = FALSE und bError = FALSE signalisiert wurde, dass ein Text zur Verfügung steht.

```
IF NOT fb.bBusy AND NOT fb.bError THEN
    bGetStringSuccess := fb.GetString(sText, SIZEOF(sText));
END_IF
```

6.1.1.7.2 Request

Der Aufruf dieser Methode triggert die asynchrone Textanfrage.

Syntax

```
METHOD Request : BOOL
VAR_INPUT
    eventClass : GUID;
END_VAR
VAR_IN_OUT CONSTANT
    text       : STRING;
END_VAR
VAR_INPUT
    nLangId    : DINT;
    ipArgs     : I_TcArguments;
END_VAR
```

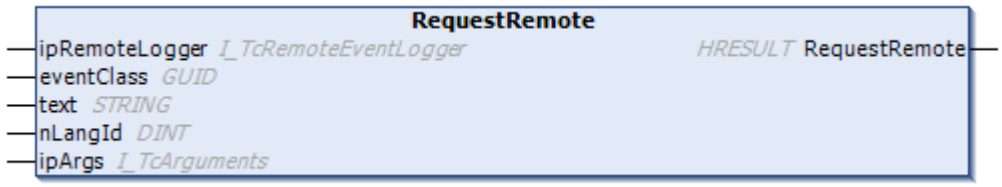
Eingänge

Name	Typ	Beschreibung
eventClass	GUID	Spezifiziert die Ereignisklasse, welche als Basis für die Abfrage verwendet werden soll.
text	STRING	Der zu übersetzende Text
nLangId	DINT	Spezifiziert die Sprach-ID: Englisch (en-US) = 1033 Deutsch (de-DE) = 1031 … Der Enum TcLcid kann verwendet werden.
ipArgs	I_TcArguments [▶ 123]	Optionale Angabe von Argumenten.

Rückgabewert

Name	Typ	Beschreibung
Request	HRESULT	Liefert mögliche Fehlerinformationen.

6.1.1.7.3 RequestRemote



Der Aufruf dieser Methode die asynchrone Anfrage auf einem entfernten Zielsystem.

Syntax

```
METHOD RequestRemote : HRESULT
VAR_INPUT
    ipRemoteLogger   : I_TcRemoteEventLogger
    eventClass       : GUID;
END_VAR
VAR_IN_OUT CONSTANT
    text             : STRING;
END_VAR
VAR_INPUT
    nLangId          : DINT;
    ipArgs           : I_TcArguments;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
ipRemoteLogger	I_TcRemoteEventLogger	Instanz vom I_TcRemoteEventLogger, der das Zielsystem beschreibt.
eventClass	GUID	Spezifiziert die Ereignisklasse, welche als Basis für die Abfrage verwendet werden soll.
text	STRING	Der zu übersetzende Text
nLangId	DINT	Spezifiziert die Sprach-ID: Englisch (en-US) = 1033 Deutsch (de-DE) = 1031 … Der Enum TcLcid kann verwendet werden.
ipArgs	I_TcArguments [▶ 123]	Optionale Angabe von Argumenten.

Rückgabewert

Name	Typ	Beschreibung
RequestRemote	HRESULT	Liefert mögliche Fehlerinformationen. S_OK, wenn die Abfrage erfolgreich ist. ADS_E_INVALIDSTATE, wenn ein invalider Zustand des Zielsystems vorliegt. ADS_E_NOMEMORY, wenn kein Speicher bereitsteht, um das Ergebnis abzulegen. ADS_E_PENDING, wenn die Abfrage noch bearbeitet wird.

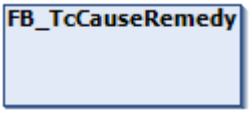
6.1.1.7.4 Clear

Diese Methode löscht das zuletzt abgefragte Ergebnis.

Syntax

```
METHOD Clear
```

6.1.1.8 FB_TcCauseRemedy



Dieser Funktionsbaustein wird verwendet, um die Ursache/Behebung zu einem Alarm darzustellen. Dieser Funktionsbaustein wird mittels FB_RequestCauseRemedy [▶ 40] genutzt.

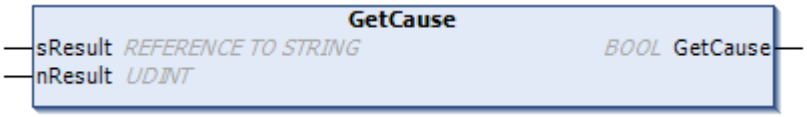
Methoden

Name	Beschreibung
GetCause [▶ 58]	Liefert die Ursache.
GetId [▶ 59]	Liefert den Index der Ursache/Behebung.
GetRemedy [▶ 59]	Liefert die Behebung.
Release [▶ 60]	Löscht den internen Speicher des Bausteins.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.1.4026.0	PC oder CX (x64, x86, Arm®)	Tc3_EventLogger (>=3.3.7.0)

6.1.1.8.1 GetCause



Diese Methode liefert die Ursache.

Syntax

```
METHOD GetCause : BOOL
VAR_INPUT
    sResult     : REFERENCE TO STRING;
    nResult     : UDINT; // buffer size in bytes
END_VAR
```

Eingänge

Name	Typ	Beschreibung
sResult	REFERENCE TO STRING	Referenz auf einen String, in dem das Ergebnis abgelegt werden soll.
nResult	UDINT	Länge des sResult

Ausgänge

Name	Typ	Beschreibung
GetCause	BOOL	Liefert den Erfolg, ob eine Ursache bereitgestellt wurde.

6.1.1.8.2 GetId

Diese Methode liefert den Index der Ursache/Behebung.

Syntax

```
METHOD GetId : BOOL
VAR_INPUT
    sResult  : REFERENCE TO STRING;
    nResult  : UDINT; // buffer size in bytes
END_VAR
```

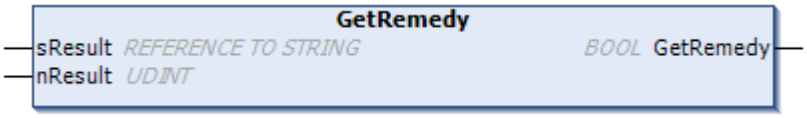
Eingänge

Name	Typ	Beschreibung
sResult	REFERENCE TO STRING	Referenz auf einen String, in dem das Ergebnis abgelegt werden soll.
nResult	UDINT	Länge des sResult

Ausgänge

Name	Typ	Beschreibung
GetId	BOOL	Liefert den Erfolg, ob eine Id bereitgestellt wurde.

6.1.1.8.3 GetRemedy



Diese Methode liefert die Behebung.

Syntax

```
METHOD GetRemedy : BOOL
VAR_INPUT
    sResult      : REFERENCE TO STRING;
    nResult      : UDINT; // buffer size in bytes
END_VAR
```

Eingänge

Name	Typ	Beschreibung
sResult	REFERENCE TO STRING	Referenz auf einen String, in dem das Ergebnis abgelegt werden soll.
nResult	UDINT	Länge des sResult

Ausgänge

Name	Typ	Beschreibung
GetRemedy	BOOL	Liefert den Erfolg, ob eine Behebung bereitgestellt wurde.

6.1.1.8.4 Release

Diese Methode löscht den internen Speicher des Bausteins.

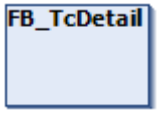
Syntax

```
METHOD Release : HRESULT
```

Ausgänge

Name	Typ	Beschreibung
Release	HRESULT	Liefert den Erfolg, ob der Speicher aufgeräumt wurde.

6.1.1.9 FB_TcDetail



Dieser Funktionsbaustein wird verwendet um die Details zu einem Ereignis darzustellen.

Dieser funktionsbaustein wird mittels FB_RequestEventClassDetails [▶ 43] oder FB_RequestEventDetails [▶ 48] verwendet.

Methoden

Name	Beschreibung
GetComment [▶ 61]	Liefert den Kommentar.
GetName [▶ 61]	Liefert den Namen.
GetText [▶ 62]	Liefert den Text.
Release [▶ 60]	Löscht den internen Speicher des Bausteins

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.1.4026.0	PC oder CX (x64, x86, Arm®)	Tc3_EventLogger (>=3.3.7.0)

6.1.1.9.1 Release

Diese Methode löscht den internen Speicher des Bausteins.

Syntax

```
METHOD Release : HRESULT
```

Ausgänge

Name	Typ	Beschreibung
Release	HRESULT	Liefert den Erfolg, ob der Speicher aufgeräumt wurde.

6.1.1.9.2 GetComment

Diese Methode liefert den Kommentar.

Syntax

```
METHOD GetComment : BOOL
VAR_INPUT
    sResult       : REFERENCE TO STRING;
    nResult       : UDINT; // buffer size in bytes
END_VAR
```

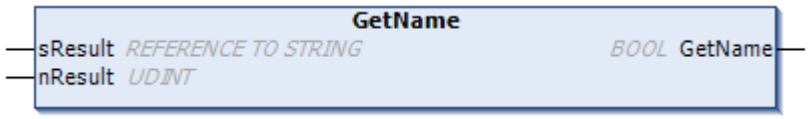
Eingänge

Name	Typ	Beschreibung
sResult	REFERENCE TO STRING	Referenz auf einen String, in dem das Ergebnis abgelegt werden soll.
nResult	UDINT	Länge des sResult

Ausgänge

Name	Typ	Beschreibung
GetCommend	BOOL	Liefert den Erfolg, ob ein Kommentar bereitgestellt wurde.

6.1.1.9.3 GetName



Diese Methode liefert den Namen.

Syntax

```
METHOD GetName : BOOL
VAR_INPUT
    sResult     : REFERENCE TO STRING;
    nResult     : UDINT; // buffer size in bytes
END_VAR
```

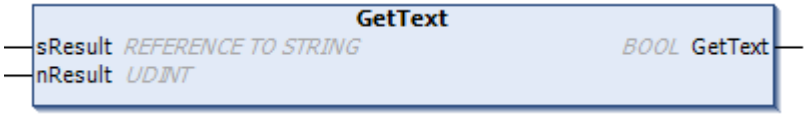
Eingänge

Name	Typ	Beschreibung
sResult	REFERENCE TO STRING	Referenz auf einen String, in dem das Ergebnis abgelegt werden soll.
nResult	UDINT	Länge des sResult

Ausgänge

Name	Typ	Beschreibung
GetName	BOOL	Liefert den Erfolg, ob ein Name bereitgestellt wurde.

6.1.1.9.4 GetText



Diese Methode liefert den Text.

Syntax

```
METHOD GetText : BOOL
VAR_INPUT
    sResult    : REFERENCE TO STRING;
    nResult    : UDINT; // buffer size in bytes
END_VAR
```

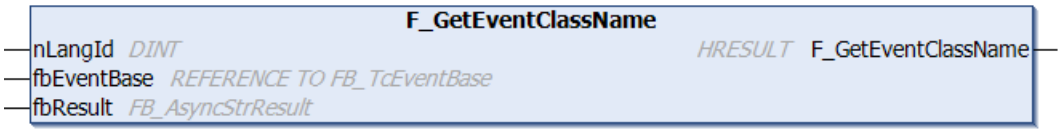
Eingänge

Name	Typ	Beschreibung
sResult	REFERENCE TO STRING	Referenz auf einen String, in dem das Ergebnis abgelegt werden soll.
nResult	UDINT	Länge des sResult

Ausgänge

Name	Typ	Beschreibung
GetText	BOOL	Liefert den Erfolg, ob ein Text bereitgestellt wurde.

6.1.1.10 F_GetEventClassName



Die Funktion triggert die asynchrone Anfrage des Namens einer Ereignisklasse.

Syntax

Definition:

```
FUNCTION F_GetEventClassName : HRESULT
VAR_INPUT
    nLangId     : DINT;
    fbEventBase : REFERENCE TO FB_TcEventBase;
END_VAR
VAR_IN_OUT
    fbResult : FB_AsyncStrResult;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
nLangId	DINT	Spezifiziert die Sprach-ID Englisch (en-US) = 1033 Deutsch (de-DE) = 1031 …
fbEventBase	REFERENCE TO FB_TcEventBase [▶ 100]	Angabe eines Event/Alarm/Message-Objektes.

/ Ein-/Ausgänge

Name	Typ	Beschreibung
fbResult	FB_AsyncStrResult [▶ 39]	Angabe einer Bausteininstanz, um die asynchrone Textanfrage zu verfolgen.

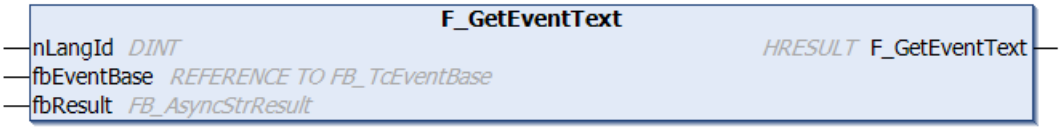
Rückgabewert

Name	Typ	Beschreibung
F_GetEventClassName	HRESULT	Liefert mögliche Fehlerinformationen.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.1.4022.20	PC oder CX (x64, x86, Arm®)	Tc3_EventLogger

6.1.1.11 F_GetEventText



Die Funktion triggert die asynchrone Anfrage eines Ereignistextes.

Syntax

Definition:

```
FUNCTION F_GetEventText : HRESULT
VAR_INPUT
    nLangId     : DINT;
    fbEventBase : REFERENCE TO FB_TcEventBase;
END_VAR
VAR_IN_OUT
    fbResult : FB_AsyncStrResult;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
nLangId	DINT	Spezifiziert die Sprach-ID Englisch (en-US) = 1033 Deutsch (de-DE) = 1031 …
fbEventBase	REFERENCE TO FB_TcEventBase [▶ 100]	Angabe eines Event/Alarm/Message-Objektes.

/ Ein-/Ausgänge

Name	Typ	Beschreibung
fbResult	FB_AsyncStrResult [▶ 39]	Angabe einer Bausteininstanz, um die asynchrone Textanfrage zu verfolgen.

Rückgabewert

Name	Typ	Beschreibung
F_GetEventText	HRESULT	Liefert mögliche Fehlerinformationen.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.1.4022.20	PC oder CX (x64, x86, Arm®)	Tc3_EventLogger

6.1.2 EventEntry-Konvertierung

6.1.2.1 AdsErr_TO_TcEventEntry

Diese Funktion konvertiert einen Standard-ADS-Fehler in ein TcEventEntry.

Syntax

Definition:

```
FUNCTION AdsErr_TO_TcEventEntry : BOOL
VAR_INPUT
    eErrorId     : E_AdsErr;
    stEventEntry : REFERENCE TO TcEventEntry;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
eErrorId	E_AdsErr	Zu konvertierender Fehlercode.
stEventEntry	REFERENCE TO TcEventEntry [▶ 139]	Gibt die resultierende Ereignisdefinition aus.

Rückgabewert

Name	Typ	Beschreibung
AdsErr_TO_TcEventEntry	BOOL	Liefert TRUE, wenn die Konvertierung erfolgreich durchgeführt wurde.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.1.4022.20	PC oder CX (x64, x86, Arm®)	Tc3_EventLogger

6.1.2.2 HRESULTAdsErr_TO_TcEventEntry

Diese Funktion konvertiert einen Standard-ADS-Fehler (HRESULT) in ein TcEventEntry.

Syntax

Definition:

```
FUNCTION HRESULTAdsErr_TO_TcEventEntry : BOOL
VAR_INPUT
    hr           : E_HRESULTAdsErr;
    stEventEntry : REFERENCE TO TcEventEntry;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
hr	E_HRESULTAdsErr	Zu konvertierender Fehlercode.
stEventEntry	REFERENCE TO TcEventEntry [▶ 139]	Gibt die resultierende Ereignisdefinition aus.

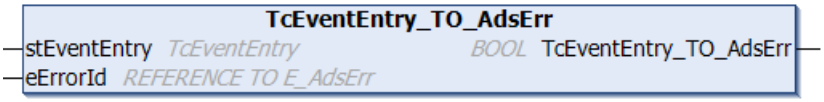
Rückgabewert

Name	Typ	Beschreibung
HRESULTAdsErr_TO_TcEventEntry	BOOL	Liefert TRUE, wenn die Konvertierung erfolgreich durchgeführt wurde. Der Aufruf schlägt fehl, wenn der Facility Code im angegebenen HRESULT unbekannt ist.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.1.4022.20	PC oder CX (x64, x86, Arm®)	Tc3_EventLogger

6.1.2.3 TcEventEntry_TO_AdsErr



Diese Funktion konvertiert ein TcEventEntry in einen Standard-ADS-Fehler.

Syntax

Definition:

```
FUNCTION TcEventEntry_TO_AdsErr : BOOL
VAR_INPUT
    stEventEntry : TcEventEntry;
    eErrorId     : REFERENCE TO E_AdsErr;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
stEventEntry	TcEventEntry [▶ 139]	Zu konvertierende Ereignisdefinition.
eErrorId	REFERENCE TO E_AdsErr	Gibt den resultierenden Fehlercode aus.

Rückgabewert

Name	Typ	Beschreibung
TcEventEntry_TO_AdsErr	BOOL	Liefert TRUE, wenn die Konvertierung erfolgreich durchgeführt wurde und FALSE, wenn die Ereignisklasse unbekannt ist.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.1.4022.20	PC oder CX (x64, x86, Arm®)	Tc3_EventLogger

6.1.2.4 TcEventEntry_TO_HRESULTAdsErr

Diese Funktion konvertiert ein TcEventEntry in einen Standard-ADS-Fehler (HRESULT).

Syntax

Definition:

```
FUNCTION TcEventEntry_TO_HRESULTAdsErr : BOOL
VAR_INPUT
    stEventEntry : TcEventEntry;
    hr           : REFERENCE TO E_HRESULTAdsErr;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
stEventEntry	TcEventEntry [▶ 139]	Zu konvertierende Ereignisdefinition.
hr	REFERENCE TO E_HRESULTAdsErr	Gibt den resultierenden Fehlercode aus.

Rückgabewert

Name	Typ	Beschreibung
TcEventEntry_TO_HRESULTAdsErr	BOOL	Liefert TRUE, wenn die Konvertierung erfolgreich durchgeführt wurde und FALSE, wenn die Ereignisklassse unbekannt ist.

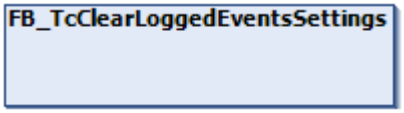
Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.1.4022.20	PC oder CX (x64, x86, Arm®)	Tc3_EventLogger

6.1.3 Filter

Die Filterfunktionalität wird an unterschiedlichen Stellen verwendet. Ein Beispiel, welches die Verwendungsmöglichkeiten beschreibt, ist das Beispiel Filter [▶ 205].

6.1.3.1 FB_TcClearLoggedEventsSettings



Dieser Funktionsbaustein bietet die Funktionalität, um festzulegen, welche Ereignisse aus dem Cache entfernt werden sollen.

Syntax

Definition:

```
FUNCTION_BLOCK FB_TcClearLoggedEventsSettings IMPLEMENTS I_TcClearLoggedEventsSettings
```

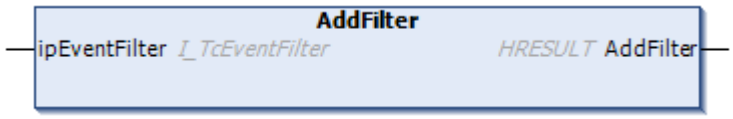
Methoden

Name	Definitionsort	Beschreibung
AddFilter [▶ 67]	Lokal	Methode, um einen Filter hinzuzufügen. Liefert bei Erfolg S_OK.
Clear [▶ 68]	Lokal	Methode zum Löschen der Einstellungen. Liefert bei Erfolg S_OK.
SetLimit [▶ 68]	Lokal	Gibt die Anzahl der zu löschenden Ereignisse an. Die Begrenzung wird nach dem Sortieren und Filtern angewendet. Liefert bei Erfolg S_OK.
SetSorting [▶ 68]	Lokal	Legt die Sortierung für die Abfrage fest. Liefert bei Erfolg S_OK.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.1.4024.17	PC oder CX (x64, x86, Arm®)	Tc3_EventLogger (>= v3.1.27.0)

6.1.3.1.1 AddFilter



Methode, um einen Filter hinzuzufügen.

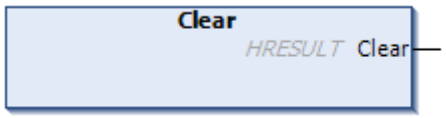
Eingänge

Name	Typ	Beschreibung
ipEventFilter	I_TcEventFilter	Instanz des zu nutzenden Filters

Rückgabewerte

Name	Typ	Beschreibung
AddFilter	HRESULT	Liefert bei Erfolg S_OK.

6.1.3.1.2 Clear



Methode zum Löschen der Einstellungen.

Rückgabewerte

Name	Typ	Beschreibung
Clear	HRESULT	Liefert bei Erfolg S_OK.

6.1.3.1.3 SetLimit

Gibt die Anzahl der zu löschenden Ereignisse an. Die Begrenzung wird nach dem Sortieren und Filtern angewendet.

Eingänge

Name	Typ	Beschreibung
eType	TcEventLimitType	Bestimmt die Referenz, ob die Begrenzung für die ersten oder letzten Ereignisse gilt.
nLimit	DINT	Spezifiziert die Anzahl (-1 = no limit)
nOffset	DINT	Optional. Definiert, wie viele Einträge übersprungen werden sollen.

Rückgabewerte

Name	Typ	Beschreibung
SetLimit	HRESULT	Liefert bei Erfolg S_OK.

6.1.3.1.4 SetSorting

Legt die Sortierung für die Abfrage fest.

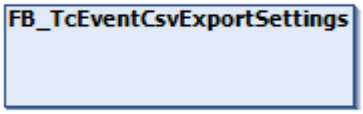
Eingänge

Name	Typ	Beschreibung
eField	TcEventField	Eigenschaft, die für die Sortierung verwendet werden soll.
eOrder	TcEventSortOrder	Definiert die Sortierreihenfolge.

Rückgabewerte

Name	Typ	Beschreibung
SetSorting	HRESULT	Liefert bei Erfolg `S_OK`.

6.1.3.2 FB_TcEventCsvExportSettings



Bietet die Funktionalität, den csv-Export zu spezifizieren.

Syntax

Definition:

```
FUNCTION_BLOCK FB_TcEventCsvExportSettings EXTENDS FB_TcEventExportSettings IMPLEMENTS
I_TcEventCsvExportSettings
```

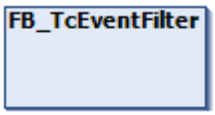
Methoden

Name	Typ	Beschreibung
bWithHeader	BOOL	Bestimmt, ob eine Kopfzeile erstellt werden soll. Standard: True
nLangId	DINT	Bestimmt die Standard-Kennung der Exportsprache. Standard: 1033
sDelimiter	STRING	Definiert das CSV-Begrenzungszeichen. Standard: Semicolon [;]

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.1.4024.17	PC oder CX (x64, x86, Arm®)	Tc3_EventLogger (>= v3.1.27.0)

6.1.3.3 FB_TcEventFilter



Stellt die Funktionalität zur Verfügung, um einen Ereignisfilter zu spezifizieren.

Die Filter werden über eine fließende Schnittstelle in Anlehnung an eine strukturierte Abfragesprache gegeben. Diese beschreibt, welche Nachrichten zutreffen sollen.

- Bedingungen können durch `.AND_OP()` und `.OR_OP()` verknüpft werden.
- Bedingungen können durch `.NOT_OP()` negiert werden.
- Bedingungen können durch Eigenschaften wie `.isAlarm()` oder beispielsweise `.EventClass.EqualsTo(<EventClass>)` definiert werden. Eine vollständige Liste der Eigenschaften befindet sich in der API-Dokumentation.
- Eine Gruppierung kann durch `.FilterExpression(<SubCondition>)` formuliert werden. Die `<SubCodition>` ist selbst wieder ein `FB_TcEventFilter` bzw. `ITcEventFilter`.

Nachdem ein Filter zusammengestellt wurde, wird er angewendet. Für das Empfangen von Nachrichten wird er beispielsweise über `FB_ListenerBase2.subscribe()` einem Empfänger zugeordnet. Der `FB_ListenerBase2` übernimmt hierdurch den Filter und gibt einen entsprechenden Rückgabewert, welcher hier beschrieben ist. Eine Änderung des Filters durch erneuten `FB_ListenerBase2.subscribe()` vornehmen.

Es können maximal 255 Filterbedingungen gesetzt werden – danach wird als Fehlermeldung ein ADS_NOMOREHDL rückgegeben.

Beispiel

Beispielsweise kann ein Filter auf diese Art zusammengestellt werden:

```
fbFilter.Severity.GreaterThan (TcEVentSeverity.Error).AND_OP().Source.Name.Like('%Main%');
```

Das Beispiel Filter [▶ 205] zeigt die Verwendung.

EtherCAT Filter

Der Empfang der EtherCAT Emergency Nachrichten ist ähnlich zu dem zuvor beschriebenen Mechanismus aufgebaut. Der Einstiegspunkt in den verketteten Methodenaufrufen ist `.EtherCATDevice()`, welches als erstes die direkte Anfrage bietet, ob es von einem EtherCAT Gerät abgesendet wurde (`IsEtherCATDevice()`). Von hier aus kann auf den Hersteller (`.VendorId()`), den ProductCode (`.ProductCode()`) sowie die Revision (`.RevisionNo()`) gefiltert werden.

Syntax

Definition:

```
FUNCTION_BLOCK FB_TcEventFilter IMPLEMENTS I_TcEventFilter, I_TcExpressionBase
```

Methoden

Name	Definitionsort	Beschreibung
Clear [▶ 71]	I_TcEventFilter	Löscht den bisherige Filterausdruck.
FilterExpression [▶ 71]	I_TcExpressionBase	Angabe einer unterlagerten Filterdefinition.
IsAlarm [▶ 72]	I_TcExpressionBase	Überprüfung, ob es sich um einen Alarm handelt.
IsMessage [▶ 72]	I_TcExpressionBase	Überprüfung, ob es sich um eine Nachricht handelt.
NOT_OP [▶ 72]	I_TcExpressionBase	Negierung der folgenden Aussage.

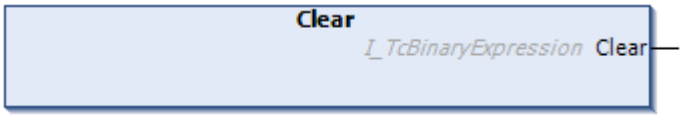
Eigenschaften

Name	Typ	Zugriff	Beschreibung
AlarmState	I_TcAlarmFilterExpression	Get	Abgleich mit einem AlarmState
EtherCATDevice	I_TcEtherCATDeviceExpression	Get	Abgleich, ob es sich um ein EtherCAT Gerät als Quelle handelt.
EventClass	I_TcGuidCompare	Get	Abgleich mit einer EventClass
EventId	I_TcUDIntCompare	Get	Abgleich mit einer EventId
JsonAttribute	I_TcJsonAttributeExpression	Get	Abgleich mit dem JsonAttribut
Severity	I_TcSeverityCompare	Get	Abgleich mit der Serverity
Source	I_TcSourceInfoExpression	Get	Abgleich mit der Quelle
TimeCleared	I_TcULIntCompare	Get	Abgleich des Clear-Zeitpunkts (nur bei Alarm)
TimeConfirmed	I_TcULIntCompare	Get	Abgleich des Confirm-Zeitpunkts (nur bei Alarm mit Quittierung)
TimeRaised	I_TcULIntCompare	Get	Abgleich des Absenders (bei Nachrichten) bzw. des Raised -Zeitpunktes (bei Alarmen)

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.1.4024.17	PC oder CX (x64, x86, Arm®)	Tc3_EventLogger (>= v3.1.27.0)

6.1.3.3.1 Clear

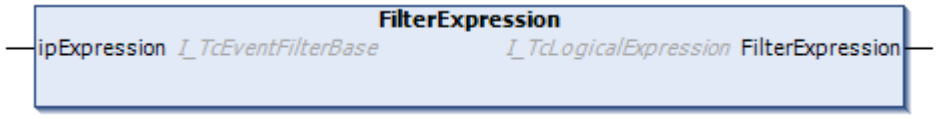


Die Methode löscht den bisherige Filterausdruck.

Rückgabewerte

Name	Typ	Beschreibung
Clear	I_TcBinaryExpression	Verknüpfunspunkt

6.1.3.3.2 FilterExpression



Die Methode gibt eine unterlagerte Filterdefinition an.

Eingänge

Name	Typ	Beschreibung
ipExpression	I_TcEventFilterBase	Verknüpfunspunkt

Rückgabewerte

Name	Typ	Beschreibung
FilterExpression	I_TcLogicalExpression	Verknüpfunspunkt

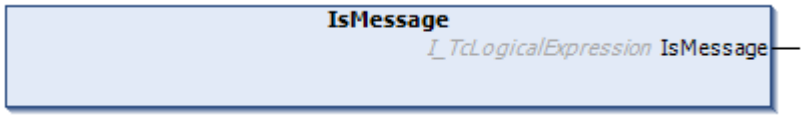
6.1.3.3.3 IsAlarm

Die Methode überprüft, ob es sich um einen Alarm handelt.

Rückgabewerte

Name	Typ	Beschreibung
IsAlarm	I_TcLogicalExpression	Verknüpfunspunkt

6.1.3.3.4 IsMessage

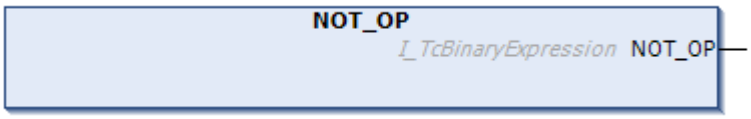


Die Methode überprüft, ob es sich um eine Nachricht handelt.

Rückgabewerte

Name	Typ	Beschreibung
IsMessage	I_TcLogicalExpression	Verknüpfunspunkt

6.1.3.3.5 NOT_OP



Die Methode negiert die folgende Aussage.

Rückgabewerte

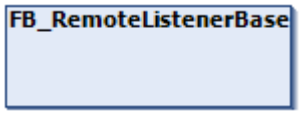
Name	Typ	Beschreibung
NOT_OP	I_TcBinaryExpression	Verknüpfunspunkt

6.1.4 RemoteEventLogger

Der Eventlogger bietet die Möglichkeit auch auf entfernte Systeme zuzugreifen, wofür hier die nötigen Funktionsbausteine beschrieben sind.
Zusätzlich besitzen Funktionsbausteine aus dem Bereich Asynchrone Textanfragen [▶ 39] eine Option um auch Texte auf einem entfernten System abzufragen.

Die Verwendung ist im Bereich Remote Zugriff auf den Eventlogger mittels PLC [▶ 36] beschrieben.

6.1.4.1 FB_RemoteListenerBase



Der Funktionsbaustein dient als Basisimplementierung eines Ereignisbeobachters eines entfernten Systems. Durch das Überschreiben der ereignisgesteuerten Methoden können neue Nachrichten und Zustandsänderungen von Alarmen erkannt werden.

Dieser Funktionsbaustein bietet den Zugriff auf den Eventlogger eines entfernten Systems und kann verwendet werden, um dort Ereignisse abzusenden oder von dort zu empfangen.

Syntax

```
FUNCTION_BLOCK FB_RemoteListenerBase IMPLEMENTS I_RemoteListener
```

Methoden

Name	Beschreibung
Execute [▶ 74]	Muss zyklisch aufgerufen werden, damit die Ereignis-Queue abgearbeitet werden kann.
Subscribe [▶ 77]	Meldet Benachrichtigungen an.
Unsubscribe [▶ 78]	Meldet Benachrichtigungen ab.

Ereignisgesteuerte Methoden

Name	Beschreibung
OnAlarmCleared [▶ 74]	Wird aufgerufen, wenn der Zustand eines Alarms von „Raised“ nach „Clear“ wechselt.
OnAlarmConfirmed [▶ 74]	Wird aufgerufen, wenn ein Alarm bestätigt wurde.
OnAlarmDisposed [▶ 75]	Wird aufgerufen, wenn eine Alarminstanz wieder frei gegeben wurde.
OnAlarmRaised [▶ 75]	Wird aufgerufen, wenn der Zustand eines Alarms von „Clear“ nach „Raised“ wechselt.
OnConnectionStateChanged [▶ 75]	Wird aufgerufen, wenn die Verbindung zum entfernten System ihren Zustand ändert.
OnDatabaseChanged [▶ 76]	Wird aufgerufen, wenn die Datenbank sich geändert hat, Z.B. wenn ein ClearLoggedEvents() aufgerufen wurde
OnEventLoggerError [▶ 76]	Wird aufgerufen, wenn bei der Kommunikation ein ADS-Fehler auftritt.
OnMessageSent [▶ 77]	Wird aufgerufen, wenn eine Nachricht abgeschickt wurde.

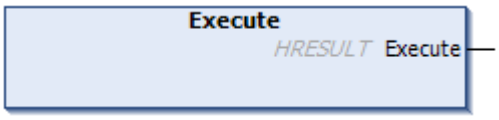
Eigenschaften

Name	Typ	Zugriff	Beschreibung
bSubscribed	BOOL	Get	Liefert TRUE, wenn der Funktionsbaustein Benachrichtigungen angemeldet hat und die Ereignisbeobachtung aktiv ist.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.1.4026.0	PC oder CX (x64, x86, Arm®)	Tc3_EventLogger (>=3.3.7.0)

6.1.4.1.1 Execute



Die Methode muss zyklisch aufgerufen werden, damit die Ereignis-Queue abgearbeitet werden kann.

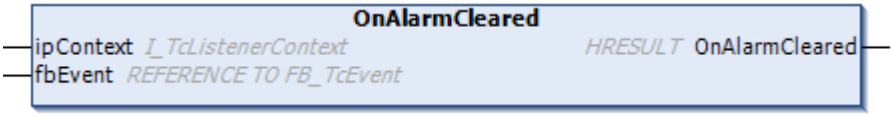
Syntax

```
METHOD Execute : HRESULT
```

Rückgabewert

Name	Typ	Beschreibung
Execute	HRESULT	Liefert S_OK, wenn der Methodenaufruf erfolgreich war, ansonsten ein HRESULT als Fehlercode.

6.1.4.1.2 OnAlarmCleared



Diese Methode wird aufgerufen, wenn der Zustand eines Alarms von Raised nach Clear wechselt.

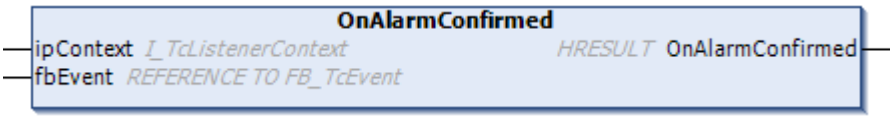
Syntax

```
METHOD OnAlarmCleared : HRESULT
VAR_INPUT
    ipContext         : I_TcListenerContext;
    fbEvent           : REFERENCE TO FB_TcEvent;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
ipContext	I_TcListenerContext	Context der unterschiedlichen Eventlogger
fbEvent	REFERENCE TO FB_TcEvent	Referenz auf den aufgetretenen Alarm. Diese Referenz darf nicht z. B. per Zuweisung kopiert werden.

6.1.4.1.3 OnAlarmConfirmed



Diese Methode wird aufgerufen, wenn ein Alarm bestätigt wurde.

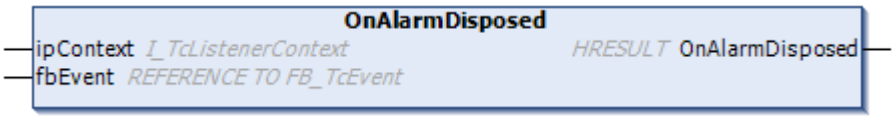
Syntax

```
METHOD OnAlarmConfirmed : HRESULT
VAR_INPUT
    ipContext           : I_TcListenerContext;
    fbEvent             : REFERENCE TO FB_TcEvent;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
ipContext	I_TcListenerContext	Context der unterschiedlichen Eventlogger
fbEvent	REFERENCE TO FB_TcEvent	Referenz auf den aufgetretenen Alarm. Diese Referenz darf nicht z. B. per Zuweisung kopiert werden.

6.1.4.1.4 OnAlarmDisposed



Diese Methode wird aufgerufen, wenn eine Alarminstanz wieder frei gegeben wurde.

Syntax

```
METHOD OnAlarmDisposed : HRESULT
VAR_INPUT
    ipContext          : I_TcListenerContext;
    fbEvent            : REFERENCE TO FB_TcEvent;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
ipContext	I_TcListenerContext	Context der unterschiedlichen Eventlogger
fbEvent	REFERENCE TO FB_TcEvent	Referenz auf den aufgetretenen Alarm. Diese Referenz darf nicht z. B. per Zuweisung kopiert werden.

6.1.4.1.5 OnAlarmRaised

Diese Methode wird aufgerufen, wenn der Zustand eines Alarms von Clear nach Raised wechselt.

Syntax

```
METHOD OnAlarmRaised : HRESULT
VAR_INPUT
    ipContext        : I_TcListenerContext;
    fbEvent          : REFERENCE TO FB_TcEvent;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
ipContext	I_TcListenerContext	Context der unterschiedlichen Eventlogger
fbEvent	REFERENCE TO FB_TcEvent	Referenz auf den aufgetretenen Alarm. Diese Referenz darf nicht z. B. per Zuweisung kopiert werden.

6.1.4.1.6 OnConnectionStateChanged

Diese Methode wird aufgerufen, wenn die Verbindung zum entfernten System ihren Zustand ändert.

Syntax

```
METHOD OnConnectionStateChanged : HRESULT
VAR_INPUT
    ipContext                   : I_TcListenerContext;
    eReason                     : TcRemoteConnectionChangeReason;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
ipContext	I_TcListenerContext	Context der unterschiedlichen Eventlogger
eReason	TcRemoteConnectionChange Reason	Die Art der Änderung des Verbindungsstatus.

6.1.4.1.7 OnDatabaseChanged

Diese Methode wird aufgerufen, wenn sich die Datenbank geändert hat, z. B. wenn ein ClearLoggedEvents() aufgerufen wurde.

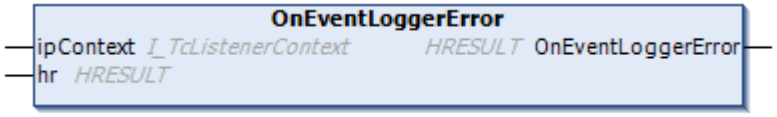
Syntax

```
METHOD OnDatabaseChanged : HRESULT
VAR_INPUT
    ipContext            : I_TcListenerContext;
    eReason              : TcDatabaseChangeReason;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
ipContext	I_TcListenerContext	Context der unterschiedlichen Eventlogger
eReason	TcDatabaseChangeReason	Einer der Werte des ENUM: Undefined, CleardByCommand, MaximumSizeReached, Deleted

6.1.4.1.8 OnEventLoggerError



Diese Methode wird aufgerufen, wenn bei der Kommunikation ein ADS-Fehler auftritt.

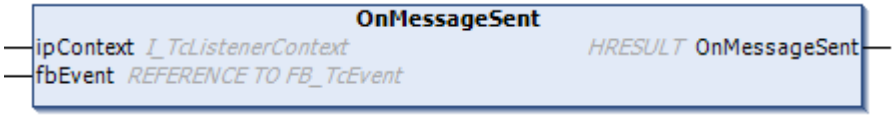
Syntax

```
METHOD OnEventLoggerError : HRESULT
VAR_INPUT
    ipContext            : I_TcListenerContext;
    hr                   : HRESULT;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
ipContext	I_TcListenerContext	Context der unterschiedlichen Eventlogger
hr	HRESULT	Der aufgetretene Fehlercode.

6.1.4.1.9 OnMessageSent



Diese Methode wird aufgerufen, wenn eine Nachricht gesendet wurde.

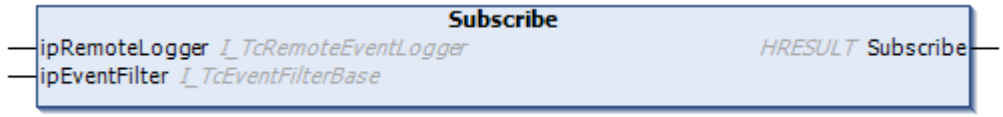
Syntax

```
METHOD OnMessageSent : HRESULT
VAR_INPUT
    ipContext        : I_TcListenerContext;
    fbEvent          : REFERENCE TO FB_TcEvent;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
ipContext	I_TcListenerContext	Context der unterschiedlichen Eventlogger
fbEvent	REFERENCE TO FB_TcEvent	Referenz auf den aufgetretenen Alarm. Diese Referenz darf nicht z. B. per Zuweisung kopiert werden.

6.1.4.1.10 Subscribe



Die Methode meldet Benachrichtigungen an.

Syntax

```
METHOD Subscribe   : HRESULT
VAR_INPUT
    ipRemoteLogger : I_TcRemoteEventLogger;
    ipEventFilter  : I_TcEventFilterBase; // optional
END_VAR
```

Eingänge

Name	Typ	Beschreibung
ipRemoteLogger	I_TcRemoteEventLogger	Referenz auf den Eventlogger, bei dem eine Benachrichtigung angemeldet werden soll.
ipEventFilter	I_TcEventFilterBase	Zeiger auf eine Instanz von FB_TcEventFilter [▶ 69], falls ein Filter aktiviert werden soll.

Rückgabewert

Name	Typ	Beschreibung
Subscribe	HRESULT	Liefert S_OK, wenn der Methodenaufruf erfolgreich war. Liefert ADS_E_EXISTS, wenn der Beobachter bereits registriert ist. Liefert ansonsten ein HRESULT als Fehlercode

6.1.4.1.11 Unsubscribe

Mit dieser Methode wird der Beobachter abgemeldet.

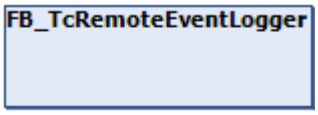
Syntax

```
METHOD Unsubscribe : HRESULT
```

Rückgabewert

Name	Typ	Beschreibung
Unsubscribe	HRESULT	Liefert S_OK, wenn der Methodenaufruf erfolgreich war. Liefert ADS_E_NOTFOUND, wenn der Beobachter nicht registriert war. Liefert ansonsten ein HRESULT als Fehlercode.

6.1.4.2 FB_TcRemoteEventLogger



Dieser Funktionsbaustein stellt den TwinCAT 3 EventLogger für ein entferntes System dar.

Syntax

```
FUNCTION_BLOCK FB_RemoteEventLogger IMPLEMENTS I_RemoteEventLogger
```

Methoden

Name	Beschreibung
ClearAlarm [▶ 79]s	Löscht aktive Alarme.
ClearLoggedEvents [▶ 80]	Löscht protokollierte Ereignisse.
ConfirmAlarms [▶ 81]	Bestätigt Alarme.
Connect [▶ 81]	Verbindet sich mit dem entfernten System.
Disconnect [▶ 82]	Löst die Verbindung mit dem entfernten System auf.
SendMessage [▶ 82]	Sendet eine Nachricht.
SendMessage2 [▶ 83]	Sendet eine Nachricht.
SendMessageEx [▶ 84]	Sendet eine Nachricht.
SendMessageEx2 [▶ 84]	Sendet eine Nachricht.

Eigenschaften

Name	Typ	Zugriff	Beschreibung
hrConnect	HRESULT	Get	Verbindungsstatus zum entfernten System
hrInit	HRESULT	Get	Fehler, die während des Aufstarten auftreten.
sNetId	T_AmsNetId	Get	AmsNetId des entfernten Systems

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.1.4026.0	PC oder CX (x64, x86, Arm®)	Tc3_EventLogger (>=3.3.7.0)

6.1.4.2.1 ClearAlarms

Methode zum Löschen aktiver Alarme. Gibt S_OK zurück, wenn erfolgreich.

Syntax

```
METHOD ClearAlarms     : HRESULT
VAR_INPUT
    nTimeStamp         : ULINT := 0;
    bResetConfirmation : BOOL := FALSE;
    ipFilter           : I_TcEventFilter;
END_VAR
```

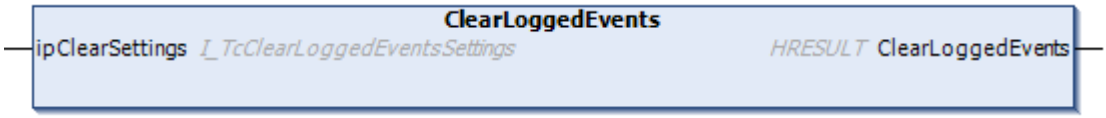
Eingänge

Name	Typ	Beschreibung
nTimeStamp	ULINT	0 setzen, um die aktuelle Zeit automatisch zu erhalten. Initial: 0
bResetConfirmation	BOOL	Wenn TRUE und der Bestätigungsstatus WaitForConfirmation ist, wird der Bestätigungsstatus auf Reset gesetzt. Andernfalls wird der Bestätigungsstatus nicht geändert. Initial: FALSE
ipFilter	I_TcEventFilter	Angeben, welche Alarme gelöscht werden sollen, ansonsten werden alle ausgelösten Alarme gelöscht.

Rückgabewert

Name	Typ	Beschreibung
ClearAlarms	HRESULT	Liefert S_OK, wenn der Methodenaufruf erfolgreich war, ansonsten ein HRESULT als Fehlercode.

6.1.4.2.2 ClearLoggedEvents



Async-Methode zum Löschen von protokollierten Ereignissen. Gibt TRUE zurück, wenn die asynchrone Anfrage nicht mehr belegt ist.

Syntax

```
METHOD ClearLoggedEvents : HRESULT
VAR_INPUT
    ipClearSettings      : I_TcClearLoggedEventsSettings;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
ipClearSettings	I_TcClearLoggedEventsSettings	Optional (Andernfalls wird der ganze Cache geleert.)

Rückgabewert

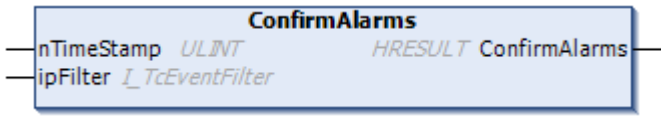
Name	Typ	Beschreibung
ClearLoggedEvents	HRESULT	Liefert S_PENDING solange die Anfrage noch nicht abgeschlossen ist. Liefert S_OK, wenn der Methodenaufruf erfolgreich war, ansonsten ein HRESULT als Fehlercode.

Verwendung der Methode

Diese Methode ist asynchron, d. h. sie benötigt mehrere Zyklen, um den Rückgabewert zu liefern. Folgende Verwendung ist vorgesehen:

```
IF bClearLoggedEvents THEN
   bClearLoggedEvents:= NOT (fbLogger.ClearLoggedEvents(0) = S_OK);
END_IF
```

6.1.4.2.3 ConfirmAlarms



Die Methode bestätigt Alarme.

Syntax

```
METHOD ConfirmAlarms : HRESULT
VAR_INPUT
    nTimeStamp       : ULINT := 0;
    ipFilter         : I_TcEventFilter;
END_VAR
```

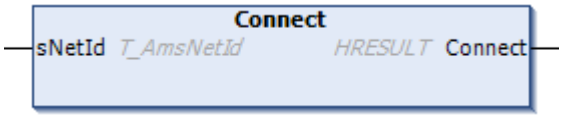
Eingänge

Name	Typ	Beschreibung
nTimeStamp	ULINT	0 setzen, um die aktuelle Zeit automatisch zu erhalten. Initial: 0
ipFilter	I_TcEventFilter	Festlegen, welche Alarme bestätigt werden sollen, ansonsten werden alle Alarme mit dem Bestätigungsstatus WaitForConfirmation bestätigt.

Rückgabewert

Name	Typ	Beschreibung
ConfirmAlarms	HRESULT	Liefert S_OK, wenn der Methodenaufruf erfolgreich war, ansonsten ein HRESULT als Fehlercode.

6.1.4.2.4 Connect



Die Methode verbindet sich mit dem entfernten System.

Syntax

```
METHOD Connect : HRESULT
VAR_INPUT
    sNetId     : T_AmsNetId; (* AmsNetId of the remote TwinCAT system.*)
END_VAR
```

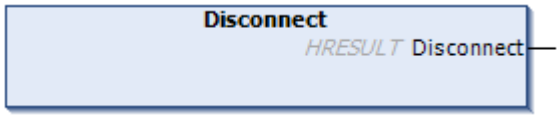
Eingänge

Name	Typ	Beschreibung
sNetId	T_AmsNetId	AmsNetId des entfernten Systems

Rückgabewert

Name	Typ	Beschreibung
Connect	HRESULT	Liefert S_OK, wenn der Methodenaufruf erfolgreich war, ansonsten ein HRESULT als Fehlercode.

6.1.4.2.5 Disconnect



Die Methode löst die Verbindung mit dem entfernten System auf.

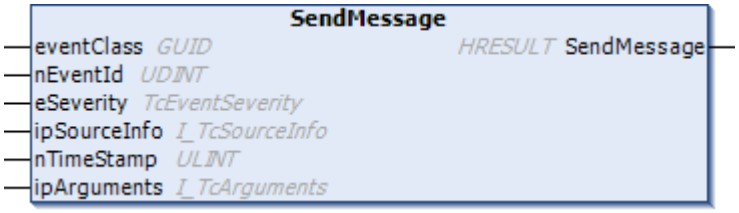
Syntax

```
METHOD Disconnect : HRESULT
```

Rückgabewert

Name	Typ	Beschreibung
Disconnect	HRESULT	Liefert S_OK, wenn der Methodenaufruf erfolgreich war, ansonsten ein HRESULT als Fehlercode.

6.1.4.2.6 SendMessage



Diese Methode sendet eine Nachricht.

Syntax

```
METHOD SendMessage : HRESULT
VAR_INPUT
    eventClass     : GUID;
    nEventId       : UDINT;
    eSeverity      : TcEventSeverity;
    ipSourceInfo   : I_TcSourceInfo := 0; // optional
    nTimeStamp     : ULINT := 0; // set 0 to get the current time automatically
    ipArguments    : I_TcArguments := 0; // optional
END_VAR
```

Eingänge

Name	Typ	Beschreibung
eventClass	GUID	GUID der Ereignisklasse
nEventId	UDINT	ID des Ereignisses
eSeverity	TcEventSeverity	Severity des Ereignisses
ipSourceInfo	I_TcSourceInfo	Schnittstellenzeiger auf die Quellinformationen. Diese Angabe ist optional. Hier kann eine Instanz vom Typ FB_TcSourceInfo [▶ 121] angegeben werden. Wird nichts (NULL) übergeben, wird eine Standardquellinformation generiert.
nTimeStamp	ULINT	0: Aktueller Zeitstempel wird verwendet. > 0: Externer Zeitstempel in 100 Nanosekunden seit dem 1. Januar 1601 (UTC)
ipArguments	I_TcArguments	Schnittstellenzeiger auf Argumente des Ereignisses. Diese Angabe ist optional. Hier kann eine Instanz vom Typ FB_TcArguments [▶ 96] angegeben werden.

Rückgabewert

Name	Typ	Beschreibung
SendMessage	HRESULT	Liefert S_OK, wenn der Methodenaufruf erfolgreich war, ansonsten ein HRESULT als Fehlercode.

6.1.4.2.7 SendMessage2

Diese Methode sendet eine Nachricht.

Syntax

```
METHOD SendMessage2 : HRESULT
VAR_INPUT
    eventClass      : GUID;
    nEventId        : UDINT;
    eSeverity       : TcEventSeverity;
    ipSourceInfo    : I_TcSourceInfo := 0; // optional
    nTimeStamp      : ULINT := 0; // set 0 to get the current time automatically
    ipArguments     : I_TcArguments := 0; // optional
END_VAR
```

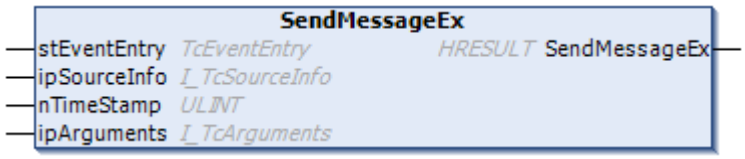
Eingänge

Name	Typ	Beschreibung
eventClass	GUID	GUID der Ereignisklasse
nEventId	UDINT	ID des Ereignisses.
eSeverity	TcEventSeverity	Severity des Ereignisses
ipSourceInfo	I_TcSourceInfo	Schnittstellenzeiger auf die Quellinformationen. Diese Angabe ist optional. Hier kann eine Instanz vom Typ FB_TcSourceInfo [▶ 121] angegeben werden. Wird nichts (NULL) übergeben, wird eine Standardquellinformation generiert.
nTimeStamp	ULINT	0: Aktueller Zeitstempel wird verwendet. > 0: Externer Zeitstempel in 100 Nanosekunden seit dem 1. Januar 1601 (UTC)
ipArguments	I_TcArguments	Schnittstellenzeiger auf Argumente des Ereignisses. Diese Angabe ist optional. Hier kann eine Instanz vom Typ FB_TcArguments [▶ 96] angegeben werden.
sJsonAttribute	STRING	String als Json Repräsentation, der zusätzlich zum Ereignistext übertragen wird.

Rückgabewert

Name	Typ	Beschreibung
SendMessage2	HRESULT	Liefert S_OK, wenn der Methodenaufruf erfolgreich war, ansonsten ein HRESULT als Fehlercode.

6.1.4.2.8 SendMessageEx



Diese Methode sendet eine Nachricht.

Syntax

```
METHOD SendMessageEx : HRESULT
VAR_INPUT
    stEventEntry     : TcEventEntry;
    ipSourceInfo     : I_TcSourceInfo := 0; // optional
    nTimeStamp       : ULINT := 0; // set 0 to get the current time automatically
    ipArguments      : I_TcArguments := 0; // optional
END_VAR
```

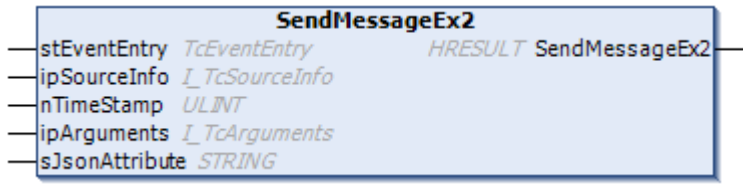
Eingänge

Name	Typ	Beschreibung
stEventEntry	TcEventEntry	Ereignisdefinition
ipSourceInfo	I_TcSourceInfo	Schnittstellenzeiger auf die Quellinformationen. Diese Angabe ist optional. Hier kann eine Instanz vom Typ FB_TcSourceInfo [▶ 121] angegeben werden. Wird nichts (NULL) übergeben, wird eine Standardquellinformation generiert.
nTimeStamp	ULINT	0: Aktueller Zeitstempel wird verwendet. > 0: Externer Zeitstempel in 100 Nanosekunden seit dem 1. Januar 1601 (UTC)
ipArguments	I_TcArguments	Schnittstellenzeiger auf Argumente des Ereignisses. Diese Angabe ist optional. Hier kann eine Instanz vom Typ FB_TcArguments [▶ 96] angegeben werden.

Rückgabewert

Name	Typ	Beschreibung
SendMessageEx	HRESULT	Liefert S_OK, wenn der Methodenaufruf erfolgreich war, ansonsten ein HRESULT als Fehlercode.

6.1.4.2.9 SendMessageEx2



Diese Methode sendet eine Nachricht.

Syntax

```
METHOD SendMessageEx : HRESULT
VAR_INPUT
    stEventEntry     : TcEventEntry;
    ipSourceInfo     : I_TcSourceInfo := 0; // optional
    nTimeStamp       : ULINT := 0; // set 0 to get the current time automatically
    ipArguments      : I_TcArguments := 0; // optional
END_VAR
```

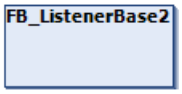
Eingänge

Name	Typ	Beschreibung
stEventEntry	TcEventEntry	Ereignisdefinition
ipSourceInfo	I_TcSourceInfo	Schnittstellenzeiger auf die Quellinformationen. Diese Angabe ist optional. Hier kann eine Instanz vom Typ FB_TcSourceInfo [▶ 121] angegeben werden. Wird nichts (NULL) übergeben, wird eine Standardquellinformation generiert.
nTimeStamp	ULINT	0: Aktueller Zeitstempel wird verwendet. > 0: Externer Zeitstempel in 100 Nanosekunden seit dem 1. Januar 1601 (UTC)
ipArguments	I_TcArguments	Schnittstellenzeiger auf Argumente des Ereignisses. Diese Angabe ist optional. Hier kann eine Instanz vom Typ FB_TcArguments [▶ 96] angegeben werden.
sJsonAttribute	STRING	String als Json Repräsentation, der zusätzlich zum Ereignistext übertragen wird.

Rückgabewert

Name	Typ	Beschreibung
SendMessageEx2	HRESULT	Liefert S_OK, wenn der Methodenaufruf erfolgreich war, ansonsten ein HRESULT als Fehlercode.

6.1.5 FB_ListenerBase2



Der Funktionsbaustein dient als Basisimplementierung eines Ereignisbeobachters.

Durch das Überschreiben der ereignisgesteuerten Methoden können neue Nachrichten und Zustandsänderungen von Alarmen erkannt werden.

Syntax

Definition:

```
FUNCTION_BLOCK FB_ListenerBase2 IMPLEMENTS I_Listener2
```

Methoden

Name	Definitionsort	Beschreibung
Execute [▶ 86]	Lokal	Muss zyklisch aufgerufen werden, damit die Ereignis-Queue abgearbeitet werden kann.
Subscribe [▶ 88]	Lokal	Meldet Benachrichtigungen an.
Unsubscribe [▶ 89]	Lokal	Meldet Benachrichtigungen ab.

Ereignisgesteuerte Methoden (Callback-Methoden)

Name	Definitionsort	Beschreibung
OnAlarmCleared [▶ 86]	I_Listener2	Wird aufgerufen, wenn der Zustand eines Alarms von „Raised“ nach „Clear“ wechselt.
OnAlarmConfirmed [▶ 87]	I_Listener2	Wird aufgerufen, wenn ein Alarm bestätigt wurde.
OnAlarmDisposed [▶ 87]	I_Listener2	Wird aufgerufen, wenn eine Alarminstanz wieder frei gegeben wurde.
OnAlarmRaised [▶ 88]	I_Listener2	Wird aufgerufen, wenn der Zustand eines Alarms von „Clear“ nach „Raised“ wechselt.
OnMessageSent [▶ 88]	I_Listener2	Wird aufgerufen, wenn eine Nachricht abgeschickt wurde.

Eigenschaften

Name	Typ	Zugriff	Definitionsort	Beschreibung
bSubscribed	BOOL	Get	Lokal	Liefert TRUE, wenn der Funktionsbaustein Benachrichtigungen angemeldet hat und die Ereignisbeobachtung aktiv ist.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.1.4024.17	PC oder CX (x64, x86, Arm®)	Tc3_EventLogger (>= v3.1.27.0)

6.1.5.1 Execute

Diese Methode muss zyklisch aufgerufen werden, damit die Ereignis-Queue abgearbeitet werden kann.

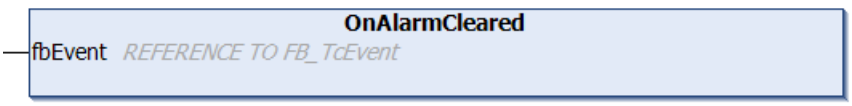
Syntax

```
METHOD Execute : HRESULT
```

Rückgabewert

Name	Typ	Beschreibung
Execute	HRESULT	Liefert S_OK, wenn der Methodenaufruf erfolgreich war, ansonsten ein HRESULT als Fehlercode

6.1.5.2 OnAlarmCleared



Diese Methode wird aufgerufen, wenn der Zustand eines Alarms von Raised nach Clear wechselt.

Syntax

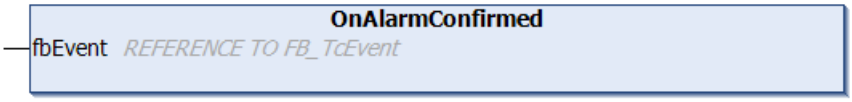
```
METHOD OnAlarmCleared : HRESULT
VAR_INPUT
    fbEvent : REFERENCE TO FB_TcEvent;
END_VAR
```

Liefert die Implementierung der Callback-Methode einen Returncode <> S_OK zurück, so werden weitere Callback-Aufrufe bis zur nächsten Ausführung pausiert.

Eingänge

Name	Typ	Beschreibung
fbEvent	REFERENCE TO FB_TcEvent [▶ 98]	Referenz auf den aufgetretenen Alarm. Diese Referenz darf nicht z.B. per Zuweisung kopiert werden.

6.1.5.3 OnAlarmConfirmed



Diese Methode wird aufgerufen, wenn ein Alarm bestätigt wurde.

Syntax

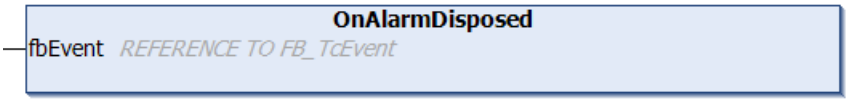
```
METHOD OnAlarmConfirmed : HRESULT
VAR_INPUT
    fbEvent : REFERENCE TO FB_TcEvent;
END_VAR
```

Liefert die Implementierung der Callback-Methode einen Returncode <> S_OK zurück, so werden weitere Callback-Aufrufe bis zur nächsten Ausführung pausiert.

Eingänge

Name	Typ	Beschreibung
fbEvent	REFERENCE TO FB_TcEvent [▶ 98]	Referenz auf den aufgetretenen Alarm. Diese Referenz darf nicht z.B. per Zuweisung kopiert werden.

6.1.5.4 OnAlarmDisposed



Diese Methode wird aufgerufen, wenn eine Alarminstanz wieder frei gegeben wurde.

Syntax

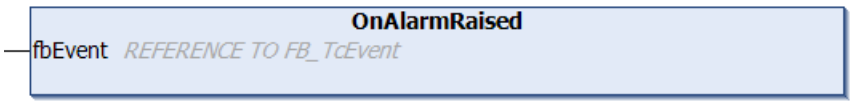
```
METHOD OnAlarmConfirmed : HRESULT
VAR_INPUT
    fbEvent : REFERENCE TO FB_TcEvent;
END_VAR
```

Liefert die Implementierung der Callback-Methode einen Returncode <> S_OK zurück, so werden weitere Callback-Aufrufe bis zur nächsten Ausführung pausiert.

Eingänge

Name	Typ	Beschreibung
fbEvent	REFERENCE TO FB_TcEvent [▶ 98]	Referenz auf den aufgetretenen Alarm. Diese Referenz darf nicht z.B. per Zuweisung kopiert werden.

6.1.5.5 OnAlarmRaised



Diese Methode wird aufgerufen, wenn der Zustand eines Alarms von Clear nach Raised wechselt.

Syntax

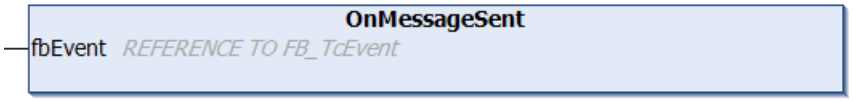
```
METHOD OnAlarmRaised : HRESULT
VAR_INPUT
    fbEvent : REFERENCE TO FB_TcEvent;
END_VAR
```

Liefert die Implementierung der Callback-Methode einen Returncode <> S_OK zurück, so werden weitere Callback-Aufrufe bis zur nächsten Ausführung pausiert.

Eingänge

Name	Typ	Beschreibung
fbEvent	REFERENCE TO FB_TcEvent [▶ 98]	Referenz auf den aufgetretenen Alarm. Diese Referenz darf nicht z.B. per Zuweisung kopiert werden.

6.1.5.6 OnMessageSent



Diese Methode wird aufgerufen, wenn eine Nachricht gesendet wurde.

Syntax

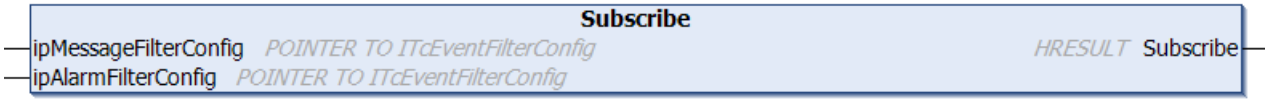
```
METHOD OnMessageSent : HRESULT
VAR_INPUT
    fbEvent : REFERENCE TO FB_TcEvent;
END_VAR
```

Liefert die Implementierung der Callback-Methode einen Returncode <> S_OK zurück, so werden weitere Callback-Aufrufe bis zur nächsten Ausführung pausiert.

Eingänge

Name	Typ	Beschreibung
fbEvent	REFERENCE TO FB_TcEvent [▶ 98]	Referenz auf das aufgetretene Ereignis. Diese Referenz darf nicht z.B. per Zuweisung kopiert werden.

6.1.5.7 Subscribe



Mit dieser Methode wird der Beobachter für Benachrichtigungen angemeldet.

Syntax

```
METHOD Subscribe : HRESULT
VAR_INPUT
    ipMessageFilterConfig : POINTER TO ITcEventFilterConfig;
    ipAlarmFilterConfig   : POINTER TO ITcEventFilterConfig;
END_VAR
```

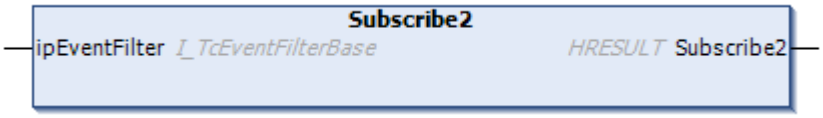
Eingänge

Name	Typ	Beschreibung
ipMessageFilterConfig	POINTER TO ITcEventFilterConfig	Zeiger auf ITcEventFilterConfig, wenn ein Filter aktiviert werden soll.
ipAlarmFilterConfig	POINTER TO ITcEventFilterConfig	Zeiger auf ITcEventFilterConfig, wenn ein Filter aktiviert werden soll.

Rückgabewert

Name	Typ	Beschreibung
Subscribe	HRESULT	Liefert S_OK, wenn der Methodenaufruf erfolgreich war. Liefert ADS_E_EXISTS, wenn der Beobachter bereits registriert ist. Liefert ansonsten ein HRESULT als Fehlercode.

6.1.5.8 Subscribe2



Diese Methode meldet Benachrichtigungen an.

Syntax

```
METHOD Subscribe2 : HRESULT
```

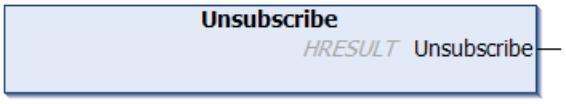
Eingang

Name	Typ	Beschreibung
ipEventFilter	I_TcEventFilterBase	Zeiger auf eine Instanz von FB_TcEventFilter [▶ 69], falls ein Filter aktiviert werden soll.

Rückgabewert

Name	Typ	Beschreibung
Subscribe2	HRESULT	Liefert bei Erfolg S_OK.

6.1.5.9 Unsubscribe



Mit dieser Methode wird der Beobachter abgemeldet.

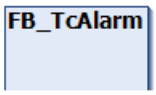
Syntax

```
METHOD Unsubscribe : HRESULT
```

Rückgabewert

Name	Typ	Beschreibung
Unsubscribe	HRESULT	Liefert S_OK, wenn der Methodenaufruf erfolgreich war. Liefert ADS_E_NOTFOUND, wenn der Beobachter nicht registriert war. Liefert ansonsten ein HRESULT als Fehlercode.

6.1.6 FB_TcAlarm



Dieser Funktionsbaustein repräsentiert einen Alarm des TwinCAT 3 EventLogger.

Syntax

Definition:

```
FUNCTION_BLOCK FB_TcAlarm EXTENDS FB_TcEventBase
```

Vererbungshierarchie

FB_TcEventBase [▶ 100]

FB_TcAlarm

Methoden

Name	Definitionsort	Beschreibung
EqualsTo [▶ 101]	Geerbt von FB_TcEventBase [▶ 100]	Vergleicht das Ereignis mit einer anderen Instanz.
EqualsToEventClass [▶ 102]	Geerbt von FB_TcEventBase [▶ 100]	Vergleicht die Ereignisklasse des Ereignisses mit einer anderen Ereignisklasse.
EqualsToEventEntry [▶ 102]	Geerbt von FB_TcEventBase [▶ 100]	Vergleicht die Ereignisklasse, die Event-ID und die Severity des Ereignisses mit denen eines anderen Ereignisses.
EqualsToEventEntryEx [▶ 103]	Geerbt von FB_TcEventBase [▶ 100]	Vergleicht die Ereignisdefinition des Ereignisses mit einer anderen Ereignisdefinition.
GetJsonAttribute [▶ 103]	Geerbt von FB_TcEventBase [▶ 100]	Liefert das Json-Attribut.
Release [▶ 104]	Geerbt von FB_TcEventBase [▶ 100]	Gibt die vom EventLogger erstellte Instanz wieder frei.
RequestEventClassName [▶ 104]	Geerbt von FB_TcEventBase [▶ 100]	Fragt den Namen der Ereignisklasse an.
RequestEventText [▶ 105]	Geerbt von FB_TcEventBase [▶ 100]	Liefert den Text zum Ereignis.
Clear [▶ 92]	Lokal	Setzt den Alarmzustand auf „Not-Raised“.
Confirm [▶ 93]	Lokal	Bestätigt den Alarm.
Create [▶ 94]	Lokal	Erstellt eine Alarminstanz im EventLogger.
CreateEx [▶ 94]	Lokal	Erstellt aus eine Ereignisdefinition eine Alarminstanz im EventLogger.
Raise [▶ 95]	Lokal	Setzt den Alarmzustand auf „Raised“.
SetJsonAttribute [▶ 96]	Lokal	Setzt das Json-Attribut.

Eigenschaften

Name	Typ	Zugriff	Definitionsort	Beschreibung
eSeverity	TcEventSeverity [▶ 140]	Get	Geerbt von FB_TcEventBase [▶ 100]	Liefert die Severity.
EventClass	GUID	Get	Geerbt von FB_TcEventBase [▶ 100]	Liefert die GUID der Ereignisklasse.
ipArguments [▶ 106]	I_TcArguments [▶ 123]	Get	Geerbt von FB_TcEventBase [▶ 100]	Liefert den Schnittstellenzeiger für die Argumente.
ipSourceInfo [▶ 106]	I_TcSourceInfo [▶ 139]	Get	Geerbt von FB_TcEventBase [▶ 100]	Als Standardverhalten wird die SourceInfo intern generiert. Sie beinhaltet dann den Symbolnamen des Funktionsbausteins, der FB_TcMessage instanziiert, als SourceName sowie die Objekt-ID der SPS-Instanz als SourceID. Wenn die Instanz von FB_TcMessage mit dem Attribut 'hide' versteckt wird, kann intern kein Symbolname für das Standardverhalten generiert werden.
nEventId	nEventId	Get	Geerbt von FB_TcEventBase [▶ 100]	Liefert die ID des Ereignisses.
stEventEntry	TcEventEntry [▶ 139]	Get	Geerbt von FB_TcEventBase [▶ 100]	Liefert die Ereignisdefinition.
bRaised	BOOL	Get	Lokal	Liefert TRUE, wenn der Alarm im Zustand „Raised“ ist.
bActive	BOOL	Get	Lokal	Liefert TRUE, wenn der Alarm im Zustand „Raised“ oder „Wait For Confirmation“ ist.
eConfirmationState	TcEventConfirmationState [▶ 140]	Get	Lokal	Liefert den Bestätigungszustand.
nTimeCleared	ULINT	Get	Lokal	Liefert den Zeitpunkt des Clear.
nTimeConfirmed	ULINT	Get	Lokal	Liefert den Zeitpunkt des Confirm.
nTimeRaised	ULINT	Get	Lokal	Liefert den Zeitpunkt des Raise.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.1.4022.20	PC oder CX (x64, x86, Arm®)	Tc3_EventLogger

6.1.6.1 Clear

Diese Methode setzt den Alarmzustand [▶ 14] auf Not-Raised.

Syntax

```
METHOD Clear : HRESULT
VAR_INPUT
    nTimeStamp         : ULINT;
    bResetConfirmation : BOOL;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
nTimeStamp	ULINT	0: Aktueller Zeitstempel wird verwendet > 0: Externer Zeitstempel in 100 Nanosekunden seit dem 1. Januar 1601 (UTC).
bResetConfirmation	BOOL	Wenn TRUE und der Bestätigungszustand WaitForConfirmation ist, wird der Bestätigungszustand auf Reset gesetzt. Ansonsten wird der Bestätigungszustand nicht verändert.

Rückgabewert

Name	Typ	Beschreibung
Clear	HRESULT	Liefert S_OK, wenn der Methodenaufruf erfolgreich war. Liefert ADS_E_INVALIDSTATE, wenn der Alarm nicht im Zustand Raised war. Liefert ansonsten ein HRESULT als Fehlercode.

6.1.6.2 Confirm

Diese Methode setzt den Bestätigungszustand [▶ 14] von WaitForConfirmation auf Confirmed.

Syntax

```
METHOD Confirm : HRESULT
VAR_INPUT
    nTimeStamp: ULINT;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
nTimeStamp	ULINT	0: Aktueller Zeitstempel wird verwendet. > 0: Externer Zeitstempel in 100 Nanosekunden seit dem 1. Januar 1601 (UTC).

Rückgabewert

Name	Typ	Beschreibung
Confirm	HRESULT	Liefert S_OK, wenn der Methodenaufruf erfolgreich war. Liefert ADS_E_INVALIDSTATE, wenn der Bestätigungszustand nicht WaitConfirmation war. Liefert ansonsten ein HRESULT als Fehlercode.

6.1.6.3 Create

Diese Methode erstellt eine Alarminstanz im EventLogger.

Syntax

```
METHOD Create : HRESULT
    eventClass        : GUID;
    nEventId          : UDINT;
    eSeverity         : TcEventSeverity;
    bWithConfirmation : BOOL;
    ipSourceInfo      : I_TcSourceInfo;
END_VAR
```

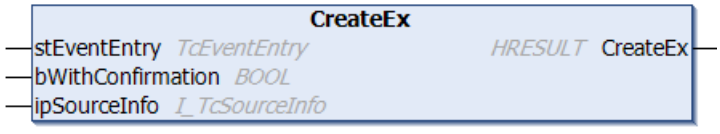
Eingänge

Name	Typ	Beschreibung
eventClass	GUID	GUID der Ereignisklasse.
nEventId	UDINT	ID des Ereignisses.
eSeverity	TcEventSeverity [▶ 140]	Severity des Ereignisses.
bWithConfirmation	BOOL	Legt fest, ob der Alarm bestätigungspflichtig ist.
ipSourceInfo	I_TcSourceInfo [▶ 139]	Schnittstellenzeiger auf die Quellinformationen. Diese Angabe ist optional. Hier kann eine Instanz vom Typ FB_TcSourceInfo [▶ 121] angegeben werden. Wird nichts (NULL) übergeben, wird eine Standardquellinformation generiert.

Rückgabewert

Name	Typ	Beschreibung
Create	HRESULT	Liefert S_OK, wenn ein neuer Alarm erfolgreich erstellt werden konnte. Liefert ERROR_ALREADY_EXISTS, wenn der Alarm bereits existiert hat. Liefert ansonsten ein HRESULT als Fehlercode

6.1.6.4 CreateEx



Diese Methode erstellt eine Alarminstanz im EventLogger.

Syntax

```
METHOD CreateEx : HRESULT
VAR_INPUT
    stEventEntry      : TcEventEntry;
    bWithConfirmation : BOOL;
    ipSourceInfo      : I_TcSourceInfo;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
stEventEntry	TcEventEntry [▶ 139]	Ereignisdefinition.
bWithConfirmation	BOOL	Legt fest, ob der Alarm bestätigungspflichtig ist.
ipSourceInfo	I_TcSourceInfo [▶ 139]	Schnittstellenzeiger auf die Quellinformationen. Diese Angabe ist optional. Hier kann eine Instanz vom Typ FB_TcSourceInfo [▶ 121] angegeben werden. Wird nichts (NULL) übergeben, wird eine Standardquellinformation generiert.

Rückgabewert

Name	Typ	Beschreibung
CreateEx	HRESULT	Liefert S_OK, wenn ein neuer Alarm erfolgreich erstellt werden konnte. Liefert ERROR_ALREADY_EXISTS, wenn der Alarm bereits existiert hat. Liefert ansonsten ein HRESULT als Fehlercode.

6.1.6.5 Raise

Die Methode setzt den Alarmzustand [▶ 14] auf Raised.

Wenn der Alarm bestätigungspflichtig ist, wird zusätzlich der Bestätigungszustand auf WaitForConfirmation gesetzt.

Syntax

```
METHOD Raise : HRESULT
VAR_INPUT
    nTimeStamp : ULINT;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
nTimeStamp	ULINT	0: Aktueller Zeitstempel wird verwendet. > 0: Externer Zeitstempel in 100 Nanosekunden seit dem 1. Januar 1601 (UTC).

Rückgabewert

Name	Typ	Beschreibung
Raise	HRESULT	Liefert S_OK, wenn der Methodenaufruf erfolgreich war. Liefert ADS_E_INVALIDSTATE, wenn der Alarm bereits im Zustand Raised war. Liefert ansonsten ein HRESULT als Fehlercode.

6.1.6.6 SetJsonAttribute

Diese Methode setzt das JSON-Attribut.

Syntax

```
METHOD SetJsonAttribute : HRESULT
VAR_IN_OUT CONSTANT
    sJsonAttribute : STRING;
END_VAR
```

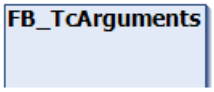
Eingänge

Name	Typ	Beschreibung
sJsonAttribute	STRING	JSON-String

Rückgabewert

Name	Typ	Beschreibung
SetJsonAttribute	HRESULT	Liefert S_OK, wenn der Methodenaufruf erfolgreich war, ansonsten ein HRESULT als Fehlercode.

6.1.7 FB_TcArguments



Mit diesem Funktionsbaustein können Argumente eines Ereignisses definiert werden. Er implementiert dafür die I_TcArguments-Schnittstelle.

Syntax

Definition:

```
FUNCTION_BLOCK FB_TcArguments IMPLEMENTS I_TcArguments
```

Schnittstellen

Typ	Beschreibung
I_TcArguments [▶ 123]	Definiert das Argumenten-Handling.

Methoden

Name	Definitionsort	Beschreibung
AddBlob [▶ 124]	I_TcArguments [▶ 123]	Fügt Binärdaten als Argument hinzu.
AddBool [▶ 125]	I_TcArguments [▶ 123]	Fügt ein Argument vom Typ BOOL hinzu.
AddByte [▶ 125]	I_TcArguments [▶ 123]	Fügt ein Argument vom Typ BYTE hinzu.
AddDint [▶ 126]	I_TcArguments [▶ 123]	Fügt ein Argument vom Typ DINT hinzu.
AddDWord [▶ 126]	I_TcArguments [▶ 123]	Fügt ein Argument vom Typ DWORD hinzu.
AddEventReferenceId [▶ 126]	I_TcArguments [▶ 123]	Fügt eine Referenz zu einem anderen Ereignis als Argument hinzu.
AddEventReferenceIdGuid [▶ 127]	I_TcArguments [▶ 123]	Fügt eine Referenz zu einem anderen Ereignis als Argument hinzu.
AddInt [▶ 127]	I_TcArguments [▶ 123]	Fügt ein Argument vom Typ INT hinzu.
AddLInt [▶ 128]	I_TcArguments [▶ 123]	Fügt ein Argument vom Typ LINT hinzu.
AddLReal [▶ 128]	I_TcArguments [▶ 123]	Fügt ein Argument vom Typ LREAL hinzu.
AddReal [▶ 129]	I_TcArguments [▶ 123]	Fügt ein Argument vom Typ REAL hinzu.
AddSInt [▶ 129]	I_TcArguments [▶ 123]	Fügt ein Argument vom Typ SINT hinzu.
AddString [▶ 130]	I_TcArguments [▶ 123]	Fügt ein Argument vom Typ STRING hinzu.
AddUDint [▶ 130]	I_TcArguments [▶ 123]	Fügt ein Argument vom Typ UDINT hinzu.
AddUInt [▶ 130]	I_TcArguments [▶ 123]	Fügt ein Argument vom Typ INT hinzu.
AddULInt [▶ 131]	I_TcArguments [▶ 123]	Fügt ein Argument vom Typ ULINT hinzu.
AddUSInt [▶ 131]	I_TcArguments [▶ 123]	Fügt ein Argument vom Typ USINT hinzu.
AddWord [▶ 132]	I_TcArguments [▶ 123]	Fügt ein Argument vom Typ WORD hinzu.
AddWString [▶ 132]	I_TcArguments [▶ 123]	Fügt ein Argument vom Typ WSTRING hinzu.
Clear [▶ 133]	I_TcArguments [▶ 123]	Entfernt alle Argumente.
IsEmpty [▶ 98]	Lokal	Prüft, ob Argumente hinzugefügt wurden.

Eigenschaften

Name	Typ	Zugriff	Beschreibung
nCount	UDINT	Get	Liefert die Anzahl der übergebenen Argumente.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.1.4022.20	PC oder CX (x64, x86, Arm®)	Tc3_EventLogger

6.1.7.1 IsEmpty

Diese Methode prüft, ob Argumente hinzugefügt wurden.

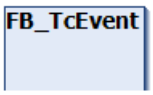
Syntax

```
METHOD IsEmpty : BOOL
```

Rückgabewert

Name	Typ	Beschreibung
IsEmpty	BOOL	Liefert TRUE, wenn keine Argumente hinzugefügt wurden.

6.1.8 FB_TcEvent



Dieser Funktionsbaustein stellt nur Lese-Methoden und -Eigenschaften zu einem Ereignis bereit.

Syntax

Definition:

```
FUNCTION_BLOCK FB_TcEvent EXTENDS FB_TcEventBase IMPLEMENTS I_TcEventBase
```

Vererbungshierarchie

FB_TcEventBase [▶ 100]

FB_TcEvent

Schnittstellen

Typ	Beschreibung
I_TcEventBase [▶ 133]	Basisschnittstelle, die Methoden und Eigenschaften eines Ereignisses definiert.

Methoden

Name	Definitionsort	Beschreibung
EqualsTo [▶ 101]	Geerbt von FB_TcEventBase [▶ 100]	Vergleicht das Ereignis mit einer anderen Instanz.
EqualsToEventClass [▶ 102]	Geerbt von FB_TcEventBase [▶ 100]	Vergleicht die Ereignisklasse des Ereignisses mit einer anderen Ereignisklasse.
EqualsToEventEntry [▶ 102]	Geerbt von FB_TcEventBase [▶ 100]	Vergleicht die Ereignisdefinition des Ereignisses mit einer anderen Ereignisdefinition.
EqualsToEventEntryEx [▶ 103]	Geerbt von FB_TcEventBase [▶ 100]	Vergleicht die Ereignisdefinition des Ereignisses mit einer anderen Ereignisdefinition.
GetJsonAttribute [▶ 103]	Geerbt von FB_TcEventBase [▶ 100]	Liefert das Json-Attribut.
Release [▶ 104]	Geerbt von FB_TcEventBase [▶ 100]	Gibt die vom EventLogger erstellte Instanz wieder frei.
RequestEventClassName [▶ 104]	Geerbt von FB_TcEventBase [▶ 100]	Fragt den Namen der Ereignisklasse an.
RequestEventText [▶ 105]	Geerbt von FB_TcEventBase [▶ 100]	Liefert den Text zum Ereignis.

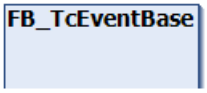
Eigenschaften

Name	Typ	Zugriff	Definitionsort	Beschreibung
eSeverity	TcEventSeverity [▶ 140]	Get	Geerbt von FB_TcEventBase [▶ 100]	Liefert die Severity.
EventClass	GUID	Get	Geerbt von FB_TcEventBase [▶ 100]	Liefert die GUID der Ereignisklasse.
ipArguments [▶ 106]	I_TcArguments [▶ 123]	Get	Geerbt von FB_TcEventBase [▶ 100]	Liefert den Schnittstellenzeiger für die Argumente.
ipSourceInfo [▶ 106]	I_TcSourceInfo [▶ 139]	Get	Geerbt von FB_TcEventBase [▶ 100]	Als Standardverhalten wird die SourceInfo intern generiert. Sie beinhaltet dann den Symbolnamen von dem Funktionsbaustein, der FB_TcMessage instanziiert, als SourceName sowie die Objekt-ID der SPS-Instanz als SourceID. Wenn die Instanz von FB_TcMessage mit dem Attribut 'hide' versteckt wird, kann intern kein Symbolname für das Standardverhalten generiert werden.
nEventId	nEventId	Get	Geerbt von FB_TcEventBase [▶ 100]	Liefert die ID des Ereignisses.
stEventEntry	TcEventEntry [▶ 139]	Get	Geerbt von FB_TcEventBase [▶ 100]	Liefert die Ereignisdefinition.
nTimestamp	ULINT	Get	Lokal	Liefert den Zeitpunkt.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.1.4022.20	PC oder CX (x64, x86, Arm®)	Tc3_EventLogger

6.1.9 FB_TcEventBase



Dieser Funktionsbaustein beinhaltet die Basisimplementierung.

Syntax

Definition:

```
FUNCTION_BLOCK FB_TcEventBase
```

Methoden

Name	Definitionsort	Beschreibung
EqualsTo [▶ 101]	Lokal	Vergleicht das Ereignis mit einer anderen Instanz.
EqualsToEventClass [▶ 102]	Lokal	Vergleicht die Ereignisklasse des Ereignisses mit einer anderen Ereignisklasse.
EqualsToEventEntry [▶ 102]	Lokal	Vergleicht die Ereignisdefinition des Ereignisses mit einer anderen Ereignisdefinition.
EqualsToEventEntryEx [▶ 103]	Lokal	Vergleicht die Ereignisdefinition des Ereignisses mit einer anderen Ereignisdefinition.
GetJsonAttribute [▶ 103]	Lokal	Liefert das Json-Attribut.
Release [▶ 104]	Lokal	Gibt die vom EventLogger erstellte Instanz wieder frei.
RequestEventClassName [▶ 104]	Lokal	Fragt den Namen der Ereignisklasse an.
RequestEventText [▶ 105]	Lokal	Liefert den Text zum Ereignis.

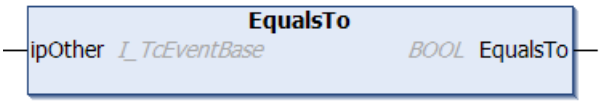
Eigenschaften

Name	Typ	Zugriff	Beschreibung
eSeverity	TcEventSeverity [▶ 140]	Get	Liefert die Severity.
EventClass	GUID	Get	Liefert die GUID der Ereignisklasse.
ipArguments [▶ 106]	I_TcArguments [▶ 123]	Get	Liefert den Schnittstellenzeiger für die Argumente.
ipSourceInfo [▶ 106]	I_TcSourceInfo [▶ 139]	Get	Als Standardverhalten wird die SourceInfo intern generiert. Sie beinhaltet dann den Symbolnamen von dem Funktionsbaustein, der FB_TcMessage instanziiert, als SourceName sowie die Objekt-ID der SPS Instanz als SourceID. Falls die Instanz von FB_TcMessage mit dem Attribut 'hide' versteckt wird, kann intern kein Symbolname für das Standardverhalten generiert werden.
nEventId	UDINT	Get	Liefert die ID des Ereignisses
nUniqueId	UDINT	Get	Liefert die Unique-ID des Ereignisses
stEventEntry	TcEventEntry [▶ 139]	Get	Liefert die Ereignisdefinition.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.1.4022.20	PC oder CX (x64, x86, Arm®)	Tc3_EventLogger

6.1.9.1 EqualsTo



Diese Methode führt einen Vergleich mit einem am Eingang angegebenen anderen Ereignis aus.

Syntax

```
METHOD EqualsTo : BOOL
VAR_INPUT
    ipOther : I_TcEventBase;
END_VAR
```

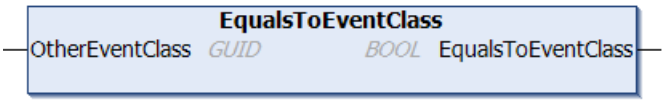
Eingänge

Name	Typ	Beschreibung
ipOther	I_TcEventBase [▶ 133]	Zu vergleichendes Ereignis

Rückgabewert

Name	Typ	Beschreibung
EqualsTo	BOOL	Liefert TRUE, wenn die Ereignisse übereinstimmen.

6.1.9.2 EqualsToEventClass



Diese Methode führt einen Vergleich mit einer am Eingang angegebenen anderen Ereignisklasse aus.

Syntax

```
METHOD EqualsToEventClass : BOOL
VAR_INPUT
    OtherEventClass : GUID;
END_VAR
```

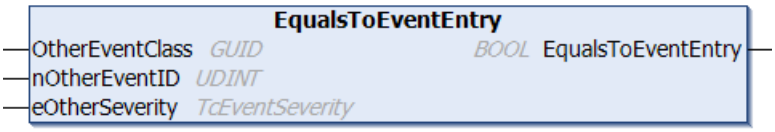
Eingänge

Name	Typ	Beschreibung
OtherEventClass	GUID	Zu vergleichende Ereignisklasse.

Rückgabewert

Name	Typ	Beschreibung
EqualsToEventClass	BOOL	Liefert TRUE, wenn die Ereignisklassen übereinstimmen.

6.1.9.3 EqualsToEventEntry



Diese Methode führt einen Vergleich mit einem am Eingang angegebenen anderen Ereignis aus.

Syntax

```
METHOD EqualsToEventEntry : BOOL
VAR_INPUT
    OtherEventClass : GUID;
    nOtherEventID   : UDINT;
    eOtherSeverity  : TcEventSeverity;
END_VAR
```

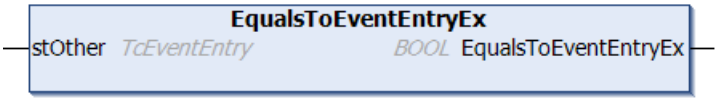
Eingänge

Name	Typ	Beschreibung
OtherEventClass	GUID	Ereignisklasse des zu vergleichenden Ereignisses.
nOtherEventID	UDINT	Event-ID des zu vergleichenden Ereignisses.
eOtherSeverity	TcEventSeverity [▶ 140]	Event-Severity des zu vergleichenden Ereignisses.

Rückgabewert

Name	Typ	Beschreibung
EqualsToEventEntry	BOOL	Liefert TRUE, wenn die Ereignisse übereinstimmen.

6.1.9.4 EqualsToEventEntryEx



Diese Methode führt einen Vergleich mit einem am Eingang angegebenen anderen Ereignis aus.

Syntax

```
METHOD EqualsToEventEntryEx : BOOL
VAR_INPUT
    stOther : TcEventEntry;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
stOther	TcEventEntry [▶ 139]	Zu vergleichendes Ereignis.

Rückgabewert

Name	Typ	Beschreibung
EqualsToEventEntryEx	BOOL	Liefert TRUE, wenn die Ereignisse übereinstimmen.

6.1.9.5 GetJsonAttribute

Diese Methode liefert das JSON-Attribut.

Syntax

```
METHOD GetJsonAttribute : HRESULT
VAR_INPUT
    sJsonAttribute : REFERENCE TO STRING;
    nJsonAttribute : UDINT;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
sJsonAttribute	REFERENCE TO STRING	Referenz auf eine Variable vom Typ String
nJsonAttribute	UDINT	Länge der String-Variable

Rückgabewert

Name	Typ	Beschreibung
GetJsonAttribute	HRESULT	Liefert S_OK, wenn der Methodenaufruf erfolgreich war. Liefert ERROR_BAD_LENGTH, wenn die Länge der Variable zu klein ist. Ansonsten wird HRESULT als Fehlercode zurückgegeben.

6.1.9.6 Release

Diese Methode gibt die vom Eventlogger erstellte Instanz wieder frei.

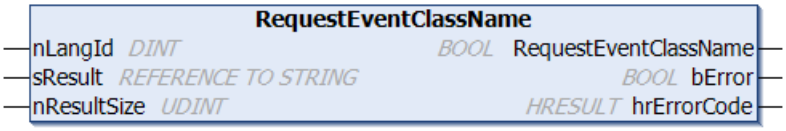
Syntax

```
METHOD Release : HRESULT
```

Rückgabewert

Name	Typ	Beschreibung
Release	HRESULT	Liefert S_OK, wenn der Methodenaufruf erfolgreich war, ansonsten ein HRESULT als Fehlercode.

6.1.9.7 RequestEventClassName



Diese Methode liefert den Namen der Ereignisklasse.

Syntax

```
METHOD RequestEventClassName : BOOL
VAR_INPUT
    nLangId     : DINT;
    sResult     : REFERENCE TO STRING;
    nResultSize : UDINT;
END_VAR
VAR_OUTPUT
    bError      : BOOL;
    hrErrorCode : HRESULT;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
nLangId	DINT	Spezifiziert die Sprach-ID Englisch (en-US) = 1033 Deutsch (de-DE) = 1031 …
sResult	REFERENCE TO STRING	Referenz auf eine Variable vom Typ String
nResultSize	UDINT	Größe der String-Variable in Bytes

Rückgabewert

Name	Typ	Beschreibung
RequestEventClassName	BOOL	Liefert TRUE, sobald die Abfrage beendet wurde. Wenn die asynchrone Abfrage noch aktiv ist, liefert der Rückgabewert FALSE. Die Methode muss so lange aufgerufen werden, bis der Rückgabewert TRUE ist.

Ausgänge

Name	Typ	Beschreibung
bError	BOOL	Liefert FALSE, wenn der Methodenaufruf erfolgreich war. Liefert TRUE, wenn ein Fehler aufgetreten ist.
hrErrorCode	HRESULT	Liefert S_OK, wenn der Methodenaufruf erfolgreich war. Im Falle eines Fehlers wird ein Fehlercode ausgegeben.

6.1.9.8 RequestEventText

Diese Methode liefert den Ereignistext.

Syntax

```
METHOD RequestEventText : BOOL
VAR_INPUT
    nLangId     : DINT;
    sResult     : REFERENCE TO STRING;
    nResultSize : UDINT;
END_VAR
VAR_OUTPUT
    bError      : BOOL;
    hrErrorCode : HRESULT;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
nLangId	DINT	Spezifiziert die Sprach-ID Englisch (en-US) = 1033 Deutsch (de-DE) = 1031 …
sResult	REFERENCE TO STRING	Referenz auf eine Variable vom Typ String
nResultSize	UDINT	Größe der String-Variable in Bytes

Rückgabewert

Name	Typ	Beschreibung
RequestEventText	BOOL	Liefert TRUE, sobald die Abfrage beendet wurde. Wenn die asynchrone Abfrage noch aktiv ist, liefert der Rückgabewert FALSE. Die Methode muss so lange aufgerufen werden, bis der Rückgabewert TRUE ist.

Ausgänge

Name	Typ	Beschreibung
bError	BOOL	Liefert FALSE, wenn der Methodenaufruf erfolgreich war. Liefert TRUE, wenn ein Fehler aufgetreten ist.
hrErrorCode	HRESULT	Liefert S_OK, wenn der Methodenaufruf erfolgreich war. Im Falle eines Fehlers wird ein Fehlercode ausgegeben.

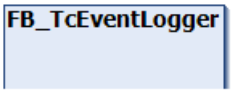
6.1.9.9 ipArguments

```
PROPERTY PUBLIC ipArguments : I_TcArguments
```

6.1.9.10 ipSourceInfo

```
PROPERTY ipSourceInfo : I_TcSourceInfo
```

6.1.10 FB_TcEventLogger



Dieser Funktionsbaustein stellt den TwinCAT 3 EventLogger selbst dar.

Syntax

Definition:

```
FUNCTION_BLOCK FB_TcEventLogger
```

Methoden

Name	Beschreibung
ClearAlarms [▶ 107]	Löscht aktive Alarme.
ClearAllAlarms [▶ 107]	Ruft Clear() für alle Alarme im Zustand Raised auf.
ClearLoggedEvents [▶ 108]	Löscht protokollierte Ereignisse.
ConfirmAlarms [▶ 109]	Bestätigt Alarme.
ConfirmAllAlarms [▶ 109]	Ruft Confirm() für alle Alarme mit dem Bestätigungszustand WaitForConfirmation auf.
ExportLoggedEvents [▶ 110]	Exportiert protokollierte Ereignisse.
GetAlarm [▶ 111]	Liefert einen Zeiger auf einen existierenden Alarm.
GetAlarmEx [▶ 111]	Liefert einen Zeiger auf einen existierenden Alarm.
IsAlarmRaised [▶ 112]	Fragt ab, ob ein Alarm im Zustand Raised ist.
IsAlarmRaisedEx [▶ 112]	Fragt ab, ob ein Alarm im Zustand Raised ist.
SendMessage [▶ 113]	Sendet eine Nachricht.
SendMessage2 [▶ 114]	Sendet eine Nachricht.
SendMessageEx [▶ 115]	Sendet eine Nachricht.
SendMessageEx2 [▶ 116]	Sendet eine Nachricht.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.1.4022.20	PC oder CX (x64, x86, Arm®)	Tc3_EventLogger

6.1.10.1 ClearAlarms

Methode zum Löschen aktiver Alarme. Gibt `S_OK` zurück, wenn erfolgreich.

Syntax

```
METHOD ClearAlarms     : HRESULT
VAR_INPUT
    nTimeStamp         : ULINT := 0;
    bResetConfirmation : BOOL  := FALSE;
    ipFilter           : I_TcEventFilter;
END_VAR
```

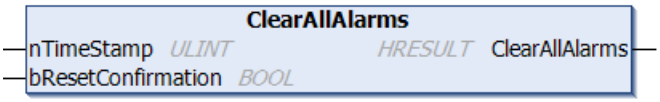
Eingänge

Name	Typ	Beschreibung
nTimeStamp	ULINT	0 setzen, um die aktuelle Zeit automatisch zu erhalten. Initial: 0
bResetConfirmation	BOOL	Wenn TRUE und der Bestätigungsstatus WaitForConfirmation ist, wird der Bestätigungsstatus auf Reset gesetzt. Andernfalls wird der Bestätigungsstatus nicht geändert. Initial: FALSE
ipFilter	I_TcEventFilter	Angeben, welche Alarme gelöscht werden sollen, ansonsten werden alle ausgelösten Alarme gelöscht.

Rückgabewerte

Name	Typ	Beschreibung
ClearAlarms	HRESULT	Liefert S_OK, wenn der Methodenaufruf erfolgreich war, ansonsten ein HRESULT als Fehlercode.

6.1.10.2 ClearAllAlarms



Diese Methode ruft für alle Alarme im Alarmzustand Raised deren Methode Clear() auf.

Syntax

```
METHOD ClearAllAlarms : HRESULT
VAR_INPUT
    nTimeStamp         : ULINT := 0;
    bResetConfirmation : BOOL := FALSE;
END_VAR
```

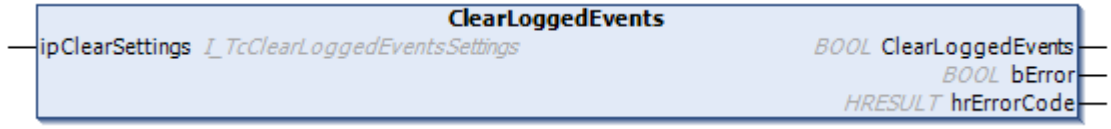
Eingänge

Name	Typ	Beschreibung
nTimeStamp	ULINT	0: Aktueller Zeitstempel wird verwendet > 0: Externer Zeitstempel in 100 Nanosekunden seit dem 1. Januar 1601 (UTC).
bResetConfirmation	BOOL	Wenn TRUE und der Bestätigungszustand WaitForConfirmation ist, wird der Bestätigungszustand auf Reset gesetzt. Ansonsten wird der Bestätigungszustand nicht verändert.

Rückgabewert

Name	Typ	Beschreibung
ClearAllAlarms	HRESULT	Liefert S_OK, wenn der Methodenaufruf erfolgreich war, ansonsten ein HRESULT als Fehlercode

6.1.10.3 ClearLoggedEvents



Async-Methode zum Löschen von protokollierten Ereignissen. Gibt TRUE zurück, wenn die asynchrone Anfrage nicht mehr belegt ist.

Syntax

```
METHOD ClearLoggedEvents : BOOL
VAR_INPUT
    ipClearSettings      : I_TcClearLoggedEventsSettings;
END_VAR
VAR_OUTPUT
    bError               : BOOL;
    hrErrorCode          : HRESULT;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
ipClearSettings	I_TcClearLoggedEventsSettings	Optional (Andernfalls wird der ganze Cache geleert.)

Rückgabewerte

Name	Typ	Beschreibung
ClearLoggedEvents	BOOL	Liefert TRUE, sobald die Anfrage abgearbeitet ist.

Ausgänge

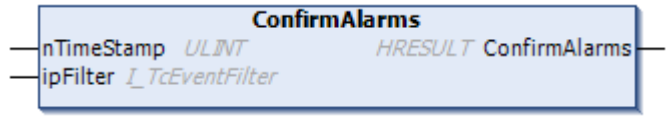
Name	Typ	Beschreibung
bError	BOOL	Liefert TRUE, falls die Abfrage fehlerhaft ist.
hrErrorCode	HRESULT	Liefert einen Fehlercode, falls die Abfrage fehlerhaft ist.

Beispiel für die Verwendung der Methode

Diese Methode ist asynchron, d. h., sie benötigt mehrere Zyklen, um den Rückgabewert zu liefern. Folgende Verwendung ist vorgesehen:

```
IF bClearLoggedEvents THEN
   bClearLoggedEvents:= NOT fbLogger.ClearLoggedEvents(0);
END_IF
```

6.1.10.4 ConfirmAlarms



Diese Methode bestätigt Alarme.

Syntax

```
METHOD ConfirmAlarms : HRESULT
VAR_INPUT
    nTimeStamp       : ULINT := 0;
    ipFilter         : I_TcEventFilter;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
nTimeStamp	ULINT	0 setzen, um die aktuelle Zeit automatisch zu erhalten. Initial: 0
ipFilter	I_TcEventFilter	Festlegen, welche Alarme bestätigt werden sollen, ansonsten werden alle Alarme mit dem Bestätigungsstatus WaitForConfirmation bestätigt.

Rückgabewerte

Name	Typ	Beschreibung
ConfirmAlarms	HRESULT	Liefert S_OK, wenn der Methodenaufruf erfolgreich war, ansonsten ein HRESULT als Fehlercode.

6.1.10.5 ConfirmAllAlarms

Diese Methode ruft für alle Alarme mit dem Bestätigungszustand WaitForConfirmation deren Methode Confirm() auf.

Syntax

```
METHOD ConfirmAllAlarms : HRESULT
VAR_INPUT
    nTimeStamp : ULINT := 0;
END_VAR
```

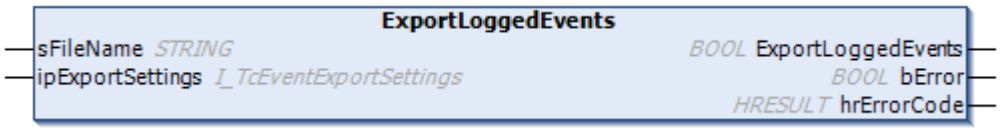
Eingänge

Name	Typ	Beschreibung
nTimeStamp	ULINT	0: Aktueller Zeitstempel wird verwendet > 0: Externer Zeitstempel in 100 Nanosekunden seit dem 1. Januar 1601 (UTC).

Rückgabewert

Name	Typ	Beschreibung
ConfirmAllAlarms	HRESULT	Liefert S_OK, wenn der Methodenaufruf erfolgreich war, ansonsten ein HRESULT als Fehlercode.

6.1.10.6 ExportLoggedEvents



Diese Methode exportiert protokollierte Ereignisse asynchron. Gibt TRUE zurück, wenn die asynchrone Abarbeitung abgeschlossen ist.

Syntax

```
METHOD ExportLoggedEvents : BOOL
VAR_IN_OUT CONSTANT
    sFileName             : STRING;
END_VAR
VAR_INPUT
ipExportSettings          : I_TcEventExportSettings;
END_VAR
VAR_OUTPUT
    bError                : BOOL;
    hrErrorCode           : HRESULT;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
ipExportSettings	I_TcEventExportSettings	Angeben, welche Ereignisse exportiert werden sollen, ansonsten werden alle Ereignisse exportiert. Hierfür kann eine Instanz von FB_TcEventCsvExportSettings [▶ 69] zugewiesen werden.

Ein-/Ausgänge

Name	Typ	Beschreibung
sFileName	STRING	Name der Zieldatei

Rückgabewerte

Name	Typ	Beschreibung
ExportLoggedEvents	BOOL	TRUE, wenn die Abarbeitung abgeschlossen ist.

Ausgänge

Name	Typ	Beschreibung
bError	BOOL	TRUE, sobald ein Fehler eintritt.
hrErrorCode	HRESULT	Gibt die Fehlerinformation aus, wenn `bError` TRUE ist.

Beispiel für die Verwendung der Methode

Diese Methode ist asynchron, d. h., sie benötigt mehrere Zyklen, um den Rückgabewert zu liefern. Folgende Verwendung ist vorgesehen:

```
IF bExportLoggedEvents THEN
  bExportLoggedEvents:= NOT fbLogger.ExportLoggedEvents(…);
END_IF
```

6.1.10.7 GetAlarm

Diese Methode liefert einen Schnittstellenzeiger auf eine existierende Instanz.

Syntax

```
METHOD GetAlarm : HRESULT
VAR_INPUT
    eventClass   : GUID;
    nEventId     : UDINT;
    ipSourceInfo : I_TcSourceInfo := 0;
    fbAlarm      : REFERENCE TO FB_TcAlarm;
END_VAR
```

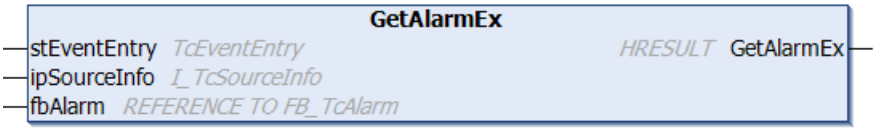
Eingänge

Name	Typ	Beschreibung
eventClass	GUID	GUID der Ereignisklasse.
nEventId	UDINT	ID des Ereignisses
ipSourceInfo	I_TcSourceInfo [▶ 139]	Zeiger auf eine ITcSourceInfo-Schnittstelle.
fbAlarm	REFERENCE TO FB_TcAlarm [▶ 90]	Zeiger auf einen Alarm.

Rückgabewert

Name	Typ	Beschreibung
GetAlarm	HRESULT	Liefet ADS_E_NOTFOUND, wenn keine Instanz gefunden wurde. Liefert S_OK, wenn der Methodenaufruf erfolgreich war, ansonsten ein HRESULT als Fehlercode.

6.1.10.8 GetAlarmEx



Diese Methode liefert einen Schnittstellenzeiger auf eine existierende Instanz.

Syntax

```
METHOD GetAlarmEx : HRESULT
VAR_INPUT
    stEventEntry : TcEventEntry;
    ipSourceInfo : I_TcSourceInfo := 0; // optional
    fbAlarm      : REFERENCE TO FB_TcAlarm;
END_VAR
```

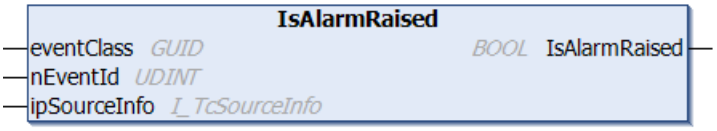
Eingänge

Name	Typ	Beschreibung
stEventEntry	TcEventEntry [▶ 139]	Ereignisdefinition
ipSourceInfo	I_TcSourceInfo [▶ 139]	Zeiger auf eine ITcSourceInfo-Schnittstelle
fbAlarm	REFERENCE TO FB_TcAlarm [▶ 90]	Zeiger auf einen Alarm

Rückgabewert

Name	Typ	Beschreibung
GetAlarmEx	HRESULT	Liefet ADS_E_NOTFOUND, wenn keine Instanz gefunden wurde. Liefert S_OK, wenn der Methodenaufruf erfolgreich war, ansonsten ein HRESULT als Fehlercode.

6.1.10.9 IsAlarmRaised



Diese Methode fragt ab, ob ein Alarm im Zustand Raised ist.

Syntax

```
METHOD IsAlarmRaised : BOOL
VAR_INPUT
    eventClass   : GUID;
    nEventId     : UDINT;
    ipSourceInfo : I_TcSourceInfo := 0;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
eventClass	GUID	GUID der Ereignisklasse.
nEventId	UDINT	ID des Ereignisses.
ipSourceInfo	I_TcSourceInfo [▶ 139]	Zeiger auf eine ITcSourceInfo-Schnittstelle.

Rückgabewert

Name	Typ	Beschreibung
IsAlarmRaised	BOOL	Liefert TRUE, wenn der Alarm im Zustand Raised ist.

6.1.10.10 IsAlarmRaisedEx

Diese Methode fragt ab, ob ein Alarm im Zustand Raised ist.

Syntax

```
METHOD IsAlarmRaisedEx : BOOL
VAR_INPUT
    stEventEntry : TcEventEntry;
    ipSourceInfo : I_TcSourceInfo := 0;
END_VAR
```

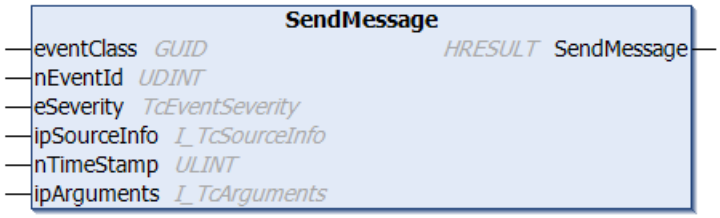
Eingänge

Name	Typ	Beschreibung
stEventEntry	UDINT	Ereignisdefinition.
ipSourceInfo	I_TcSourceInfo [▶ 139]	Zeiger auf eine ITcSourceInfo-Schnittstelle.

Rückgabewert

Name	Typ	Beschreibung
IsAlarmRaisedEx	BOOL	Liefert TRUE, wenn der Alarm im Zustand Raised ist.

6.1.10.11 SendMessage



Diese Methode sendet eine Nachricht.

Syntax

```
METHOD SendMessage : HRESULT
VAR_INPUT
    eventClass    : GUID;
    nEventId      : UDINT;
    eSeverity     : TcEventSeverity;
    ipSourceInfo  : I_TcSourceInfo := 0;
    nTimeStamp    : ULINT := 0;
    ipArguments   : I_TcArguments := 0;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
eventClass	GUID	GUID der Ereignisklasse.
nEventId	UDINT	ID des Ereignisses.
eSeverity	TcEventSeverity [▶ 140]	Severity des Ereignisses.
ipSourceInfo	I_TcSourceInfo [▶ 139]	Schnittstellenzeiger auf die Quellinformationen. Diese Angabe ist optional. Hier kann eine Instanz vom Typ FB_TcSourceInfo [▶ 121] angegeben werden. Wird nichts (NULL) übergeben, wird eine Standardquellinformation generiert.
nTimeStamp	ULINT	0: Aktueller Zeitstempel wird verwendet. > 0: Externer Zeitstempel in 100 Nanosekunden seit dem 1. Januar 1601 (UTC).
ipArguments	I_TcArguments [▶ 123]	Schnittstellenzeiger auf Argumente des Ereignisses. Diese Angabe ist optional. Hier kann eine Instanz vom Typ FB_TcArguments [▶ 96] angegeben werden.

Rückgabewert

Name	Typ	Beschreibung
SendMessage	HRESULT	Liefert S_OK, wenn der Methodenaufruf erfolgreich war, ansonsten ein HRESULT als Fehlercode.

6.1.10.12 SendMessage2

Diese Methode sendet eine Nachricht.

Syntax

```
METHOD SendMessage2 : HRESULT
VAR_INPUT
    eventClass      : GUID;
    nEventId        : UDINT;
    eSeverity       : TcEventSeverity;
    ipSourceInfo    : I_TcSourceInfo := 0;
    nTimeStamp      : ULINT := 0;
    ipArguments     : I_TcArguments := 0;
END_VAR
VAR_IN_OUT CONSTANT
    sJsonAttribute  : STRING;
END_VAR
```

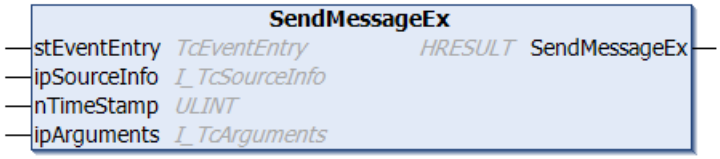
Eingänge

Name	Typ	Beschreibung
eventClass	GUID	GUID der Ereignisklasse
nEventId	UDINT	ID des Ereignisses
eSeverity	TcEventSeverity	Severity des Ereignisses
ipSourceInfo	I_TcSourceInfo	Schnittstellenzeiger auf die Quellinformationen. Diese Angabe ist optional. Hier kann eine Instanz vom Typ FB_TcSourceInfo [▶ 121] angegeben werden. Wird nichts (NULL) übergeben, wird eine Standardquellinformation generiert.
nTimeStamp	ULINT	0: Aktueller Zeitstempel wird verwendet. > 0: Externer Zeitstempel in 100 Nanosekunden seit dem 1. Januar 1601 (UTC)
ipArguments	I_TcArguments	Schnittstellenzeiger auf Argumente des Ereignisses. Diese Angabe ist optional. Hier kann eine Instanz vom Typ FB_TcArguments [▶ 96] angegeben werden.
sJsonAttribute	STRING	String als Json Repräsentation, der zusätzlich zum Ereignistext übertragen wird.

Rückgabewerte

Name	Typ	Beschreibung
SendMessage2	HRESULT	Liefert S_OK, wenn der Methodenaufruf erfolgreich war, ansonsten ein HRESULT als Fehlercode.

6.1.10.13 SendMessageEx



Diese Methode sendet eine Nachricht.

Syntax

```
METHOD SendMessageEx : HRESULT
VAR_INPUT
    stEventEntry : TcEventEntry;
    ipSourceInfo : I_TcSourceInfo := 0;
    nTimeStamp   : ULINT := 0;
    ipArguments  : I_TcArguments := 0;
END_VAR
```

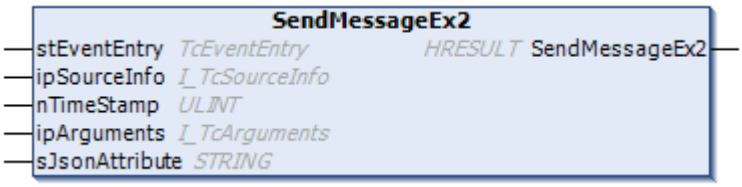
Eingänge

Name	Typ	Beschreibung
stEventEntry	TcEventEntry [▶ 139]	Ereignisdefinition.
ipSourceInfo	I_TcSourceInfo [▶ 139]	Schnittstellenzeiger auf die Quellinformationen. Diese Angabe ist optional. Hier kann eine Instanz vom Typ FB_TcSourceInfo [▶ 121] angegeben werden. Wird nichts (NULL) übergeben, wird eine Standardquellinformation generiert.
nTimeStamp	ULINT	0: Aktueller Zeitstempel wird verwendet > 0: Externer Zeitstempel in 100 Nanosekunden seit dem 1. Januar 1601 (UTC).
ipArguments	I_TcArguments [▶ 123]	Schnittstellenzeiger auf Argumente des Ereignisses. Diese Angabe ist optional. Hier kann eine Instanz vom Typ FB_TcArguments [▶ 96] angegeben werden.

Rückgabewert

Name	Typ	Beschreibung
SendMessageEx	HRESULT	Liefert S_OK, wenn der Methodenaufruf erfolgreich war, ansonsten ein HRESULT als Fehlercode.

6.1.10.14 SendMessageEx2



Diese Methode sendet eine Nachricht.

Syntax

```
METHOD SendMessageEx2 : HRESULT
VAR_INPUT
    stEventEntry      : TcEventEntry;
    ipSourceInfo      : I_TcSourceInfo := 0;
    nTimeStamp        : ULINT := 0;
    ipArguments       : I_TcArguments := 0;
END_VAR
VAR_IN_OUT CONSTANT
    sJsonAttribute    : STRING;
END_VAR
```

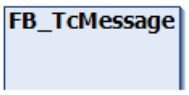
Eingänge

Name	Typ	Beschreibung
stEventEntry	TcEventEntry	Ereignisdefinition
ipSourceInfo	I_TcSourceInfo	Schnittstellenzeiger auf die Quellinformationen. Diese Angabe ist optional. Hier kann eine Instanz vom Typ FB_TcSourceInfo [▶ 121] angegeben werden. Wird nichts (NULL) übergeben, wird eine Standardquellinformation generiert.
nTimeStamp	ULINT	0: Aktueller Zeitstempel wird verwendet. > 0: Externer Zeitstempel in 100 Nanosekunden seit dem 1. Januar 1601 (UTC)
ipArguments	I_TcArguments	Schnittstellenzeiger auf Argumente des Ereignisses. Diese Angabe ist optional. Hier kann eine Instanz vom Typ FB_TcArguments [▶ 96] angegeben werden.
sJsonAttribute	STRING	String als Json Repräsentation, der zusätzlich zum Ereignistext übertragen wird.

Rückgabewerte

Name	Typ	Beschreibung
SendMessageEx2	HRESULT	Liefert S_OK, wenn der Methodenaufruf erfolgreich war, ansonsten ein HRESULT als Fehlercode.

6.1.11 FB_TcMessage



Dieser Funktionsbaustein repräsentiert eine Nachricht vom TwinCAT 3 EventLogger.

Syntax

Definition:

```
FUNCTION_BLOCK FB_TcMessage EXTENDS FB_TcEventBase IMPLEMENTS I_TcMessage
```

Vererbungshierarchie

FB_TcEventBase [▶ 100]

FB_TcMessage

Schnittstellen

Typ	Beschreibung
I_TcMessage [▶ 138]	Stellt Methoden und Eigenschaften für das Nachrichten-Handling bereit.

Methoden

Name	Definitionsort	Beschreibung
EqualsTo [▶ 101]	Geerbt von FB_TcEventBase [▶ 100]	Vergleicht das Ereignis mit einer anderen Instanz.
EqualsToEventClass [▶ 102]	Geerbt von FB_TcEventBase [▶ 100]	Vergleicht die Ereignisklasse des Ereignisses mit einer anderen Ereignisklasse.
EqualsToEventEntry [▶ 102]	Geerbt von FB_TcEventBase [▶ 100]	Vergleicht die Ereignisdefinition des Ereignisses mit einer anderen Ereignisdefinition.
EqualsToEventEntryEx [▶ 103]	Geerbt von FB_TcEventBase [▶ 100]	Vergleicht die Ereignisdefinition des Ereignisses mit einer anderen Ereignisdefinition.
GetJsonAttribute [▶ 103]	Geerbt von FB_TcEventBase [▶ 100]	Liefert das Json-Attribut.
Release [▶ 104]	Geerbt von FB_TcEventBase [▶ 100]	Gibt die vom EventLogger erstellte Instanz wieder frei.
RequestEventClassName [▶ 104]	Geerbt von FB_TcEventBase [▶ 100]	Fragt den Namen der Ereignisklasse an.
RequestEventText [▶ 105]	Geerbt von FB_TcEventBase [▶ 100]	Liefert den Text zum Ereignis.
Create [▶ 119]	Lokal	Erstellt eine Nachrichtinstanz im EventLogger.
CreateEx [▶ 120]	Lokal	Erstellt aus eine Ereignisdefinition eine Nachrichtinstanz im EventLogger.
SetJsonAttribute [▶ 120]	Lokal	Setzt das Json-Attribut.
Send [▶ 138]	I_TcMessage [▶ 138]	Sendet eine Nachricht.

Eigenschaften

Name	Typ	Zugriff	Definitionsort	Beschreibung
eSeverity	TcEventSeverity [▶ 140]	Get	Geerbt von FB_TcEventBase [▶ 100]	Liefert die Severity.
EventClass	GUID	Get	Geerbt von FB_TcEventBase [▶ 100]	Liefert die GUID der Ereignisklasse.
ipArguments [▶ 106]	I_TcArguments [▶ 123]	Get	Geerbt von FB_TcEventBase [▶ 100]	Liefert den Schnittstellenzeiger für die Argumente.
ipSourceInfo [▶ 106]	I_TcSourceInfo [▶ 139]	Get	Geerbt von FB_TcEventBase [▶ 100]	Als Standardverhalten wird die SourceInfo intern generiert. Sie beinhaltet dann den Symbolnamen von dem Funktionsbaustein, der FB_TcMessage instanziiert, als SourceName sowie die Objekt-ID der SPS-Instanz als SourceID. Wenn die Instanz von FB_TcMessage mit dem Attribut 'hide' versteckt wird, kann intern kein Symbolname für das Standardverhalten generiert werden.
nEventId	nEventId	Get	Geerbt von FB_TcEventBase [▶ 100]	Liefert die ID des Ereignisses.
stEventEntry	TcEventEntry [▶ 139]	Get	Geerbt von FB_TcEventBase [▶ 100]	Liefert die Ereignisdefinition.
nTimeSent	ULINT	Get	Lokal	Liefer den Zeitpunkt des Send.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.1.4022.20	PC oder CX (x64, x86, Arm®)	Tc3_EventLogger

6.1.11.1 Create

Diese Methode erstellt eine Nachrichtinstanz im EventLogger.

Syntax

```
METHOD Create : HRESULT
VAR_INPUT
    eventClass   : GUID;
    nEventId     : UDINT;
    eSeverity    : TcEventSeverity;
    ipSourceInfo : I_TcSourceInfo := 0;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
eventClass	GUID	GUID der Ereignisklasse.
nEventId	UDINT	ID des Ereignisses.
eSeverity	TcEventSeverity [▶ 140]	Definiert die Severity.
ipSourceInfo	I_TcSourceInfo [▶ 139]	Schnittstellenzeiger auf die Quellinformationen. Diese Angabe ist optional. Hier kann eine Instanz vom Typ FB_TcSourceInfo [▶ 121] angegeben werden. Wird nichts (NULL) übergeben, wird eine Standardquellinformation generiert.

Rückgabewert

Name	Typ	Beschreibung
Create	HRESULT	Liefert S_OK, wenn der Methodenaufruf erfolgreich war, ansonsten ein HRESULT als Fehlercode.

6.1.11.2 CreateEx

Diese Methode erstellt aus eine Ereignisdefinition eine Nachrichtinstanz im EventLogger.

Syntax

```
METHOD PUBLIC CreateEx : HRESULT
VAR_INPUT
    stEventEntry : TcEventEntry;
    ipSourceInfo : I_TcSourceInfo := 0;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
stEventEntry	TcEventEntry [▶ 139]	Ereignisdefinition.
ipSourceInfo	I_TcSourceInfo [▶ 139]	Schnittstellenzeiger auf die Quellinformationen. Diese Angabe ist optional. Hier kann eine Instanz vom Typ FB_TcSourceInfo [▶ 121] angegeben werden. Wird nichts (NULL) übergeben, wird eine Standardquellinformation generiert.

Rückgabewert

Name	Typ	Beschreibung
CreateEx	HRESULT	Liefert S_OK, wenn der Methodenaufruf erfolgreich war, ansonsten ein HRESULT als Fehlercode.

6.1.11.3 SetJsonAttribute

Diese Methode setzt das JSON-Attribut.

Syntax

```
METHOD SetJsonAttribute : HRESULT
VAR_IN_OUT CONSTANT
    sJsonAttribute : STRING;
END_VAR
```

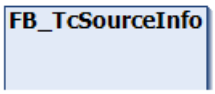
Eingänge

Name	Typ	Beschreibung
sJsonAttribute	STRING	JSON-String

Rückgabewert

Name	Typ	Beschreibung
SetJsonAttribute	HRESULT	Liefert S_OK, wenn der Methodenaufruf erfolgreich war, ansonsten ein HRESULT als Fehlercode.

6.1.12 FB_TcSourceInfo



Mit diesem Funktionsbaustein kann die Quellinformation eines Ereignisses definiert werden.

Syntax

Definition:

```
FUNCTION_BLOCK FB_TcSourceInfo IMPLEMENTS I_TcSourceInfo
```

Schnittstellen

Typ	Beschreibung
I_TcSourceInfo [▶ 139]	Stellt Lese-Methoden und -Eigenschaften einer Quellinformation bereit.

Methoden

Name	Definitionsort	Beschreibung
Clear [▶ 122]	Lokal	Setzt die Quellinformation zurück.
ExtendName [▶ 122]	Lokal	Fügt den übergebenen String an den Namen an.
ResetToDefault [▶ 123]	Lokal	Setzt die Eigenschaften auf Standardwerte. sName wird mit dem Symbolnamen des instanziierenden Funktionsbausteins initialisiert. nId wird mit der Objekt-ID der SPS Instanz initialisiert. Wenn die Instanz von FB_TcSourceInfo mit dem Attribut 'hide' versteckt wird, kann intern kein Symbolname für das Standardverhalten generiert werden.
EqualsTo [▶ 139]	I_TcSourceInfo [▶ 139]	Vergleicht eine Instanz mit einer anderen Instanz.

Eigenschaften

Name	Typ	Zugriff	Definitionsort	Beschreibung
guid	GUID	Get	I_TcSourceInfo [▶ 139]	Liefert die GUID der Quelleinfo zurück.
guid	GUID	SET	Lokal	Setzt die GUID als Quellinfo.
nId	UDINT	Get	I_TcSourceInfo [▶ 139]	Liefert die ID der Quellinfo.
nId	UDINT	SET	Lokal	Setzt die ID der Quellinfo.
sName	STRING(ParameterList.cSourceNameSize-1)	Get	I_TcSourceInfo [▶ 139]	Liefert den Namen der Quellinfo.
sName	STRING(ParameterList.cSourceNameSize-1)	SET	Lokal	Setzt den Namen der Quellinfo

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.1.4022.20	PC oder CX (x64, x86, Arm®)	Tc3_EventLogger

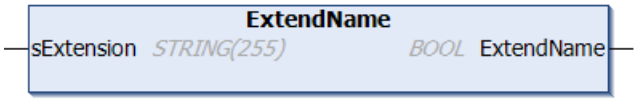
6.1.12.1 Clear

Diese Methode setzt die Quellinformation zurück.

Syntax

```
METHOD Clear
```

6.1.12.2 ExtendName



Diese Methode erweitert den Namen.

Syntax

```
METHOD ExtendName : BOOL
VAR_INPUT
    sExtension : STRING(255);
END_VAR
```

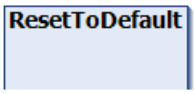
Eingänge

Name	Typ	Beschreibung
sExtension	STRING(255)	Text der rechts angehängt werden soll.

Rückgabewert

Name	Typ	Beschreibung
ExtendName	BOOL	Liefert TRUE, wenn die Verkettung erfolgreich war. Liefert FALSE, wenn die resultierende Zeichenfolge länger ist als die Ausgabe-Zeichenfolge und nicht in den gegebenen Ausgangspuffer passt. Der Speicherbedarf der resultierenden Zeichenfolge ist dann größer als der der Ausgabe-Zeichenfolge. Die Zeichenfolge wird dann abgeschnitten.

6.1.12.3 ResetToDefault



Diese Methode setzt die Quellinformation auf Standardwerte.

Standardwerte:

sName wird mit dem Symbolnamen des instanziierenden Funktionsbausteins initialisiert.

nId wird mit der Objekt-ID der SPS-Instanz initialisiert.

Wenn die Instanz von FB_TcSourceInfo mit dem Attribut 'hide' versteckt wird, kann intern kein Symbolname für das Standardverhalten generiert werden.

Syntax

```
METHOD ResetToDefault
```

6.2 Schnittstellen

6.2.1 I_TcArguments

Diese Schnittstelle definiert Methoden für das Argumenten-Handling.

Vererbungshierarchie

__SYSTEM.IQueryInterface

I_TcArguments

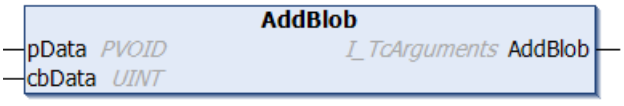
Methoden

Name	Beschreibung
AddBlob [▶ 124]	Fügt Binärdaten als Argument hinzu.
AddBool [▶ 125]	Fügt ein Argument vom Typ BOOL hinzu.
AddByte [▶ 125]	Fügt ein Argument vom Typ BYTE hinzu.
AddDint [▶ 126]	Fügt ein Argument vom Typ DINT hinzu.
AddDWord [▶ 126]	Fügt ein Argument vom Typ DWORD hinzu.
AddEventReferenceId [▶ 126]	Fügt eine Referenz zu einem anderen Ereignis als Argument hinzu.
AddEventReferenceIdGuid [▶ 127]	Fügt eine Referenz zu einem anderen Ereignis als Argument hinzu.
AddInt [▶ 127]	Fügt ein Argument vom Typ INT hinzu.
AddLInt [▶ 128]	Fügt ein Argument vom Typ LINT hinzu.
AddLReal [▶ 128]	Fügt ein Argument vom Typ LREAL hinzu.
AddReal [▶ 129]	Fügt ein Argument vom Typ REAL hinzu.
AddSInt [▶ 129]	Fügt ein Argument vom Typ SINT hinzu.
AddString [▶ 130]	Fügt ein Argument vom Typ STRING hinzu.
AddUDint [▶ 130]	Fügt ein Argument vom Typ UDINT hinzu.
AddUInt [▶ 130]	Fügt ein Argument vom Typ INT hinzu.
AddULInt [▶ 131]	Fügt ein Argument vom Typ ULINT hinzu.
AddUSInt [▶ 131]	Fügt ein Argument vom Typ USINT hinzu.
AddWord [▶ 132]	Fügt ein Argument vom Typ WORD hinzu.
AddWString [▶ 132]	Fügt ein Argument vom Typ WSTRING hinzu.
Clear [▶ 133]	Entfernt alle Argumente.

Eigenschaften

Name	Typ	Zugriff	Beschreibung
nCount	UDINT	Get	Liefert die Anzahl der übergebenen Argumente.

6.2.1.1 AddBlob



Diese Methode fügt Binärdaten als Argument hinzu.

Syntax

```
METHOD AddBlob : I_TcArguments
VAR_INPUT
    pData  : PVOID;
    cbData : UINT;
END_VAR
```

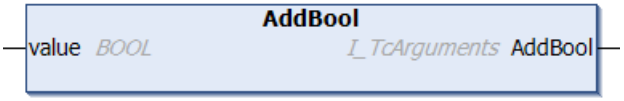
Eingänge

Name	Typ	Beschreibung
pData	PVOID	Zeiger auf das erste Byte von dem Binärdatum.
cbData	UINT	Länge des Binärdatums in Bytes.

Rückgabewert

Name	Typ	Beschreibung
AddBlob	I_TcArguments [▶ 123]	Liefert den I_TcArgument-Zeiger wieder zurück.

6.2.1.2 AddBool



Diese Methode fügt ein Argument vom Typ BOOL hinzu.

Syntax

```
METHOD AddBool : I_TcArguments
VAR_INPUT
    value : BOOL;
END_VAR
```

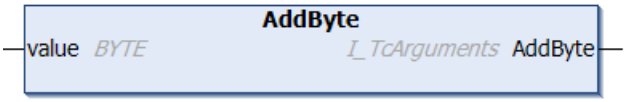
Eingänge

Name	Typ	Beschreibung
value	BOOL	Wert, der hinzugefügt werden soll.

Rückgabewert

Name	Typ	Beschreibung
AddBool	I_TcArguments [▶ 123]	Liefert den I_TcArgument-Zeiger wieder zurück.

6.2.1.3 AddByte



Diese Methode fügt ein Argument vom Typ BYTE hinzu.

Syntax

```
METHOD AddByte : I_TcArguments
VAR_INPUT
    value : BYTE;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
value	BYTE	Wert, der hinzugefügt werden soll.

Rückgabewert

Name	Typ	Beschreibung
AddByte	I_TcArguments [▶ 123]	Liefert den I_TcArgument-Zeiger wieder zurück.

6.2.1.4 AddDint

Diese Methode fügt ein Argument vom Typ DINT hinzu.

Syntax

```
METHOD AddDINT : I_TcArguments
VAR_INPUT
    value : DINT;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
value	DINT	Wert, der hinzugefügt werden soll.

Rückgabewert

Name	Typ	Beschreibung
AddDINT	I_TcArguments [▶ 123]	Liefert den I_TcArgument-Zeiger wieder zurück.

6.2.1.5 AddDWord

Diese Methode fügt ein Argument vom Typ DWORD hinzu.

Syntax

```
METHOD AddDWord : I_TcArguments
VAR_INPUT
    value : DWORD;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
value	DWORD	Wert, der hinzugefügt werden soll.

Rückgabewert

Name	Typ	Beschreibung
AddDWord	I_TcArguments [▶ 123]	Liefert den I_TcArgument-Zeiger wieder zurück.

6.2.1.6 AddEventReferenceId

Diese Methode fügt eine Referenz zu einem anderen Ereignis als Argument hinzu.

Syntax

```
METHOD AddEventReferenceId : I_TcArguments
VAR_INPUT
    nEventId : UDINT;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
nEventId	UDINT	ID des Ereignisses.

Rückgabewert

Name	Typ	Beschreibung
AddEventReferenceId	I_TcArguments [▶ 123]	Liefert den I_TcArgument-Zeiger wieder zurück.

6.2.1.7 AddEventReferenceIdGuid

Diese Methode fügt eine Referenz zu einem anderen Ereignis als Argument hinzu.

Syntax

```
METHOD AddEventReferenceIdGuid : I_TcArguments
VAR_INPUT
    nEventId   : UDINT;
    EventClass : GUID;
END_VAR
```

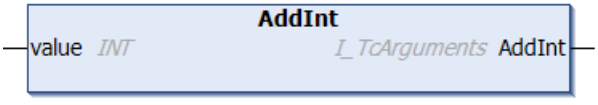
Eingänge

Name	Typ	Beschreibung
nEventId	UDINT	ID des Ereignisses.
EventClass	GUID	GUID der Ereignisklasse.

Rückgabewert

Name	Typ	Beschreibung
AddEventReferenceIdGuid	I_TcArguments [▶ 123]	Liefert den I_TcArgument-Zeiger wieder zurück.

6.2.1.8 AddInt



Diese Methode fügt ein Argument vom Typ INT hinzu.

Syntax

```
METHOD AddINT : I_TcArguments
VAR_INPUT
    value : INT;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
value	INT	Wert, der hinzugefügt werden soll.

Rückgabewert

Name	Typ	Beschreibung
AddInt	I_TcArguments [▶ 123]	Liefert den I_TcArgument-Zeiger wieder zurück.

6.2.1.9 AddLInt

Diese Methode fügt ein Argument vom Typ LINT hinzu.

Syntax

```
METHOD AddLInt : I_TcArguments
VAR_INPUT
    value : LINT;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
value	LINT	Wert, der hinzugefügt werden soll.

Rückgabewert

Name	Typ	Beschreibung
AddLInt	I_TcArguments [▶ 123]	Liefert den I_TcArgument-Zeiger wieder zurück.

6.2.1.10 AddLReal

Diese Methode fügt ein Argument vom Typ LREAL hinzu.

Syntax

```
METHOD AddLReal : I_TcArguments
VAR_INPUT
    value : LREAL;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
value	LREAL	Wert, der hinzugefügt werden soll.

Rückgabewert

Name	Typ	Beschreibung
AddLReal	I_TcArguments [▶ 123]	Liefert den I_TcArgument-Zeiger wieder zurück.

6.2.1.11 AddReal

Diese Methode fügt ein Argument vom Typ REAL hinzu.

Syntax

```
METHOD AddReal : I_TcArguments
VAR_INPUT
    value : REAL;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
value	REAL	Wert, der hinzugefügt werden soll.

Rückgabewert

Name	Typ	Beschreibung
AddReal	I_TcArguments [▶ 123]	Liefert den I_TcArgument-Zeiger wieder zurück.

6.2.1.12 AddSInt

Diese Methode fügt ein Argument vom Typ SINT hinzu.

Syntax

```
METHOD AddSInt : I_TcArguments
VAR_INPUT
    value : SInt;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
value	SINT	Wert, der hinzugefügt werden soll.

Rückgabewert

Name	Typ	Beschreibung
AddSInt	I_TcArguments [▶ 123]	Liefert den I_TcArgument-Zeiger wieder zurück.

6.2.1.13 AddString

Diese Methode fügt ein Argument vom Typ STRING hinzu.

Syntax

```
METHOD AddString : I_TcArguments
VAR_IN_OUT CONSTANT
    value : STRING;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
value	STRING	Wert, der hinzugefügt werden soll.

Rückgabewert

Name	Typ	Beschreibung
AddString	I_TcArguments [▶ 123]	Liefert den I_TcArgument-Zeiger wieder zurück.

6.2.1.14 AddUDint

Diese Methode fügt ein Argument vom Typ UDINT hinzu.

Syntax

```
METHOD AddUDInt : I_TcArguments
VAR_INPUT
    value : UDINT;
END_VAR
```

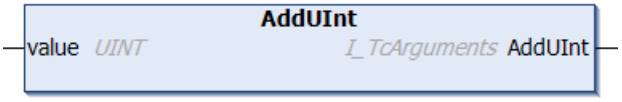
Eingänge

Name	Typ	Beschreibung
value	UDINT	Wert, der hinzugefügt werden soll.

Rückgabewert

Name	Typ	Beschreibung
AddUDInt	I_TcArguments [▶ 123]	Liefert den I_TcArgument-Zeiger wieder zurück.

6.2.1.15 AddUInt



Diese Methode fügt ein Argument vom Typ INT hinzu.

Syntax

```
METHOD AddUInt : I_TcArguments
VAR_INPUT
    value : UINT;
END_VAR
```

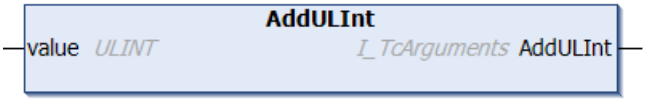
Eingänge

Name	Typ	Beschreibung
value	UINT	Wert, der hinzugefügt werden soll.

Rückgabewert

Name	Typ	Beschreibung
AddUInt	I_TcArguments [▶ 123]	Liefert den I_TcArgument-Zeiger wieder zurück.

6.2.1.16 AddULInt



Diese Methode fügt ein Argument vom Typ ULINT hinzu.

Syntax

```
METHOD AddULInt : I_TcArguments
VAR_INPUT
    value : ULINT;
END_VAR
```

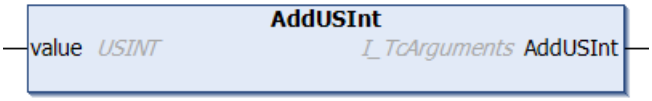
Eingänge

Name	Typ	Beschreibung
value	ULINT	Wert, der hinzugefügt werden soll.

Rückgabewert

Name	Typ	Beschreibung
AddULInt	I_TcArguments [▶ 123]	Liefert den I_TcArgument-Zeiger wieder zurück.

6.2.1.17 AddUSInt



Diese Methode fügt ein Argument vom Typ USINT hinzu.

Syntax

```
METHOD AddUSInt : I_TcArguments
VAR_INPUT
    value : USINT;
END_VAR
```

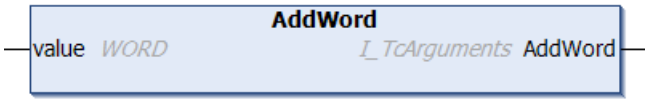
Eingänge

Name	Typ	Beschreibung
value	USINT	Wert, der hinzugefügt werden soll.

Rückgabewert

Name	Typ	Beschreibung
AddUSInt	I_TcArguments [▶ 123]	Liefert den I_TcArgument-Zeiger wieder zurück.

6.2.1.18 AddWord



Diese Methode fügt ein Argument vom Typ WORD hinzu.

Syntax

```
METHOD AddWord : I_TcArguments
VAR_INPUT
    value : WORD;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
value	WORD	Wert, der hinzugefügt werden soll.

Rückgabewert

Name	Typ	Beschreibung
AddWord	I_TcArguments [▶ 123]	Liefert den I_TcArgument-Zeiger wieder zurück.

6.2.1.19 AddWString

Diese Methode fügt ein Argument vom Typ WSTRING hinzu.

Syntax

```
METHOD AddWString : I_TcArguments
VAR_IN_OUT CONSTANT
    value : WSTRING;
END_VAR
```

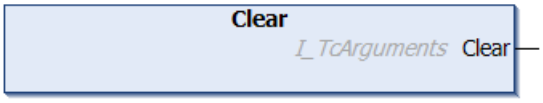
Eingänge

Name	Typ	Beschreibung
value	WSTRING	Wert, der hinzugefügt werden soll.

Rückgabewert

Name	Typ	Beschreibung
AddWString	I_TcArguments [▶ 123]	Liefert den I_TcArgument-Zeiger wieder zurück.

6.2.1.20 Clear



Diese Methode entfernt alle Argumente.

Syntax

```
METHOD Clear : I_TcArguments
```

Rückgabewert

Name	Typ	Beschreibung
Clear	I_TcArguments [▶ 123]	Liefert den I_TcArgument-Zeiger wieder zurück.

6.2.2 I_TcEventBase

In dieser Basisschnittstelle sind Methoden und Eigenschaften eines Ereignisses definiert.

Methoden

Name	Beschreibung
EqualsTo [▶ 134]	Vergleicht das Ereignis mit einer anderen Instanz.
EqualsToEventClass [▶ 134]	Vergleicht die Ereignisklasse des Ereignisses mit einer anderen Ereignisklasse.
EqualsToEventEntryEx [▶ 135]	Vergleicht die Ereignisdefinition des Ereignisses mit einer anderen Ereignisdefinition.
GetJsonAttribute [▶ 136]	Liefert das Json-Attribut.
RequestEventClassName [▶ 136]	Fragt den Namen der Ereignisklasse an.
RequestEventText [▶ 137]	Liefert den Text zum Ereignis.

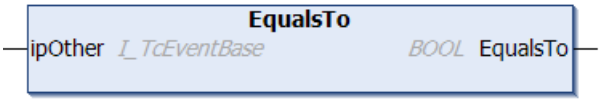
Eigenschaften

Name	Typ	Zugriff	Beschreibung
eSeverity	TcEventSeverity [▶ 140]	Get	Liefert die Severity.
EventClass	GUID	Get	Liefert die GUID der Ereignisklasse.
ipSourceInfo	I_TcSourceInfo [▶ 139]	Get	Liefert einen Zeiger auf die Quelldefinition.
nEventId	UDINT	Get	Liefert die ID des Ereignisses.
stEventEntry	TcEventEntry [▶ 139]	Get	Liefert die Ereignisdefinition.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.1.4022.20	PC oder CX (x64, x86, Arm®)	Tc3_EventLogger

6.2.2.1 EqualsTo



Diese Methode führt einen Vergleich mit einem am Eingang angegebenen anderen Ereignis aus.

Syntax

```
METHOD EqualsTo : BOOL
VAR_INPUT
    ipOther : I_TcEventBase;
END_VAR
```

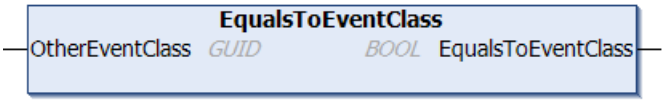
Eingänge

Name	Typ	Beschreibung
ipOther	I_TcEventBase [▶ 133]	Zu vergleichendes Ereignis

Rückgabewert

Name	Typ	Beschreibung
EqualsTo	BOOL	Liefert TRUE, wenn die Ereignisse übereinstimmen.

6.2.2.2 EqualsToEventClass



Diese Methode führt einen Vergleich mit einer am Eingang angegebenen anderen Ereignisklasse aus.

Syntax

```
METHOD EqualsToEventClass : BOOL
VAR_INPUT
    OtherEventClass : GUID;
END_VAR
```

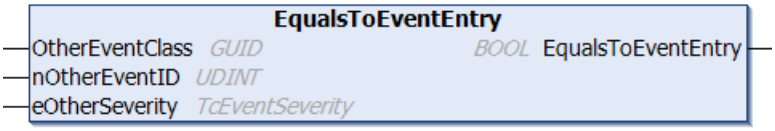
Eingänge

Name	Typ	Beschreibung
OtherEventClass	GUID	Zu vergleichende Ereignisklasse.

Rückgabewert

Name	Typ	Beschreibung
EqualsToEventClass	BOOL	Liefert TRUE, wenn die Ereignisklassen übereinstimmen.

6.2.2.3 EqualsToEventEntry



Diese Methode führt einen Vergleich mit einem am Eingang angegebenen anderen Ereignis aus.

Syntax

```
METHOD EqualsToEventEntry : BOOL
VAR_INPUT
    OtherEventClass : GUID;
    nOtherEventID   : UDINT;
    eOtherSeverity  : TcEventSeverity;
END_VAR
```

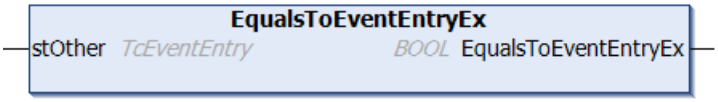
Eingänge

Name	Typ	Beschreibung
OtherEventClass	GUID	Ereignisklasse des zu vergleichenden Ereignisses.
nOtherEventID	UDINT	Event-ID des zu vergleichenden Ereignisses.
eOtherSeverity	TcEventSeverity [▶ 140]	Event-Severity des zu vergleichenden Ereignisses.

Rückgabewert

Name	Typ	Beschreibung
EqualsToEventEntry	BOOL	Liefert TRUE, wenn die Ereignisse übereinstimmen.

6.2.2.4 EqualsToEventEntryEx



Diese Methode führt einen Vergleich mit einem am Eingang angegebenen anderen Ereignis aus.

Syntax

```
METHOD EqualsToEventEntryEx : BOOL
VAR_INPUT
    stOther : TcEventEntry;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
stOther	TcEventEntry [▶ 139]	Zu vergleichendes Ereignis.

Rückgabewert

Name	Typ	Beschreibung
EqualsToEventEntryEx	BOOL	Liefert TRUE, wenn die Ereignisse übereinstimmen.

6.2.2.5 GetJsonAttribute

Diese Methode liefert das JSON-Attribut.

Syntax

```
METHOD GetJsonAttribute : HRESULT
VAR_INPUT
    sJsonAttribute : REFERENCE TO STRING;
    nJsonAttribute : UDINT;
END_VAR
```

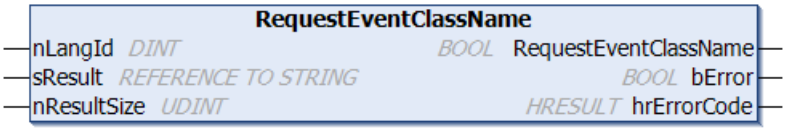
Eingänge

Name	Typ	Beschreibung
sJsonAttribute	REFERENCE TO STRING	Referenz auf eine Variable vom Typ String
nJsonAttribute	UDINT	Länge der String-Variable

Rückgabewert

Name	Typ	Beschreibung
GetJsonAttribute	HRESULT	Liefert S_OK, wenn der Methodenaufruf erfolgreich war. Liefert ERROR_BAD_LENGTH, wenn die Länge der Variable zu klein ist. Ansonsten wird HRESULT als Fehlercode zurückgegeben.

6.2.2.6 RequestEventClassName



Diese Methode liefert den Namen der Ereignisklasse.

Syntax

```
METHOD RequestEventClassName : BOOL
VAR_INPUT
    nLangId     : DINT;
    sResult     : REFERENCE TO STRING;
    nResultSize : UDINT;
END_VAR
VAR_OUTPUT
    bError      : BOOL;
    hrErrorCode : HRESULT;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
nLangId	DINT	Spezifiziert die Sprach-ID Englisch (en-US) = 1033 Deutsch (de-DE) = 1031 …
sResult	REFERENCE TO STRING	Referenz auf eine Variable vom Typ String
nResultSize	UDINT	Größe der String-Variable in Bytes

Rückgabewert

Name	Typ	Beschreibung
RequestEventClassName	BOOL	Liefert TRUE, sobald die Abfrage beendet wurde. Wenn die asynchrone Abfrage noch aktiv ist, liefert der Rückgabewert FALSE. Die Methode muss so lange aufgerufen werden, bis der Rückgabewert TRUE ist.

Ausgänge

Name	Typ	Beschreibung
bError	BOOL	Liefert FALSE, wenn der Methodenaufruf erfolgreich war. Liefert TRUE, wenn ein Fehler aufgetreten ist.
hrErrorCode	HRESULT	Liefert S_OK, wenn der Methodenaufruf erfolgreich war. Im Falle eines Fehlers wird ein Fehlercode ausgegeben.

6.2.2.7 RequestEventText

Diese Methode liefert den Ereignistext.

Syntax

```
METHOD RequestEventText : BOOL
VAR_INPUT
    nLangId     : DINT;
    sResult     : REFERENCE TO STRING;
    nResultSize : UDINT;
END_VAR
VAR_OUTPUT
    bError      : BOOL;
    hrErrorCode : HRESULT;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
nLangId	DINT	Spezifiziert die Sprach-ID Englisch (en-US) = 1033 Deutsch (de-DE) = 1031 …
sResult	REFERENCE TO STRING	Referenz auf eine Variable vom Typ String
nResultSize	UDINT	Größe der String-Variable in Bytes

Rückgabewert

Name	Typ	Beschreibung
RequestEventText	BOOL	Liefert TRUE, sobald die Abfrage beendet wurde. Wenn die asynchrone Abfrage noch aktiv ist, liefert der Rückgabewert FALSE. Die Methode muss so lange aufgerufen werden, bis der Rückgabewert TRUE ist.

Ausgänge

Name	Typ	Beschreibung
bError	BOOL	Liefert FALSE, wenn der Methodenaufruf erfolgreich war. Liefert TRUE, wenn ein Fehler aufgetreten ist.
hrErrorCode	HRESULT	Liefert S_OK, wenn der Methodenaufruf erfolgreich war. Im Falle eines Fehlers wird ein Fehlercode ausgegeben.

6.2.3 I_TcMessage

Diese Schnittstelle stellt Methoden und Eigenschaften für das Nachrichten-Handling bereit.

Vererbungshierarchie

I_TcEventBase [▶ 133]

I_TcMessage

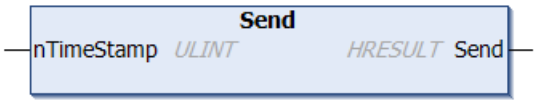
Methoden

Name	Beschreibung
Send [▶ 138]	Sendet eine Nachricht

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.1.4022.20	PC oder CX (x64, x86, Arm®)	Tc3_EventLogger

6.2.3.1 Send



Diese Methode sendet die Nachricht.

Syntax

```
METHOD Send : HRESULT
VAR_INPUT
    nTimeStamp: ULINT;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
nTimeStamp	ULINT	0: Aktueller Zeitstempel wird verwendet > 0: Externer Zeitstempel in 100 Nanosekunden seit dem 1. Januar 1601 (UTC).

Rückgabewert

Name	Typ	Beschreibung
Send	FB_ HRESULT	Liefert S_OK, wenn der Methodenaufruf erfolgreich war, ansonsten ein HRESULT als Fehlercode

6.2.4 I_TcSourceInfo

Diese Schnittstelle definiert Eigenschaften für eine Quellinformation.

Methoden

Name	Beschreibung
EqualsTo [▶ 139]	Vergleicht eine Instanz mit Quellinformationen mit einer anderen Instanz.

Eigenschaften

Name	Typ	Zugriff	Beschreibung
guid	GUID	Get	Liefert die GUID der Quelleinfo zurück.
nId	UDINT	Get	Liefert die ID der Quelleinfo zurück.
sName	STRING(ParameterList.cSourceNameSize-1)	Get	Liefert den Namen der Quelleinfo zurück.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.1.4022.20	PC oder CX (x64, x86, Arm®)	Tc3_EventLogger

6.2.4.1 EqualsTo

Diese Methode vergleicht eine Instanz mit Quellinformationen mit einer anderen Instanz.

Syntax

```
METHOD EqualsTo : BOOL
VAR_INPUT
    ipOther : I_TcSourceInfo;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
ipOther	I_TcSourceInfo [▶ 139]	Zu vergleichende Quellinformationen

Rückgabewert

Name	Typ	Beschreibung
EqualsTo	BOOL	Liefert TRUE, wenn die Quellinformationen übereinstimmen.

6.3 Datentypen

6.3.1 TcEventEntry

Definiert ein Ereignis (Event) mittels Ereignisklasse, Ereignis-ID und Severity.

Syntax

Definition:

```
TYPE TcEventEntry :
STRUCT
    uuidEventClass : GUID;
    nEventId       : UDINT;
    eSeverity      : TcEventSeverity;
END_STRUCT
END_TYPE
```

Parameter

Name	Typ	Beschreibung
uuidEventClass	GUID	GUID der Ereignisklasse.
nEventId	UDINT	ID des Ereignisses.
eSeverity	TcEventSeverity	Event-Severity definiert den Schweregrad des Ereignisses.

6.3.2 TcEventSeverity

Definiert die Severity des Ereignisses.

Syntax

Definition:

```
{attribute 'qualified_only'}
TYPE TcEventSeverity : (
    Verbose  := 0,
    Info     := 1,
    Warning  := 2,
    Error    := 3,
    Critical := 4);
END_TYPE
```

Parameter

	Name	Beschreibung
4	Critical	Kritisch
3	Error	Fehler
2	Warning	Warnung
1	Info	Information
0	Verbose	Erweiterte Ausgabe

6.3.3 TcEventConfirmationState

Definiert den Bestätigungszustand eines Alarms.

Syntax

Definition:

```
{attribute 'qualified_only'}
TYPE TcEventConfirmationState : (
   NotSupported := 0,
   NotRequired  := 1,
   WaitForConfirmation := 2,
   Confirmed := 3,
   Reset     := 4);
END_TYPE
```

Parameter

Name	Beschreibung
Confirmed	Bestätigt
NotRequired	Bestätigung im aktuellen Zustand nicht nötig. (Alarm aktuell nicht im Zustand Raised.)
NotSupported	Wurde ohne Bestätigung initialisiert.
Reset	Initialzustand
WaitForConfirmation	Wartet auf Bestätigung.

6.4 Globale Listen

6.4.1 Global_Constants

```
VAR_GLOBAL CONSTANT
    EMPTY_EVENT_CLASS : GUID := (Data1:=16#0, Data2:=16#0, Data3:=16#0, Data4:=[16#0,16#0,16#0,16#0,
16#0,16#0,16#0,16#0]);
    EMPTY_EVENT_ID    : UDINT := 16#0;
    EMPTY_SEVERITY    : TcEventSeverity := TcEventSeverity.Verbose;
    SUCCESS_EVENT     : TcEventEntry := ( uuidEventClass := EMPTY_EVENT_CLASS, nEventID := EMPTY_EVE
NT_ID, eSeverity := EMPTY_SEVERITY );
END_VAR
```

Name	Typ	Initialwert
EMPTY_EVENT_CLASS	GUID	STRUCT(Data1:=16#0, Data2:=16#0, Data3:=16#0, Data4:=[16#0,16#0,16#0,16#0,16#0,16#0,16#0,16#0])
EMPTY_EVENT_ID	UDINT	16#0
EMPTY_SEVERITY	TcEventSeverity [▶ 140]	TcEventSeverity.Verbose
SUCCESS_EVENT	TcEventEntry [▶ 139]	STRUCT(uuidEventClass := EMPTY_EVENT_CLASS, nEventID := EMPTY_EVENT_ID, eSeverity := EMPTY_SEVERITY)

6.4.2 GVL

```
{attribute 'qualified_only'}
VAR_GLOBAL
    nLangId_OnlineMonitoring : DINT := 1033;
END_VAR
```

Name	Typ	Initialwert	Beschreibung
nLangId_OnlineMonitoring	DINT	1033	Sprachkennung für das Online Monitoring Englisch (en-US) = 1033 Deutsch (de-DE) = 1031 …

6.4.3 Parameterlist

```
{attribute 'qualified_only'}
VAR_GLOBAL CONSTANT
    cSourceNameSize : UDINT(81..10000) := 256;
END_VAR
```

Name	Typ	Initialwert	Beschreibung
cSourceNameSize	UDINT(81..10000)	256	Größe in Bytes für den Namen der Quellinformation. Empfohlen sind maximal 512 Bytes.

6.4.4 Global_Version

Alle Bibliotheken haben eine bestimmte Version. Diese Version ist u. a. im SPS-Bibliotheksrepository zu sehen.
Eine globale Konstante enthält die Information über die Bibliotheksversion (vom Typ ST_LibVersion):

Global_Version

```
VAR_GLOBAL CONSTANT
    stLibVersion_Tc3_EventLogger : ST_LibVersion;
END_VAR
```

Um zu sehen, ob die Version, die Sie haben auch die Version ist, die Sie brauchen, benutzen Sie die Funktion F_CmpLibVersion (definiert in der Tc2_System-Bibliothek).

7 C++ API

7.1 Schnittstellen

7.1.1 ITcEvent

Diese Schnittstelle stellt allgemeine Methoden für ITcAlarm und ITcMessage bereit. Sie wird in den Callbacks der Schnittstellen ITcAlarmListener und ITcMessageListener verwendet.

Syntax

```
TCOM_DECL_INTERFACE("4A9CB0E9-8969-4B85-B567-605110511200", ITcEvent)
```

Methoden

Name	Beschreibung
GetEventClass [▶ 143]	Liefert die GUID der Ereignisklasse.
GetEventId [▶ 143]	Liefert die ID des Ereignisses.
GetSeverity [▶ 144]	Liefert die Severity des Ereignisses.
GetSourceInfo [▶ 144]	Liefert die SourceInfo.
GetJsonAttribute [▶ 144]	Liefert das JSON-Attribut.
GetText [▶ 145]	Liefert den Text, asynchron.
GetEventClassName [▶ 145]	Liefert den Ereignisklassen-Namen, asynchron.

7.1.1.1 GetEventClass

Liefert die GUID der Ereignisklasse.

Syntax

```
virtual HRESULT TCOMAPI GetEventClass (GUID eventClass)
```

Parameter

Name	Typ	Beschreibung
eventClass	REFERENCE TO GUID	Referenz auf die GUID der Ereignisklasse.

Rückgabewert

Typ	Beschreibung
HRESULT	Liefert S_OK, wenn der Methodenaufruf erfolgreich war, ansonsten ein HRESULT als Fehlercode.

7.1.1.2 GetEventId

Liefert die ID des Ereignisses.

Syntax

```
virtual HRESULT TCOMAPI GetEventId (UDINT EventId)
```

Parameter

Name	Typ	Beschreibung
eventId	REFERENCE TO UDINT	Referenz auf die ID des Ereignisses.

Rückgabewert

Typ	Beschreibung
HRESULT	Liefert S_OK, wenn der Methodenaufruf erfolgreich war, ansonsten ein HRESULT als Fehlercode.

7.1.1.3 GetSeverity

Liefert die Severity des Ereignisses.

Syntax

```
virtual HRESULT TCOMAPI GetSeverity (TcEventSeverity severity)
```

Parameter

Name	Typ	Beschreibung
severity	REFERENCE TO TcEventSeverity	Referenz auf die Severity eines Ereignisses.

Rückgabewert

Typ	Beschreibung
HRESULT	Liefert S_OK, wenn der Methodenaufruf erfolgreich war, ansonsten ein HRESULT als Fehlercode.

7.1.1.4 GetSourceInfo

Liefert die SourceInfo.

Syntax

```
virtual HRESULT TCOMAPI GetSourceInfo (ITcSourceInfo pipSourceInfo)
```

Parameter

Name	Typ	Beschreibung
pipSourceInfo	POINTER TO ITcSourceInfo	Referenz auf die SourceInfo-Schnittstelle.

Rückgabewert

Typ	Beschreibung
HRESULT	Liefert S_OK, wenn der Methodenaufruf erfolgreich war, ansonsten ein HRESULT als Fehlercode.

7.1.1.5 GetJsonAttribute

Liefert das JSON-Attribut.

Syntax

```
virtual HRESULT TCOMAPI GetJsonAttribute (STRING sJsonAttribute, UDINT nJsonAttribute)
```

Parameter

Name	Typ	Beschreibung
sJsonAttribute	REFERENCE TO STRING	Referenz auf den JSON-String
nJsonAttribute	REFERENCE TO UDINT	Referenz auf die Länge des Json-Attributs.

Rückgabewert

Typ	Beschreibung
HRESULT	Liefert S_OK, wenn der Methodenaufruf erfolgreich war, ansonsten ein HRESULT als Fehlercode.

7.1.1.6 GetText

Liefert den Text, asynchron.

Syntax

```
virtual HRESULT TCOMAPI GetText (DINT nLangId, ITcAsyncStringResult pipResult)
```

Parameter

Name	Typ	Beschreibung
nLangId	DINT	Sprach-ID (LCID) der angefragten Sprache.
pipResult	POINTER TO ITcAsyncStringResult	Referenz auf einen ITcAsyncStringResult-Zeiger.

Rückgabewert

Typ	Beschreibung
HRESULT	Liefert S_OK, wenn der Methodenaufruf erfolgreich war, ansonsten ein HRESULT als Fehlercode.

7.1.1.7 GetEventClassName

Liefert den Ereignisklassen-Namen, asynchron.

Syntax

```
virtual HRESULT TCOMAPI GetClassName (DINT nLangId, ITcAsyncStringResult pipResult)
```

Parameter

Name	Typ	Beschreibung
nLangId	DINT	Sprach-ID (LCID) der angefragten Sprache.
pipResult	POINTER TO ITcAsyncStringResult	Referenz auf einen ITcAsyncStringResult-Zeiger.

Rückgabewert

Typ	Beschreibung
HRESULT	Liefert S_OK, wenn der Methodenaufruf erfolgreich war, ansonsten ein HRESULT als Fehlercode.

7.1.2 ITcMessage

Diese Schnittstelle repräsentiert eine Nachricht vom TwinCAT 3 EventLogger.

Syntax

```
TCOM_DECL_INTERFACE("6474ED2C-E483-454E-A67D-233E6D337C08", ITcMessage)
```

Methoden

Name	Beschreibung
SetJsonAttribute [▶ 146]	Setzt das JSON-Attribut.
GetArguments [▶ 146]	Liefert den Schnittstellenzeiger für die Argumente.
Send [▶ 147]	Sendet die Nachricht.

7.1.2.1 SetJsonAttribute

Setzt das JSON-Attribut.

Syntax

```
virtual HRESULT TCOMAPI SetJsonAttribute (STINRG sJsonAttribute)
```

Parameter

Name	Typ	Beschreibung
sJsonAttribute	STRING	JSON-String

Rückgabewert

Typ	Beschreibung
HRESULT	Liefert S_OK, wenn der Methodenaufruf erfolgreich war, ansonsten ein HRESULT als Fehlercode.

7.1.2.2 GetArguments

Liefert den Schnittstellenzeiger für die Argumente.

Syntax

```
virtual HRESULT TCOMAPI GetArguments (ITcArguments pipArguments)
```

Parameter

Name	Typ	Beschreibung
pipArguments	POINTER to ITcArguments	Referenz auf die ITcArguments-Schnittstelle

Rückgabewert

Typ	Beschreibung
HRESULT	Liefert S_OK, wenn der Methodenaufruf erfolgreich war, ansonsten ein HRESULT als Fehlercode.

7.1.2.3 Send

Sendet die Nachricht.

Syntax

```
virtual HRESULT TCOMAPI Send (ULINT timeStamp)
```

Parameter

Name	Typ	Beschreibung
timeStamp	ULINT	> 0: Externer Zeitstempel in 100 Nanosekunden seit dem 1. Januar 1601 (UTC).

Rückgabewert

Typ	Beschreibung
HRESULT	Liefert S_OK, wenn der Methodenaufruf erfolgreich war, ansonsten ein HRESULT als Fehlercode.

7.1.3 ITcAlarm

Diese Schnittstelle repräsentiert einen Alarm vom TwinCAT 3 EventLogger.

Syntax

```
TCOM_DECL_INTERFACE("EC6D4FF7-5805-4DDB-A316-27894E77D644", ITcAlarm)
```

Methoden

Name	Beschreibung
SetJsonAttribute [▶ 147]	Setzt das JSON-Attribut.
GetArguments [▶ 148]	Liefert den Schnittstellenzeiger für die Argumente.
GetIsRaised [▶ 148]	Liefert TRUE, wenn der Alarm im Zustand Raised ist.
Raise [▶ 148]	Setzt den Alarmzustand auf Raised.
Clear [▶ 149]	Setzt den Alarmzustand auf Not-Raised.
GetConfirmationState [▶ 149]	Liefert den Bestätigungszustand.
Confirm [▶ 150]	Setzt den Alarmzustand auf Confirmed.

7.1.3.1 SetJsonAttribute

Setzt das Json-Attribut.

Syntax

```
virtual HRESULT TCOMAPI SetJsonAttribute (STRING sJsonAttribute)
```

Parameter

Name	Typ	Beschreibung
sJsonAttribute	STRING	JSON-String

Rückgabewert

Typ	Beschreibung
HRESULT	Liefert S_OK, wenn der Methodenaufruf erfolgreich war, ansonsten ein HRESULT als Fehlercode.

7.1.3.2 GetArguments

Liefert den Schnittstellenzeiger für die Argumente.

Syntax

```
virtual HRESULT TCOMAPI GetArguments (ITcArguments pipArguments)
```

Parameter

Name	Typ	Beschreibung
pipArguments	POINTER to ITcArguments	Referenz auf die ITcArguments-Schnittstelle

Rückgabewert

Typ	Beschreibung
HRESULT	Liefert S_OK, wenn der Methodenaufruf erfolgreich war, ansonsten ein HRESULT als Fehlercode.

7.1.3.3 GetIsRaised

Liefert TRUE im Parameter bIsRaised, wenn der Alarm im Zustand Raised ist.

Syntax

```
virtual HRESULT TCOMAPI GetIsRaised (BOOL32 bIsRaised)
```

Parameter

Name	Typ	Beschreibung
bIsRaised	REFERENCE TO BOOL32	Referenz auf den Zustand. Liefert TRUE, wenn der Alarm im Zustand Raised ist.

Rückgabewert

Typ	Beschreibung
HRESULT	Liefert S_OK, wenn der Methodenaufruf erfolgreich war, ansonsten ein HRESULT als Fehlercode.

7.1.3.4 Raise

Setzt den Alarmzustand [▶ 14] auf Raised.

Wenn der Alarm bestätigungspflichtig ist, wird zusätzlich der Bestätigungszustand auf WaitForConfirmation gesetzt.

Syntax

```
virtual HRESULT TCOMAPI Raise (ULINT timeStamp)
```

Parameter

Name	Typ	Beschreibung
timeStamp	ULINT	> 0: Externer Zeitstempel in 100 Nanosekunden seit dem 1. Januar 1601 (UTC).

Rückgabewert

Typ	Beschreibung
HRESULT	Liefert S_OK, wenn der Methodenaufruf erfolgreich war, ansonsten ein HRESULT als Fehlercode.

7.1.3.5 Clear

Setzt den Alarmzustand [▶ 14] auf Not-Raised.

Syntax

```
virtual HRESULT TCOMAPI Clear (ULINT timeStamp, BOOL32 bResetConfirmation)
```

Parameter

Name	Typ	Beschreibung
timeStamp	ULINT	> 0: Externer Zeitstempel in 100 Nanosekunden seit dem 1. Januar 1601 (UTC).
bResetConfirmation	BOOL32	Wenn TRUE und der Bestätigungszustand WaitForConfirmation ist, wird der Bestätigungszustand auf Reseted gesetzt. Ansonsten wird der Bestätigungszustand nicht verändert.

Rückgabewert

Typ	Beschreibung
HRESULT	Liefert S_OK, wenn der Methodenaufruf erfolgreich war, ansonsten ein HRESULT als Fehlercode.

7.1.3.6 GetConfirmationState

Liefert den Bestätigungszustand [▶ 14].

Syntax

```
virtual HRESULT TCOMAPI GetConfirmationState (… state)
```

Parameter

Name	Typ	Beschreibung
state	REFERENCE TO TcEventConfirmationState	Liefert den Bestätigungszustand.

Rückgabewert

Typ	Beschreibung
HRESULT	Liefert S_OK, wenn der Methodenaufruf erfolgreich war, ansonsten ein HRESULT als Fehlercode.

7.1.3.7 Confirm

Setzt den Bestätigungszustand [▶ 14] von WaitingForConfirmation auf Confirmed.

Syntax

```
virtual HRESULT TCOMAPI Confirm (ULINT timeStamp)
```

Parameter

Name	Typ	Beschreibung
timeStamp	ULINT	> 0: Externer Zeitstempel in 100 Nanosekunden seit dem 1. Januar 1601 (UTC).

Rückgabewert

Typ	Beschreibung
HRESULT	Liefert S_OK, wenn der Methodenaufruf erfolgreich war, ansonsten ein HRESULT als Fehlercode.

7.1.4 ITcEventLogger

Diese Schnittstelle stellt den TwinCAT 3 EventLogger selbst dar.

Syntax

```
TCOM_DECL_INTERFACE("B2D5D4E2-07F6-44F4-A292-92CA8035AA86", ITcEventLogger)
```

Benötigte include:

```
#include "TcRouterInterfaces.h"
#include "TcEventLoggerInterfaces.h"
```

Methoden

Name	Beschreibung
CreateMessage [▶ 151]	Erstellt eine Instanz, die ITcMessage implementiert.
CreateAlarm [▶ 152]	Erstellt eine Instanz, die ITcAlarm implementiert.
GetAlarm [▶ 152]	Liefert einen Zeiger auf einen existierenden Alarm.
IsAlarmRaised [▶ 153]	Fragt ab, ob ein Alarm im Zustand Raised ist.
ConfirmAllAlarms [▶ 153]	Ruft Confirm() für alle Alarme mit dem Bestätigungszustand WaitForConfirmation auf.
ClearAllAlarms [▶ 153]	Ruft Clear() für alle Alarme im Zustand Raised auf.
SendTcMessage [▶ 154]	Sendet eine Nachricht.
AddMessageListener [▶ 154]	Meldet einen Nachrichtenbeobachter an.
RemoveMessageListener [▶ 155]	Meldet einen Nachrichtenbeobachter ab.
NotifyMessageListener [▶ 155]	Arbeitet eine Queue für einen Nachrichtenbeobachter ab.
AddAlarmListener [▶ 156]	Meldet einen Alarmbeobachter an.
RemoveAlarmListener [▶ 156]	Meldet einen Alarmbeobachter ab.
NotifyAlarmListener [▶ 156]	Arbeitet eine Queue für einen Alarmbeobachter ab.
GetEventText [▶ 157]	Liefert eine Text zu einem Event.
GetEventClassName [▶ 157]	Liefert den Klassennamen zu einem Ereignis.
CreateArguments [▶ 158]	Erstellt eine Instanz, die ITcArguments implementiert.

7.1.4.1 CreateMessage

Erstellt eine Instanz, die ITcMessage implementiert.

Syntax

```
virtual HRESULT TCOMAPI CreateMessage (GUID eventClass, UDINT eventId, GUID severit, ITcSourceInfo i
pSourceInfo, ITcMessage pipMessage)
```

Parameter

Name	Typ	Beschreibung
eventClass	REFERENCE TO GUID	Referenz auf die GUID der Ereignisklasse.
eventId	REFERENCE TO UDINT	Referenz auf die ID des Ereignisses.
severity	REFERENCE TO TcEventSeverity	Referenz auf die Severity eines Ereignisses.
ipSourceInfo	ITcSourceInfo	Zeiger auf die ITcSourceInfo-Schnittstelle.
pipMessage	ITcMessage [▶ 146]	Zeiger auf einen ITcMessage-Zeiger.

Rückgabewert

Typ	Beschreibung
HRESULT	Liefert S_OK, wenn der Methodenaufruf erfolgreich war, ansonsten ein HRESULT als Fehlercode.

7.1.4.2 CreateAlarm

Erstellt eine Instanz, die ITcAlarm implementiert.

Syntax

```
virtual HRESULT TCOMAPI CreateAlarm (GUID eventClass, UDINT eventId, GUID severity, BOOL32 bWithConf
irmation, ITcSourceInfo ipSourceInfo, ITcAlarm pipAlarm)
```

Parameter

Name	Typ	Beschreibung
eventClass	REFERENCE TO GUID	Referenz auf die GUID der Ereignisklasse.
eventId	REFERENCE TO UDINT	Referenz auf die ID des Ereignisses.
severity	REFERENCE TO TcEventSeverity	Referenz auf die Severity eines Ereignisses.
bWithConfirmation	BOOL32	Legt fest, ob der Alarm bestätigungspflichtig ist.
ipSourceInfo	ITcSourceInfo	Zeiger auf die ITcSourceInfo-Schnittstelle.
pipAlarm	ITcAlarm [▶ 147]	Zeiger auf einen ITcAlarm-Zeiger.

Rückgabewert

Typ	Beschreibung
HRESULT	Liefert S_OK, wenn ein neuer Alarm erfolgreich erstellt werden konnte. Liefert ERROR_ALREADY_EXISTS, wenn der Alarm bereits existiert. Im Fehlerfall wird ein HRESULT als Fehlercode zurückgegeben.

7.1.4.3 GetAlarm

Liefert einen Schnittstellenzeiger auf eine existierende Instanz.

Syntax

```
virtual HRESULT TCOMAPI GetAlarm (GUID eventClass, UDINT eventId, ITcSourceInfo ipSourceInfo, ITcAla
rm pipAlarm)
```

Parameter

Name	Typ	Beschreibung
eventClass	REFERENCE TO GUID	Referenz auf die GUID der Ereignisklasse.
eventId	REFERENCE TO UDINT	Referenz auf die ID des Ereignisses.
ipSourceInfo	ITcSourceInfo	Zeiger auf die ITcSourceInfo-Schnittstelle.
pipAlarm	ITcAlarm [▶ 147]	Zeiger auf einen ITcAlarm-Zeiger.

Rückgabewert

Typ	Beschreibung
HRESULT	Liefet ADS_E_NOTFOUND wenn keine Instanz gefunden wurde. S_OK wenn alles erfolgreich war, ansonsten ein HRESULT als Fehlercode.

7.1.4.4 IsAlarmRaised

Fragt ab, ob ein Alarm im Zustand Raised ist.

Syntax

```
virtual HRESULT TCOMAPI IsAlarmRaised (GUID eventClass, UDINT eventId, BOOL32 bIsRaised, ITcSourceInfo ipSourceInfo)
```

Parameter

Name	Typ	Beschreibung
eventClass	REFERENCE TO GUID	Referenz auf die GUID der Ereignisklasse.
eventId	REFERENCE TO UDINT	Referenz auf die ID des Ereignisses.
bIsRaised	REFERENCE TO BOOL32	Referenz auf den Zustand. Liefert TRUE, wenn der Alarm im Zustand Raised ist.
ipSourceInfo	ITcSourceInfo	Zeiger auf die ITcSourceInfo-Schnittstelle.

Rückgabewert

Typ	Beschreibung
HRESULT	Liefert S_OK, wenn der Methodenaufruf erfolgreich war, ansonsten ein HRESULT als Fehlercode.

7.1.4.5 ConfirmAllAlarms

Ruft Confirm() für alle Alarme mit dem Bestätigungszustand WaitForConfirmation auf.

Syntax

```
virtual HRESULT TCOMAPI ConfirmAllAlarms (ULINT timeStamp)
```

Parameter

Name	Typ	Beschreibung
timeStamp	ULINT	> 0: Externer Zeitstempel in 100 Nanosekunden seit dem 1. Januar 1601 (UTC).

Rückgabewert

Typ	Beschreibung
HRESULT	Liefert S_OK, wenn der Methodenaufruf erfolgreich war, ansonsten ein HRESULT als Fehlercode.

7.1.4.6 ClearAllAlarms

Ruft Clear() für alle Alarme im Zustand Raised auf.

Syntax

```
virtual HRESULT TCOMAPI ClearAllAlarms (ULINT timestamp, BOOL32 bResetConfirmation)
```

Parameter

Name	Typ	Beschreibung
timeStamp	ULINT	> 0: Externer Zeitstempel in 100 Nanosekunden seit dem 1. Januar 1601 (UTC).
bResetConfirmation	BOOL32	Wenn TRUE und der Bestätigungszustand WaitForConfirmation ist, wird der Bestätigungszustand auf Reseted gesetzt. Ansonsten wird der Bestätigungszustand nicht verändert.

Rückgabewert

Typ	Beschreibung
HRESULT	Liefert S_OK, wenn der Methodenaufruf erfolgreich war, ansonsten ein HRESULT als Fehlercode.

7.1.4.7 SendTcMessage

Sendet eine Nachricht.

Syntax

```
virtual HRESULT TCOMAPI SendTcMessage (GUID eventClass, UDINT eventId, GUID severity, ITcSourceInfo ipSourceInfo, ULINT timeStamp, ITcArguments ipSerializedArguments)
```

Parameter

Name	Typ	Beschreibung
eventClass	REFERENCE TO GUID	Referenz auf die GUID der Ereignisklasse.
eventId	REFERENCE TO UDINT	Referenz auf die ID des Ereignisses.
severity	REFERENCE TO TcEventSeverity	Referenz auf die Severity eines Ereignisses.
ipSourceInfo	ITcSourceInfo	Zeiger auf die ITcSourceInfo-Schnittstelle.
timeStamp	ULINT	> 0: Externer Zeitstempel in 100 Nanosekunden seit dem 1. Januar 1601 (UTC).
ipSerializedArguments	ITcArguments	Zeiger auf die ITcArguments-Schnittstelle.

Rückgabewert

Typ	Beschreibung
HRESULT	Liefert S_OK, wenn der Methodenaufruf erfolgreich war, ansonsten ein HRESULT als Fehlercode.

7.1.4.8 AddMessageListener

Meldet einen Nachrichtenbeobachter an.

Syntax

```
virtual HRESULT TCOMAPI AddMessageListener (ITcMessageListener ipListener, ITcEventFilterConfig pipF
ilterConfig)
```

Parameter

Name	Typ	Beschreibung
ipListener	ITcMessageListener	Zeiger auf die ITcMessageListener-Schnittstelle.
pipFilterConfig	ITcEventFilterConfig	Zeiger auf einen ITcEventFilterConfig-Zeiger.

Rückgabewert

Typ	Beschreibung
HRESULT	Liefert S_OK, wenn der Methodenaufruf erfolgreich war, ansonsten ein HRESULT als Fehlercode.

7.1.4.9 RemoveMessageListener

Meldet einen Nachrichtenbeobachter ab.

Syntax

```
virtual HRESULT TCOMAPI RemoveMessageListener (ITcMessageListener ipListener)
```

Parameter

Name	Typ	Beschreibung
ipListener	ITcMessageListener	Zeiger auf die ITcMessageListener-Schnittstelle.

Rückgabewert

Typ	Beschreibung
HRESULT	Liefert S_OK, wenn der Methodenaufruf erfolgreich war, ansonsten ein HRESULT als Fehlercode.

7.1.4.10 NotifyMessageListener

Arbeitet eine Queue für den Nachrichtenbeobachter ab.

Syntax

```
virtual HRESULT TCOMAPI NotifyMessageListener (ITcMessageListener ipListener)
```

Parameter

Name	Typ	Beschreibung
ipListener	ITcMessageListener	Zeiger auf die ITcMessageListener-Schnittstelle.

Rückgabewert

Typ	Beschreibung
HRESULT	Liefert S_OK, wenn der Methodenaufruf erfolgreich war, ansonsten ein HRESULT als Fehlercode.

7.1.4.11 AddAlarmListener

Meldet einen Alarmbeobachter an.

Syntax

```
virtual HRESULT TCOMAPI AddAlarmListener (ITcMessageListener ipListener, ITcEventFilterConfig pipFil
terConfig)
```

Parameter

Name	Typ	Beschreibung
ipListener	ITcMessageListener	Zeiger auf die ITcMessageListener-Schnittstelle.
pipFilterConfig	ITcEventFilterConfig	Zeiger auf einen ITcEventFilterConfig-Zeiger.

Rückgabewert

Typ	Beschreibung
HRESULT	Liefert S_OK, wenn der Methodenaufruf erfolgreich war, ansonsten ein HRESULT als Fehlercode.

7.1.4.12 RemoveAlarmListener

Meldet einen Alarmbeobachter ab.

Syntax

```
virtual HRESULT TCOMAPI RemoveAlarmListener (ITcMessageListener ipListener)
```

Parameter

Name	Typ	Beschreibung
ipListener	ITcMessageListener	Zeiger auf die ITcMessageListener-Schnittstelle.

Rückgabewert

Typ	Beschreibung
HRESULT	Liefert S_OK, wenn der Methodenaufruf erfolgreich war, ansonsten ein HRESULT als Fehlercode.

7.1.4.13 NotifyAlarmListener

Arbeitet eine Queue für einen Alarmbeobachter ab.

Syntax

```
virtual HRESULT TCOMAPI NotifyMessageListener (ITcMessageListener ipListener)
```

Parameter

Name	Typ	Beschreibung
ipListener	ITcMessageListener	Zeiger auf die ITcMessageListener-Schnittstelle.

Rückgabewert

Typ	Beschreibung
HRESULT	Liefert S_OK, wenn der Methodenaufruf erfolgreich war, ansonsten ein HRESULT als Fehlercode.

7.1.4.14 GetEventText

Liefert eine Text zu einem Ereignis.

Syntax

```
virtual HRESULT TCOMAPI GetEventText (GUID eventClass, UDINT eventId, ITcSourceInfo ipSourceInfo, ITcArguments ipArguments, DINT nLangId, ITcAsyncStringResult pipResult)
```

Parameter

Name	Typ	Beschreibung
eventClass	REFERENCE TO GUID	Referenz auf die GUID der Ereignisklasse.
eventId	REFERENCE TO UDINT	Referenz auf die ID des Ereignisses.
ipSourceInfo	ITcSourceInfo	Zeiger auf die ITcSourceInfo-Schnittstelle.
ipArguments	ITcArguments	Zeiger auf die ITcArguments-Schnittstelle.
nLangId	DINT	Sprach-ID (LCID) der angefragten Sprache.
pipResult	POINTER TO ITcAsyncStringResult	Referenz auf einen ITcAsyncStringResult-Zeiger.

Rückgabewert

Typ	Beschreibung
HRESULT	Liefert S_OK, wenn der Methodenaufruf erfolgreich war, ansonsten ein HRESULT als Fehlercode.

7.1.4.15 GetEventClassName

Liefert den Klassennamen zu einem Ereignis.

Syntax

```
virtual HRESULT TCOMAPI GetEventClassName (GUID eventClass, DINT nLangId, ITcAsyncStringResult pipResult)
```

Parameter

Name	Typ	Beschreibung
eventClass	REFERENCE TO GUID	Referenz auf die GUID der Ereignisklasse.
nLangId	DINT	Sprach-ID (LCID) der angefragten Sprache.
pipResult	POINTER TO ITcAsyncStringResult	Referenz auf einen ITcAsyncStringResult-Zeiger.

Rückgabewert

Typ	Beschreibung
HRESULT	Liefert S_OK, wenn der Methodenaufruf erfolgreich war, ansonsten ein HRESULT als Fehlercode.

7.1.4.16 CreateArguments

Erstellt eine Instanz, die ITcArguments implementiert.

Syntax

```
virtual HRESULT TCOMAPI CreateArguments (ITcArguments ipArguments)
```

Parameter

Name	Typ	Beschreibung
ipArguments	ITcArguments	Zeiger auf die ITcArguments-Schnittstelle.

Rückgabewert

Typ	Beschreibung
HRESULT	Liefert S_OK, wenn der Methodenaufruf erfolgreich war, ansonsten ein HRESULT als Fehlercode.

7.2 Datentypen

7.2.1 TcEventEntry

Definiert ein Ereignis (Event) mittels Ereignisklasse, Ereignis-ID und Severity.

Syntax

Definition:

```
typedef struct
{
    GUID             uuidEventClass;
    UDINT            nEventId;
    TcEventSeverity  eSeverity;
}TcEventEntry;
```

Parameter

Name	Typ	Beschreibung
uuidEventClass	GUID	GUID der Ereignisklasse.
nEventId	UDINT	ID des Ereignisses.
eSeverity	TcEventSeverity	Event-Severity definiert den Schweregrad des Ereignisses.

7.2.2 TcEventSeverity

Definiert die Severity des Ereignisses.

Syntax

Definition:

```
typedef enum
{
    Verbose   = 0,
    Info      = 1,
    Warning   = 2,
    Error     = 3,
    Critical  = 4
}TcEventSeverity;
```

7.2.3 TcEventConfirmationState

Definiert den Bestätigungszustand eines Alarms.

Syntax

Definition:

```
typedef enum
{
   NotSupported        = 0,
   NotRequired         = 1,
   WaitForConfirmation = 2,
   Confirmed           = 3,
   Reset               = 4
}TcEventConfirmationState;
```

8 Usermode API

Der EventLogger bietet eine Schnittstelle an, um aus Usermode-Programmen sowohl Nachrichten abzusenden als auch abgesendete Events zu empfangen.

Beckhoff.TwinCAT.TcEventLoggerAdsProxy.Net von NuGet.org

Die API steht auf NuGet.org über das Paket Beckhoff.TwinCAT.TcEventLoggerAdsProxy.Net zur Einbindung in Projekte bereit. Zum einfachen Einstieg finden Sie dort Beispiel-Code in der README-Datei, der lediglich in ein .NET Projekt kopiert werden muss.

COM basierte Schnittstelle wird ersetzt

Die COM basierte Schnittstelle, welche hier zuvor beschrieben ist, wird bis TwinCAT 3.1 4024 unterstützt. Aufgrund der genutzten Technologie kann sie nicht unter TwinCAT/BSD angeboten werden.
Die auf NuGet.org bereitgestellte API ist der Nachfolger und bietet durch eine äquivalente API eine einfache Portierung für Kundenanwendungen.

8.1 Klassen

8.1.1 TcEventLogger

Diese Klasse stellt die Verbindung zu einem TwinCAT 3 Eventlogger dar.

Syntax

```
public class: ITcEventLogger2, _ITcEventLoggerEvents
```

Konstruktor

Initialisiert eine neue Instanz der Klasse TcEventLogger Class.

```
public TcEventLoggerClass();
```

Aufruf:

```
TcEventLogger logger : new TcEventLogger();
```

Schnittstellen

Name	Beschreibung
ITcEventLogger2 [▶ 185]	Schnittstelle, um Kommandos an den Eventlogger zu senden.
_ITcEventLoggerEvents [▶ 169]	Diese Schnittstelle stellt die Benachrichtigungen für auftretende Ereignisse bereit.

Methoden

Name	Modifizierer	Rückgabetyp	Definitionsort	Beschreibung
ClearAllAlarms [▶ 185]	public virtual	void	ITcEventLogger/ ITcEventLogger2 [▶ 185]	Ruft Clear() für alle Alarme im Zustand „Raised“ auf.
ClearLoggedEvents [▶ 186]	public virtual	void	ITcEventLogger/ ITcEventLogger2 [▶ 185]	Ruft Clear() für alle Events, die aktuell im Zustand „Raised“ sind, auf.
ConfirmAllAlarms [▶ 186]	public virtual	void	ITcEventLogger/ ITcEventLogger2 [▶ 185]	Ruft Confirm() für alle Alarme mit dem Bestätigungszustand „WaitForConfirmation“ auf.
Connect [▶ 186]	public virtual	void	ITcEventLogger/ ITcEventLogger2 [▶ 185]	Verbindet sich mit dem Eventlogger des TwinCAT Systems auf einem gegebenen System.
Disconnect [▶ 186]	public virtual	void	ITcEventLogger/ ITcEventLogger2 [▶ 185]	Hebt die Verbindung mit einem Eventlogger des TwinCAT Systems auf.
GetEventClassName [▶ 186]	public virtual	string	ITcEventLogger/ ITcEventLogger2 [▶ 185]	Liefert den Klassennamen zu einem Ereignis.
GetLoggedEvents [▶ 187]	public virtual	TcEventLogger AdsProxyLib.TcLoggedEvent Collection	ITcEventLogger/ ITcEventLogger2 [▶ 185]	Fragt die im Cache befindlichen Events ab.
ITcEventLogger2_SendTcMessage [▶ 164]	public virtual	void	ITcEventLogger2 [▶ 185]	Sendet eine Nachricht.

Ereignisse

Name	Rückgabetyp	Definitionsort	Beschreibung
AlarmCleared	_ITcEventLogger Events_AlarmClearedEventHandler	_ITcEventLoggerEvents [▶ 169]	Wird aufgerufen, wenn ein Alarm in den Zustand „Not Raised" wechselt.
AlarmConfirmed	_ITcEventLogger Events_AlarmConfirmedEventHandler	_ITcEventLoggerEvents [▶ 169]	Wird aufgerufen, wenn ein Alarm in den Zustand „Confirmed" wechselt.
AlarmRaised	_ITcEventLogger Events_AlarmRaisedEventHandler	_ITcEventLoggerEvents [▶ 169]	Wird aufgerufen, wenn ein Alarm in den Zustand „Raised" wechselt.
MessageSent	_ITcEventLogger Events_MessageSentEventHandler	_ITcEventLoggerEvents [▶ 169]	Wird aufgerufen, wenn eine Nachricht gesendet wurde.

Eigenschaften

Name	Modifizierer	Typ	Zugriff	Definitionsort	Beschreibung
ActiveAlarms [▶ 187]	public virtual	TcAlarmCollection	get	ITcEventLogger/ ITcEventLogger2 [▶ 185]	Liefert eine Collection mit allen aktuell aktiven Alarmen zurück.
IsConnected [▶ 187]	public virtual	bool	get	ITcEventLogger/ ITcEventLogger2 [▶ 185]	Stellt die Verbindung dar, die mittels Connect aufgebaut wurde. Sollte regelmäßig überprüft werden.

8.1.1.1 ITcEventLogger_ClearAllAlarms

Diese Methode setzt alle Alarme, die im Zustand „Raised" sind, auf „Not Raised".

Syntax

```
public virtual void ITcEventLogger_ClearAllAlarms([bool bResetConfirmation = True])
```

Parameter

Name	Typ	Beschreibung
bResetConfirmation	bool	Bestimmt, ob die Bestätigungen für Alarme ausgelöst werden sollen.

8.1.1.2 ITcEventLogger_ClearLoggedEvents

Diese Methode löscht den Cache der Events.
Hierbei werden die aktuellen Alarmzustände nicht geändert.

Syntax

```
public virtual void ITcEventLogger_ClearLoggedEvents()
```

8.1.1.3 ITcEventLogger_ConfirmAllAlarms

Diese Methode bestätigt alle Alarme, die bestätigt werden müssen, die also im Zustand „WaitForConfirmation“ sind.

Syntax

```
public virtual void ITcEventLogger_ConfirmAllAlarms()
```

8.1.1.4 ITcEventLogger_Connect

Diese Methode verbindet das Objekt mit einem Eventlogger auf einem Laufzeitsystem anhand der AmsNetId.

Syntax

```
public virtual void ITcEventLogger_Connect([string Address = localhost])
```

Parameter

Name	Typ	Beschreibung
Address	string	Definiert die AmsNetId.

8.1.1.5 ITcEventLogger_Disconnect

Diese Methode trennt die Verbindung des Objekts mit dem Eventlogger.

Syntax

```
public virtual void ITcEventLogger_Disconnect()
```

8.1.1.6 ITcEventLogger_GetEventClassName

Diese Methode liefert zu einer gegebenen Ereignisklassen-GUID und Sprache den passenden EventClass-Namen.

Syntax

```
public virtual string ITcEventLogger_GetEventClassName(System.Guid EventClass, int nLangId)
```

Parameter

Name	Typ	Beschreibung
EventClass	System.Guid	GUID der Ereignisklasse.
langId	int	LangID, in der der Name geliefert werden soll.

Rückgabewert

Name	Typ	Beschreibung
ITcEventLogger_GetEventClassName	string	Name der Ereignisklasse.

8.1.1.7 ITcEventLogger_GetLoggedEvents

Diese Methode liefert eine Collection von gespeicherten, letzten Ereignissen.

Syntax

```
public virtual TcLoggedEventCollection ITcEventLogger_GetLoggedEvents(uint nMaxEntiries)
```

Parameter

Name	Typ	Beschreibung
nMaxEntiries	uint	Anzahl der maximal zu liefernden Ereignisse.

Rückgabewert

Name	Typ	Beschreibung
ITcEventLogger_GetLoggedEvents	TcLoggedEventCollection	Eine Sammlung der letzten Ereignisse.

8.1.1.8 ITcEventLogger_GetText

Diese Methode liefert den Text eines Ereignisses.

Syntax

```
public virtual string ITcEventLogger_GetText(System.Guid EventClass, uint EventId, uint objectId,
TcEventLoggerAdsProxyLib.TcEventArgumentsInfo pArgInfo, System.IntPtr pArgData, int nLangId)
```

Parameter

Name	Typ	Beschreibung
EventClass	System.Guid	GUID der Ereignisklasse.
EventId	uint	ID des Ereignisses
objectId	uint	
pArgInfo	TcEventLoggerAdsProxyLib.TcEventArgumentsInfo	
pArgData	System.IntPtr	
langId	int	LangID, in der der Name geliefert werden soll.

Rückgabewert

Name	Typ	Beschreibung
ITcEventLogger_GetText	string	Text des Ereignisses.

8.1.1.9 ITcEventLogger_ActiveAlarms

Dieses Eigenschaftsfeld liefert eine Collection von den aktuellen aktiven Alarmen zurück (aktiver Zustand ist dabei „Raised“ oder „WaitingForConfirmation“).

Syntax

```
public virtual TcEventLoggerAdsProxyLib.TcAlarmCollection ITcEventLogger_ActiveAlarms
```

8.1.1.10 ITcEventLogger_IsConnected

Diese Eigenschaft zeigt an, ob das TcEventLogger-Objekt aktuell mit einem Zielsystem verbunden ist. Dieses sollte regelmäßig überprüft werden, um auf einen Verbindungsverlust reagieren zu können.

Syntax

```
public virtual bool ITcEventLogger_IsConnected
```

8.1.1.11 ITcEventLogger2_SendTcMessage

Diese Methode sendet eine Nachricht.

Syntax

```
public virtual void SendTcMessage(System.Guid EventClass, uint EventId,
TcEventLoggerAdsProxyLib.SeverityLevelEnum severity, string JsonAttribute,
TcEventLoggerAdsProxyLib.TcSourceInfo pSourceInfo, TcEventLoggerAdsProxyLib.TcArguments pArguments)
```

Parameter

Name	Typ	Beschreibung
EventClass	System.Guid	GUID der Ereignisklassen
EventId	uint	ID des Ereignisses
Severity	TcEventLoggerAdsProxyLib.SeverityLevelEnum [▶ 199]	Severity des Ereignisses
JsonAttribute	string	JSON-Attribut
pSourceInfo	TcEventLoggerAdsProxyLib.TcSourceInfo	Referenz auf die Quellen-Informationen
pArguments	TcArguments	Referenz auf die Argumente.

8.1.2 TcArguments

Mit dieser Klasse können Argumente eines Ereignisses definiert werden. Dafür wird die ITcArguments-Schnittstelle implementiert.

Syntax

```
public class: ITcArguments
```

Konstruktor

Initialisiert eine neue Instanz der Klasse TcArguments.

```
public TcArguments();
```

Aufruf:

```
TcArguments args : new TcArguments();
```

Schnittstellen

Name	Beschreibung
ITcArguments	Schnittstelle, um die Argumente zu beschreiben.

Methoden

Name	Modifizierer	Rückgabetyp	Definitionsort	Beschreibung
Add [▶ 165]	public virtual	void	ITcEventLogger/ ITcArguments	Fügt ein Argument hinzu.
AddV [▶ 166]	public virtual	void	ITcEventLogger/ ITcArguments	Fügt einen Array von Argumenten hinzu.
Clear [▶ 166]	public virtual	void	ITcEventLogger/ ITcArguments	Löscht alle Argumente.
GetEnumerator [▶ 166]	public virtual	System.Collections.IEnumerator	ITcEventLogger/ ITcArguments	Liefert eine Aufzählung über die Argumente.
Remove [▶ 166]	public virtual	void	ITcEventLogger/ ITcArguments	Löscht ein Argument.
Set [▶ 179]	public virtual	void	ITcEventLogger/ ITcArguments	Setzt ein Argument.

Eigenschaften

Name	Modifizierer	Typ	Zugriff	Definitionsort	Beschreibung
Count [▶ 167]	public virtual	int	get	ITcEventLogger/ ITcArguments	Anzahl der Argumente.
this [uint]	public virtual	ITcLoggedEvent	get	ITcEventLogger/ ITcArguments	Das Ereignis, auf welches sich die Argumente beziehen.

8.1.2.1 Add

Diese Methode fügt ein Argument hinzu.

Syntax

```
public virtual void Add(object Item)
```

Parameter

Name	Typ	Beschreibung
Item	object	Das hinzuzufügende Argument.

8.1.2.2 AddV

Diese Methode fügt einen Array von Argumenten hinzu.

Syntax

```
public virtual void AddV(params object[] args)
```

Parameter

Name	Typ	Beschreibung
args	params object[]	Array der hinzuzufügenden Argumente.

8.1.2.3 Clear

Diese Methode löscht alle Argumente.

Syntax

```
public virtual void Clear()
```

8.1.2.4 GetEnumerator

Diese Methode liefert eine Aufzählung über die Argumente.

Syntax

```
public virtual System.Collections.IEnumerator GetEnumerator()
```

Rückgabewert

Name	Typ	Beschreibung
GetEnumerator	System.Collections.IEnumerator	Aufzählung über die Argumente.

8.1.2.5 Remove

Diese Methode löscht ein Argument.

Syntax

```
public virtual void Remove(uint Index)
```

Parameter

Name	Typ	Beschreibung
Index	uint	Index des zu löschenden Arguments.

8.1.2.6 Set

Diese Methode setzt ein Argument.

Syntax

```
public virtual void Set(uint Index, object Item)
```

Parameter

Name	Typ	Beschreibung
Index	uint	Index des zu setzenden Arguments.
Item	object	Das neue Argument.

8.1.2.7 Count

Diese Eigenschaft liefert die Anzahl.

Syntax

```
public virtual int Count
```

8.1.2.8 this [uint]

Diese Eigenschaft ist das Ereignis, auf welches sich die Argumente beziehen.

Syntax

```
public virtual TcEventLoggerAdsProxyLib.TcArgumentEntry this[uint Index]
```

8.1.3 TcSourceInfo

Diese Klasse beschreibt die Quelleninformation eines Ereignisses.

Syntax

```
public class: ITcSourceInfo
```

Konstruktor

Initialisiert eine neue Instanz der Klasse TcSourceInfo.

```
public TcArguments();
```

Aufruf:

```
TcSourceInfo souceInfo: new SourceInfo();
```

Methoden

Name	Modifizierer	Rückgabetyp	Definitionsort	Beschreibung
IsSourceInfoTypeSupported [▶ 168]	public virtual	bool	ITcEventLogger/ ITcSourceInfo	Abfragemöglichkeit für die Typen der Quelleninformation.

Eigenschaften

Name	Modifizierer	Typ	Zugriff	Definitionsort	Beschreibung
Count [▶ 168]	public virtual	int	get	ITcEventLogger/ ITcSourceInfo	Anzahl der SourceInfoTypen.
Guid [▶ 168]	public virtual	System.Guid	get, set	ITcEventLogger/ ITcSourceInfo	GUID der Quelle.
Id [▶ 168]	public virtual	uint	get, set	ITcEventLogger/ ITcSourceInfo	Id der Quelle.
Name [▶ 169]	public virtual	string	get, set	ITcEventLogger/ ITcSourceInfo	Name der Quelle.

8.1.3.1 GetData

Diese Methode liefert die Daten zu einer Quelle.

Syntax

```
public virtual void GetData(uint Index, out TcEventLoggerAdsProxyLib.TcSourceInfoTypeEnum pInfoType,
 System.IntPtr pData, out uint cbData)
```

Parameter

Name	Typ	Beschreibung
Index	uint	Index der Quelle.
pInfoType	TcEventLoggerAdsProxyLib.TcSourceInfoTypeEnum [▶ 200]	Referenz auf die Typ-Information.
pData	System.IntPtr	Referenz auf die Daten.
cbData	uint	Länge der Daten.

8.1.3.2 IsSourceInfoTypeSupported

Mit dieser Methode kann geprüft werden, ob der Quelleninformationstyp definiert wurde.

Syntax

```
public virtual bool IsSourceInfoTypeSupported(TcSourceInfoTypeEnum infoType)
```

Parameter

Name	Typ	Beschreibung
infoType	TcSourceInfoTypeEnum [▶ 200]	Anfrage Typ der Quelle

Rückgabewert

Name	Typ	Beschreibung
IsSourceInfoTypeSupported	bool	Information, ob unterstützt.

8.1.3.3 Count

Diese Eigenschaft liefert die Anzahl.

Syntax

```
public virtual int Count
```

8.1.3.4 Guid

Diese Eigenschaft liefert die GUID der Quelle.

Syntax

```
public virtual System.Guid Guid
```

8.1.3.5 Id

Diese Eigenschaft liefert die Id.

Syntax

```
public virtual uint Id
```

8.1.3.6 Name

Diese Eigenschaft liefert den Namen.

Syntax

```
public virtual string Name
```

8.2 Schnittstellen

8.2.1 _ITcEventLoggerEvents

Diese Schnittstelle stellt die Benachrichtigungen für auftretende Ereignisse bereit.

Syntax

```
public interface _ITcEventLoggerEvents
```

Methoden

Name	Modifizierer	Rückgabetyp	Beschreibung
AlarmCleared [▶ 169]	public virtual	void	Wird aufgerufen, wenn ein Alarm in den Zustand „Not Raised" wechselt.
AlarmConfirmed [▶ 169]	public virtual	void	Wird aufgerufen, wenn ein Alarm in den Zustand „Confirmed" wechselt.
AlarmRaised [▶ 170]	public virtual	void	Wird aufgerufen, wenn ein Alarm in den Zustand „Raised" wechselt.
MessageSend [▶ 170]	public virtual	void	Wird aufgerufen, wenn eine Nachricht gesendet wurde.

8.2.1.1 AlarmCleared

Wird aufgerufen, wenn ein Alarm in den Zustand „Not Raised" wechselt.

Syntax

```
public virtual void AlarmCleared(TcAlarm evtObj, bool bRemove)
```

Parameter

Name	Typ	Beschreibung
evtObj	TcAlarm	Der Alarm
bRemove	bool	TRUE = Der Alarm ist im Zustand „Not-Raised“ und ist nicht im Zustand „Wait for Confirmation“. FALSE = Der Alarm wartet weiterhin auf eine Confirmation.

8.2.1.2 AlarmConfirmed

Wird aufgerufen, wenn ein Alarm in den Zustand „Confirmed" wechselt.

Syntax

```
public virtual void AlarmConfirmed(TcAlarm evtObj, bool bRemove)
```

Parameter

Name	Typ	Beschreibung
evtObj	TcAlarm	Der Alarm
bRemove	bool	TRUE, wenn der Alarm nicht im Zustand „Raised“ ist. FALSE, wenn der Alarm im Zustand „Raised“ ist.

8.2.1.3 AlarmRaised

Wird aufgerufen, wenn ein Alarm in den Zustand „Raised" wechselt.

Syntax

```
public virtual void AlarmRaised(TcAlarm evtObj)
```

Parameter

Name	Typ	Beschreibung
evtObj	TcAlarm	Der Alarm

8.2.1.4 MessageSend

Wird aufgerufen, wenn eine Nachricht gesendet wurde.

Syntax

```
void MessageSent(TcMessage evtObj)
```

Parameter

Name	Typ	Beschreibung
evtObj	TcMessage	Die Nachricht

8.2.2 ITcAlarm3

Diese Schnittstelle repräsentiert einen Alarm.

Syntax

```
public interface ITcAlarm3
```

Methoden

Name	Modifizierer	Rückgabetyp	Beschreibung
Confirm [▶ 171]	public virtual	void	Bestätigt den Alarm.
GetCauseRemedy [▶ 173]	Public virtual	TcCauseRemedyCollection	Liefert Informationen zur Ursache und Hilfe (wenn diese definiert sind).
GetDetails [▶ 174]	Public virtual	TcDetailCollection	Liefert die Details.
GetEventClassName [▶ 172]	public virtual	string	Liefert den Ereignisklassen-Namen.
GetText [▶ 173]	public virtual	string	Liefert den Ereignis-Text inkl. der Argumente.
IsSourceInfoTypeSupported [▶ 173]	public virtual	bool	Abfragemöglichkeit für die Typen der Quellen-Information.

Eigenschaften

Name	Modifizierer	Typ	Zugriff	Beschreibung
ConfirmationState [▶ 173]	public virtual	ConfirmationStateEnum	get	Liefert den Confirmation-State [▶ 198].
EventClass [▶ 174]	public virtual	System.Guid	get	Liefert die Ereignisklassen-GUID.
EventId [▶ 174]	public virtual	uint	get	Liefert die Ereignis-ID.
EventType [▶ 174]	public virtual	EventTypeEnum	get	Liefert den Typ des Ereignisses [▶ 198].
FileTimeCleared [▶ 174]	public virtual	long	get	Zeitstempel, wann der Alarm in den Zustand Cleared überführt wurde.
FileTimeConfirmed [▶ 174]	public virtual	long	get	Zeitstempel, wann der Alarm bestätigt wurde.
FileTimeRaised [▶ 175]	public virtual	long	get	Zeitstempel, wann der Alarm in den Zustand Raised überführt wurde.
IsRaised [▶ 175]	public virtual	bool	get	Zeigt an, ob der Alarm in den Zustand Raised überführt worden ist.
JsonAttribute [▶ 175]	public virtual	string	get	Das JSON-Attribut.
SeverityLevel [▶ 175]	public virtual	SeverityLevelEnum	get	Das Severity-Level [▶ 199].
SourceGuid [▶ 175]	public virtual	System.Guid	get	Liefert die GUID der Quelle.
SourceId [▶ 175]	public virtual	uint	get	Liefert die ID der Quelle.
SourceName [▶ 175]	public virtual	string	get	Liefert den Namen der Quelle.
TimeCleared [▶ 175]	public virtual	System.DateTime	get	Zeitstempel, wann der Alarm in den Zustand Cleared überführt wurde.
TimeConfirmed [▶ 176]	public virtual	System.DateTime	get	Zeitstempel, wann der Alarm bestätigt wurde.
TimeRaised [▶ 176]	public virtual	Sytem.DateTime	get	Zeitstempel, wann der Alarm in den Zustand Raised überführt wurde.

8.2.2.1 Confirm

Diese Methode bestätigt den Alarm.

Syntax

```
public virtual void Confirm()
```

8.2.2.2 GetArgumentData

Nur zur internen Realisierung

Diese Methode ist nur für die interne Realisierung und nicht für die Verwendung durch Applikationen vorgesehen. Zur programmatischen Nutzung können die Argumente als JsonAttribute zusätzlich angehangen werden.

Diese Methode liefert die Daten der Argumente.

Syntax

```
public virtual void GetArgumentData(ref System.IntPtr ppArgData, uint size)
```

Parameter

Name	Typ	Beschreibung
ppArgData	ref System.IntPtr	Referenz auf die Daten.
size	Uint	Größe

8.2.2.3 GetArgumentInfo

Nur zur internen Realisierung

Diese Methode ist nur für die interne Realisierung und nicht für die Verwendung durch Applikationen vorgesehen. Zur programmatischen Nutzung können die Argumente als JsonAttribute zusätzlich angehangen werden.

Diese Methode liefert die (Typ-)Infos zu den Argumenten.

Syntax

```
public virtual void GetArgumentInfo(ref TcEventArgumentsInfo pArgInfo)
```

Parameter

Name	Typ	Beschreibung
pArgInfo	ref TcEventArgumentsInfo	Referenz auf die bereitgestellten Informationen.

8.2.2.4 GetEventClassName

Diese Methode liefert den Ereignisklassen-Namen.

Syntax

```
public virtual string GetEventClassName(int langId)
```

Parameter

Name	Typ	Beschreibung
langId	int	LangID der Sprache

Rückgabewert

Name	Typ	Beschreibung
GetEventClassName	string	Namen der Ereignisklasse

8.2.2.5 GetText

Diese Methode liefert den Ereignis-Text inkl. der Argumente.

Syntax

```
public virtual string GetText(int langId)
```

Parameter

Name	Typ	Beschreibung
langId	int	LangID der Sprache

Rückgabewert

Name	Typ	Beschreibung
GetText	string	Ereignis-Text

8.2.2.6 IsSourceInfoTypeSupported

Mit dieser Methode kann geprüft werden, ob der Quelleninformationstyp definiert wurde.

Syntax

```
public virtual bool IsSourceInfoTypeSupported(TcSourceInfoTypeEnum infoType)
```

Parameter

Name	Typ	Beschreibung
infoType	TcSourceInfoTypeEnum [▶ 200]	Anfrage Typ der Quelle

Rückgabewert

Name	Typ	Beschreibung
IsSourceInfoTypeSupported	bool	Information, ob unterstützt.

8.2.2.7 GetCauseRemedy

Diese Methode liefert die Ursachen-/Hilfe-Informationen, falls diese definiert wurden.

Syntax

```
public virtual TcCauseRemedyCollection GetCauseRemedy(int langId)
```

Parameter

Name	Typ	Beschreibung
langId	int	LangID der Sprache.

Rückgabewert

Name	Typ	Beschreibung
GetCauseRemedy	TcCauseRemedyCollection	Collection der Cause/Remedy Informationen.

8.2.2.8 ConfirmationState

Diese Eigenschaft liefert den Confirmation-State.

Syntax

```
Public virtual ConfirmationStateEnum ConfirmationState
```

8.2.2.9 GetDetails

Diese Methode liefert die Details.

Syntax

```
public virtual TcDetailCollection GetDetails(int langId)
```

Parameter

Name	Typ	Beschreibung
langId	int	LangID der Sprache

Rückgabewert

Name	Typ	Beschreibung
GetDetails	TcDetailCollection	Collection der Details

8.2.2.10 EventClass

Diese Eigenschaft liefert die Ereignisklassen-GUID.

Syntax

```
public virtual System.Guid EventClass
```

8.2.2.11 EventId

Diese Eigenschaft liefert die Ereignis-ID.

Syntax

```
public virtual uint EventId
```

8.2.2.12 EventType

Diese Eigenschaft liefert den Typ des Ereignisses.

Syntax

```
public virtual EventTypeEnum EventType
```

8.2.2.13 FileTimeCleared

Diese Eigenschaft liefert den Zeitstempel, wann der Alarm in den Zustand Cleared überführt wurde.

Syntax

```
public virtual long FileTimeCleared
```

8.2.2.14 FileTimeConfirmed

Diese Eigenschaft liefert den Zeitstempel, wann der Alarm in den Zustand Confirmed überführt wurde.

Syntax

```
public virtual long FileTimeConfirmed
```

8.2.2.15 FileTimeRaised

Diese Eigenschaft liefert den Zeitstempel, wann der Alarm in den Zustand Raised überführt wurde.

Syntax

```
public virtual long FileTimeRaised
```

8.2.2.16 IsRaised

Diese Eigenschaft zeigt an, ob der Alarm in den Zustand Raised überführt worden ist.

Syntax

```
public virtual bool IsRaised
```

8.2.2.17 JsonAttribute

Diese Eigenschaft liefert das JSON-Attribut.

Syntax

```
public virtual string JsonAttribute
```

8.2.2.18 SeverityLevel

Diese Eigenschaft liefert das Severity-Level.

Syntax

```
public virtual TcEventLoggerAdsProxyLib.SeverityLevelEnum SeverityLevel
```

8.2.2.19 SourceGuid

Diese Eigenschaft liefert die GUID der Quelle.

Syntax

```
public virtual System.Guid SourceGuid
```

8.2.2.20 SourceId

Diese Eigenschaft liefert die ID der Quelle.

Syntax

```
public virtual uint SourceId
```

8.2.2.21 SourceName

Diese Eigenschaft liefert den Namen der Quelle.

Syntax

```
public virtual string SourceName
```

8.2.2.22 TimeCleared

Diese Eigenschaft liefert den Zeitstempel, wann der Alarm in den Zustand Gecleared überführt wurde.

Syntax

```
public virtual System.DateTime TimeCleared
```

8.2.2.23 TimeConfirmed

Diese Eigenschaft liefert den Zeitstempel, wann der Alarm in den Zustand Confirmed überführt wurde.

Syntax

```
public virtual System.DateTime TimeConfirmed
```

8.2.2.24 TimeRaised

Diese Eigenschaft liefert den Zeitstempel, wann der Alarm in den Zustand Raised überführt wurde.

Syntax

```
public virtual System.DateTime TimeRaised
```

8.2.3 ITcArgumentEntry

Diese Schnittstelle beschreibt ein Argument.

Syntax

```
public interface ITcArgumentEntry
```

Methoden

Name	Modifizierer	Rückgabetyp	Beschreibung
Get [▶ 176]	public virtual	dynamic	Liefert den Wert als dynamischen Typ.
GetBoolean [▶ 177]	public virtual	bool	Liefert den Wert als Boolean.
GetDouble [▶ 177]	public virtual	void	Liefert den Wert als Double.
GetFloat [▶ 177]	public virtual	float	Liefert den Wert als Float.
GetInt16 [▶ 178]	public virtual	short	Liefert den Wert als Short.
GetInt32 [▶ 178]	public virtual	int	Liefert den Wert als Int.
GetInt64 [▶ 178]	public virtual	long	Liefert den Wert als Long.
GetInt8 [▶ 178]	public virtual	sbyte	Liefert den Wert als SByte.
GetString [▶ 178]	public virtual	string	Liefert den Wert als String.
GetUInt16 [▶ 179]	public virtual	ushort	Liefert den Wert als UShort.
GetUInt32 [▶ 179]	public virtual	uint	Liefert den Wert als UInt.
GetUInt64 [▶ 179]	public virtual	ulong	Liefert den Wert als ULong.
GetUInt8 [▶ 179]	public virtual	byte	Liefert den Wert als Byte.
Set [▶ 179]	public virtual	void	Setzt den Wert.

8.2.3.1 Get

Diese Methode liefert den Wert als dynamischen Typ.

Syntax

```
public virtual Object Get()
```

Rückgabewert

Name	Typ	Beschreibung
Get	Objekt	Rückgabe ist ein Objekt vom Typ entsprechend des Arguments.

8.2.3.2 GetBoolean

Diese Methode liefert den Wert als Boolean.

Syntax

```
public virtual bool GetBoolean()
```

Rückgabewert

Name	Typ	Beschreibung
GetBoolean	bool	Rückgabe

8.2.3.3 GetData

Diese Methode liefert den Wert als Referenz auf Daten.

Syntax

```
public virtual void GetData(out TcEventArgumentTypeEnum pInfoType, System.IntPtr pData, out uint
cbData)
```

Parameter

Name	Typ	Beschreibung
pInfoType	TcEventArgumentTypeEnum [▶ 199]	Referenz auf den Typ der Daten.
pData	System.IntPtr	Referenz auf die Daten.
cbData	uint	Länge der Daten.

8.2.3.4 GetDouble

Diese Methode liefert den Wert als Double.

Syntax

```
public virtual double GetDouble
```

Rückgabewert

Name	Typ	Beschreibung
GetDouble	double	Rückgabe

8.2.3.5 GetFloat

Diese Methode liefert den Wert als Float.

Syntax

```
public virtual float GetFloat()
```

Rückgabewert

Name	Typ	Beschreibung
GetFloat	float	Rückgabe

8.2.3.6 GetInt16

Diese Methode liefert den Wert als Short.

Syntax

```
public virtual short GetInt16()
```

Rückgabewert

Name	Typ	Beschreibung
GetInt16	short	Rückgabe

8.2.3.7 GetInt32

Diese Methode liefert den Wert als Int.

Syntax

```
public virtual int GetInt32()
```

Rückgabewert

Name	Typ	Beschreibung
GetInt32	int	Rückgabe

8.2.3.8 GetInt64

Diese Methode liefert den Wert als Long.

Syntax

```
public virtual long GetInt64()
```

Rückgabewert

Name	Typ	Beschreibung
GetInt64	long	Rückgabe

8.2.3.9 GetInt8

Diese Methode liefert den Wert als SByte.

Syntax

```
public virtual sbyte GetInt8()
```

Rückgabewert

Name	Typ	Beschreibung
GetInt8	sbyte	Rückgabe

8.2.3.10 GetString

Diese Methode liefert den Wert als String.

Syntax

```
public virtual string GetString()
```

Rückgabewert

Name	Typ	Beschreibung
GetString	string	Rückgabe

8.2.3.11 GetUInt16

Diese Methode liefert den Wert als UShort.

Syntax

```
public virtual ushort GetUInt16()
```

Rückgabewert

Name	Typ	Beschreibung
GetUInt16	ushort	Rückgabe

8.2.3.12 GetUInt32

Diese Methode liefert den Wert als UInt.

Syntax

```
public virtual uint GetUInt32()
```

Rückgabewert

Name	Typ	Beschreibung
GetUInt32	uint	Rückgabe

8.2.3.13 GetUInt64

Diese Methode liefert den Wert als ULong.

Syntax

```
public virtual ulong GetUInt64()
```

Rückgabewert

Name	Typ	Beschreibung
GetUInt64	ulong	Rückgabe

8.2.3.14 GetUInt8

Diese Methode liefert den Wert als Byte.

Syntax

```
public virtual byte GetUInt8()
```

Rückgabewert

Name	Typ	Beschreibung
GetUInt8	byte	Rückgabe

8.2.3.15 Set

Diese Methode setzt den Wert.

Syntax

```
public virtual void Set(object Item)
```

Parameter

Name	Typ	Beschreibung
Item	object	Neues Argument. Ein Objekt vom Typ entsprechend des Arguments.

8.2.4 ITcCauseRemedy

Diese Schnittstelle beschreibt die Cause/Remedy Informationen eines Ereignisses.

Syntax

```
public interface ITcCauseRemedy
```

Eigenschaften

Name	Modifizierer	Typ	Zugriff	Beschreibung
Cause [▶ 180]	public virtual	string	get	Die Ursache.
Id [▶ 180]	public virtual	uint	get	Die ID.
Remedy [▶ 180]	public virtual	string	get	Die Behebung.

8.2.4.1 Cause

Diese Eigenschaft liefert die Ursache.

Syntax

```
public virtual string Cause
```

8.2.4.2 Id

Diese Eigenschaft liefert die Id.

Syntax

```
public virtual uint Id
```

8.2.4.3 Remedy

Diese Eigenschaft liefert die Behebung.

Syntax

```
public virtual string Remedy
```

8.2.5 ITcDetail

Diese Schnittstelle beschreibt die Details eines Ereignisses.

Syntax

```
public interface ITcDetail
```

Eigenschaften

Name	Modifizierer	Typ	Zugriff	Beschreibung
Comment [▶ 181]	Public virtual	string	get	Der Kommentar
Name [▶ 181]	public virtual	string	get	Der Name
text [▶ 181]	Public virtual	string	get	Der Text

8.2.5.1 Comment

Diese Eigenschaft liefert den Kommentar zu einem Detail.

Syntax

```
public virtual string Comment
```

8.2.5.2 Name

Diese Eigenschaft liefert den Namen.

Syntax

```
public virtual string Name
```

8.2.5.3 text

Diese Eigenschaft liefert den Text zu einem Detail.

Syntax

```
public virtual string text
```

8.2.6 ITcEvent

Diese Schnittstelle beschreibt ein Ereignis. Dieses Ereignis kann entweder eine Message oder ein Alarm sein.

Syntax

```
public interface ITcEvent
```

Methoden

Name	Modifizierer	Rückgabetyp	Beschreibung
GetArgumentData [▶ 182]	public virtual	void	Liefert die Daten der Argumente.
GetArgumentInfo [▶ 182]	public virtual	void	Liefert die (Typ-)Infos zu den Argumenten.
GetEventClassName [▶ 183]	public virtual	string	Liefert den Ereignisklassen-Namen.
GetText [▶ 183]	public virtual	string	Liefert den Ereignis-Text inkl. der Argumente.
IsSourceInfoTypeSupported [▶ 183]	public virtual	bool	Abfragemöglichkeit für die Typen der Quellen-Information.

Eigenschaften

Name	Modifizierer	Typ	Zugriff	Beschreibung
EventClass [▶ 184]	public virtual	System.Guid	get	Liefert die Ereignisklassen-GUID.
EventId [▶ 184]	public virtual	uint	get	Liefert die Ereignis-ID.
EventType [▶ 184]	public virtual	EventTypeEnum	get	Liefert den Typ des Ereignisses [▶ 198].
FileTimeRaised [▶ 184]	public virtual	long	get	Zeitstempel, wann das Ereignis gesendet wurde.
SeverityLevel [▶ 184]	public virtual	SeverityLevelEnum	get	Das Severity-Level [▶ 199].
SourceGuid [▶ 184]	public virtual	System.Guid	get	Liefert die GUID der Quelle.
SourceId [▶ 184]	public virtual	uint	get	Liefert die ID der Quelle.
SourceName [▶ 185]	public virtual	string	get	Liefert den Namen der Quelle.
TimeRaised [▶ 185]	public virtual	System.DateTime	get	Zeitstempel, wann das Ereignis gesendet wurde.

8.2.6.1 GetArgumentData

Nur zur internen Realisierung

Diese Methode ist nur für die interne Realisierung und nicht für die Verwendung durch Applikationen vorgesehen. Zur programmatischen Nutzung können die Argumente als JsonAttribute zusätzlich angehangen werden.

Diese Methode liefert die Daten der Argumente.

Syntax

```
public virtual void GetArgumentData(ref System.IntPtr ppArgData, uint size)
```

Parameter

Name	Typ	Beschreibung
ppArgData	ref System.IntPtr	Referenz auf die Daten.
size	Uint	Größe

8.2.6.2 GetArgumentInfo

Nur zur internen Realisierung

Diese Methode ist nur für die interne Realisierung und nicht für die Verwendung durch Applikationen vorgesehen. Zur programmatischen Nutzung können die Argumente als JsonAttribute zusätzlich angehangen werden.

Diese Methode liefert die (Typ-)Infos zu den Argumenten.

Syntax

```
public virtual void GetArgumentInfo(ref TcEventArgumentsInfo pArgInfo)
```

Parameter

Name	Typ	Beschreibung
pArgInfo	ref TcEventArgumentsInfo	Referenz auf die bereitgestellten Informationen.

8.2.6.3 GetEventClassName

Diese Methode liefert den Ereignisklassen-Namen.

Syntax

```
public virtual string GetEventClassName(int langId)
```

Parameter

Name	Typ	Beschreibung
langId	int	LangID der Sprache

Rückgabewert

Name	Typ	Beschreibung
GetEventClassName	string	Namen der Ereignisklasse

8.2.6.4 GetText

Diese Methode liefert den Ereignis-Text inkl. der Argumente.

Syntax

```
public virtual string GetText(int langId)
```

Parameter

Name	Typ	Beschreibung
langId	int	LangID der Sprache

Rückgabewert

Name	Typ	Beschreibung
GetText	string	Ereignis-Text

8.2.6.5 IsSourceInfoTypeSupported

Mit dieser Methode kann geprüft werden, ob der Quelleninformationstyp definiert wurde.

Syntax

```
public virtual bool IsSourceInfoTypeSupported(TcSourceInfoTypeEnum infoType)
```

Parameter

Name	Typ	Beschreibung
infoType	TcSourceInfoTypeEnum [▶ 200]	Anfrage Typ der Quelle

Rückgabewert

Name	Typ	Beschreibung
IsSourceInfoTypeSupported	bool	Information, ob unterstützt.

8.2.6.6 EventClass

Diese Eigenschaft liefert die Ereignisklassen-GUID.

Syntax

```
public virtual System.Guid EventClass
```

8.2.6.7 EventId

Diese Eigenschaft liefert die Ereignis-ID.

Syntax

```
public virtual uint EventId
```

8.2.6.8 EventType

Diese Eigenschaft liefert den Typ des Ereignisses.

Syntax

```
public virtual EventTypeEnum EventType
```

8.2.6.9 FileTimeRaised

Diese Eigenschaft liefert den Zeitstempel, wann der Alarm in den Zustand Raised überführt wurde.

Syntax

```
public virtual long FileTimeRaised
```

8.2.6.10 SeverityLevel

Diese Eigenschaft liefert das Severity-Level.

Syntax

```
public virtual TcEventLoggerAdsProxyLib.SeverityLevelEnum SeverityLevel
```

8.2.6.11 SourceGuid

Diese Eigenschaft liefert die GUID der Quelle.

Syntax

```
public virtual System.Guid SourceGuid
```

8.2.6.12 SourceId

Diese Eigenschaft liefert die ID der Quelle.

Syntax

```
public virtual uint SourceId
```

8.2.6.13 SourceName

Diese Eigenschaft liefert den Namen der Quelle.

Syntax

```
public virtual string SourceName
```

8.2.6.14 TimeRaised

Diese Eigenschaft liefert den Zeitstempel, wann der Alarm in den Zustand Raised überführt wurde.

Syntax

```
public virtual System.DateTime TimeRaised
```

8.2.7 ITcEventLogger2

Schnittstelle, um Kommandos an den Eventlogger zu senden.

Syntax

```
public interface ITcEventLogger2
```

Methoden

Name	Modifizierer	Rückgabetyp	Beschreibung
ClearAllAlarms [▶ 185]	public virtual	void	Ruft Clear() für alle Alarme im Zustand „Raised“ auf.
ClearLoggedEvents [▶ 186]	public virtual	void	Löscht den Cache.
ConfirmAllAlarms [▶ 186]	public virtual	void	Ruft Confirm() für alle Alarme mit dem Bestätigungszustand „WaitForConfirmation“ auf.
Connect [▶ 186]	public virtual	void	Verbindet sich mit dem Eventlogger des TwinCAT Systems auf einem gegebenen System.
Disconnect [▶ 186]	public virtual	void	Hebt die Verbindung mit einem Eventlogger des TwinCAT Systems auf.
GetEventClassName [▶ 186]	public virtual	string	Liefert den Klassennamen zu einem Ereignis.
GetLoggedEvents [▶ 187]	public virtual	TcLoggedEventCollection	Fragt die im Cache befindlichen Events ab.

Eigenschaften

Name	Modifizierer	Typ	Zugriff	Beschreibung
ActiveAlarms [▶ 187]	public virtual	TcAlarmCollection	get	Liefert eine Collection mit allen aktuell aktiven Alarmen zurück.
IsConnected [▶ 187]	public virtual	bool	get	Stellt die Verbindung dar, die mittels Connect aufgebaut wurde. Sollte regelmäßig überprüft werden.

8.2.7.1 ClearAllAlarms

Ruft Clear() für alle Alarme im Zustand „Raised“ auf.

Syntax

```
public virtual void ClearAllAlarms([bool bResetConfirmation = True])
```

Parameter

Name	Typ	Beschreibung
bResetConfirmation	bool	Gibt an, ob die Alarme auch in den Zustand „Confirmed“ überführt werden sollen.

8.2.7.2 ClearLoggedEvents

Löscht den Cache.

Syntax

```
Public virtual void ClearLoggedEvents()
```

8.2.7.3 ConfirmAllAlarms

Ruft Confirm() für alle Alarme mit dem Bestätigungszustand „WaitForConfirmation“ auf.

Syntax

```
public virtual void ConfirmAllAlarms()
```

8.2.7.4 Connect

Verbindet sich mit dem Eventlogger des TwinCAT Systems auf einem gegebenen System.

Syntax

```
public virtual void Connect([string Address = localhost])
```

Parameter

Name	Typ	Beschreibung
Adress	string	AMS Net ID

8.2.7.5 Disconnect

Hebt die Verbindung mit einem Eventlogger des TwinCAT Systems auf.

Syntax

```
public virtual void Disconnect()
```

8.2.7.6 GetEventClassName

Liefert den Klassennamen zu einem Ereignis.

Syntax

```
public virtual string GetEventClassName(System.Guid EventClass, int nLangId)
```

Parameter

Name	Typ	Beschreibung
langId	int	LangID für die Sprache
EventClass	System.Guid	GUID der Ereignisklasse

Rückgabewert

Name	Typ	Beschreibung
GetEventClassName	string	

8.2.7.7 GetLoggedEvents

Fragt die im Cache befindlichen Events ab.

Syntax

```
public virtual TcEventLoggerAdsProxyLib.TcLoggedEventCollection GetLoggedEvents(uint nMaxEntiries)
```

Parameter

Name	Typ	Beschreibung
nMaxEntiries	uint	Anzahl der zu liefernden Ereignisse.

Rückgabewert

Name	Typ	Beschreibung
GetLoggedEvents	TcEventLoggerAdsProxyLib.TcLoggedEventCollection	

8.2.7.8 ActiveAlarms

Liefert eine Collection mit allen aktuell aktiven Alarmen zurück.

Syntax

```
public virtual TcEventLoggerAdsProxyLib.TcAlarmCollection ActiveAlarms
```

8.2.7.9 IsConnected

Stellt die Verbindung dar, die mittels Connect aufgebaut wurde. Sollte regelmäßig überprüft werden.

Syntax

```
public virtual bool IsConnected
```

8.2.7.10 ITcEventLogger2_SendTcMessage

Diese Methode sendet eine Nachricht.

Syntax

```
public virtual void SendTcMessage(System.Guid EventClass, uint EventId,
TcEventLoggerAdsProxyLib.SeverityLevelEnum severity, string JsonAttribute,
TcEventLoggerAdsProxyLib.TcSourceInfo pSourceInfo, TcEventLoggerAdsProxyLib.TcArguments pArguments)
```

Parameter

Name	Typ	Beschreibung
EventClass	System.Guid	GUID der Ereignisklassen
EventId	uint	ID des Ereignisses
Severity	TcEventLoggerAdsProxyLib.SeverityLevelEnum [▶ 199]	Severity des Ereignisses
JsonAttribute	string	JSON-Attribut
pSourceInfo	TcEventLoggerAdsProxyLib.TcSourceInfo	Referenz auf die Quellen-Informationen
pArguments	TcArguments	Referenz auf die Argumente.

8.2.8 ITcLoggedEvent4

Diese Schnittstelle beschreibt ein gespeichertes Ereignis. Dieses Ereignis kann entweder eine Message oder ein Alarm sein.

Syntax

```
public interface ITcLoggedEvent4
```

Methoden

Name	Modifizierer	Rückgabetyp	Beschreibung
GetArgumentData [▶ 190]	public virtual	void	Liefert die Daten der Argumente.
GetArgumentInfo [▶ 190]	public virtual	void	Liefert die (Typ-)Infos zu den Argumenten.
GetCauseRemedy [▶ 190]	public virtual	TcCauseRemedyCollection	Liefert die Cause/Remedy Informationen.
GetDetails [▶ 191]	public virtual	TcDetailCollection	Liefert die Details.
GetEventClassName [▶ 191]	public virtual	string	Liefert den Ereignisklassen-Namen.
GetText [▶ 191]	public virtual	string	Liefert den Ereignis-Text inkl. der Argumente.
IsSourceInfoTypeSupported [▶ 191]	public virtual	bool	Abfragemöglichkeit für die Typen der Quellen-Information.

Eigenschaften

Name	Modifizierer	Typ	Zugriff	Beschreibung
EventClass [▶ 192]	public virtual	System.Guid	get	Liefert die Ereignisklassen-GUID.
EventId [▶ 192]	public virtual	uint	get	Liefert die Ereignis-ID.
EventType [▶ 192]	public virtual	EventTypeEnum	get	Liefert den Typ des Ereignisses [▶ 198].
FileTimeCleared [▶ 192]	public virtual	long	get	Nur Alarm: Zeitstempel, wann der Alarm in den Zustand Cleared überführt wurde.
FileTimeConfirmed [▶ 192]	public virtual	long	get	Nur Alarm: Zeitstempel, wann der Alarm bestätigt wurde.
FileTimeRaised [▶ 192]	public virtual	long	get	Zeitstempel, wann das Ereignis gesendet wurde/der Alarm in den Zustand Raised überführt wurde.
JsonAttribute [▶ 192]	public virtual	string	get	Das JSON-Attribut.
SeverityLevel [▶ 193]	public virtual	SeverityLevelEnum	get	Das Severity-Level [▶ 199].
SourceGuid [▶ 193]	public virtual	System.Guid	get	Liefert die GUID der Quelle.
SourceId [▶ 193]	public virtual	uint	get	Liefert die ID der Quelle.
SourceName [▶ 193]	public virtual	string	get	Liefert den Namen der Quelle.
TimeCleared [▶ 193]	public virtual	System.DateTime	get	Nur Alarm: Zeitstempel, wann der Alarm in den Zustand Cleared überführt wurde.
TimeConfirmed [▶ 193]	public virtual	System.DateTime	get	Nur Alarm: Zeitstempel, wann der Alarm bestätigt wurde.
TimeRaised [▶ 193]	public virtual	System.DateTime	get	Zeitstempel, wann das Ereignis gesendet wurde/der Alarm in den Zustand Raised überführt wurde.
WithConfirmation [▶ 193]	public virtual	bool	get	Nur Alarm: Wird eine Bestätigung erfordert.

8.2.8.1 GetArgumentData

Nur zur internen Realisierung

Diese Methode ist nur für die interne Realisierung und nicht für die Verwendung durch Applikationen vorgesehen. Zur programmatischen Nutzung können die Argumente als JsonAttribute zusätzlich angehangen werden.

Diese Methode liefert die Daten der Argumente.

Syntax

```
public virtual void GetArgumentData(ref System.IntPtr ppArgData, uint size)
```

Parameter

Name	Typ	Beschreibung
ppArgData	ref System.IntPtr	Referenz auf die Daten.
size	Uint	Größe

8.2.8.2 GetArgumentInfo

Nur zur internen Realisierung

Diese Methode ist nur für die interne Realisierung und nicht für die Verwendung durch Applikationen vorgesehen. Zur programmatischen Nutzung können die Argumente als JsonAttribute zusätzlich angehangen werden.

Diese Methode liefert die (Typ-)Infos zu den Argumenten.

Syntax

```
public virtual void GetArgumentInfo(ref TcEventArgumentsInfo pArgInfo)
```

Parameter

Name	Typ	Beschreibung
pArgInfo	ref TcEventArgumentsInfo	Referenz auf die bereitgestellten Informationen.

8.2.8.3 GetCauseRemedy

Diese Methode liefert die Ursachen-/Hilfe-Informationen, falls diese definiert wurden.

Syntax

```
public virtual TcCauseRemedyCollection GetCauseRemedy(int langId)
```

Parameter

Name	Typ	Beschreibung
langId	int	LangID der Sprache.

Rückgabewert

Name	Typ	Beschreibung
GetCauseRemedy	TcCauseRemedyCollection	Collection der Cause/Remedy Informationen.

8.2.8.4 GetDetails

Diese Methode liefert die Details.

Syntax

```
public virtual TcDetailCollection GetDetails(int langId)
```

Parameter

Name	Typ	Beschreibung
langId	int	LangID der Sprache

Rückgabewert

Name	Typ	Beschreibung
GetDetails	TcDetailCollection	Collection der Details

8.2.8.5 GetEventClassName

Diese Methode liefert den Ereignisklassen-Namen.

Syntax

```
public virtual string GetEventClassName(int langId)
```

Parameter

Name	Typ	Beschreibung
langId	int	LangID der Sprache

Rückgabewert

Name	Typ	Beschreibung
GetEventClassName	string	Namen der Ereignisklasse

8.2.8.6 GetText

Diese Methode liefert den Ereignis-Text inkl. der Argumente.

Syntax

```
public virtual string GetText(int langId)
```

Parameter

Name	Typ	Beschreibung
langId	int	LangID der Sprache

Rückgabewert

Name	Typ	Beschreibung
GetText	string	Ereignis-Text

8.2.8.7 IsSourceInfoTypeSupported

Mit dieser Methode kann geprüft werden, ob der Quelleninformationstyp definiert wurde.

Syntax

```
public virtual bool IsSourceInfoTypeSupported(TcSourceInfoTypeEnum infoType)
```

Parameter

Name	Typ	Beschreibung
infoType	TcSourceInfoTypeEnum [▶ 200]	Anfrage Typ der Quelle

Rückgabewert

Name	Typ	Beschreibung
IsSourceInfoTypeSupported	bool	Information, ob unterstützt.

8.2.8.8 EventClass

Diese Eigenschaft liefert die Ereignisklassen-GUID.

Syntax

```
public virtual System.Guid EventClass
```

8.2.8.9 EventId

Diese Eigenschaft liefert die Ereignis-ID.

Syntax

```
public virtual uint EventId
```

8.2.8.10 EventType

Diese Eigenschaft liefert den Typ des Ereignisses.

Syntax

```
public virtual EventTypeEnum EventType
```

8.2.8.11 FileTimeCleared

Diese Eigenschaft liefert den Zeitstempel, wann der Alarm in den Zustand Cleared überführt wurde.

Syntax

```
public virtual long FileTimeCleared
```

8.2.8.12 FileTimeConfirmed

Diese Eigenschaft liefert den Zeitstempel, wann der Alarm in den Zustand Confirmed überführt wurde.

Syntax

```
public virtual long FileTimeConfirmed
```

8.2.8.13 FileTimeRaised

Diese Eigenschaft liefert den Zeitstempel, wann der Alarm in den Zustand Raised überführt wurde.

Syntax

```
public virtual long FileTimeRaised
```

8.2.8.14 JsonAttribute

Diese Eigenschaft liefert das JSON-Attribut.

Syntax

```
public virtual string JsonAttribute
```

8.2.8.15 SeverityLevel

Diese Eigenschaft liefert das Severity-Level.

Syntax

```
public virtual TcEventLoggerAdsProxyLib.SeverityLevelEnum SeverityLevel
```

8.2.8.16 SourceGuid

Diese Eigenschaft liefert die GUID der Quelle.

Syntax

```
public virtual System.Guid SourceGuid
```

8.2.8.17 SourceId

Diese Eigenschaft liefert die ID der Quelle.

Syntax

```
public virtual uint SourceId
```

8.2.8.18 SourceName

Diese Eigenschaft liefert den Namen der Quelle.

Syntax

```
public virtual string SourceName
```

8.2.8.19 TimeCleared

Diese Eigenschaft liefert den Zeitstempel, wann der Alarm in den Zustand Gecleared überführt wurde.

Syntax

```
public virtual System.DateTime TimeCleared
```

8.2.8.20 TimeConfirmed

Diese Eigenschaft liefert den Zeitstempel, wann der Alarm in den Zustand Confirmed überführt wurde.

Syntax

```
public virtual System.DateTime TimeConfirmed
```

8.2.8.21 TimeRaised

Diese Eigenschaft liefert den Zeitstempel, wann der Alarm in den Zustand Raised überführt wurde.

Syntax

```
public virtual System.DateTime TimeRaised
```

8.2.8.22 WithConfirmation

Diese Eigenschaft beschreibt, ob eine Bestätigung erfordert wird.

Syntax

```
public virtual bool WithConfirmation
```

8.2.9 ITcMessage3

Diese Schnittstelle repräsentiert eine Nachricht.

Syntax

```
public interface ITcMessage
```

Methoden

Name	Modifizierer	Rückgabetyp	Beschreibung
GetArgumentData [▶ 195]	public virtual	void	Liefert die Daten der Argumente.
GetArgumentInfo [▶ 195]	public virtual	void	Liefert die (Typ-)Infos zu den Argumenten.
GetCauseRemedy [▶ 195]	public virtual	TcCauseRemedyCollection	Liefert die Cause/Remedy Informationen.
GetDetails [▶ 196]	public virtual	TcDetailCollection	Liefert die Details.
GetEventClassName [▶ 196]	public virtual	string	Liefert den Ereignisklassen-Namen
GetText [▶ 196]	public virtual	string	Liefert den Ereignis-Text inkl. der Argumente.
IsSourceInfoTypeSupported [▶ 196]	public virtual	bool	Abfragemöglichkeit für die Typen der Quellen-Information.

Eigenschaften

Name	Modifizierer	Typ	Zugriff	Beschreibung
EventClass [▶ 197]	public virtual	System.Guid	get	Liefert die Ereignisklassen-GUID.
EventId [▶ 197]	public virtual	uint	get	Liefert die Ereignis-ID.
EventType [▶ 197]	public virtual	EventTypeEnum	get	Liefert den Typ des Ereignisses [▶ 198].
FileTimeRaised [▶ 197]	public virtual	long	get	Zeitstempel, wann die Nachricht gesendet wurde.
JsonAttribute [▶ 197]	public virtual	string	get	Das JSON-Attribut.
SeverityLevel [▶ 197]	public virtual	SeverityLevelEnum	get	Das Severity-Level [▶ 199].
SourceGuid [▶ 197]	public virtual	System.Guid	get	Liefert die GUID der Quelle.
SourceId [▶ 198]	public virtual	uint	get	Liefert die ID der Quelle.
SourceName [▶ 198]	public virtual	string	get	Liefert den Namen der Quelle.
TimeRaised [▶ 198]	public virtual	System.DateTime	get	Zeitstempel, wann die Nachricht gesendet wurde.

8.2.9.1 GetArgumentData

Nur zur internen Realisierung

Diese Methode ist nur für die interne Realisierung und nicht für die Verwendung durch Applikationen vorgesehen. Zur programmatischen Nutzung können die Argumente als JsonAttribute zusätzlich angehangen werden.

Diese Methode liefert die Daten der Argumente.

Syntax

```
public virtual void GetArgumentData(ref System.IntPtr ppArgData, uint size)
```

Parameter

Name	Typ	Beschreibung
ppArgData	ref System.IntPtr	Referenz auf die Daten.
size	Uint	Größe

8.2.9.2 GetArgumentInfo

Nur zur internen Realisierung

Diese Methode ist nur für die interne Realisierung und nicht für die Verwendung durch Applikationen vorgesehen. Zur programmatischen Nutzung können die Argumente als JsonAttribute zusätzlich angehangen werden.

Diese Methode liefert die (Typ-)Infos zu den Argumenten.

Syntax

```
public virtual void GetArgumentInfo(ref TcEventArgumentsInfo pArgInfo)
```

Parameter

Name	Typ	Beschreibung
pArgInfo	ref TcEventArgumentsInfo	Referenz auf die bereitgestellten Informationen.

8.2.9.3 GetCauseRemedy

Diese Methode liefert die Ursachen-/Hilfe-Informationen, falls diese definiert wurden.

Syntax

```
public virtual TcCauseRemedyCollection GetCauseRemedy(int langId)
```

Parameter

Name	Typ	Beschreibung
langId	int	LangID der Sprache.

Rückgabewert

Name	Typ	Beschreibung
GetCauseRemedy	TcCauseRemedyCollection	Collection der Cause/Remedy Informationen.

8.2.9.4 GetDetails

Diese Methode liefert die Details.

Syntax

```
public virtual TcDetailCollection GetDetails(int langId)
```

Parameter

Name	Typ	Beschreibung
langId	int	LangID der Sprache

Rückgabewert

Name	Typ	Beschreibung
GetDetails	TcDetailCollection	Collection der Details

8.2.9.5 GetEventClassName

Diese Methode liefert den Ereignisklassen-Namen.

Syntax

```
public virtual string GetEventClassName(int langId)
```

Parameter

Name	Typ	Beschreibung
langId	int	LangID der Sprache

Rückgabewert

Name	Typ	Beschreibung
GetEventClassName	string	Namen der Ereignisklasse

8.2.9.6 GetText

Diese Methode liefert den Ereignis-Text inkl. der Argumente.

Syntax

```
public virtual string GetText(int langId)
```

Parameter

Name	Typ	Beschreibung
langId	int	LangID der Sprache

Rückgabewert

Name	Typ	Beschreibung
GetText	string	Ereignis-Text

8.2.9.7 IsSourceInfoTypeSupported

Mit dieser Methode kann geprüft werden, ob der Quelleninformationstyp definiert wurde.

Syntax

```
public virtual bool IsSourceInfoTypeSupported(TcSourceInfoTypeEnum infoType)
```

Parameter

Name	Typ	Beschreibung
infoType	TcSourceInfoTypeEnum [▶ 200]	Anfrage Typ der Quelle

Rückgabewert

Name	Typ	Beschreibung
IsSourceInfoTypeSupported	bool	Information, ob unterstützt.

8.2.9.8 EventClass

Diese Eigenschaft liefert die Ereignisklassen-GUID.

Syntax

```
public virtual System.Guid EventClass
```

8.2.9.9 EventId

Diese Eigenschaft liefert die Ereignis-ID.

Syntax

```
public virtual uint EventId
```

8.2.9.10 EventType

Diese Eigenschaft liefert den Typ des Ereignisses.

Syntax

```
public virtual EventTypeEnum EventType
```

8.2.9.11 FileTimeRaised

Diese Eigenschaft liefert den Zeitstempel, wann der Alarm in den Zustand Raised überführt wurde.

Syntax

```
public virtual long FileTimeRaised
```

8.2.9.12 JsonAttribute

Diese Eigenschaft liefert das JSON-Attribut.

Syntax

```
public virtual string JsonAttribute
```

8.2.9.13 SeverityLevel

Diese Eigenschaft liefert das Severity-Level.

Syntax

```
public virtual TcEventLoggerAdsProxyLib.SeverityLevelEnum SeverityLevel
```

8.2.9.14 SourceGuid

Diese Eigenschaft liefert die GUID der Quelle.

Syntax

```
public virtual System.Guid SourceGuid
```

8.2.9.15 SourceId

Diese Eigenschaft liefert die ID der Quelle.

Syntax

```
public virtual uint SourceId
```

8.2.9.16 SourceName

Diese Eigenschaft liefert den Namen der Quelle.

Syntax

```
public virtual string SourceName
```

8.2.9.17 TimeRaised

Diese Eigenschaft liefert den Zeitstempel, wann der Alarm in den Zustand Raised überführt wurde.

Syntax

```
public virtual System.DateTime TimeRaised
```

8.3 Datentypen

8.3.1 ConfirmationStateEnum

Definiert den Bestätigungszustand eines Alarms.

Syntax

```
public enum ConfirmationStateEnum
{
    Confirmed,
    NotRequired,
    NotSupported,
    Reset,
    WaitForConfirmation
}
```

Parameter

Name	Beschreibung
Confirmed	Bestätigt
NotRequired	Bestätigung im aktuellen Zustand nicht nötig. (Alarm aktuell nicht im Zustand Raised.)
NotSupported	Wurde ohne Bestätigung initialisiert.
Reset	Initialzustand
WaitForConfirmation	Wartet auf Bestätigung.

8.3.2 EventTypeEnum

Type Definition, ob ein TcEvent vom Typ Alarm oder Message ist.

Syntax

```
public enum EventTypeEnum
{
    Alarm,
    Message
}
```

Parameter

Name	Beschreibung
Alarm	Der TcEvent ist vom Typ TcAlarm.
Message	Der TcEvent ist vom Typ TcMessage.

8.3.3 SeverityLevelEnum

Diese Enumeration definiert die „Severity“ des Ereignisses. Es ist eine geordnete Liste.

Syntax

```
public enum SeverityLevelEnum
{
    Critical,
    Error,
    Warning,
    Info,
    Verbose
}
```

Parameter

	Name	Beschreibung
4	Critical	Kritisch
3	Error	Fehler
2	Warning	Warnung
1	Info	Information
0	Verbose	Erweiterte Ausgabe

8.3.4 TcEventArgumentTypeEnum

Type-Definition, von welchem Typ ein TcArgument ist.

Syntax

```
public enum TcEventArgumentTypeEnum
{
    Blob,
    Boolean,
    Char,
    Double,
    E_AdsnotificationStream,
    EventReference,
    ExternalTimeStamp,
    Float,
    FormatString,
    Int16,
    Int32,
    Int64,
    Int8,
    StringType,
    UInt16,
    UInt32,
    UInt64,
    UInt8,
    Undefinied,
    UTF8EncodedString,
```

```
    WChar,
    WStringType
}
```

8.3.5 TcSourceInfoTypeEnum

Definition, welcher Eintrag in einem TcSourceInfo identifiziert wird.

Syntax

```
public enum TcSourceInfoTypeEnum
{
    SourceGuid,
    SourceId,
    SourceName
}
```

Parameter

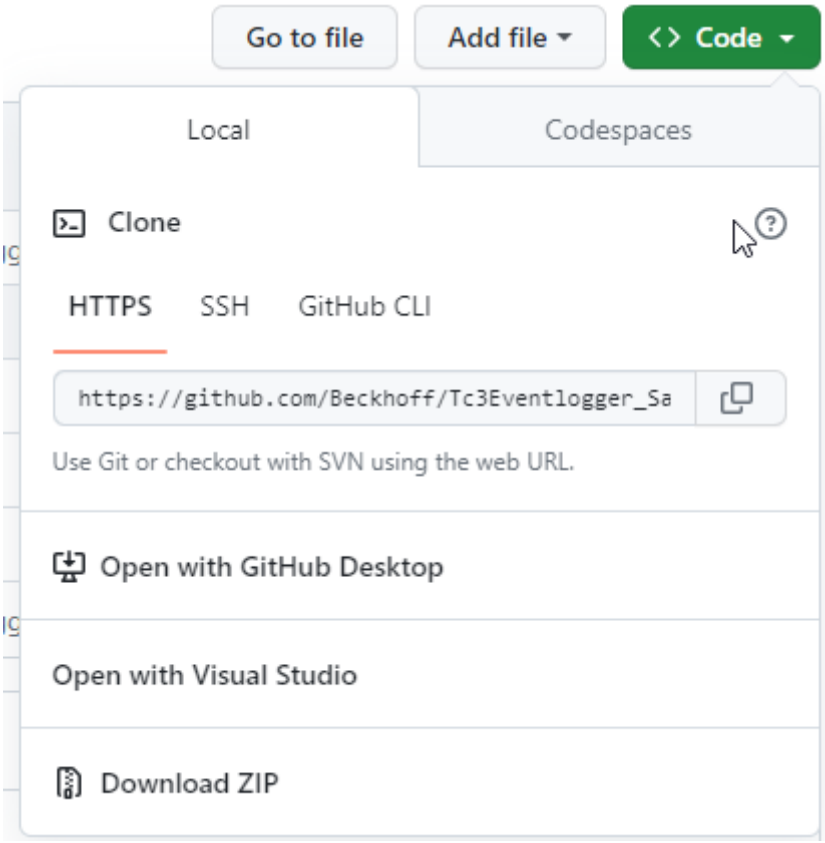
Name	Beschreibung
SourceGuid	Die Source GUID des TcSourceInfo.
SourceId	Die Source ID des TcSourceInfo. Beispielsweise die TcCOM Objekt ID.
SourceName	Der SourceName des TcSourceInfo. Beispielsweise der InstanzPfad innerhalb einer PLC.

9 Beispiele

An dieser Stelle werden Beispiele für die Nutzung des Eventloggers bereitgestellt.

Diese Beispiele stellen einfach nachzuvollziehende Demonstrationen für den Umgang mit dem TwinCAT 3 Eventlogger dar.

Beispielcode und -konfigurationen für dieses Produkt können über das entsprechende Repository auf GitHub bezogen werden: https://github.com/Beckhoff/Tc3Eventlogger_Samples. Sie haben dort die Möglichkeit das Repository zu clonen oder ein ZIP File mit dem Sample herunterzuladen.



Die Beispiele SPS [▶ 201] sowie C++ [▶ 206] beziehen sich auf die Echtzeitprogrammier-Schnittstellen von TwinCAT.

Für Usermode-Programme steht eine .NET Schnittstelle [▶ 212] als bereit.

9.1 SPS

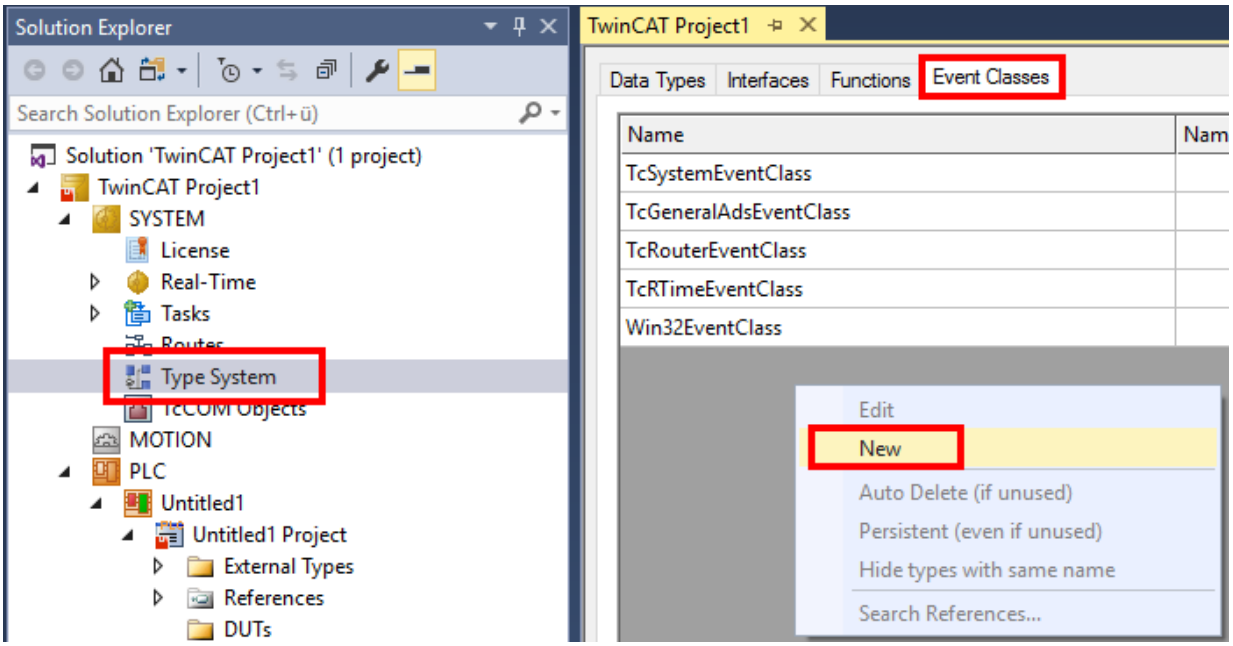
9.1.1 Tutorial

Dieses Tutorial verdeutlicht die Arbeitsschritte von einem leeren TwinCAT-Projekt bis hin zu einer abgesendeten Meldung. Es zeigt die im Abschnitt Technische Einführung [▶ 13] beschriebenen Eigenschaften des TwinCAT 3 EventLoggers anschaulich im Arbeitsablauf.

Ereignisklasse im Typsystem von TwinCAT anlegen

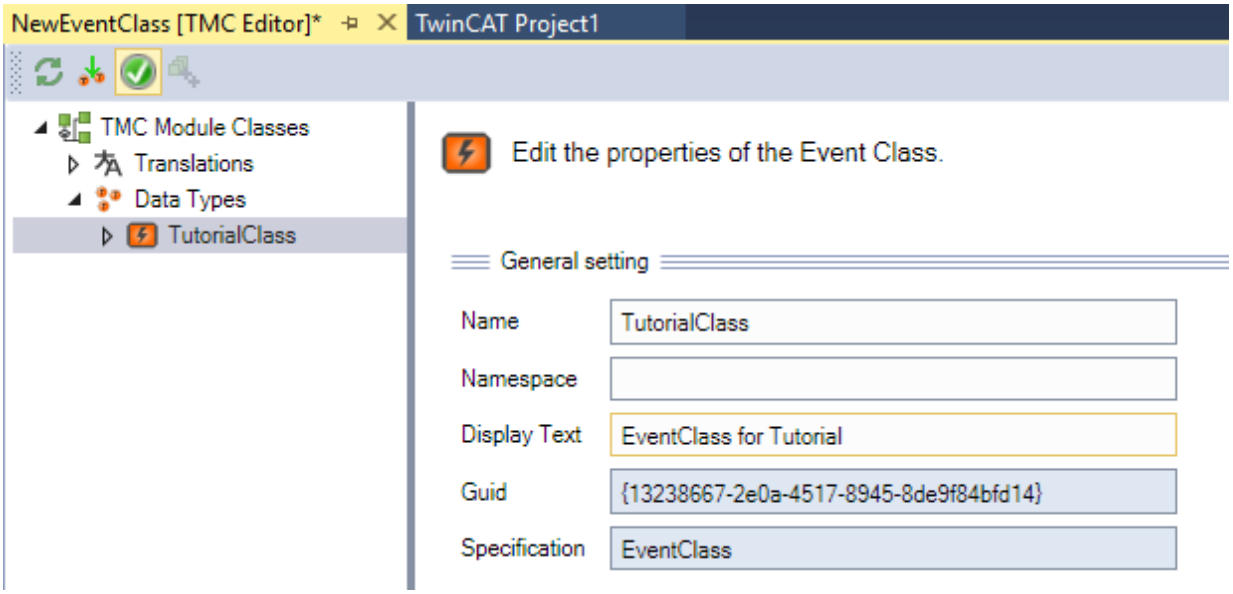
✓ Ein Standard-TwinCAT-SPS-Projekt existiert.

1. Klicken Sie im SYSTEM-Teilbaum doppelt auf **Type System** und wählen Sie in dem sich öffnenden Editor die Registerkarte **Event Classes**. Öffnen Sie das Kontextmenü und wählen Sie den Befehl **New**.

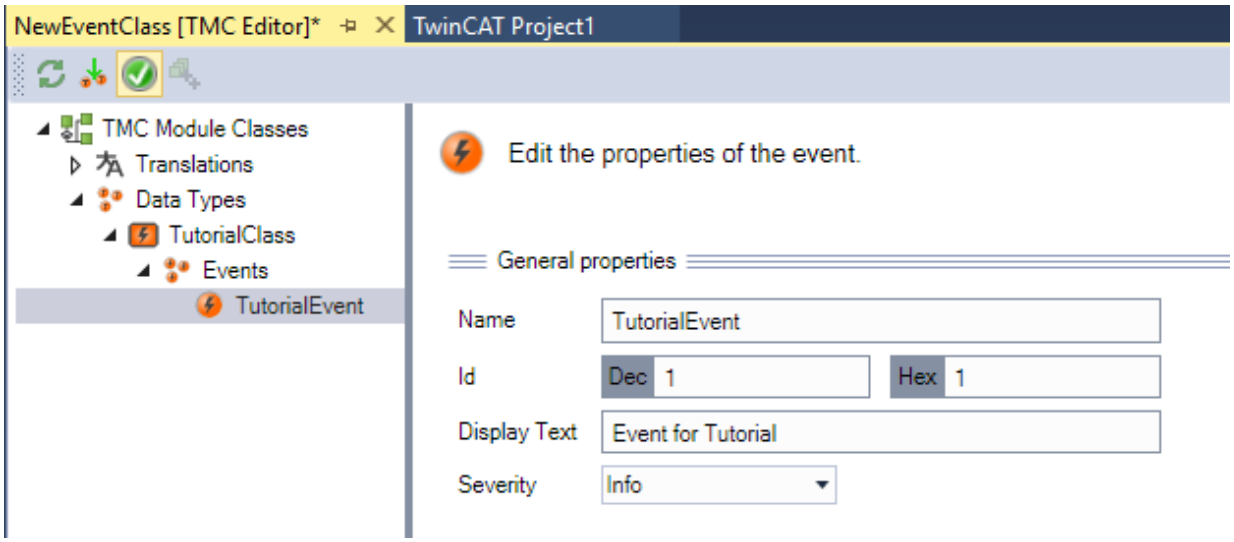


⇨ Der TMC Editor öffnet sich.

2. Geben Sie der Ereignisklasse einen Namen und geben Sie einen Display-Text an.



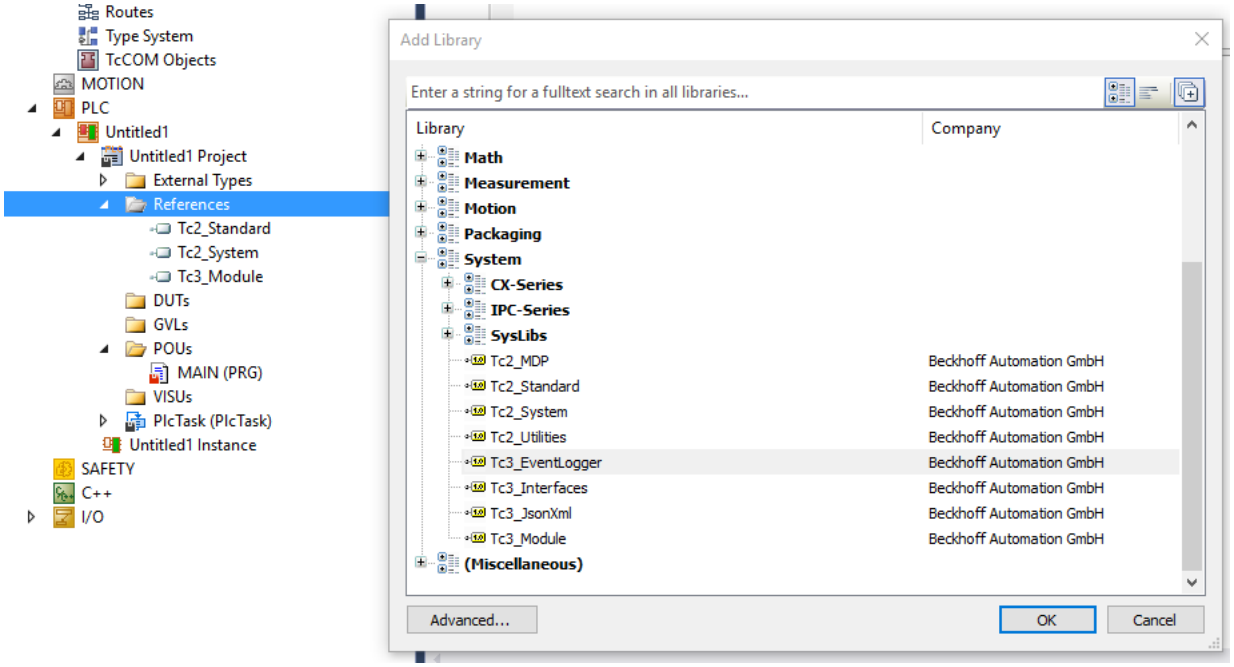
3. Unterhalb der der Ereignisklasse ist bereits ein Ereignis angelegt. Geben Sie dem Ereignis einen Namen und geben Sie einen Dispaly-Text und die Severity an.



4. Speichern und ggf. schließen Sie die Ereignisklasse.
   ⇨ Der Quellcode wird in der SPS bereitgestellt und ist unter dem Symbol TC_EVENTS erreichbar.

Bibliothek TC3_EventLogger hinzufügen

5. Wählen Sie im Kontextmenü des Objekts **References** den Befehl **Add Library**.
   ⇨ Der Dialog **Add Library** öffnet sich.
6. Wählen Sie die Bibliothek aus und bestätigen Sie den Dialog.



⇨ Die Bibliothek wird dem SPS-Projekt hinzugefügt.

SPS-Programm erstellen

1. Öffnen Sie mit einem Doppelklick das MAIN-Programm des SPS-Projekts im Editor.
2. Deklarieren und initialisieren Sie die Variablen bInit und bSend und deklarieren Sie eine Instanz des Funktionsbausteins FB_TcMessage:

```
PROGRAM MAIN
VAR
    bInit : BOOL := TRUE;
    bSend : BOOL := TRUE;
```

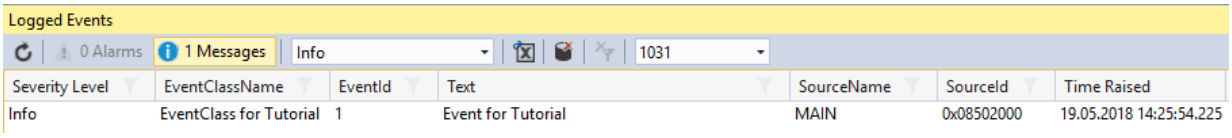
```
    fbMsg : FB_TcMessage;
END_VAR
```

3. Implementieren Sie den Sendevorgang wie im Code dargestellt. Die Nachricht wird einmalig mittels der CreateEx-Methode initialisiert. Da die Initialisierung dynamische Ressourcen benötigt, sollte sie nicht zyklisch erfolgen. Die initialisierte Nachricht wird anschließend mit der Send-Methode gesendet.

```
IF bInit THEN
    bInit := FALSE;
    fbMsg.CreateEx(TC_EVENTS.TutorialClass.TutorialEvent, 0);
END_IF

IF bSend THEN
    bSend := FALSE;
    fbMsg.Send(0);
END_IF
```

4. Erstellen Sie das SPS-Projekt und starten Sie die SPS.

⇨ Das Ergebnis wird im Fenster LoggedEvents des TwinCAT 3 Engineerings anzeigt.



9.1.2 Beispiel ResultMessage

Dieses Beispiel zeigt die Verwendung des TwinCAT 3 Eventloggers mit Funktionsbausteinen. Es demonstriert zum einen, wie ein Ausgang an einem Funktionsbaustein dazu verwendet werden kann, die Event-Informationen als erweiterte Rückgabe zu nutzen. Zum anderen demonstriert es, wie eine Parametrisierung vorgenommen werden kann, um eine Ausgabe der Meldungen über den TwinCAT 3 Eventlogger nur in bestimmten Fällen durchzuführen.

Download des Beispiels: https://github.com/Beckhoff/Tc3Eventlogger_Samples/tree/main/PLC/Tc3EventLogger_ResultMessageSample

Das Bespiel besteht aus zwei Funktionsbausteinen:

- FB_MathCalculation: Dieser Funktionsbaustein bietet zwei Methoden und zwei Properties an, die Meldungen immer am Ausgang ipResultMessage ausgeben und zusätzlich über den EventLogger absenden, wenn ein Tracelevel überschritten ist.
  - Methode Addition(): Addiert zwei Zahlen und sendet bei einem Überlauf eine Nachricht
  - Methode Divison(): Dividiert zwei Zahlen nach Prüfung. Sendet eine Nachricht, wenn eine Division durch 0 erfolgt.
  - Property bTraceLevelDefault: Gibt an, ob das Tracelevel lokal am Funktionsbaustein beachtet werden soll, oder ob ein Library Tracelevel verwendet werden soll, welcher im Beispiel in der GVL vorhanden ist.
  - Property eTraceLevel: Die Methoden senden die Nachricht nur über den EventLogger ab, wenn die Severity größer oder gleich diesem Property ist.
- FB_Control: Dieser Funktionsbaustein zeigt die Verwendung des FB_MathCalculation-Bausteins innerhalb eines anderen Bausteins. Dabei nutzt die Execute-Methode des FB_Control die FB_MathCalculation.Divison() und behandelt die Nachricht als Fehlercode selbst weiter.

9.1.3 Beispiel Listener

Dieses Beispiel zeigt die Verwendung des TwinCAT 3 EventLoggers in Bezug auf Nachrichten und Alarme. Gleichzeitig wird das Empfangen von Nachrichten in einem zweiten Projekt gezeigt.

Download des Beispiels: https://github.com/Beckhoff/Tc3Eventlogger_Samples/tree/main/PLC/Tc3EventLogger_ListenerSample

Publisher-Projekt

Im Publisher-Projekt werden einfache BOOL-Variablen als Trigger verwendet:

- bSendMessage, um eine Nachricht abzusetzen.
- bRaiseAlarm, um einen Alarm zu setzen.
- bClearAlarm, um einen Alarm zurückzunehmen.
- bConfirmAlarm, um einen Alarm zu quittieren.

Zusätzlich gibt es die Möglichkeit, das JSON-Attribut zu setzen, um dieses bei beiden Nachrichten mitzusenden.

Listener-Projekt

Im Listener-Projekt ist ein Funktionsbaustein FB_Listener enthalten, der den in der Tc3_EventLogger enthaltenen Baustein FB_ListenerBase erweitert. Der Baustein implementiert hierbei die Funktionen zum Empfang der Nachrichten:

- OnMessageSent: Wenn eine Nachricht versendet wurde, wird der EventLogger diese Methode als Callback aufrufen. Die Methode zählt die Anzahl der Nachrichten mit.
- OnAlarmRaised/OnAlarmCleared/OnAlarmConfirmed: Wenn der Alarm den Zustand ändert, wird der EventLogger diese Methode als Callback aufrufen. Die Methoden zählen jeweils die Anzahl der Zustandsänderungen mit.
- Um den Empfang der Nachrichten zu initiieren, ist eine Execute-Methode an dem Baustein implementiert.
- Der Text der letzten empfangenen Nachricht kann abgeholt werden.
- Der Funktionsbaustein FB_ListenerTest nutzt den FB_Listener. Hierbei registriert er einmalig die zu empfangende Ereignisklasse. Eine weitere existierende Ereignisklasse wird nicht empfangen, wodurch die Filterfunktionalität demonstriert wird.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.1.4024.17	PC oder CX (x64, x86, Arm®)	Tc3_EventLogger (>= v3.1.27.0)

9.1.4 Beispiel Filter

Dieses Beispiel zeigt die Verwendung des TwinCAT 3 EventLoggers in Bezug auf das Empfangen von Nachrichten. Hierbei wird ein Fokus auf die Filterfunktionen gesetzt, um zielgerichtet die richtigen Nachrichten zu verarbeiten.

Download des Beispiels: https://github.com/Beckhoff/Tc3Eventlogger_Samples/tree/main/PLC/Tc3EventLogger_FilterSample

Das Beispiel besteht aus vier Komponenten:

- Es werden eine Reihe von unterschiedlichen Nachrichten abgesendet, wodurch die Selektion der Nachrichten in unterschiedlichen Filtern demonstriert wird.
- Eine Komponente zeigt, wie aus dem Cache Nachrichten verworfen werden können, welche über einen Filter spezifiziert werden.
- Eine andere Komponente zeigt den Export von im Cache hinterlegten Nachrichten in eine CSV-Datei. Auch hierbei wird über die Filter programmiert, welche Nachrichten ausgewählt werden sollen.
- Eine weitere Komponente zeigt das allgemeine Empfangen von in der Echtzeit gesendeten Nachrichten sowie das Empfangen von EtherCAT Emergency Nachrichten, welche empfangen werden sollen.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.1.4024.17	PC oder CX (x64, x86, Arm®)	Tc3_EventLogger (>= v3.1.27.0)

9.1.5 Beispiel Resource Import

Dieses Beispiel zeigt, wie eine Resource Datei im ECPX Format mit dem TwinCAT 3 Eventlogger verwendet werden kann.

Download des Beispiels: https://github.com/Beckhoff/Tc3Eventlogger_Samples/tree/main/PLC/Tc3EventLogger_ResouceImportSample/TwinCAT%20ResourceImport

Dafür wird in dem Beispiel eine neue Implementierung des FB_SimpleAdsLogEvent Funktionsbausteins erstellt. Dieses gebraucht die aus Excel erzeugte Mapping Funktion F_MapSource, welche über die TcPOU Datei importiert werden kann. Die Implementierung zeigt auch das Mapping der Funktionsweise:

- **bSetEvent**: Bei steigender Flanke wird ein Ereignis als Alarm gesetzt (`raise`) und bei fallender Flanke wieder gelöscht (`cleared`)
- **bQuit**: Falls ReqMustCon gesetzt war, wird auch ein Alarm als quittier fähig angelegt. Über bQuit kann diese Quittierung vorgenommen werden.

Folgende Schritte werden vorgeschlagen

✓ Eine Resource.ecpx Datei liegt vor und entsprechende Alarme sollen über den TwinCAT 3 Eventlogger gesendet werden. Das Excel Addin ist installiert.

1. Importieren Sie die Datei in Excel [▶ 21] mittels „Import“.
   ⇨ Aus jeder „Source“ wird eine Ereignisklasse angelegt, in der sich die Events finden. Auch die Übersetzungen sind in der Translation übernommen wordn
2. Die Ereignisklassen können von nun an in Excel weiterentwickelt werden.
3. Durch „Generate TMC-File“ wird eine TMC Datei erzeugt.
4. Diese kann in einer TwinCAT Solution mittels Rechtsklick auf System->TypeSystem „Add existing item“ eingebunden werden.
5. Mittels „Create PLC mapping function” kann eine PLC-Function erzeugt werden.
6. Diese kann in einem PLC Projekt auf DUT mit einem Rechtsklick „Add“->“Existing Item“ importiert werden.
7. Das Beispiel stellt eine Beispielimplementeirung für einen Funktionsbaustein bereit, der die gleichen Ein-/Ausgänge hat, wie der FB_SimpleAdsLogEvent. Somit muss lediglich die Deklaration aber nicht das Verhalten des Ablaufes verändert werden.

⇨ Die Events werden durch den bereitgestellten Funktionsbaustein über den TwinCAT 3 Eventlogger gesendet.
Hierfür muss das Verhalten der beispielhaften Implementierung ggf angepasst werden und beispielsweise das Mapping der Severity adaptiert werden. Vgl. Importieren von Resource Dateien [▶ 37]

Das Beispiel zeigt das Ergebnis und stellt auch beispielhaft die Ausgangsdatei im ECPX Format, erzeugten TMC / Function Datei bereit.

9.2 C++

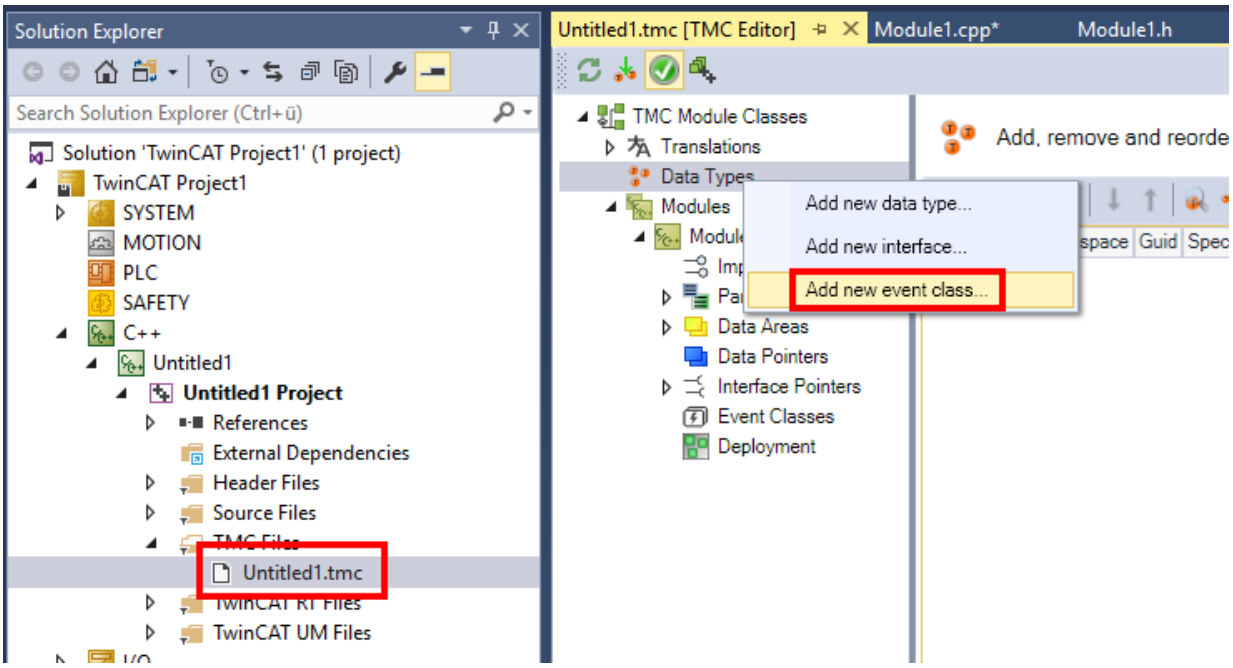
9.2.1 Tutorial

Dieses Tutorial verdeutlicht die Arbeitsschritte von einem leeren TwinCAT-Projekt bis hin zu einer abgesendeten Meldung. Es zeigt die im Abschnitt Technische Einführung [▶ 13] beschriebenen Eigenschaften des TwinCAT 3 EventLoggers anschaulich im Arbeitsablauf.

Ereignisklasse in den Datentypen der TMC-Datei des C++-Projekts anlegen

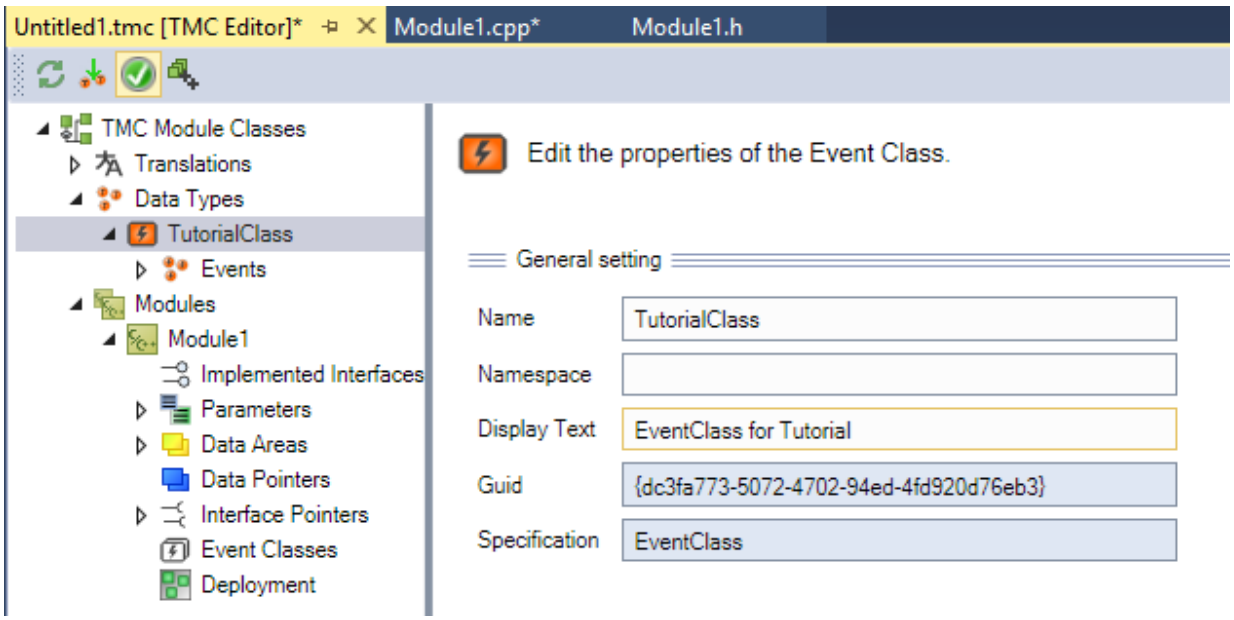
✓ Ein neues TwinCAT-C++-Projekt mit Modul aus dem Wizard „TwinCAT Module Class with Cyclic IO” exisitert.

1. Klicken Sie im C++-Projekt doppelt auf die TMC-Datei, um den TMC Editor zu öffnen. Wählen Sie im Kontextmenü von **Data Types** den Befehl **Add new event class…**

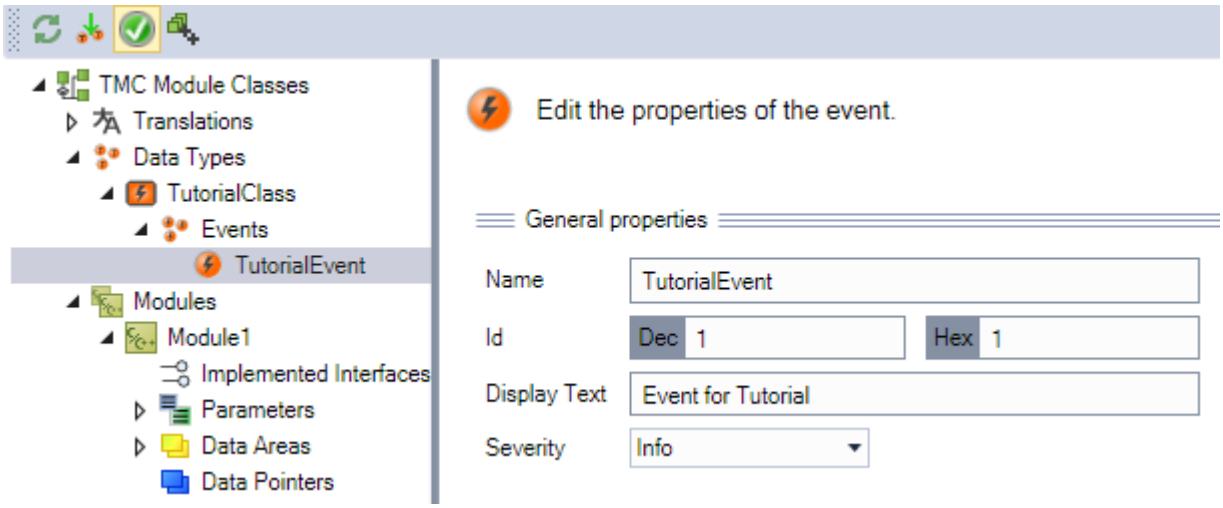


⇨ Der TMC Editor öffnet die Ereignisklasse.

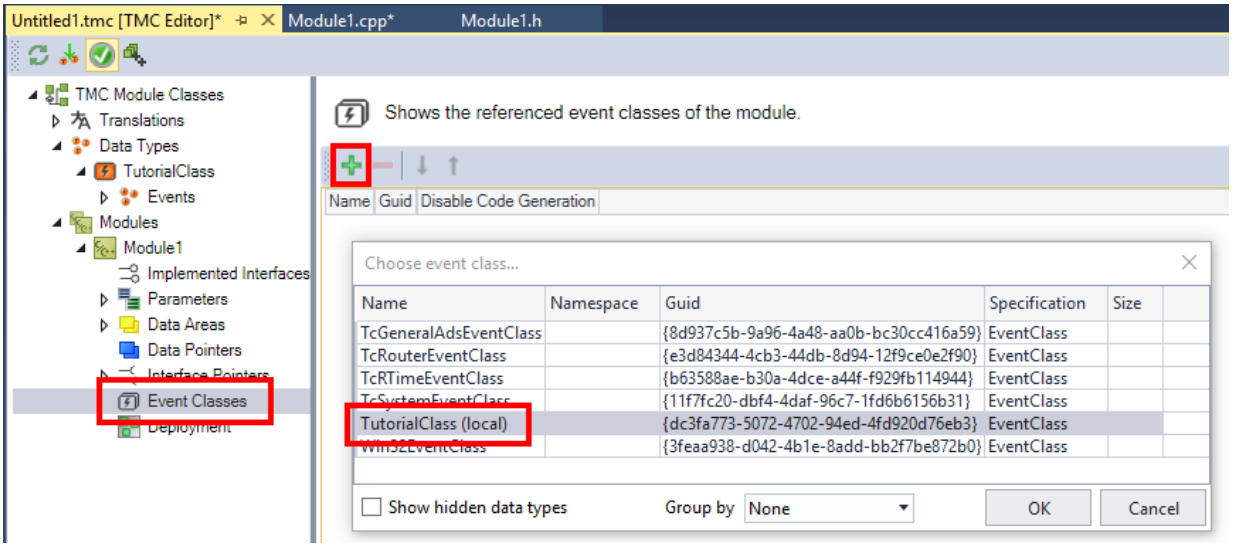
2. Geben Sie der Ereignisklasse einen Namen und geben Sie optional einen Display-Text an.



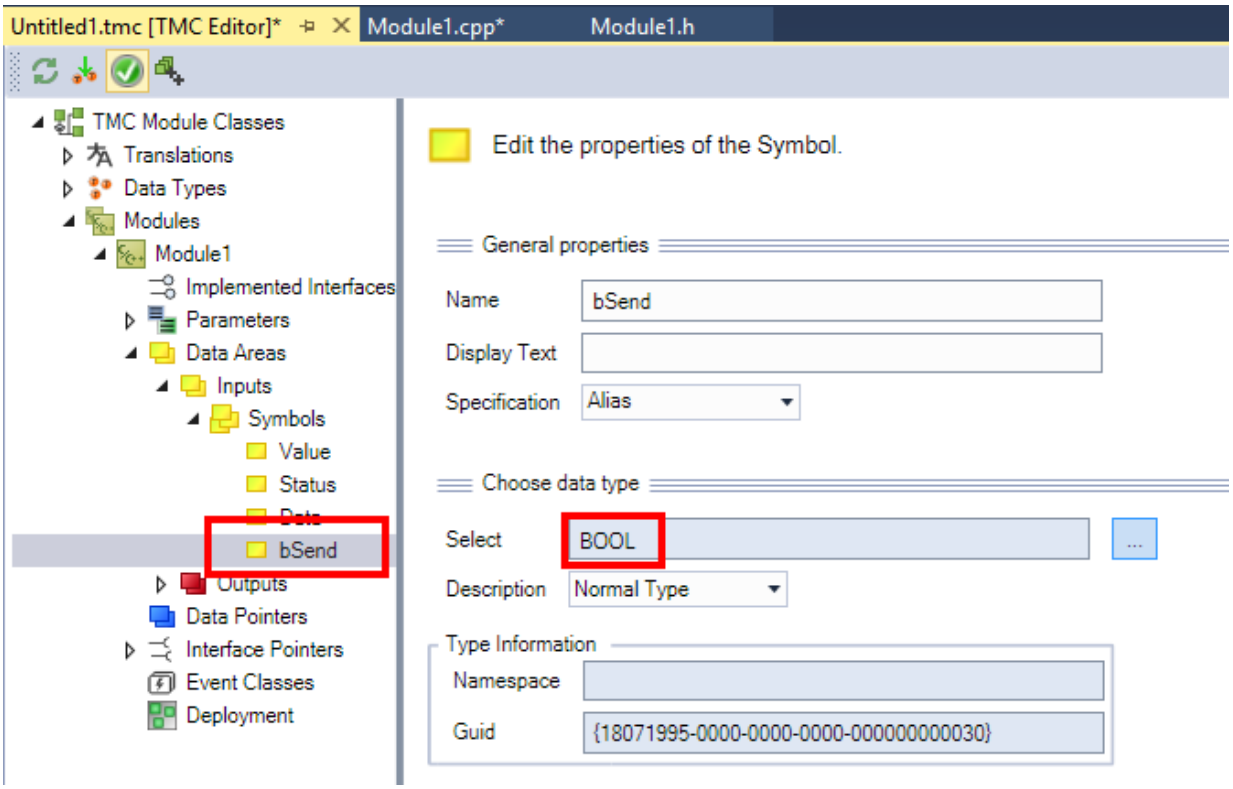
3. Unterhalb der Ereignisklasse ist bereits ein Ereignis angelegt. Geben Sie dem Ereignis einen Namen und geben Sie einen Dispaly-Text und die Severity an.



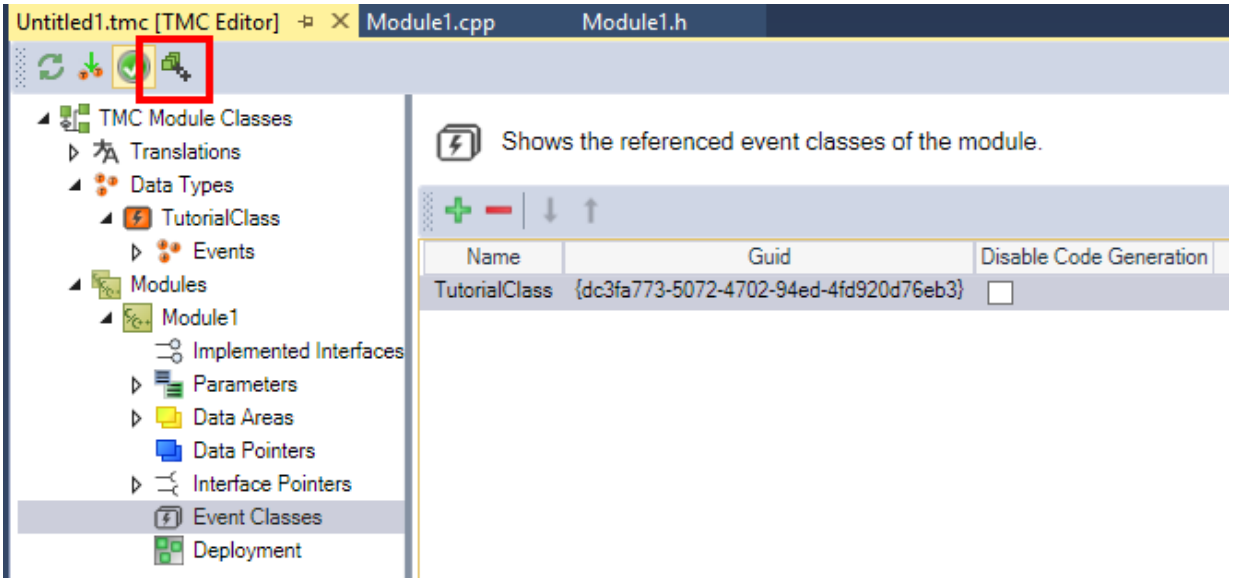
4. Nutzen Sie die Ereignisklasse in dem zuvor angelegten Modul.



5. Legen Sie zusätzlich einen Eingang bSend anlegen, um das Ereignis bei steigender Flanke zu senden.



6. Generieren Sie den Quellcode des Moduls.



C++-Programm erstellen

7. Fügen Sie im C++-Programm Header zu der Untitled1Interfaces.h hinzu:

```
#include "TcRouterInterfaces.h"
#include "TcEventLoggerInterfaces.h"
```

8. In der Module1.h benötigen Sie lokal folgende Deklarationen:

```
UINT m_counter;
ITcEventLoggerPtr m_spEventLogger;
ITcMessagePtr m_spMessage;
BOOL m_OldSend;
```

9. Initialisieren Sie im Konstruktor des Moduls in der Module1.cpp folgende Werte:

```
CModule1::CModule1()
    : m_Trace(m_TraceLevelMax, m_spSrv)
, m_counter(0)
{
```

```
///<AutoGeneratedContent id="MemberInitialization">
    m_TraceLevelMax = tlAlways;
    memset(&m_Parameter, 0, sizeof(m_Parameter));
    memset(&m_Inputs, 0, sizeof(m_Inputs));
    memset(&m_Outputs, 0, sizeof(m_Outputs));
///</AutoGeneratedContent>
    m_spEventLogger = 0;
    m_spMessage = 0;
    m_OldSend = FALSE;
    m_Inputs.bSend = TRUE;
}
```

10. Führen Sie in der Methode SetObjStateSO() zusätzlich folgende Initialisierung durch:

```
// TODO: Add any additional initialization
m_spEventLogger.SetOID(OID_TCEVENTLOGGER);
hr = FAILED(hr) ? hr : m_spSrv->TcQuerySmartObjectInterface(m_spEventLogger);
hr = FAILED(hr) ? hr : m_spEventLogger-
>CreateMessage(TcEvents::TutorialClass::EventClass, TcEvents::TutorialClass::TutorialEvent.nEven
tId, TcEvents::TutorialClass::TutorialEvent.eSeverity, 0, &m_spMessage);
```

11. Wenn der Aufruf der Methode AddModuleToCaller() fehlschlägt, führen Sie eine Deinitialisierung durch:

```
// Cleanup if transition failed at some stage
if ( FAILED(hr) )
{
    RemoveModuleFromCaller();
    m_spEventLogger = NULL;
    m_spMessage = NULL;
}
```

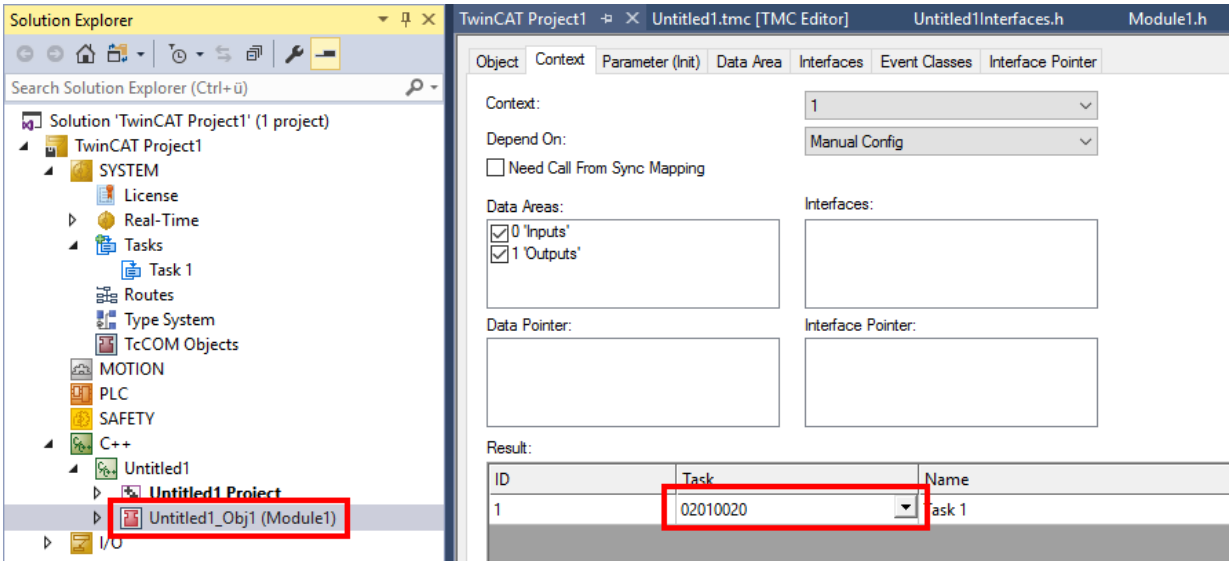
12. Führen Sie in der Methode SetObjStateOS() eine Deinitialisierung durch:

```
// TODO: Add any additional deinitialization
m_spEventLogger = NULL;
m_spMessage = NULL;
```

13. Erweitern Sie den zyklischen Code des Moduls um das Versenden der Nachricht:

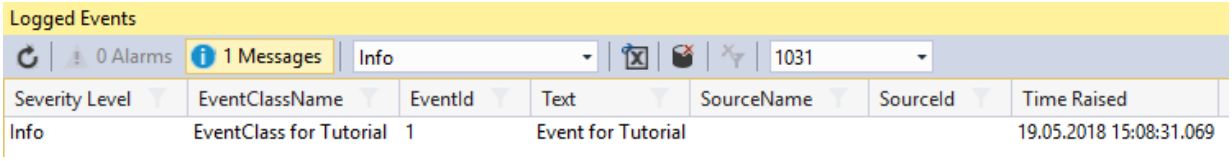
```
// TODO: Replace the sample with your cyclic code
if (m_Inputs.bSend && ! m_OldSend) // raising edge
{
   m_spMessage->Send(0);
}
m_OldSend = m_Inputs.bSend;
```

14. Legen Sie eine Modulinstanz an und verknüpfen Sie diese mit einem Task.



15. Erstellen Sie das C++-Projekt und starten Sie TwinCAT.

⇨ Das Ergebnis wird im Fenster LoggedEvents des TwinCAT 3 Engineerings angezeigt.



9.2.2 Beispiel Start-Stop

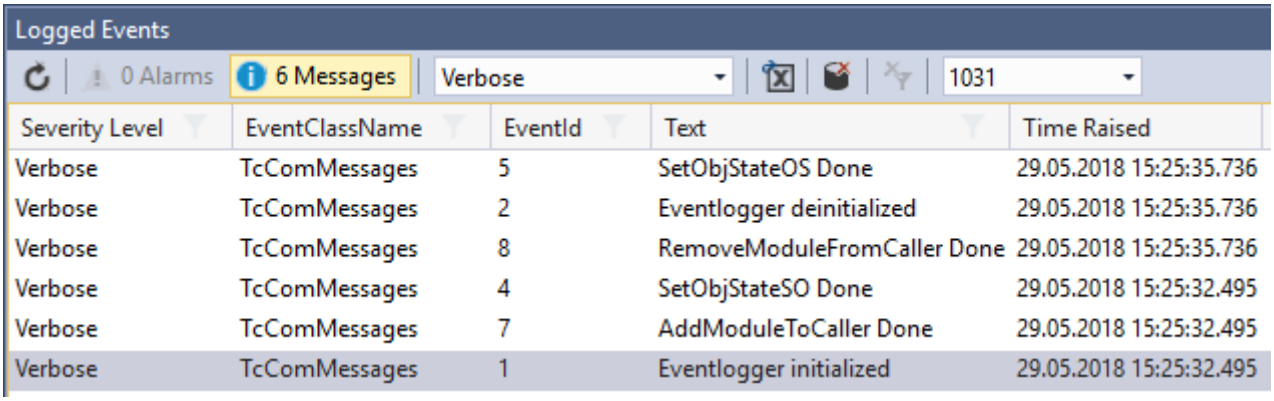
Dieses Beispiel zeigt die Verwendung des TwinCAT 3 EventLoggers beim Starten und Stoppen von TwinCAT. Es wird eine Nachricht mit Argumenten verwendet, die unterschiedliche Status des C++-Moduls abbilden.

Download des Beispiels: https://github.com/Beckhoff/Tc3Eventlogger_Samples/tree/main/C%2B%2B(RT)/Tc3EventLogger_CppStartStop_Sample

In den Methoden der Statemachines finden sich die Verwendungen der Nachricht:

- SetObjStatePS(): Der EventLogger wird von TwinCAT hochgefahren. Eine Nutzung ist in diesem State damit nicht möglich.
- SetObjStateSO(): Hier werden die Referenz auf den EventLogger durch TcQuerySmartObjectInterface bezogen und die Nachrichten durch CreateMessage() initiiert. Eine entsprechende Meldung wird abgesendet. Mit AddModuleToCaller() wird eine Nachricht abgesendet. Hier wird am Ende der Transition ebenfalls eine Nachricht abgesendet.
- CycleUpdate():In dem Beispiel werden keine Nachrichten abgeschickt. Das Verhalten des Moduls zum OP-State entspricht damit dem eines „Cyclic IO“-Moduls.
- SetObjStateOS(): Mit RemoveModuleFromCaller() wird eine Nachricht abgesendet. Hier wird am Ende der Transition ebenfalls eine Nachricht abgesendet. Danach werden die Referenzen zur Nachricht und zum EventLogger auf NULL gesetzt, was ebenfalls durch eine Nachricht mitgeteilt wird.
- SetObjStateSP(): Der EventLogger wird von TwinCAT heruntergefahren. Eine Nutzung ist in diesem State damit nicht möglich.

Es ergeben sich die folgenden Nachrichten:



9.2.3 Beispiel Listener

Dieses Beispiel zeigt die Verwendung des TwinCAT 3 EventLoggers in Bezug auf Nachrichten und Alarme.

Es besteht aus einem Modul, das eine Nachricht und einen Alarm senden kann und einem Modul, das diese Nachrichten empfängt.

Download des Beispiels: https://github.com/Beckhoff/Tc3Eventlogger_Samples/tree/main/C%2B%2B(RT)/Tc3EventLogger_CppListener_Sample

Publisher-Modul

Im Publisher-Modul werden einfache BOOL-Variablen als Trigger an der Input-Data-Area verwendet:

- bSendMessage, um eine Nachricht abzusetzen
- bRaiseAlarm, um einen Alarm zu setzen
- bClearAlarm, um einen Alarm zurückzunehmen
- bConfirmAlarm, um einen Alarm zu quittieren

Zusätzlich gibt es die Möglichkeit das JSON-Attribut zu setzen und zu entfernen, um bei beiden Nachrichten dieses mitzusenden.

Listener-Modul

Das Listener-Modul implementiert sowohl die ITCMessageListener- als auch die ITcAlarmListener-Schnittstelle Die hierdurch spezifizierten Methoden werden als Callback vom EventLogger aufgerufen.

- OnMessageSent: Wenn eine Nachricht versendet wurde, wird der EventLogger diese Methode als Callback aufrufen. Die Methode zählt die Anzahl der Nachrichten mit.
- OnAlarmRaised/OnAlarmCleared/OnAlarmConfirmed: Wenn der Alarm den Zustand ändert, wird der EventLogger diese Methode als Callback aufrufen. Die Methoden zählen jeweils die Anzahl der Zustandsänderungen mit.

Das Listener-Modul registriert sich dabei beim Starten in der Methode SetObjStateSO() beim EventLogger für die entsprechenden Ereignisklassen.

9.3 Usermode API

Beckhoff.TwinCAT.TcEventLoggerAdsProxy.Net von NuGet.org

Die API steht auf NuGet.org über das Paket Beckhoff.TwinCAT.TcEventLoggerAdsProxy.Net zur Einbindung in Projekte bereit. Zum einfachen Einstieg finden Sie dort Beispiel-Code in der README-Datei, der lediglich in ein .NET Projekt kopiert werden muss.

> **COM basierte Schnittstelle wird ersetzt**
>
> Die COM basierte Schnittstelle, welche hier zuvor beschrieben ist, wird bis TwinCAT 3.1 4024 unterstützt. Aufgrund der genutzten Technologie kann sie nicht unter TwinCAT/BSD angeboten werden.
> Die auf NuGet.org bereitgestellte API ist der Nachfolger und bietet durch eine äquivalente API eine einfache Portierung für Kundenanwendungen.

Sehen Sie dazu auch

Usermode API [▶ 160]

10 Anhang

10.1 ADS Return Codes

Gruppierung der Fehlercodes:
Globale Fehlercodes: 0x0000 [▶ 213]... (0x9811_0000 ...)
Router Fehlercodes: 0x0500 [▶ 213]... (0x9811_0500 ...)
Allgemeine ADS Fehler: 0x0700 [▶ 214]... (0x9811_0700 ...)
RTime Fehlercodes: 0x1000 [▶ 216]... (0x9811_1000 ...)

Globale Fehlercodes

Hex	Dec	HRESULT	Name	Beschreibung
0x0	0	0x98110000	ERR_NOERROR	Kein Fehler.
0x1	1	0x98110001	ERR_INTERNAL	Interner Fehler.
0x2	2	0x98110002	ERR_NORTIME	Keine Echtzeit.
0x3	3	0x98110003	ERR_ALLOCLOCKEDMEM	Zuweisung gesperrt - Speicherfehler.
0x4	4	0x98110004	ERR_INSERTMAILBOX	Postfach voll – Es konnte die ADS Nachricht nicht versendet werden. Reduzieren der Anzahl der ADS Nachrichten pro Zyklus bringt Abhilfe.
0x5	5	0x98110005	ERR_WRONGRECEIVEHMSG	Falsches HMSG.
0x6	6	0x98110006	ERR_TARGETPORTNOTFOUND	Ziel-Port nicht gefunden – ADS Server ist nicht gestartet, nicht erreichbar oder nicht installiert.
0x7	7	0x98110007	ERR_TARGETMACHINENOTFOUND	Zielrechner nicht gefunden – AMS Route wurde nicht gefunden.
0x8	8	0x98110008	ERR_UNKNOWNCMDID	Unbekannte Befehl-ID.
0x9	9	0x98110009	ERR_BADTASKID	Ungültige Task-ID.
0xA	10	0x9811000A	ERR_NOIO	Kein IO.
0xB	11	0x9811000B	ERR_UNKNOWNAMSCMD	Unbekannter AMS-Befehl.
0xC	12	0x9811000C	ERR_WIN32ERROR	Win32 Fehler.
0xD	13	0x9811000D	ERR_PORTNOTCONNECTED	Port nicht verbunden.
0xE	14	0x9811000E	ERR_INVALIDAMSLENGTH	Ungültige AMS-Länge.
0xF	15	0x9811000F	ERR_INVALIDAMSNETID	Ungültige AMS Net ID.
0x10	16	0x98110010	ERR_LOWINSTLEVEL	Installations-Level ist zu niedrig –TwinCAT 2 Lizenzfehler.
0x11	17	0x98110011	ERR_NODEBUGINTAVAILABLE	Kein Debugging verfügbar.
0x12	18	0x98110012	ERR_PORTDISABLED	Port deaktiviert – TwinCAT System Service nicht gestartet.
0x13	19	0x98110013	ERR_PORTALREADYCONNECTED	Port bereits verbunden.
0x14	20	0x98110014	ERR_AMSSYNC_W32ERROR	AMS Sync Win32 Fehler.
0x15	21	0x98110015	ERR_AMSSYNC_TIMEOUT	AMS Sync Timeout.
0x16	22	0x98110016	ERR_AMSSYNC_AMSERROR	AMS Sync Fehler.
0x17	23	0x98110017	ERR_AMSSYNC_NOINDEXINMAP	Keine Index-Map für AMS Sync vorhanden.
0x18	24	0x98110018	ERR_INVALIDAMSPORT	Ungültiger AMS-Port.
0x19	25	0x98110019	ERR_NOMEMORY	Kein Speicher.
0x1A	26	0x9811001A	ERR_TCPSEND	TCP Sendefehler.
0x1B	27	0x9811001B	ERR_HOSTUNREACHABLE	Host nicht erreichbar.
0x1C	28	0x9811001C	ERR_INVALIDAMSFRAGMENT	Ungültiges AMS Fragment.
0x1D	29	0x9811001D	ERR_TLSSEND	TLS Sendefehler – Secure ADS Verbindung fehlgeschlagen.
0x1E	30	0x9811001E	ERR_ACCESSDENIED	Zugriff Verweigert – Secure ADS Zugriff verweigert.

Router Fehlercodes

Hex	Dec	HRESULT	Name	Beschreibung
0x500	1280	0x98110500	ROUTERERR_NOLOCKEDMEMORY	Lockierter Speicher kann nicht zugewiesen werden.
0x501	1281	0x98110501	ROUTERERR_RESIZEMEMORY	Die Größe des Routerspeichers konnte nicht geändert werden.
0x502	1282	0x98110502	ROUTERERR_MAILBOXFULL	Das Postfach hat die maximale Anzahl der möglichen Meldungen erreicht.
0x503	1283	0x98110503	ROUTERERR_DEBUGBOXFULL	Das Debug Postfach hat die maximale Anzahl der möglichen Meldungen erreicht.
0x504	1284	0x98110504	ROUTERERR_UNKNOWNPORTTYPE	Der Porttyp ist unbekannt.
0x505	1285	0x98110505	ROUTERERR_NOTINITIALIZED	Router ist nicht initialisiert.
0x506	1286	0x98110506	ROUTERERR_PORTALREADYINUSE	Die Portnummer ist bereits vergeben.
0x507	1287	0x98110507	ROUTERERR_NOTREGISTERED	Der Port ist nicht registriert.
0x508	1288	0x98110508	ROUTERERR_NOMOREQUEUES	Die maximale Portanzahl ist erreicht.
0x509	1289	0x98110509	ROUTERERR_INVALIDPORT	Der Port ist ungültig.
0x50A	1290	0x9811050A	ROUTERERR_NOTACTIVATED	Der Router ist nicht aktiv.
0x50B	1291	0x9811050B	ROUTERERR_FRAGMENTBOXFULL	Das Postfach hat die maximale Anzahl für fragmentierte Nachrichten erreicht.
0x50C	1292	0x9811050C	ROUTERERR_FRAGMENTTIMEOUT	Fragment Timeout aufgetreten.
0x50D	1293	0x9811050D	ROUTERERR_TOBEREMOVED	Port wird entfernt.

Allgemeine ADS Fehlercodes

Hex	Dec	HRESULT	Name	Beschreibung
0x700	1792	0x98110700	ADSERR_DEVICE_ERROR	Allgemeiner Gerätefehler.
0x701	1793	0x98110701	ADSERR_DEVICE_SRVNOTSUPP	Service wird vom Server nicht unterstützt.
0x702	1794	0x98110702	ADSERR_DEVICE_INVALIDGRP	Ungültige Index-Gruppe.
0x703	1795	0x98110703	ADSERR_DEVICE_INVALIDOFFSET	Ungültiger Index-Offset.
0x704	1796	0x98110704	ADSERR_DEVICE_INVALIDACCESS	Lesen oder Schreiben nicht gestattet. Mehrere Ursachen sind möglich. Beispielsweise beim Anlegen von Routen, dass ein falsches Passwort angegeben wurde.
0x705	1797	0x98110705	ADSERR_DEVICE_INVALIDSIZE	Parametergröße nicht korrekt.
0x706	1798	0x98110706	ADSERR_DEVICE_INVALIDDATA	Ungültige Daten-Werte.
0x707	1799	0x98110707	ADSERR_DEVICE_NOTREADY	Gerät nicht betriebsbereit.
0x708	1800	0x98110708	ADSERR_DEVICE_BUSY	Gerät beschäftigt.
0x709	1801	0x98110709	ADSERR_DEVICE_INVALIDCONTEXT	Ungültiger Kontext vom Betriebssystem - Kann durch Verwendung von ADS Bausteinen in unterschiedlichen Tasks auftreten. Abhilfe kann die Multitasking-Syncronisation in der SPS geben.
0x70A	1802	0x9811070A	ADSERR_DEVICE_NOMEMORY	Nicht genügend Speicher.
0x70B	1803	0x9811070B	ADSERR_DEVICE_INVALIDPARM	Ungültige Parameter-Werte.
0x70C	1804	0x9811070C	ADSERR_DEVICE_NOTFOUND	Nicht gefunden (Dateien,...).
0x70D	1805	0x9811070D	ADSERR_DEVICE_SYNTAX	Syntax-Fehler in Datei oder Befehl.
0x70E	1806	0x9811070E	ADSERR_DEVICE_INCOMPATIBLE	Objekte stimmen nicht überein.
0x70F	1807	0x9811070F	ADSERR_DEVICE_EXISTS	Objekt ist bereits vorhanden.
0x710	1808	0x98110710	ADSERR_DEVICE_SYMBOLNOTFOUND	Symbol nicht gefunden.
0x711	1809	0x98110711	ADSERR_DEVICE_SYMBOLVERSIONINVALID	Symbol-Version ungültig – Kann durch einen Online-Change auftreten. Erzeuge einen neuen Handle.
0x712	1810	0x98110712	ADSERR_DEVICE_INVALIDSTATE	Gerät (Server) ist im ungültigen Zustand.
0x713	1811	0x98110713	ADSERR_DEVICE_TRANSMODENOTSUPP	AdsTransMode nicht unterstützt.
0x714	1812	0x98110714	ADSERR_DEVICE_NOTIFYHNDINVALID	Notification Handle ist ungültig.
0x715	1813	0x98110715	ADSERR_DEVICE_CLIENTUNKNOWN	Notification-Client nicht registriert.
0x716	1814	0x98110716	ADSERR_DEVICE_NOMOREHDLS	Keine weiteren Handles verfügbar.
0x717	1815	0x98110717	ADSERR_DEVICE_INVALIDWATCHSIZE	Größe der Notification zu groß.
0x718	1816	0x98110718	ADSERR_DEVICE_NOTINIT	Gerät nicht initialisiert.
0x719	1817	0x98110719	ADSERR_DEVICE_TIMEOUT	Gerät hat einen Timeout.
0x71A	1818	0x9811071A	ADSERR_DEVICE_NOINTERFACE	Interface Abfrage fehlgeschlagen.
0x71B	1819	0x9811071B	ADSERR_DEVICE_INVALIDINTERFACE	Falsches Interface angefordert.
0x71C	1820	0x9811071C	ADSERR_DEVICE_INVALIDCLSID	Class-ID ist ungültig.
0x71D	1821	0x9811071D	ADSERR_DEVICE_INVALIDOBJID	Object-ID ist ungültig.
0x71E	1822	0x9811071E	ADSERR_DEVICE_PENDING	Anforderung steht aus.
0x71F	1823	0x9811071F	ADSERR_DEVICE_ABORTED	Anforderung wird abgebrochen.
0x720	1824	0x98110720	ADSERR_DEVICE_WARNING	Signal-Warnung.
0x721	1825	0x98110721	ADSERR_DEVICE_INVALIDARRAYIDX	Ungültiger Array-Index.
0x722	1826	0x98110722	ADSERR_DEVICE_SYMBOLNOTACTIVE	Symbol nicht aktiv.
0x723	1827	0x98110723	ADSERR_DEVICE_ACCESSDENIED	Zugriff verweigert. Mehrere Ursachen sind möglich. Beispielsweise, dass eine Unidirectionale ADS Route in die umgekehrte Richtung verwendet wird.
0x724	1828	0x98110724	ADSERR_DEVICE_LICENSENOTFOUND	Fehlende Lizenz.
0x725	1829	0x98110725	ADSERR_DEVICE_LICENSEEXPIRED	Lizenz abgelaufen.
0x726	1830	0x98110726	ADSERR_DEVICE_LICENSEEXCEEDED	Lizenz überschritten.
0x727	1831	0x98110727	ADSERR_DEVICE_LICENSEINVALID	Lizenz ungültig.
0x728	1832	0x98110728	ADSERR_DEVICE_LICENSESYSTEMID	Lizenzproblem: System-ID ist ungültig.
0x729	1833	0x98110729	ADSERR_DEVICE_LICENSENOTIMELIMIT	Lizenz nicht zeitlich begrenzt.
0x72A	1834	0x9811072A	ADSERR_DEVICE_LICENSEFUTUREISSUE	Lizenzproblem: Zeitpunkt in der Zukunft.
0x72B	1835	0x9811072B	ADSERR_DEVICE_LICENSETIMETOLONG	Lizenz-Zeitraum zu lang.
0x72C	1836	0x9811072C	ADSERR_DEVICE_EXCEPTION	Exception beim Systemstart.
0x72D	1837	0x9811072D	ADSERR_DEVICE_LICENSEDUPLICATED	Lizenz-Datei zweimal gelesen.
0x72E	1838	0x9811072E	ADSERR_DEVICE_SIGNATUREINVALID	Ungültige Signatur.
0x72F	1839	0x9811072F	ADSERR_DEVICE_CERTIFICATEINVALID	Zertifikat ungültig.
0x730	1840	0x98110730	ADSERR_DEVICE_LICENSEOEMNOTFOUND	Public Key vom OEM nicht bekannt.
0x731	1841	0x98110731	ADSERR_DEVICE_LICENSERESTRICTED	Lizenz nicht gültig für diese System.ID.

Hex	Dec	HRESULT	Name	Beschreibung
0x732	1842	0x98110732	ADSERR_DEVICE_LICENSEDEMODENIED	Demo-Lizenz untersagt.
0x733	1843	0x98110733	ADSERR_DEVICE_INVALIDFNCID	Funktions-ID ungültig.
0x734	1844	0x98110734	ADSERR_DEVICE_OUTOFRANGE	Außerhalb des gültigen Bereiches.
0x735	1845	0x98110735	ADSERR_DEVICE_INVALIDALIGNMENT	Ungültiges Alignment.
0x736	1846	0x98110736	ADSERR_DEVICE_LICENSEPLATFORM	Ungültiger Plattform Level.
0x737	1847	0x98110737	ADSERR_DEVICE_FORWARD_PL	Kontext – Weiterleitung zum Passiv-Level.
0x738	1848	0x98110738	ADSERR_DEVICE_FORWARD_DL	Kontext – Weiterleitung zum Dispatch-Level.
0x739	1849	0x98110739	ADSERR_DEVICE_FORWARD_RT	Kontext – Weiterleitung zur Echtzeit.
0x740	1856	0x98110740	ADSERR_CLIENT_ERROR	Clientfehler.
0x741	1857	0x98110741	ADSERR_CLIENT_INVALIDPARM	Dienst enthält einen ungültigen Parameter.
0x742	1858	0x98110742	ADSERR_CLIENT_LISTEMPTY	Polling-Liste ist leer.
0x743	1859	0x98110743	ADSERR_CLIENT_VARUSED	Var-Verbindung bereits im Einsatz.
0x744	1860	0x98110744	ADSERR_CLIENT_DUPLINVOKEID	Die aufgerufene ID ist bereits in Benutzung.
0x745	1861	0x98110745	ADSERR_CLIENT_SYNCTIMEOUT	Timeout ist aufgetreten – Die Gegenstelle antwortet nicht im vorgegebenen ADS Timeout. Die Routeneinstellung der Gegenstelle kann falsch konfiguriert sein.
0x746	1862	0x98110746	ADSERR_CLIENT_W32ERROR	Fehler im Win32 Subsystem.
0x747	1863	0x98110747	ADSERR_CLIENT_TIMEOUTINVALID	Ungültiger Client Timeout-Wert.
0x748	1864	0x98110748	ADSERR_CLIENT_PORTNOTOPEN	Port nicht geöffnet.
0x749	1865	0x98110749	ADSERR_CLIENT_NOAMSADDR	Keine AMS Adresse.
0x750	1872	0x98110750	ADSERR_CLIENT_SYNCINTERNAL	Interner Fehler in Ads-Sync.
0x751	1873	0x98110751	ADSERR_CLIENT_ADDHASH	Überlauf der Hash-Tabelle.
0x752	1874	0x98110752	ADSERR_CLIENT_REMOVEHASH	Schlüssel in der Tabelle nicht gefunden.
0x753	1875	0x98110753	ADSERR_CLIENT_NOMORESYM	Keine Symbole im Cache.
0x754	1876	0x98110754	ADSERR_CLIENT_SYNCRESINVALID	Ungültige Antwort erhalten.
0x755	1877	0x98110755	ADSERR_CLIENT_SYNCPORTLOCKED	Sync Port ist verriegelt.
0x756	1878	0x98110756	ADSERR_CLIENT_REQUESTCANCELLED	Die Anfrage wurde abgebrochen.

RTime Fehlercodes

Hex	Dec	HRESULT	Name	Beschreibung
0x1000	4096	0x98111000	RTERR_INTERNAL	Interner Fehler im Echtzeit-System.
0x1001	4097	0x98111001	RTERR_BADTIMERPERIODS	Timer-Wert nicht gültig.
0x1002	4098	0x98111002	RTERR_INVALIDTASKPTR	Task-Pointer hat den ungültigen Wert 0 (null).
0x1003	4099	0x98111003	RTERR_INVALIDSTACKPTR	Stack-Pointer hat den ungültigen Wert 0 (null).
0x1004	4100	0x98111004	RTERR_PRIOEXISTS	Die Request Task Priority ist bereits vergeben.
0x1005	4101	0x98111005	RTERR_NOMORETCB	Kein freier TCB (Task Control Block) verfügbar. Maximale Anzahl von TCBs beträgt 64.
0x1006	4102	0x98111006	RTERR_NOMORESEMAS	Keine freien Semaphoren zur Verfügung. Maximale Anzahl der Semaphoren beträgt 64.
0x1007	4103	0x98111007	RTERR_NOMOREQUEUES	Kein freier Platz in der Warteschlange zur Verfügung. Maximale Anzahl der Plätze in der Warteschlange beträgt 64.
0x100D	4109	0x9811100D	RTERR_EXTIRQALREADYDEF	Ein externer Synchronisations-Interrupt wird bereits angewandt.
0x100E	4110	0x9811100E	RTERR_EXTIRQNOTDEF	Kein externer Sync-Interrupt angewandt.
0x100F	4111	0x9811100F	RTERR_EXTIRQINSTALLFAILED	Anwendung des externen Synchronisierungs-Interrupts ist fehlgeschlagen.
0x1010	4112	0x98111010	RTERR_IRQLNOTLESSOREQUAL	Aufruf einer Service-Funktion im falschen Kontext
0x1017	4119	0x98111017	RTERR_VMXNOTSUPPORTED	Intel VT-x Erweiterung wird nicht unterstützt.
0x1018	4120	0x98111018	RTERR_VMXDISABLED	Intel VT-x Erweiterung ist nicht aktiviert im BIOS.
0x1019	4121	0x98111019	RTERR_VMXCONTROLSMISSING	Fehlende Funktion in Intel VT-x Erweiterung.
0x101A	4122	0x9811101A	RTERR_VMXENABLEFAILS	Aktivieren von Intel VT-x schlägt fehl.

Spezifische positive HRESULT Return Codes:

HRESULT	Name	Beschreibung
0x0000_0000	S_OK	Kein Fehler.
0x0000_0001	S_FALSE	Kein Fehler. Bsp.: erfolgreiche Abarbeitung, bei der jedoch ein negatives oder unvollständiges Ergebnis erzielt wurde.
0x0000_0203	S_PENDING	Kein Fehler. Bsp.: erfolgreiche Abarbeitung, bei der jedoch noch kein Ergebnis vorliegt.
0x0000_0256	S_WATCHDOG_TIMEOUT	Kein Fehler. Bsp.: erfolgreiche Abarbeitung, bei der jedoch eine Zeitüberschreitung eintrat.

TCP Winsock-Fehlercodes

Hex	Dec	Name	Beschreibung
0x274C	10060	WSAETIMEDOUT	Verbindungs Timeout aufgetreten - Fehler beim Herstellen der Verbindung, da die Gegenstelle nach einer bestimmten Zeitspanne nicht ordnungsgemäß reagiert hat, oder die hergestellte Verbindung konnte nicht aufrecht erhalten werden, da der verbundene Host nicht reagiert hat.
0x274D	10061	WSAECONNREFUSED	Verbindung abgelehnt - Es konnte keine Verbindung hergestellt werden, da der Zielcomputer dies explizit abgelehnt hat. Dieser Fehler resultiert normalerweise aus dem Versuch, eine Verbindung mit einem Dienst herzustellen, der auf dem fremden Host inaktiv ist—das heißt, einem Dienst, für den keine Serveranwendung ausgeführt wird.
0x2751	10065	WSAEHOSTUNREACH	Keine Route zum Host - Ein Socketvorgang bezog sich auf einen nicht verfügbaren Host.
Weitere Winsock-Fehlercodes: Win32-Fehlercodes			

10.2 Support und Service

Beckhoff und seine weltweiten Partnerfirmen bieten einen umfassenden Support und Service, der eine schnelle und kompetente Unterstützung bei allen Fragen zu Beckhoff Produkten und Systemlösungen zur Verfügung stellt.

Downloadfinder

Unser Downloadfinder beinhaltet alle Dateien, die wir Ihnen zum Herunterladen anbieten. Sie finden dort Applikationsberichte, technische Dokumentationen, technische Zeichnungen, Konfigurationsdateien und vieles mehr.

Die Downloads sind in verschiedenen Formaten erhältlich.

Beckhoff Niederlassungen und Vertretungen

Wenden Sie sich bitte an Ihre Beckhoff Niederlassung oder Ihre Vertretung für den lokalen Support und Service zu Beckhoff Produkten!

Die Adressen der weltweiten Beckhoff Niederlassungen und Vertretungen entnehmen Sie bitte unserer Internetseite: www.beckhoff.com

Dort finden Sie auch weitere Dokumentationen zu Beckhoff Komponenten.

Beckhoff Support

Der Support bietet Ihnen einen umfangreichen technischen Support, der Sie nicht nur bei dem Einsatz einzelner Beckhoff Produkte, sondern auch bei weiteren umfassenden Dienstleistungen unterstützt:

- Support
- Planung, Programmierung und Inbetriebnahme komplexer Automatisierungssysteme
- umfangreiches Schulungsprogramm für Beckhoff Systemkomponenten

Hotline:	+49 5246 963-157
E-Mail:	support@beckhoff.com

Beckhoff Service

Das Beckhoff Service-Center unterstützt Sie rund um den After-Sales-Service:

- Vor-Ort-Service
- Reparaturservice
- Ersatzteilservice
- Hotline-Service

Hotline:	+49 5246 963-460
E-Mail:	service@beckhoff.com

Beckhoff Unternehmenszentrale

Beckhoff Automation GmbH & Co. KG

Hülshorstweg 20
33415 Verl
Deutschland

Telefon:	+49 5246 963-0
E-Mail:	info@beckhoff.com
Internet:	www.beckhoff.com

Trademark statements

Beckhoff®, ATRO® , EtherCAT®, EtherCAT G®, EtherCAT G10®, EtherCAT P®, MX-System®, Safety over EtherCAT®, TC/BSD®, TwinCAT®, TwinCAT/BSD®, TwinSAFE®, XFC®, XPlanar® and XTS® are registered and licensed trademarks of Beckhoff Automation GmbH.

Third-party trademark statements

Arm, Arm9 and Cortex are trademarks or registered trademarks of Arm Limited (or its subsidiaries or affiliates) in the US and/or elsewhere.

Intel, the Intel logo, Intel Core, Xeon, Intel Atom, Celeron and Pentium are trademarks of Intel Corporation or its subsidiaries.

Microsoft, Microsoft Azure, Microsoft Edge, PowerShell, Visual Studio, Windows and Xbox are trademarks of the Microsoft group of companies.

Mehr Informationen:
www.beckhoff.com/te1000

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