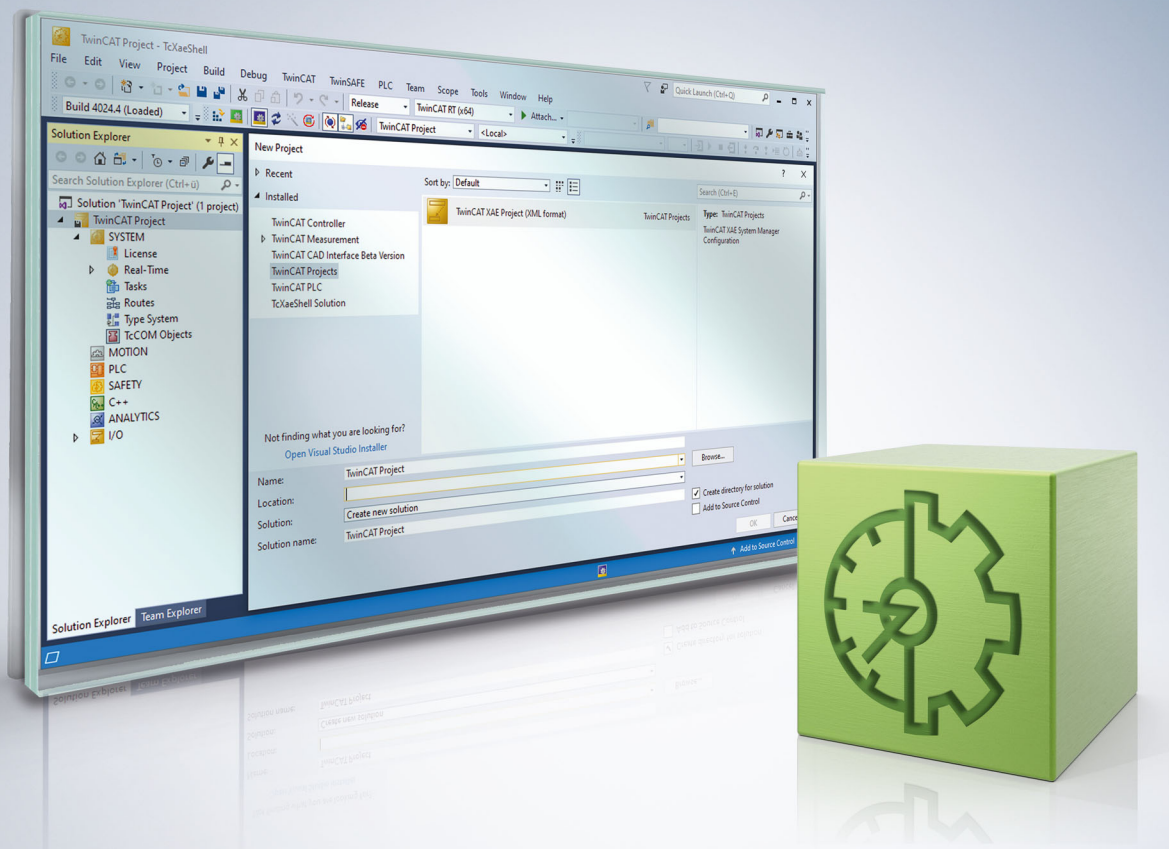
BECKHOFF New Automation Technology

Manual | EN

TE1000

TwinCAT 3 | EventLogger



2022-12-20 | Version: 1.10.0

Table of contents

1 Foreword

1.1 Notes on the documentation

This description is only intended for the use of trained specialists in control and automation engineering who are familiar with applicable national standards.
It is essential that the documentation and the following notes and explanations are followed when installing and commissioning the components.
It is the duty of the technical personnel to use the documentation published at the respective time of each installation and commissioning.

The responsible staff must ensure that the application or use of the products described satisfy all the requirements for safety, including all the relevant laws, regulations, guidelines and standards.

Disclaimer

The documentation has been prepared with care. The products described are, however, constantly under development.
We reserve the right to revise and change the documentation at any time and without prior announcement.
No claims for the modification of products that have already been supplied may be made on the basis of the data, diagrams and descriptions in this documentation.

Trademarks

Beckhoff®, TwinCAT®, TwinCAT/BSD®, TC/BSD®, EtherCAT®, EtherCAT G®, EtherCAT G10®, EtherCAT P®, Safety over EtherCAT®, TwinSAFE®, XFC®, XTS® and XPlanar® are registered trademarks of and licensed by Beckhoff Automation GmbH.
Other designations used in this publication may be trademarks whose use by third parties for their own purposes could violate the rights of the owners.

Patent Pending

The EtherCAT Technology is covered, including but not limited to the following patent applications and patents:
EP1590927, EP1789857, EP1456722, EP2137893, DE102015105702
with corresponding applications or registrations in various other countries.

EtherCAT® is a registered trademark and patented technology, licensed by Beckhoff Automation GmbH, Germany

Copyright

1.2 Safety instructions

Safety regulations

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

Exclusion of liability

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

Personnel qualification

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

Description of symbols

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

⚠ DANGER

Serious risk of injury!

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

⚠ WARNING

Risk of injury!

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

⚠ CAUTION

Personal injuries!

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

NOTICE

Damage to the environment or devices

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

Tip or pointer

This symbol indicates information that contributes to better understanding.

1.3 Notes on information security

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

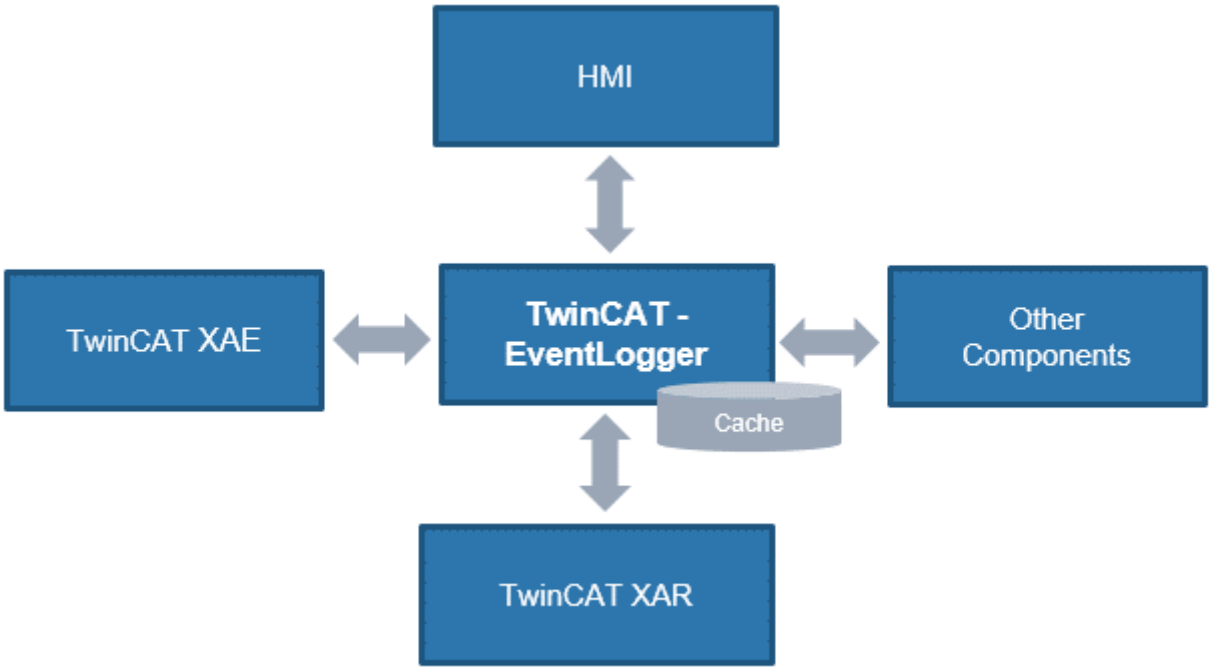
In addition, the recommendations from Beckhoff regarding appropriate protective measures should be observed. Further information regarding information security and industrial security can be found in our https://www.beckhoff.com/secguide.

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

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

2 Overview

The TwinCAT 3 EventLogger provides an interface for the exchange of messages between TwinCAT components and non-TwinCAT components.



This documentation is aimed at the user of the TwinCAT 3 EventLogger. Information about how the TwinCAT 3 EventLogger is used by different TwinCAT 3 components can be found in the respective product documentations.

Product components

All the components of the TwinCAT 3 EventLogger are present in the respective basic installation.

Various interfaces are included in the TwinCAT 3 Engineering for programming. The necessary modules and libraries are part of the corresponding runtime.

Use of the TwinCAT 3 EventLogger is free of license fees.

3 System requirements

Technical data	Requirement
Operating system	Windows 7/10, Windows Embedded Standard 7, Windows CE 7, TwinCAT/BSD
Target platform	PC architecture (x86, x64 and ARM)
TwinCAT version	TwinCAT 3.1 Build 4022.20 or higher
Required TwinCAT setup level	TwinCAT 3 XAE, XAR
Required TwinCAT license	Any runtime license (PLC, C++)
Visual Studio version	Parts of the TwinCAT 3 EventLogger are usable from Visual Studio 2013.

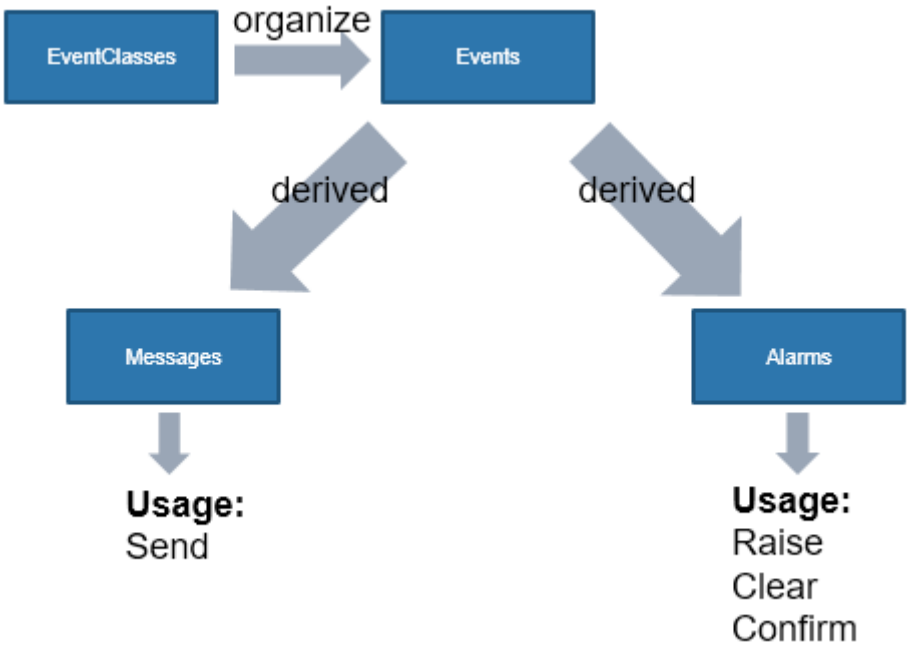
4 Limitations

- Events have a maximum size of 8 kB during transmission. When using the TwinCAT 3 EventLogger, make sure that this limit is observed. It refers to all elements that are transmitted and described in this documentation, including the dynamic elements (Attributes, SourceName, JSON Attributes).
- The interface for receiving events in real-time stores a maximum of 1024 events temporarily until they have to be retrieved.
  If they are not retrieved in time, events are lost.
- The TwinCAT 3 EventLogger offers a connection to the TwinCAT HMI (TF2xxx). The TwinCAT PLC HMI (TF18xx), conversely, cannot receive the events.

5 Technical introduction

The TwinCAT 3 EventLogger transmits so-called events. An event is a message or an alarm.

This technical introduction focuses on the data that are transmitted as the contents of an event, because they are necessary for the understanding of the process. The transmission and reception of an event are explained in detail in the API descriptions for PLC and C++ as well as in the samples.



Events

An event itself is not used directly, but rather the derived types "messages" or "alarms".

An event provides the following common elements of messages and alarms:

EventClass (GUID)	Event classes are groups of events (possibly for a topic).
Event-ID (UDINT)	The event is clearly identified within an event class by an Event-ID.
Text (String)	Description of the event. The description is intended for people and can therefore also be internationalized. Arguments can be inserted at runtime to make the text individually adaptable.
Source Info	Source of the occurrence of an event. Source consists of three elements. These can be used as desired, but there is a corresponding recommendation as to how they should be used. • **Source-ID (INT)**: TcCOM object ID • **Source Name (STRING)**: Path within the TcCOM object. e.g. path to a function block in a PLC project • **Source-GUID (GUID)**: Can be used, for example, to identify a project or a (sub-) product.
JSON Attribute (STRING)	Can be used as desired. Whereas the "text" (see above) is intended for people as recipients, the JSON attribute is intended for programmatic reception. A JSON string can easily be created (serialized) and processed (deserialized). TwinCAT offers the JsonXml library in real-time for this (see Documentation PLC library Tc3_JsonXml).

In addition to these elements, events possess further elements, which are described in the TMC Editor [▶ 15].

Messages

Messages are stateless. They are sent when called and the corresponding registered components are delivered.

Identification

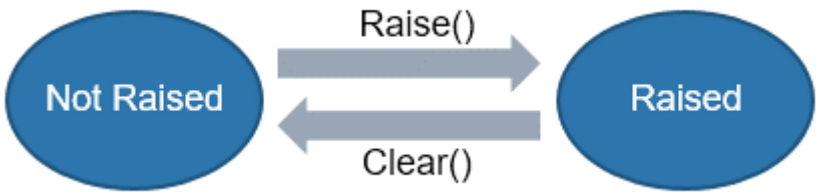
Messages are identified by the EventClass and Event-ID.

Alarm

As opposed to a message, an alarm is not stateless.

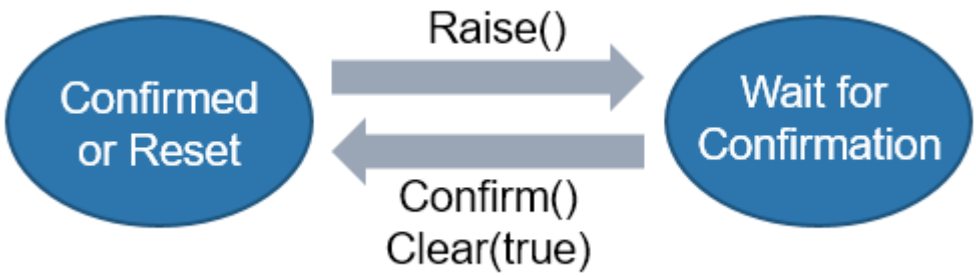
It has the following alarm states:

- Not-Raised
- Raised



In addition, a confirmation can be demanded. Distinction is made between the following confirmation states:

- WaitForConfirmation
- Confirmed or Reset



Calling the corresponding methods causes an event to be transmitted and sets the alarm to the respective state.

- If a confirmation takes place via Confirm(), the state is set to Confirmed.
- If a confirmation takes place via Clear(TRUE), the state is set to Reset.

If the call of a method is invalid in the current state, this will be indicated by a return value.

When shutting down TwinCAT (RUN → CONFIG transition), a dispose that sets a Clear time stamp is executed internally for all alarms in the Raised state. There is no confirmation for these alarms.

Identification

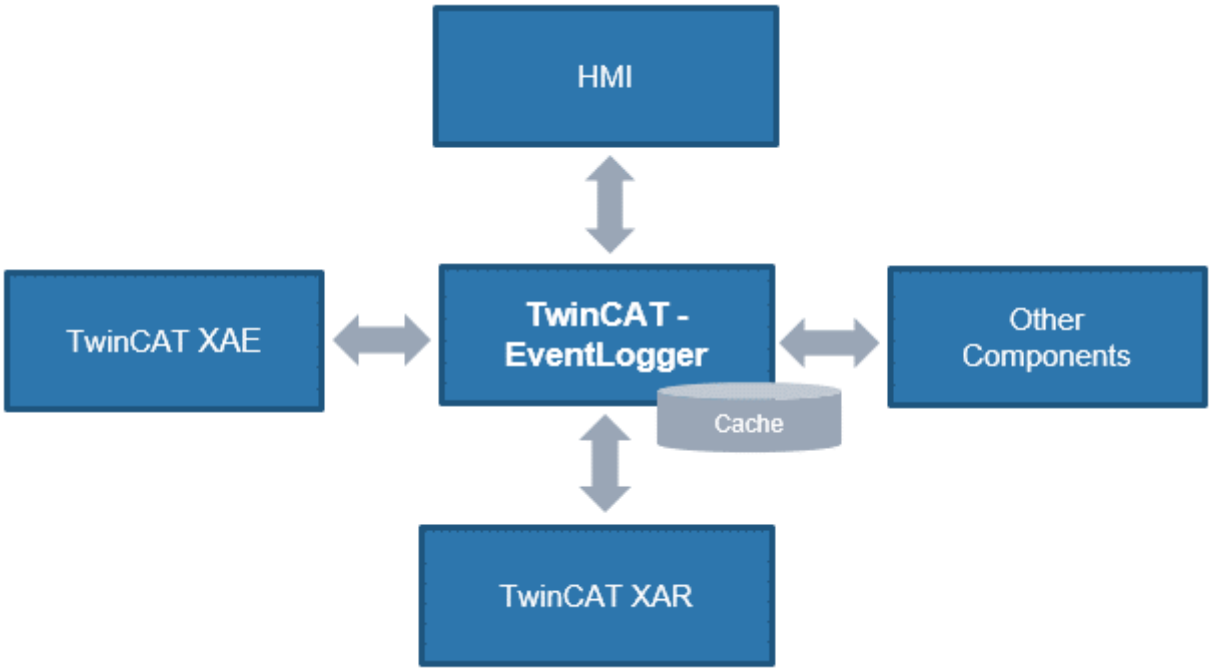
The TwinCAT 3 EventLogger identifies an alarm using the EventClass, Event-ID and Source Info. Thus an alarm (combination of EventClass and Event-ID) can be used at different points in a program. For example, an alarm "storage empty" can be used for different storages, since different "Source Info" is provided at runtime (see also Handling sources [▶ 28]).

Architecture

The TwinCAT 3 EventLogger transmits events centrally between other components. These components include the real-time programming interfaces PLC or C++ as the primary source of events.

During the development the messages can be displayed in the TwinCAT Engineering (XAE).

An HMI, for example, can receive messages and display them accordingly. The customer can create further components of his own for receiving events.



The TwinCAT 3 EventLogger keeps a limited number of the last events in a cache. This can be queried from the Engineering, for example after a restart, in order to enable a diagnosis to be made. The cache is dependent on the secured shutdown of the computer.

Cache not persistent under Windows CE

Starting with TwinCAT 3.1 Build 4024.25 the cache of events is not persisted under Windows CE systems for performance reasons.

Workflow

The general workflow for establishing asynchronous communication between components is as follows:

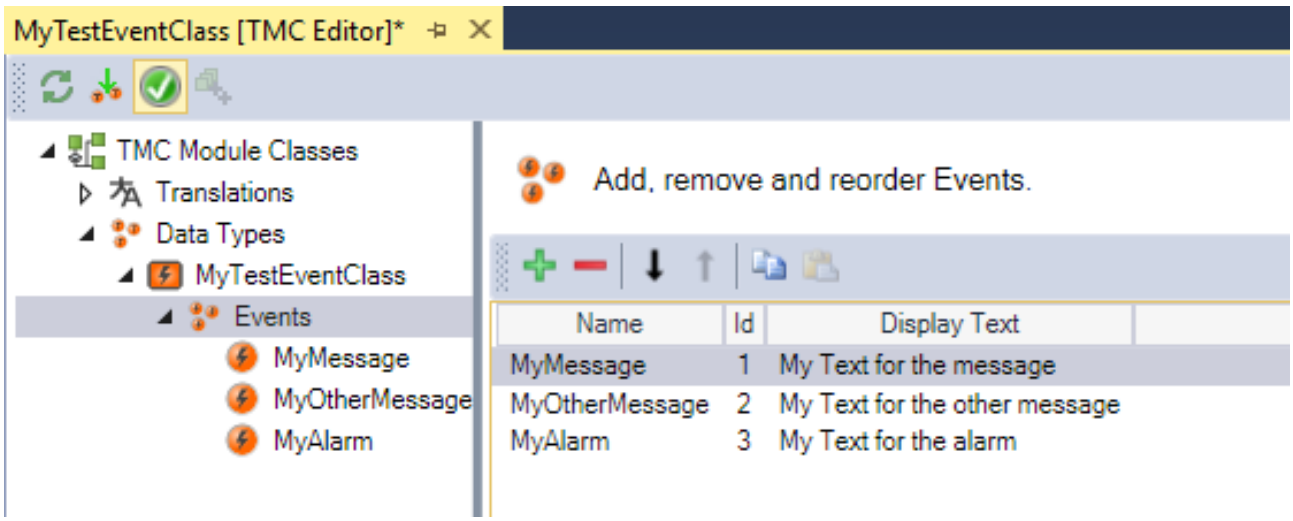
1. Creation of a new TwinCAT project.
2. Definition of event classes and events in the TwinCAT type system.
3. Execution of the automatic code generation in order to provide the source code for the real-time programming language in TwinCAT.
4. Implementation of the use and thus the sending and receiving of events.

See also:

- Documentation TwinCAT 3 type system
- API descriptions PLC [▶ 32] and C++ [▶ 101]
- Samples [▶ 157]

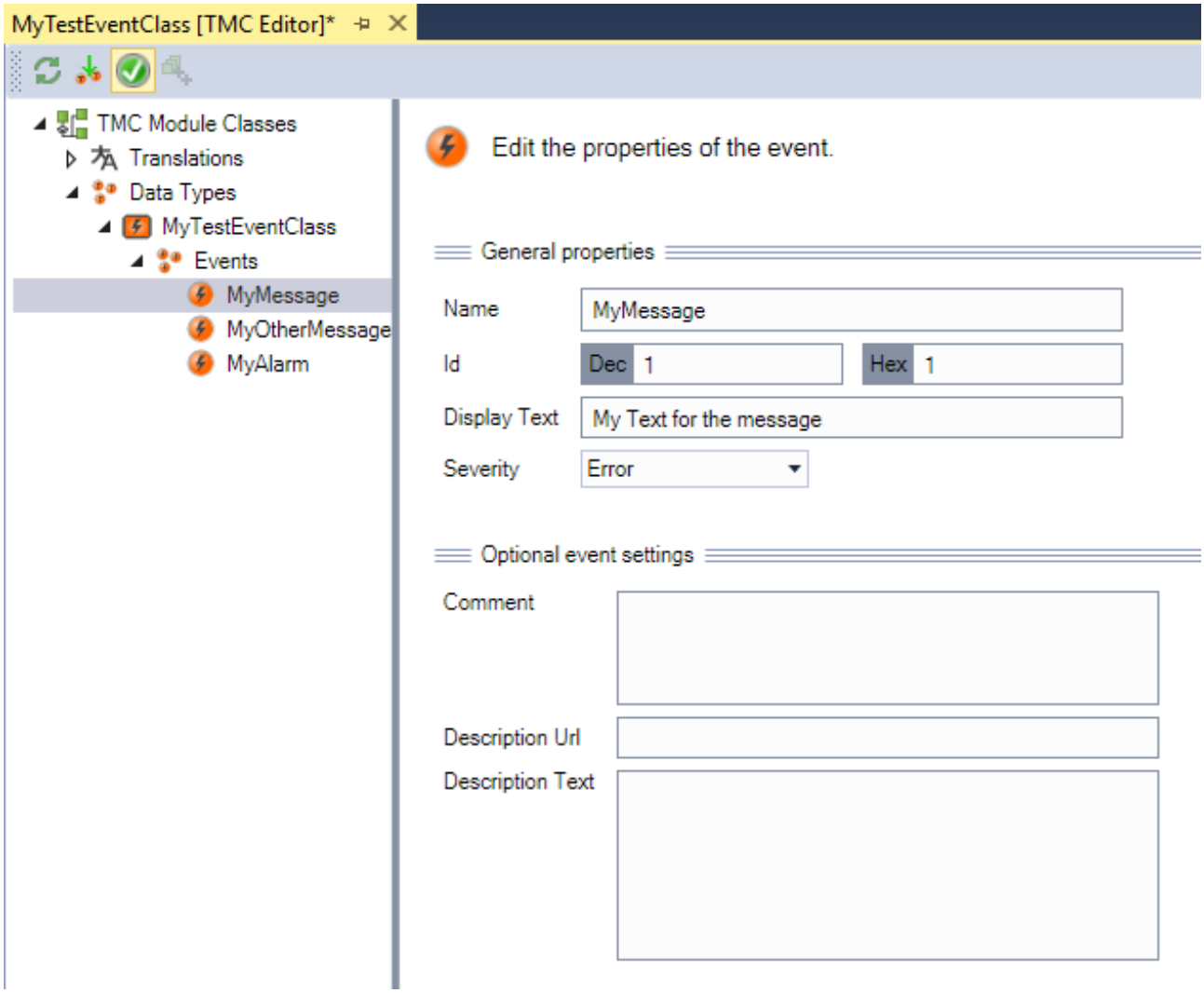
5.1 Event

Events are described within the event class.



The events can be configured accordingly through the subelements.

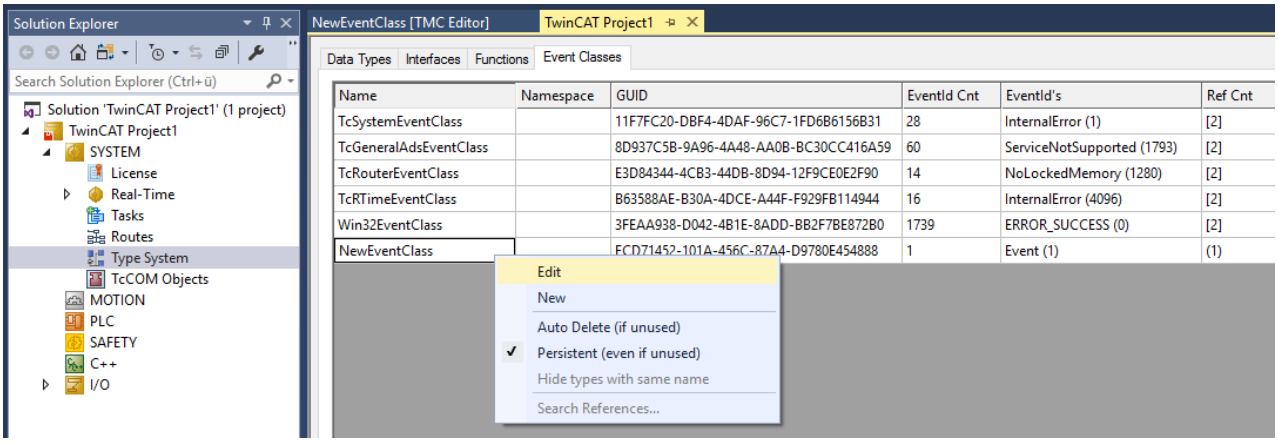
Properties of events



Name	This name designates the event, e.g. in the generated source code.
ID	Clearly identifies the event within the event class.
Display Text	This text is used for the event class for the display. It can be internationalized (see Internationalization/translations [▶ 21]).
Severity	Severity of the event. This is provided by the generated source code and thus represents the default behavior; however, a different value can also be used there: • Verbose • Info • Warning • Error • Critical

5.2 Event class

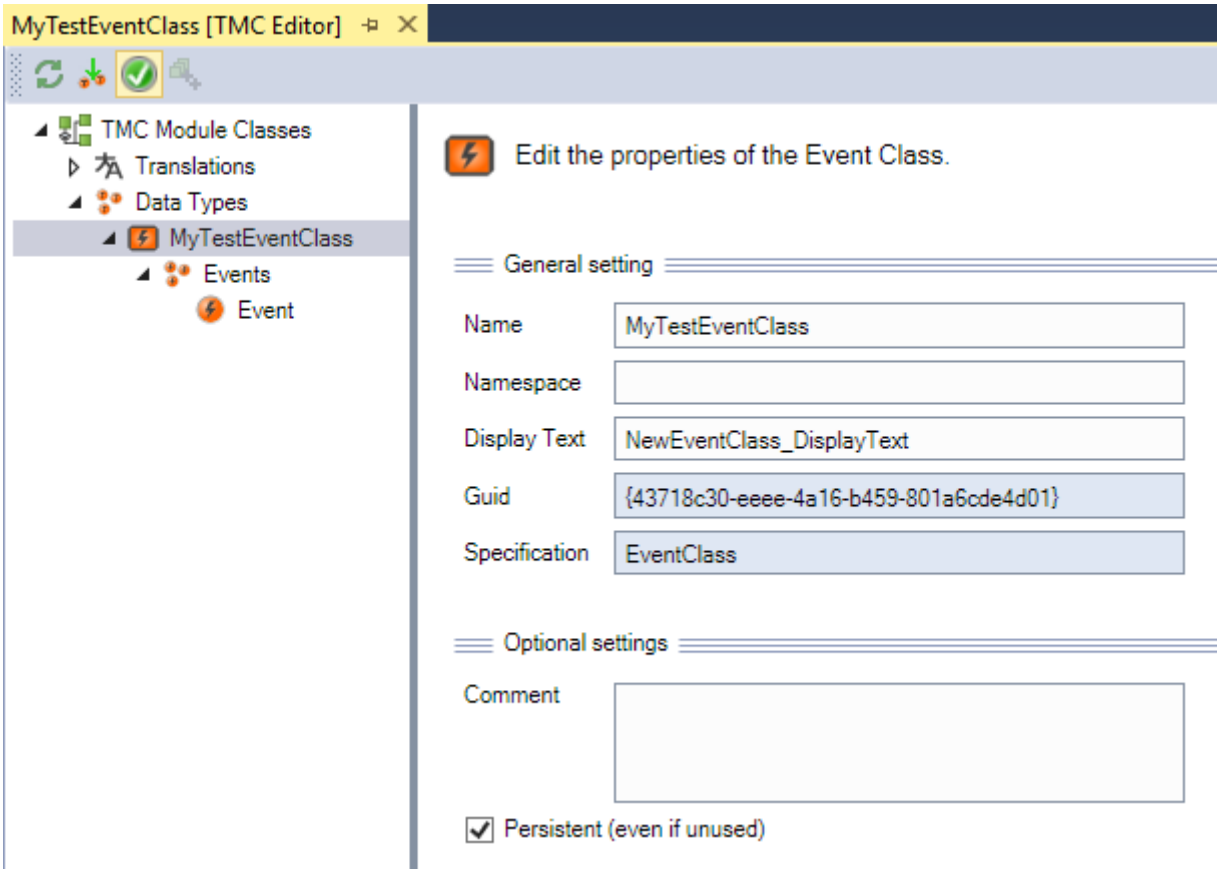
Event classes are groups of events (possibly for a theme) and, in the sense of the TwinCAT type system, data types that can be used in different modules. For this reason they are created as data types in the TwinCAT type system (System > Type System > Event Classes).



All known event classes are listed on the **Event Classes** tab in the TwinCAT type system. The TMC editor, in which the event classes can be defined and edited, can be opened via the context menu commands **Edit** and **New**.

In addition to the project event classes, TwinCAT also provides further event classes such as ADS return codes and system events.

Properties of event classes



Name	This name designates the event class, for example in the generated source code.
Namespace	Like all data types, event classes may also belong to a namespace.
DisplayText	This text is used for the event class for the display. It can be internationalized (see Internationalization/translations [▶ 21]).
Guid	Like all data types it identifies the currently described event class. It is calculated automatically and changes in the event of changes within the event class.

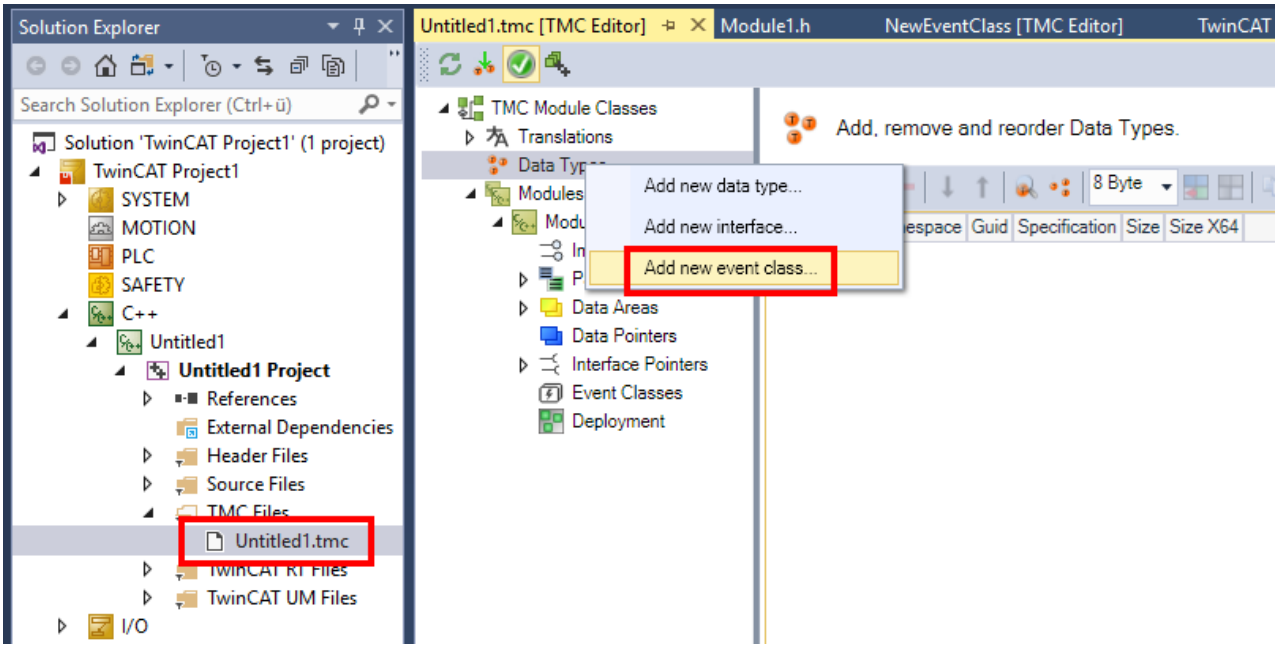
Event classes in PLC libraries

If event classes are used in a PLC library, they should also be part of the library in order to ensure that the data types are contained in the TwinCAT type system of every application that uses the PLC library.

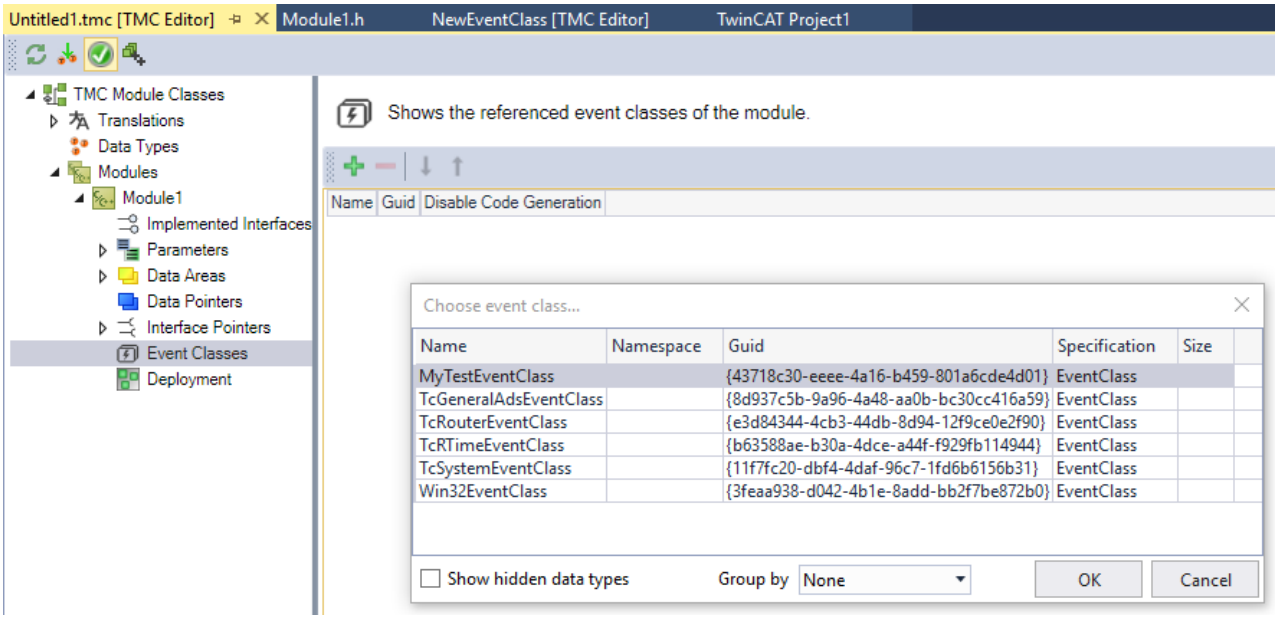
To make this possible the event classes must be "pinned" in the PLC library. To do this, select the command **Pin Global Data Type** in the context menu of the created data type under **External Types** in the PLC library object.

Event classes in C++ projects

Analogous to the other TwinCAT data types there is a possibility for C++ projects to define event classes locally in the C++ project.



The event classes are declared for use in the corresponding C++ modules (irrespective of where they were defined). If an event class is added to a C++ module, it is automatically embedded in the TMC file of the module.



5.3 Code generation of the event definition

Source code is generated from the definition of event classes with events, both in the PLC and in C++.

The "names" of the events and event classes are used in the code generation. An event class can thus be used across different versions of an event class through its name.

The "severity" is provided by the code generation. The programmer thus has the possibility to set this individually when creating the events. The severity as described in the TMC editor [▶ 15] should thus be considered the default behavior. The severity can deviate from this in the specific application case.

PLC

A GVL TC_EVENTS is created in the PLC and contains the event classes as subelements and is updated following changes (saving/closing the TMC editor). These global constants in turn contain the events themselves as subelements together with the individual elements EventId, Severity and the UUID of the event class to which they belong.

These elements can be used by means of IntelliSense for the parameters, e.g. with Create()/CreateEx().

Logged-in PLC

The code is not updated if a connection to a PLC exists (login). The update then takes place following a logout.

Via "OnlineChange" the changes to an event class can be applied if the option "Update boot project" is selected.

C++

TcCOM modules must use the event classes, i.e. they must be entered as used in the TMC editor. A code generation then creates a namespace, "TcEvents", as part of the <DriverName>Services.h- file. The namespace can be used by means of IntelliSense for the parameters of the event classes/events, e.g. with CreateMessage()/CreateAlarm().

Compatibility

The C++ source code generation requires Visual Studio 2013 or newer.

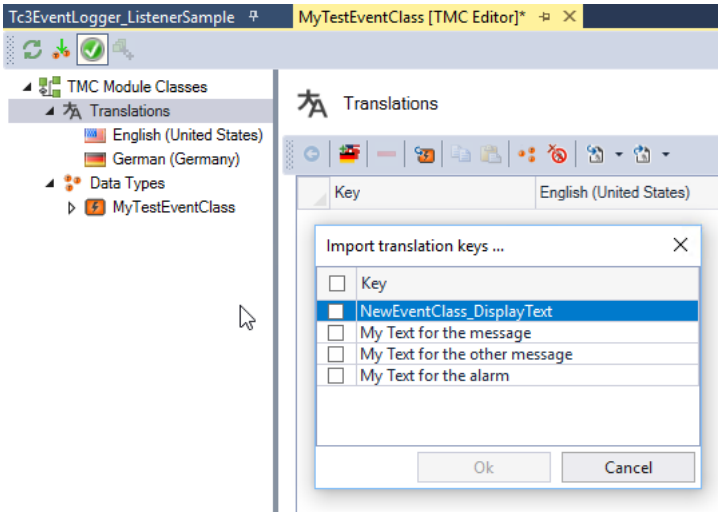
5.4 Internationalization/translations

The texts of the events ("DisplayText") for display in HMIs, for example, can be internationalized.

The "Translations" section in the TMC editor is provided for this. The section describes a table in whose rows the keys are assigned to the texts in different languages.

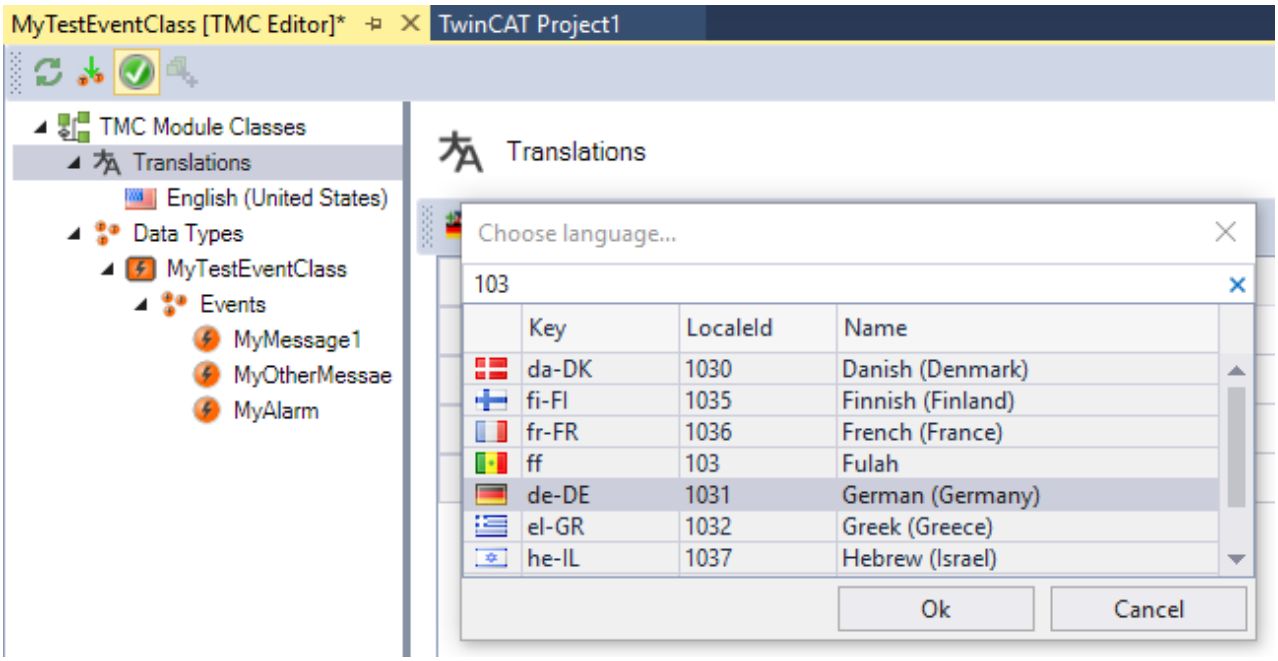
Keys to be translated

The keys are automatically determined from the texts ("Display Text") of the events and event classes. It is possible to describe individually whether the keys are to be adopted into the translation:



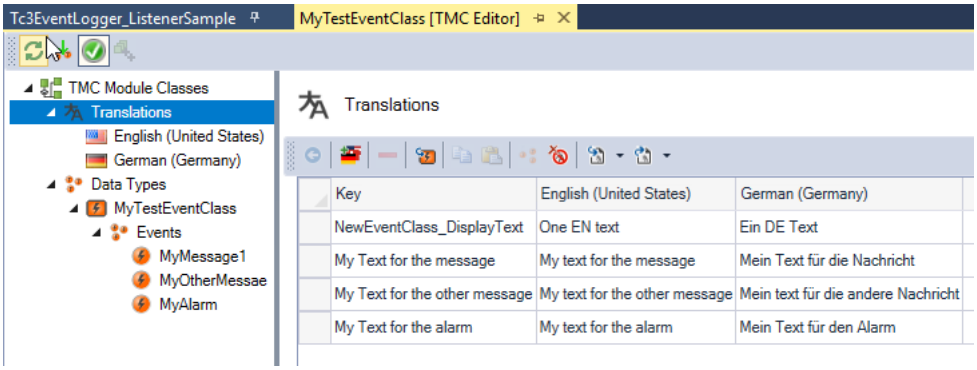
Languages to be included

Languages can be selected and added. If a language is requested at runtime for which no text is stored, the English text will be used.

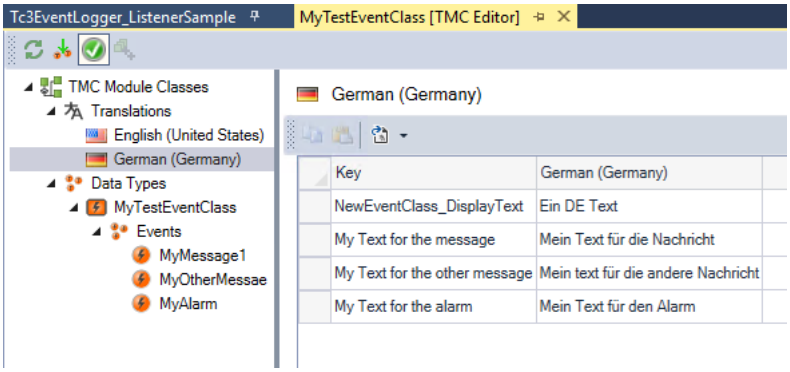


Translations

The translations can be placed in the table below "Translations".

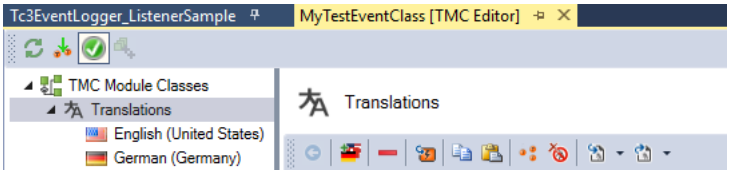


Alternatively the languages can be edited separately on the respective sub-nodes.



Additional functions

Further functions are available for the internationalization:



The following functions are available:

- Add a language: Adds a language, as described above.
- Remove selected translations: Removes a key with the translations.
- Import Translation Keys: Imports the key that is used from the event classes as described above.
- Copy: Copies the translation.
- Paste: Pastes a translation.
- Show Types: Displays the use of the selected key.
- Delete unused: Deletes the key that is not used in any EventClass from the table.
- Import: Imports the translations from an XML or CSV file.
- Export: Exports the translations to an XML or CSV file.

The import and export functions relate to an XML/CSV format, which can be investigated by way of an example through an export.

Translations outside of TwinCAT (XML)

The translation information is stored in its own area of the configuration files, which can also be edited outside of the TwinCAT XAE.

5.5 Target system

The TwinCAT 3 EventLogger can be configured on the target system by means of registry entries.

The following keys are usable below *HKEY_LOCAL_MACHINE\SOFTWARE\[WOW6432Node\]Beckhoff\TwinCAT3*:

Time stamp		
\EventLogger\TimestampSource	DWORD	0 = CurPentiumTime [default] 2 = CurSystemTime
\EventLogger\TimestampBase	DWORD	0 = SystemTime [default] 1 = ExternalTimeHard 2 = ExternalTimeMedium 3 = ExternalTimeSoft For 1 to 3, refer to the documentation for "Corrected time stamps" https://infosys.beckhoff.com/index.php?content=../content/1033/corrected_timestamps/index.html.
Maximum size of the message cache		
\EventLogger\MaxDatabaseSize	DWORD	20 [default] in MB On reaching the limit, half of the messages will be discarded.
Message cache location		
\EventLogger\DatabaseDir (From > TwinCAT 3.1 Build 4024.22)	STRING	:memory: = main memory (messages are deleted with the transition to CONFIG and also reboot etc.) <path> = file system folder for storing the LoggedEvents.db file If the key is not present, the database is placed in the boot folder. Under Windows CE this is used to store the database non-persistently.
Save messages in the 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
Logging TwinCAT system errors **See also** https://infosys.beckhoff.com/index.php?content=../content/1031/tceventlogger/12332566027.html		
HKEY_LOCAL_MACHINE\SOFTWARE\[WOW6432Node\]Beckhoff\TwinCAT3\System[LogMessageType]	DWORD	3 = acceptance of the system errors into the Tc3 Eventlogger 4 = acceptance of the system errors into the Tc3 Eventlogger and the error list of the XAE.

If the keys do not exist, they must be created with the specified type.

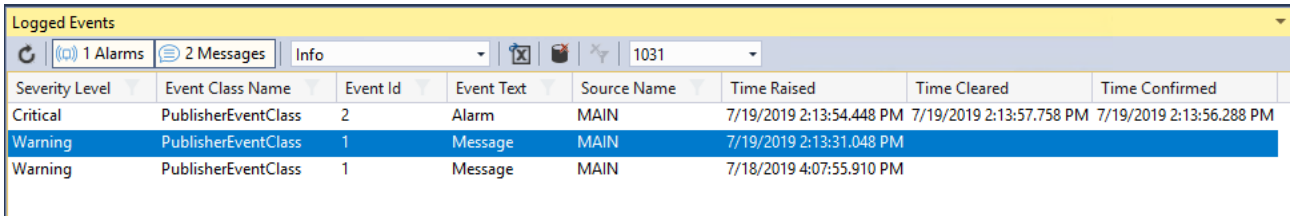
5.6 Engineering

TwinCAT Logged Events window

Compatibility

The **TwinCAT Logged Events** window is available from Visual Studio 2013.

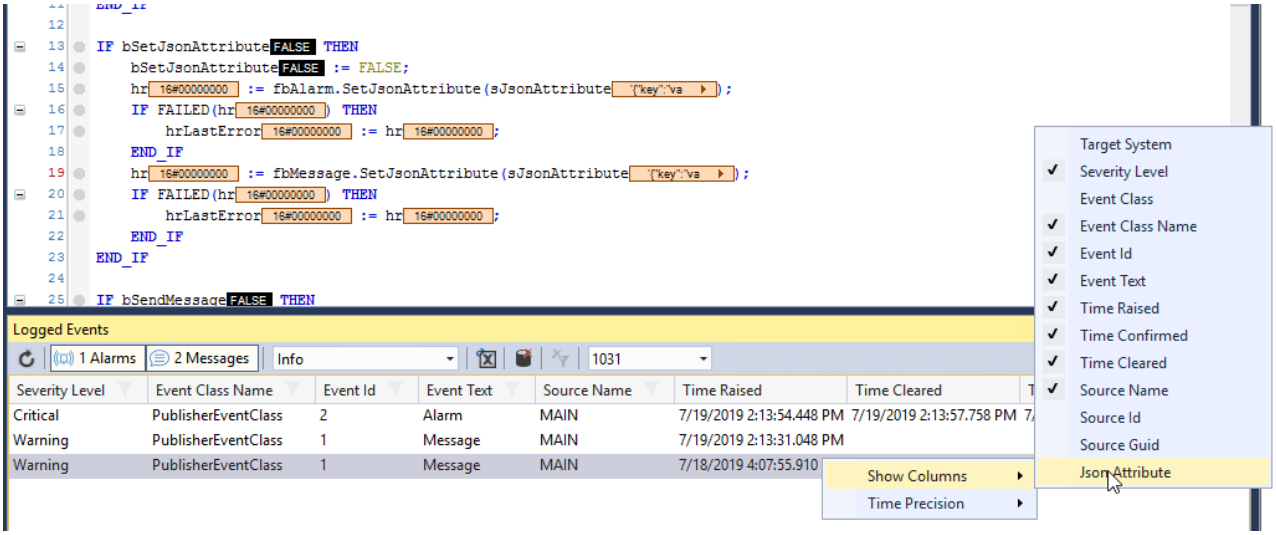
The target system events can be loaded from the cache database mentioned above and displayed via the **Logged Window**. The window is opened in the TwinCAT 3 Engineering (XAE) via **View > Other Windows > TwinCAT Logged Events.**



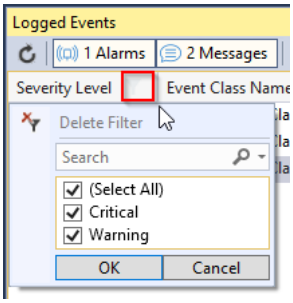
The toolbar in the window provides the following functions:

	Loads the events from the selected target system.
	You can configure whether alarms or messages should be displayed via the **Alarms** or **Messages** button respectively.
	The severity from which the events should be displayed can be selected via the drop-down menu.
	Provides an export of the data in a CVS format, where the information that is currently displayed is exported.
	Deletes (after a prompt) the cache database on the target system.
	The language can be selected or entered via the drop-down menu.

The columns in the window and the temporal resolution can be configured using the commands in the context menu:

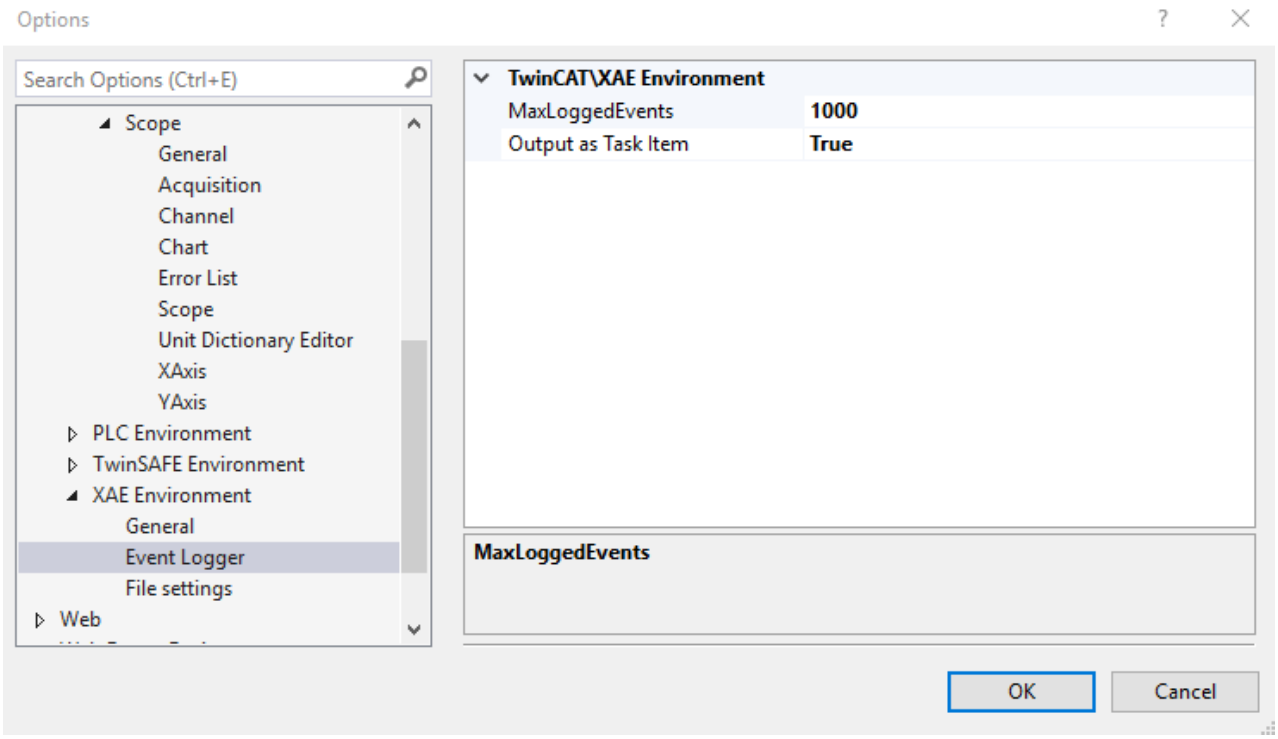


Entries can be selected via the filter function:



TwinCAT options

The TwinCAT engineering settings in the TwinCAT options (**Tools > Options**) provide basic settings for the TwinCAT 3 EventLogger.



MaxLoggedEvents	Maximum number of messages displayed in the **TwinCAT Logged Events** window.
Output as Task Item	Display of the events in the **Error List** window. The display in this window makes sense because it does not have to be loaded, but instead takes place synchronously. The display is not designed for a large number of messages, however, so this means that the option can also be used to deactivate the display if necessary.

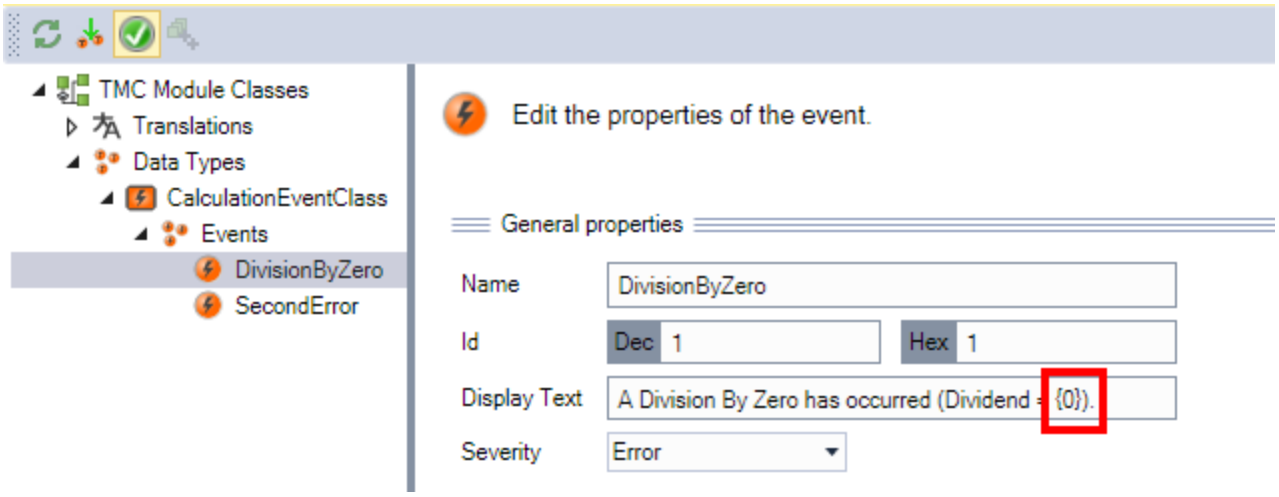
5.7 Arguments

The texts of the events can be individualized by programming with "arguments".

A marking with the notation `{n}` is used for this during the description of the events in the TMC editor, where `n` is an ascending number starting from 0.

Up to 128 arguments with a maximum size of 1024 bytes can be used at one event.

In the TMC editor, for example, such a Display Text is used for an event:



This can then be used in the source code as follows:

PLC

Arguments can be handled in the PLC as follows:

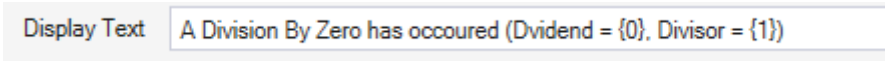
```
fbMsg : FB_TcMessage;
```

```
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
```

The arguments must thereby be defined after Create()/CreateEx(), but before Send().

Several arguments can be specified in concatenated form.




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

In this case `fDividend` is set in place of `{0}` and `fDivisor` in place of `{1}`.

C++

Arguments can be handled in the C++ as follows:

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

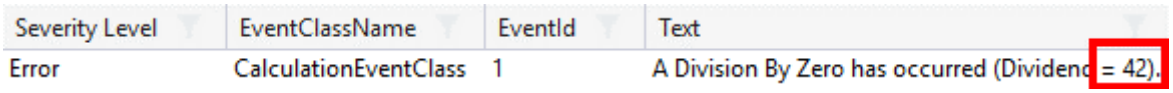
The arguments must thereby be defined after CreateMessage()/CreateAlarm(), but before Send().

For this, `TcEventLoggerTemplate.h` must be included in `<ProjectName>Interfaces.h`.

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

Output

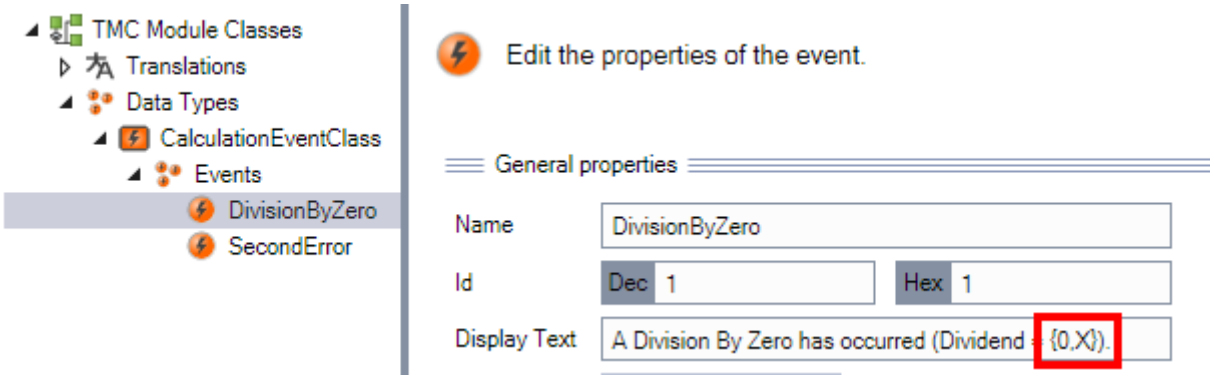
The output is accordingly:



This notation can also be used as text within the translations.

Formatting

The output of the arguments can also be formatted. To do this the syntax `{n,<Format>}` is used accordingly in the TMC editor:



The following formats are available:

Type	Format	Description
Numerical value	d / D	Decimal representation
	e / E	Exponential representation
	x / X	Hexadecimal representation
	f / F	Fixed point

Note that, for example, a REAL cannot be represented as "d" or "x" etc.

Additionally the syntax `{eventID}` (for the event ID) as well as `{eventclass}` (for the guid of the event class) are available to output the corresponding information as part of the text.

5.8 Handling sources

The same events can occur at different points in a program. The source of an event is described in the programming by the "Source Info" and sent as well when transmitting.

The SourceInfo consists of three parts (see Events [▶ 13]).

The source is also specified when creating the event in both programming languages.

PLC

The FB_TcSourceInfo is used for this in the PLC.

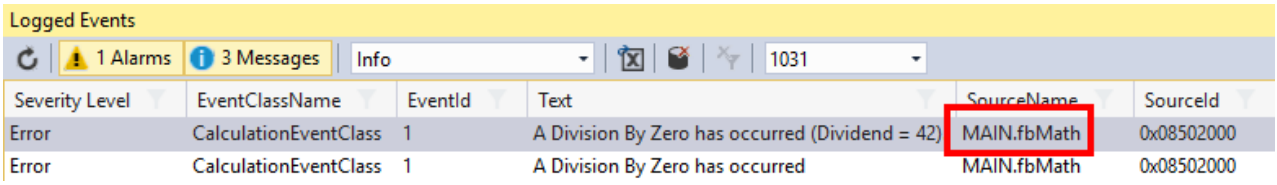
```
VAR
    fbResult : FB_TcMessage;
    fbSource : FB_TcSourceInfo; // optional
```

This is parameterized accordingly before Create()/CreateEx() is called:

```
//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.
```

Alternatively a zero can be assigned to the corresponding parameter on calling Create()/CreateEx() in order to use the internal standard source information of the PLC. If **no** explicit SourceInfo is then specified, the symbol path is output where the event is instanced as SourceName and the object ID of the PLC instance as SourceID.



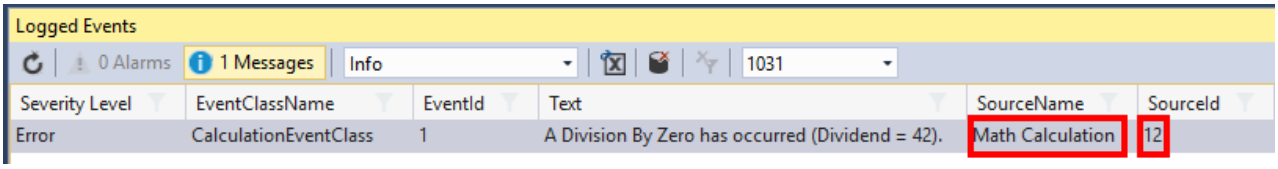
C++

TcSourceInfo is used in C++ and can be transferred, for example, in the following way with CreateMessage()/CreateAlarm():

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

Output

This SourceInfo can be shown accordingly in the LoggedEvents window:



5.9 JSON attributes

As described in the introductory part of the Technical introduction [▶ 13] section, there is a possibility to transmit an additional JSON attribute with a message.

The JsonXml library (PLC library Tc3_JsonXml) can be used to generate the JSON when creating and when receiving.

PLC

The JSON attribute can be specified before the Send() but after the Create()/CreateEx().

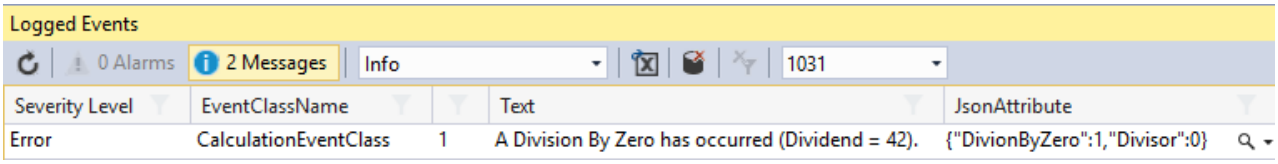
```
4   //fbSource.Clear();
5   //fbSource.sName := 'Math Calculation';
6   IF NOT fbResult.EqualsToEventEntryEx(stOther:=TC_EVENTS.CalculationEventClass.DivisionByZero) THEN
7       hr := fbResult.CreateEx(TC_EVENTS.CalculationEventClass.DivisionByZero, 0 ); //This uses dyn ressources and shouldn't be called cyclically.
8       IF FAILED(hr) THEN
9           hrLastInternalError := hr;
10      END_IF
11
12      //set Arguments if required
13      //fbResult.ipArguments.Clear();
14
15      fbResult.SetJsonAttribute('{"DivionByZero":1,"Divisor":0}');
16
17      IF fbResult.eSeverity >= eTraceLevel THEN
18          hr := fbResult.Send(0);
19          IF FAILED(hr) THEN
20              hrLastInternalError := hr;
```

C++

The JSON attribute can be specified before the Send() but after the CreateMessage()/CreateAlarm().

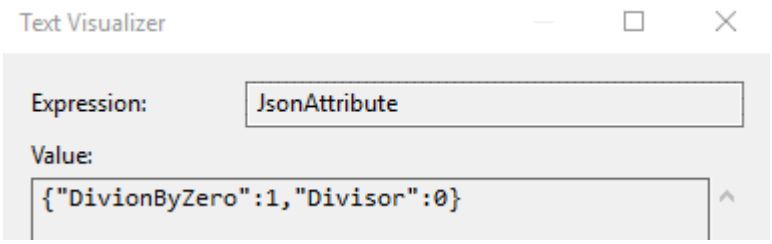
```
237
238         m_spMessage->SetJsonAttribute(m_pjsonAttribute);
239         hr = m_spMessage->Send(0);
240
```

Output



The Logged Events window provides two visualizations for the JSON attributes, which can be selected via the drop-down menu inside the information column and opened by clicking on the magnifying glass:

Text Visualizer



JSON Visualizer

5.10 Query filter

NOTICE

From TwinCAT 3.1, build 4024.17

The filters described here are available from version TwinCAT 3.1 Build 4024.17.

When processing messages such as receiving, the question arises as to which messages are to be considered at the corresponding point.

An API is provided to formulate the desired messages. For all incoming messages, the API describes which ones are relevant, resulting in a filter.

This API is available for use in various places:

- Receiving messages in real-time via the Listener interface.
- Receiving messages that occur based on EtherCAT emergency messages of the IO system.

- Deleting messages from the cache.
- Exporting messages to a file ("CSV Export“).

The practical application of the filters is illustrated in Example filter [▶ 160] and Example Listener [▶ 160].

FB_TcEventFilter [▶ 41] is the entry point in terms of usage.

5.10.1 Return values

When processing the filters, the correctness is checked in the respective application. Any errors are indicated by corresponding return values.

These are documented here:

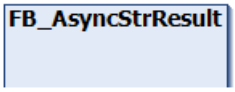
ADS_E_NOTINIT	0x98117018	ADS connection not initialized
E_POINTER	0x80004003L	Invalid pointer
E_NOTIMPL	0x80004001L	Function is not implemented.
E_OUTOFMEMORY	0x8007000EL	Insufficient memory
ADS_E_INVALIDPARM	0x9811700B	Validation error
ADS_E_NOMEMORY	0x9811700A	Insufficient memory
ADS_E_INVALIDSTATE	0x98117012	The system is in a state in which it cannot process the filters. For example, ExportLoggedEvents is not in OP state.
ADS_E_INVALIDDATA	0x98117006	AddJsonAttributeExpression path is invalid.

6 PLC API

6.1 Functions and function blocks

6.1.1 Asynchronous text requests

6.1.1.1 FB_AsyncStrResult



This function block enables the asynchronous request for a text.

Syntax

Definition:

```
FUNCTION_BLOCK FB_AsyncStrResult
```

Methods

Name	Description
GetString [▶ 32]	As soon as bBusy is FALSE and provided no error has occurred (bError = FALSE), the requested text can be fetched with this method.

Properties

Name	Type	Access	Description
bBusy	BOOL	Get	TRUE as long as the processing is not yet completed.
bError	BOOL	Get	TRUE when an error occurs.
hrErrorCode	HRESULT	Get	Outputs the error information if bError is TRUE.

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT v3.1.4022.20	PC or CX (x64, x86, ARM)	Tc3_EventLogger

6.1.1.1.1 GetString

As soon as bBusy is FALSE and provided no error has occurred (bError = FALSE), the requested text can be fetched with this method.

Syntax

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

Inputs

Name	Type	Description
sResult	REFERENCE TO STRING	Buffer variable for the requested text
nResult	UDINT	Buffer size in bytes

Return value

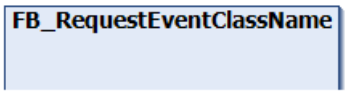
Name	Type	Description
GetString	BOOL	Returns TRUE if the text could be assigned. Returns FALSE if the text could not be completely assigned because the specified buffer variable is too small.

Example

The method may only be called if bBusy = FALSE and bError = FALSE signal that text is available.

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

6.1.1.2 FB_RequestEventClassName



This function block enables the asynchronous request for the name of an event class.

Syntax

Definition:

```
FUNCTION_BLOCK FB_RequestEventClassName
```

Methods

Name	Description
GetString [▶ 34]	As soon as bBusy is FALSE and provided no error has occurred (bError = FALSE), the requested text can be fetched with this method.
Request [▶ 34]	Calling this method triggers the asynchronous text request.

Properties

Name	Type	Access	Description
bBusy	BOOL	Get	TRUE as long as the processing is not yet completed.
bError	BOOL	Get	TRUE when an error occurs.
hrErrorCode	HRESULT	Get	Outputs the error information if bError is TRUE.

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT v3.1.4022.20	PC or CX (x64, x86, ARM)	Tc3_EventLogger

6.1.1.2.1 GetString

As soon as bBusy is FALSE and provided no error has occurred (bError = FALSE), the requested text can be fetched with this method.

Syntax

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

Inputs

Name	Type	Description
sResult	REFERENCE TO STRING	Buffer variable for the requested text
nResult	UDINT	Buffer size in bytes

Return value

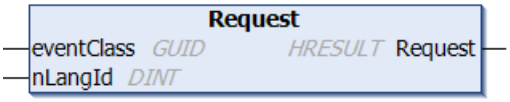
Name	Type	Description
GetString	BOOL	Returns TRUE if the text could be assigned. Returns FALSE if the text could not be completely assigned because the specified buffer variable is too small.

Example

The method may only be called if bBusy = FALSE and bError = FALSE signal that text is available.

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

6.1.1.2.2 Request



Calling this method triggers the asynchronous text request.

Syntax

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

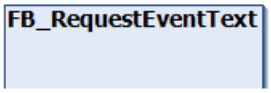
Inputs

Name	Type	Description
eventClass	GUID	GUID of the event class.
nLangId	DINT	Specifies the language ID English (en-US) = 1033 German (de-DE) = 1031

Return value

Name	Type	Description
Request	HRESULT	Returns possible error information.

6.1.1.3 FB_RequestEventText



This function block enables the asynchronous request for an event text in the desired language.

Syntax

Definition:

```
FUNCTION_BLOCK FB_RequestEventText
```

Methods

Name	Description
GetString [▶ 35]	As soon as bBusy is FALSE and provided no error has occurred (bError = FALSE), the requested text can be fetched with this method.
Request [▶ 36]	Calling this method triggers the asynchronous text request.

Properties

Name	Type	Access	Description
bBusy	BOOL	Get	TRUE as long as the processing is not yet completed.
bError	BOOL	Get	TRUE when an error occurs.
hrErrorCode	HRESULT	Get	Outputs the error information if bError is TRUE.

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT v3.1.4022.20	PC or CX (x64, x86, ARM)	Tc3_EventLogger

6.1.1.3.1 GetString

As soon as bBusy is FALSE and provided no error has occurred (bError = FALSE), the requested text can be fetched with this method.

Syntax

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

Inputs

Name	Type	Description
sResult	REFERENCE TO STRING	Buffer variable for the requested text
nResult	UDINT	Buffer size in bytes

Return value

Name	Type	Description
GetString	BOOL	Returns TRUE if the text could be assigned. Returns FALSE if the text could not be completely assigned because the specified buffer variable is too small.

Example

The method may only be called if bBusy = FALSE and bError = FALSE signal that text is available.

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

6.1.1.3.2 Request

Calling this method triggers the asynchronous text request.

Syntax

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

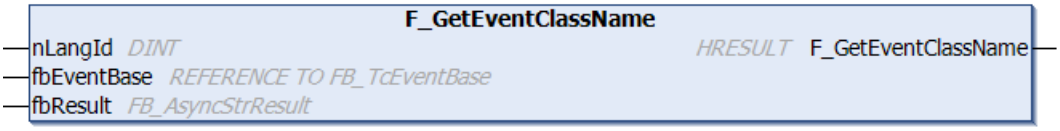
Inputs

Name	Type	Description
eventClass	GUID	Specifies the event class.
nEventId	UDINT	ID of the event.
nLangId	DINT	Specifies the language ID English (en-US) = 1033 German (de-DE) = 1031 …
ipArgs	I_TcArguments [▶ 81]	Optional specification of arguments.

Return value

Name	Type	Description
Request	HRESULT	Returns possible error information.

6.1.1.4 F_GetEventClassName



The function triggers the asynchronous request for the name of an event class.

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
```

Inputs

Name	Type	Description
nLangId	DINT	Specifies the language ID English (en-US) = 1033 German (de-DE) = 1031 …
fbEventBase	REFERENCE TO FB_TcEventBase [▶ 61]	Specification of an event/alarm/message object.

/ Inputs/outputs

Name	Type	Description
fbResult	FB_AsyncStrResult [▶ 32]	Specification of a function block instance in order to track an asynchronous text request.

Return value

Name	Type	Description
F_GetEventClassName	HRESULT	Returns possible error information.

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT v3.1.4022.20	PC or CX (x64, x86, ARM)	Tc3_EventLogger

6.1.1.5 F_GetEventText

The function triggers the asynchronous request for an event text.

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
```

Inputs

Name	Type	Description
nLangId	DINT	Specifies the language ID English (en-US) = 1033 German (de-DE) = 1031 …
fbEventBase	REFERENCE TO FB_TcEventBase [▶ 61]	Specification of an event/alarm/message object.

Inputs/outputs

Name	Type	Description
fbResult	FB_AsyncStrResult [▶ 32]	Specification of a function block instance in order to track an asynchronous text request.

Return value

Name	Type	Description
F_GetEventText	HRESULT	Returns possible error information.

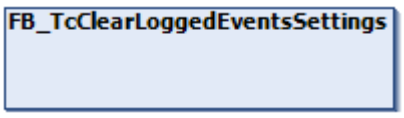
Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT v3.1.4022.20	PC or CX (x64, x86, ARM)	Tc3_EventLogger

6.1.2 Filter

The filter functionality is used in different places. A sample that describes the possible uses is covered in the Example filter [▶ 160].

6.1.2.1 FB_TcClearLoggedEventsSettings



Provides the functionality to specify which events should be removed from the cache.

Syntax

Definition:

```
FUNCTION_BLOCK FB_TcClearLoggedEventsSettings IMPLEMENTS I_TcClearLoggedEventsSettings
```

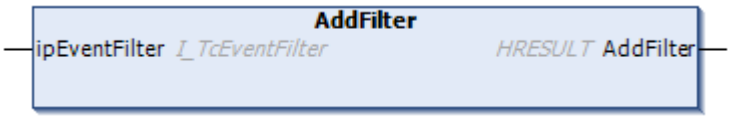
Methods

Name	Definition location	Description
AddFilter	Local	Method for adding a filter. Returns S_OK if successful.
Clear	Local	Method for clearing the settings. Returns S_OK if successful.
SetLimit	Local	Indicates the number of events to be cleared. The limit is applied after sorting and filtering. Returns S_OK if successful.
SetSorting	Local	Sets the sort order for the query. Returns S_OK if successful.

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT v3.1.4024.17	PC or CX (x64, x86, ARM)	Tc3_EventLogger (>= v3.1.27.0)

6.1.2.1.1 AddFilter



Method for adding a filter.

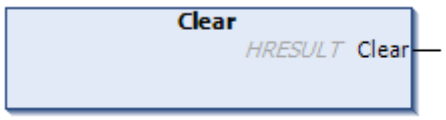
Inputs

Name	Type	Description
ipEventFilter	I_TcEventFilter	Instance of the filter to be used

Return values

Name	Type	Description
AddFilter	HRESULT	Returns S_OK if successful.

6.1.2.1.2 Clear



Method for clearing the settings.

Return values

Name	Type	Description
Clear	HRESULT	Returns S_OK if successful.

6.1.2.1.3 SetLimit

Indicates the number of events to be cleared. The limit is applied after sorting and filtering.

Inputs

Name	Type	Description
eType	TcEventLimitType	Determines the reference whether the limit applies to the first or last events.
nLimit	DINT	Specifies the number (-1 = no limit)
nOffset	DINT	Optional. Defines how many entries are to be skipped.

Return values

Name	Type	Description
SetLimit	HRESULT	Returns `S_OK` if successful.

6.1.2.1.4 SetSorting

Sets the sort order for the query.

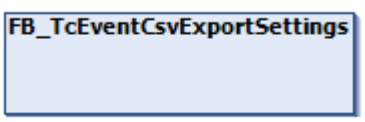
Inputs

Name	Type	Description
eField	TcEventField	Property to be used for sorting.
eOrder	TcEventSortOrder	Defines the sort order.

Return values

Name	Type	Description
SetSorting	HRESULT	Returns `S_OK` if successful.

6.1.2.2 FB_TcEventCsvExportSettings



Provides the functionality to specify the csv export.

Syntax

Definition:

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

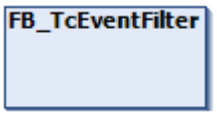
Methods

Name	Type	Description
bWithHeader	BOOL	Determines whether a header should be created. Standard: True
nLangId	DINT	Determines the default identifier of the export language. Standard: 1033
sDelimiter	STRING	Defines the CSV delimiter. Standard: Semicolon [;]

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT v3.1.4024.17	PC or CX (x64, x86, ARM)	Tc3_EventLogger (>= v3.1.27.0)

6.1.2.3 FB_TcEventFilter



Provides the functionality to specify an event filter.

The filters are provided via a floating interface following a structured query language. It describes which messages should apply.

- Conditions can be linked through `.AND_OP()` and `.OR_OP()`.
- Conditions can be negated through `.NOT_OP`.
- Conditions can be defined through properties such as `.isAlarm()` or `.EventClass.EqualsTo(<EventClass>)`, for example. A complete list of properties can be found in the API documentation.
- A grouping can be formulated through `.FilterExpression(<SubCondition>)`. The `<SubCondition>` itself is a `FB_TcEventFilter` or `ITcEventFilter`.

A filter is applied once it has been compiled. To receive messages it is assigned to a recipient via `FB_ListenerBase2.subscribe()`, for example. In this way `FB_ListenerBase2` takes over the filter and provides a corresponding return value, which is described here. The filter can be amended by repeating `FB_ListenerBase2.subscribe()`.

Sample

A filter can be assembled in the following way, for example:

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

The Example filter [▶ 160] illustrates the usage.

EtherCAT filter

The mechanism for receiving EtherCAT emergency messages is similar to that described above. The entry point in the chained method calls is `.EtherCATDevice()`, which first provides a direct query to ascertain if it was sent from an EtherCAT device (`IsEtherCATDevice()`). From here you can filter based on the manufacturer (`.VendorId()`), the product code (`.ProductCode()`) or the revision (`.RevisionNo()`).

Syntax

Definition:

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

Methods

Name	Definition location	Description
Clear	I_TcEventFilter	Clears the previous filter expression.
FilterExpression	I_TcExpressionBase	Specification of a subordinate filter definition.
IsAlarm	I_TcExpressionBase	Checking whether it is an alarm.
IsMessage	I_TcExpressionBase	Checking whether it is a message.
NOT_OP	I_TcExpressionBase	Negation of the subsequent statement.

Properties

Name	Type	Access	Description
AlarmState	I_TcAlarmFilterExpression	Get	Checking with an AlarmState
EtherCATDevice	I_TcEtherCATDeviceExpression	Get	Checking whether the source is an EtherCAT device.
EventClass	I_TcGuidCompare	Get	Checking with an EventClass
EventId	I_TcUDIntCompare	Get	Checking with an EventId
JsonAttribute	I_TcJsonAttributeExpression	Get	Checking with the JsonAttribute
Severity	I_TcSeverityCompare	Get	Checking with Serverity
Source	I_TcSourceInfoExpression	Get	Checking with the source
TimeCleared	I_TcULIntCompare	Get	Checking of the clear time (only in the event of an alarm)
TimeConfirmed	I_TcULIntCompare	Get	Checking of the confirm time (only for alarm with acknowledgement)
TimeRaised	I_TcULIntCompare	Get	Checking of the sender (for messages) or the raised time (for alarms)

Requirements

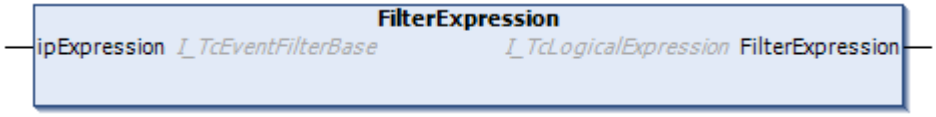
Development environment	Target platform	PLC libraries to include
TwinCAT v3.1.4024.17	PC or CX (x64, x86, ARM)	Tc3_EventLogger (>= v3.1.27.0)

6.1.2.3.1 Clear

Return values

Name	Type	Description
Clear	I_TcBinaryExpression	

6.1.2.3.2 FilterExpression



Inputs

Name	Type	Description
ipExpression	I_TcEventFilterBase	

Return values

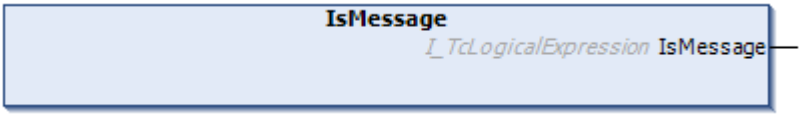
Name	Type	Description
FilterExpression	I_TcLogicalExpression	

6.1.2.3.3 IsAlarm

Return values

Name	Type	Description
IsAlarm	I_TcLogicalExpression	

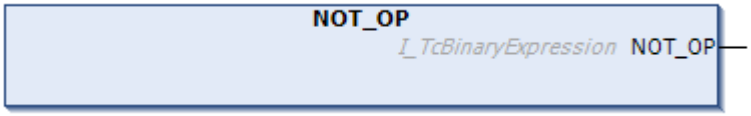
6.1.2.3.4 IsMessage



Return values

Name	Type	Description
IsMessage	I_TcLogicalExpression	

6.1.2.3.5 NOT_OP



Return values

Name	Type	Description
NOT_OP	I_TcBinaryExpression	

6.1.3 EventEntry conversion

6.1.3.1 AdsErr_TO_TcEventEntry

This function converts a standard ADS error into a TcEventEntry.

Syntax

Definition:

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

Inputs

Name	Type	Description
eErrorId	E_AdsErr	Error code to be converted.
stEventEntry	REFERENCE TO TcEventEntry [▶ 97]	Outputs the resulting event definition.

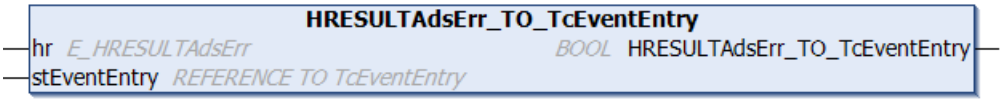
Return value

Name	Type	Description
AdsErr_TO_TcEventEntry	BOOL	Returns TRUE if the conversion was carried out successfully.

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT v3.1.4022.20	PC or CX (x64, x86, ARM)	Tc3_EventLogger

6.1.3.2 HRESULTAdsErr_TO_TcEventEntry



This function converts a standard ADS error (HRESULT) into a TcEventEntry.

Syntax

Definition:

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

Inputs

Name	Type	Description
hr	E_HRESULTAdsErr	Error code to be converted.
stEventEntry	REFERENCE TO TcEventEntry [▶ 97]	Outputs the resulting event definition.

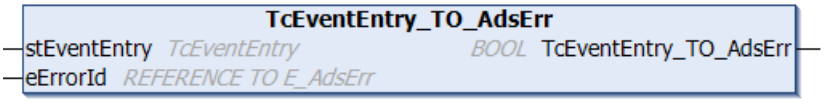
Return value

Name	Type	Description
HRESULTAdsErr_TO_TcEventEntry	BOOL	Returns TRUE if the conversion was carried out successfully. The call fails if the facility code in the specified HRESULT is unknown.

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT v3.1.4022.20	PC or CX (x64, x86, ARM)	Tc3_EventLogger

6.1.3.3 TcEventEntry_TO_AdsErr



This function converts a TcEventEntry into a standard ADS error.

Syntax

Definition:

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

Inputs

Name	Type	Description
stEventEntry	TcEventEntry [▶ 97]	Event definition to be converted.
eErrorId	REFERENCE TO E_AdsErr	Outputs the resulting error code.

Return value

Name	Type	Description
TcEventEntry_TO_AdsErr	BOOL	Returns TRUE if the conversion was carried out successfully and FALSE if the event class is unknown.

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT v3.1.4022.20	PC or CX (x64, x86, ARM)	Tc3_EventLogger

6.1.3.4 TcEventEntry_TO_HRESULTAdsErr

This function converts a TcEventEntry into a standard ADS error (HRESULT).

Syntax

Definition:

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

Inputs

Name	Type	Description
stEventEntry	TcEventEntry [▶ 97]	Event definition to be converted.
hr	REFERENCE TO E_HRESULTAdsErr	Outputs the resulting error code.

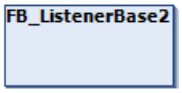
Return value

Name	Type	Description
TcEventEntry_TO_HRESULTAdsErr	BOOL	Returns TRUE if the conversion was carried out successfully and FALSE if the event class is unknown.

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT v3.1.4022.20	PC or CX (x64, x86, ARM)	Tc3_EventLogger

6.1.4 FB_ListenerBase2



The function block serves as the basic implementation of an event listener.

New messages and state changes of alarms can be recognized through the overwriting of the event-driven methods.

Syntax

Definition:

```
FUNCTION_BLOCK FB_ListenerBase2 IMPLEMENTS I_Listener2
```

Methods

Name	Definition location	Description
Execute [▶ 47]	Local	Must be called cyclically so that the event queue can be processed.
Subscribe [▶ 49]	Local	Subscribes messages.
Unsubscribe [▶ 50]	Local	Unsubscribes messages.

Event-driven methods (callback methods)

Name	Definition location	Description
OnAlarmCleared [▶ 47]	I_Listener2	Called when the state of an alarm changes from "Raised" to "Clear".
OnAlarmConfirmed [▶ 48]	I_Listener2	Called when an alarm has been confirmed.
OnAlarmDisposed [▶ 48]	I_Listener2	Called when an alarm instance has been released again.
OnAlarmRaised [▶ 49]	I_Listener2	Called when the state of an alarm changes from "Clear" to "Raised".
OnMessageSent [▶ 49]	I_Listener2	Called when a message has been sent.

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT v3.1.4024.17	PC or CX (x64, x86, ARM)	Tc3_EventLogger (>= v3.1.27.0)

6.1.4.1 Execute

This method must be called cyclically so that the event queue can be processed.

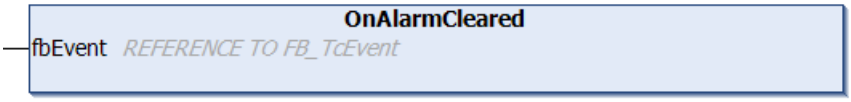
Syntax

```
METHOD Execute : HRESULT
```

Return value

Name	Type	Description
Execute	HRESULT	Returns S_OK if the method call was successful, otherwise an HRESULT as the error code.

6.1.4.2 OnAlarmCleared



This method is called if the state of an alarm changes from Raised to Clear.

Syntax

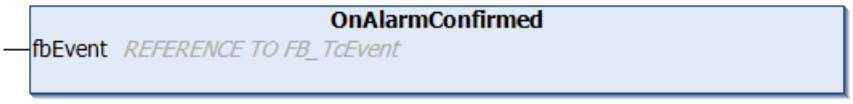
```
METHOD OnAlarmCleared : HRESULT
VAR_INPUT
    fbEvent : REFERENCE TO FB_TcEvent;
END_VAR
```

If the implementation of the callback method returns a return code <> S_OK, further callback calls will be paused until the next execution.

Inputs

Name	Type	Description
fbEvent	REFERENCE TO FB_TcEvent [▶ 59]	Reference to the alarm that has occurred. This reference must not be copied, e.g. through assignment.

6.1.4.3 OnAlarmConfirmed



This method is called when an alarm has been confirmed.

Syntax

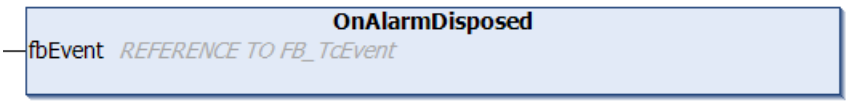
```
METHOD OnAlarmConfirmed : HRESULT
VAR_INPUT
    fbEvent : REFERENCE TO FB_TcEvent;
END_VAR
```

If the implementation of the callback method returns a return code <> S_OK, further callback calls will be paused until the next execution.

Inputs

Name	Type	Description
fbEvent	REFERENCE TO FB_TcEvent [▶ 59]	Reference to the alarm that has occurred. This reference must not be copied, e.g. through assignment.

6.1.4.4 OnAlarmDisposed



This method is called when an alarm instance has been released again.

Syntax

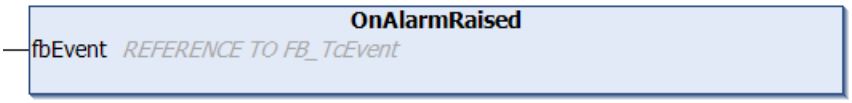
```
METHOD OnAlarmConfirmed : HRESULT
VAR_INPUT
    fbEvent : REFERENCE TO FB_TcEvent;
END_VAR
```

If the implementation of the callback method returns a return code <> S_OK, further callback calls will be paused until the next execution.

Inputs

Name	Type	Description
fbEvent	REFERENCE TO FB_TcEvent [▶ 59]	Reference to the alarm that has occurred. This reference must not be copied, e.g. through assignment.

6.1.4.5 OnAlarmRaised



This method is called if the state of an alarm changes from Clear to Raised.

Syntax

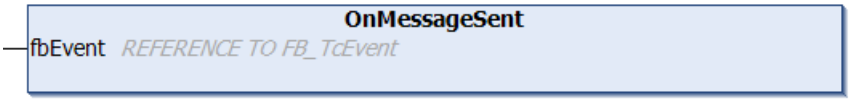
```
METHOD OnAlarmRaised : HRESULT
VAR_INPUT
    fbEvent : REFERENCE TO FB_TcEvent;
END_VAR
```

If the implementation of the callback method returns a return code <> S_OK, further callback calls will be paused until the next execution.

Inputs

Name	Type	Description
fbEvent	REFERENCE TO FB_TcEvent [▶ 59]	Reference to the alarm that has occurred. This reference must not be copied, e.g. through assignment.

6.1.4.6 OnMessageSent



This method is called when a message has been sent.

Syntax

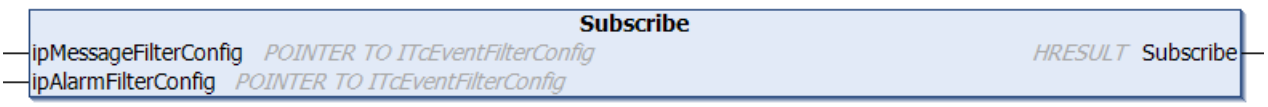
```
METHOD OnMessageSent : HRESULT
VAR_INPUT
    fbEvent : REFERENCE TO FB_TcEvent;
END_VAR
```

If the implementation of the callback method returns a return code <> S_OK, further callback calls will be paused until the next execution.

Inputs

Name	Type	Description
fbEvent	REFERENCE TO FB_TcEvent [▶ 59]	Reference to the event that has occurred. This reference must not be copied, e.g. through assignment.

6.1.4.7 Subscribe



The listener is subscribed for messages with this method.

Syntax

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

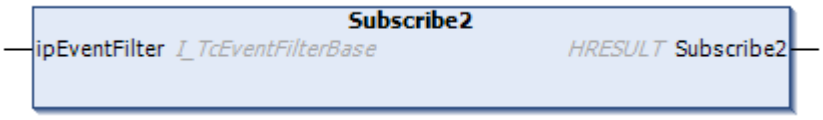
Inputs

Name	Type	Description
ipMessageFilterConfig	POINTER TO ITcEventFilterConfig	Pointer to ITcEventFilterConfig if a filter is to be activated.
ipAlarmFilterConfig	POINTER TO ITcEventFilterConfig	Pointer to ITcEventFilterConfig if a filter is to be activated.

Return value

Name	Type	Description
Subscribe	HRESULT	Returns S_OK if the method call was successful. Returns ADS_E_EXISTS if the listener is already subscribed. Otherwise returns HRESULT as the error code.

6.1.4.8 Subscribe2



Syntax

```
METHOD Subscribe2 : HRESULT
```

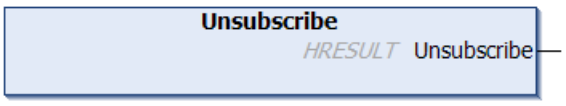
Input

Name	Type	Description
ipEventFilter	I_TcEventFilterBase	Pointer to an instance of FB_TcEventFilter [▶ 41], if a filter is to be activated.

Return value

Name	Type	Description
Subscribe2	HRESULT	Delivers if successful S_OK.

6.1.4.9 Unsubscribe



The listener is unsubcribed with this method.

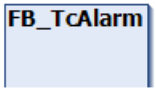
Syntax

```
METHOD Unsubscribe : HRESULT
```

Return value

Name	Type	Description
Unsubscribe	HRESULT	Returns S_OK if the method call was successful. Returns ADS_E_NOTFOUND if the listener was not subscribed. Otherwise returns HRESULT as the error code.

6.1.5 FB_TcAlarm



This function block represents an alarm of the TwinCAT 3 EventLogger.

Syntax

Definition:

```
FUNCTION_BLOCK FB_TcAlarm EXTENDS FB_TcEventBase
```

Inheritance hierarchy

FB_TcEventBase [▶ 61]

FB_TcAlarm

Methods

Name	Definition location	Description
EqualsTo [▶ 62]	Inherited from FB_TcEventBase [▶ 61]	Compares the event with another instance.
EqualsToEventClass [▶ 63]	Inherited from FB_TcEventBase [▶ 61]	Compares the event class of the event with another event class.
EqualsToEventEntry [▶ 63]	Inherited from FB_TcEventBase [▶ 61]	Compares the event class, the event ID and the severity of the event with those of another event.
EqualsToEventEntryEx [▶ 64]	Inherited from FB_TcEventBase [▶ 61]	Compares the event definition of the event with another event definition.
GetJsonAttribute [▶ 64]	Inherited from FB_TcEventBase [▶ 61]	Returns the Json attribute.
Release [▶ 65]	Inherited from FB_TcEventBase [▶ 61]	Releases the instance created by the EventLogger again.
RequestEventClassName [▶ 65]	Inherited from FB_TcEventBase [▶ 61]	Requests the name of the event class.
RequestEventText [▶ 66]	Inherited from FB_TcEventBase [▶ 61]	Returns the text for the event.
Clear [▶ 53]	Local	Sets the alarm state to "Not Raised".
Confirm [▶ 54]	Local	Confirms the alarm.
Create [▶ 55]	Local	Creates an alarm instance in the EventLogger.
CreateEx [▶ 55]	Local	Creates an alarm instance in the EventLogger from an event definition.
Raise [▶ 56]	Local	Sets the alarm state to "Raised".
SetJsonAttribute [▶ 57]	Local	Sets the Json attribute.

Properties

Name	Type	Access	Definition location	Description
eSeverity	TcEventSeverity [▶ 98]	Get	Inherited from FB_TcEventBase [▶ 61]	Returns the severity.
EventClass	GUID	Get	Inherited from FB_TcEventBase [▶ 61]	Returns the GUID of the event class.
ipArguments [▶ 67]	I_TcArguments [▶ 81]	Get	Inherited from FB_TcEventBase [▶ 61]	Returns the interface pointer for the arguments.
ipSourceInfo [▶ 67]	I_TcSourceInfo [▶ 97]	Get	Inherited from FB_TcEventBase [▶ 61]	The SourceInfo is created internally as the default behavior. It then contains the symbol name of the function block that instances FB_TcMessage as SourceName and the object ID of the PLC instance as SourceID. If the instance of FB_TcMessage is hidden with the attribute "hide", no symbol name can be created internally for the default behavior.
nEventId	nEventId	Get	Inherited from FB_TcEventBase [▶ 61]	Returns the ID of the event.
stEventEntry	TcEventEntry [▶ 97]	Get	Inherited from FB_TcEventBase [▶ 61]	Returns the event definition.
bRaised	BOOL	Get	Local	Returns TRUE if the alarm is in the "raised" state.
bActive	BOOL	Get	Local	Returns TRUE if the alarm is in the "Raised" or "Wait For Confirmation" state.
eConfirmationState	TcEventConfirmationState [▶ 98]	Get	Local	Returns the confirmation state.
nTimeCleared	ULINT	Get	Local	Returns the time of the Clear.
nTimeConfirmed	ULINT	Get	Local	Returns the time of the Confirm.
nTimeRaised	ULINT	Get	Local	Returns the time of the Raise.

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT v3.1.4022.20	PC or CX (x64, x86, ARM)	Tc3_EventLogger

6.1.5.1 Clear

This method sets the alarm state [▶ 14] to Not Raised.

Syntax

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

Inputs

Name	Type	Description
nTimeStamp	ULINT	0: Current time stamp is used > 0: External time stamp in 100 nanoseconds since January 1st, 1601 (UTC).
bResetConfirmation	BOOL	If TRUE and the confirmation state is WaitForConfirmation, the confirmation state is set to Reset. Otherwise the confirmation state is not changed.

Return value

Name	Type	Description
Clear	HRESULT	Returns S_OK if the method call was successful. Returns ADS_E_INVALIDSTATE if the alarm was not in the Raised state. Otherwise returns HRESULT as the error code.

6.1.5.2 Confirm

Sets the confirmation state [▶ 14] of WaitingForConfirmation to Confirmed.

Syntax

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

Inputs

Name	Type	Description
nTimeStamp	ULINT	0: Current time stamp is used > 0: External time stamp in 100 nanoseconds since January 1st, 1601 (UTC).

Return value

Name	Type	Description
Confirm	HRESULT	Returns S_OK if the method call was successful. Returns ADS_E_INVALIDSTATE if the confirmation state was not WaitConfirmation. Otherwise returns HRESULT as the error code.

6.1.5.3 Create

This method creates an alarm instance in the EventLogger.

Syntax

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

Inputs

Name	Type	Description
eventClass	GUID	GUID of the event class.
nEventId	UDINT	ID of the event.
eSeverity	TcEventSeverity [▶ 98]	Severity of the event.
bWithConfirmation	BOOL	Defines whether the alarm requires mandatory confirmation.
ipSourceInfo	I_TcSourceInfo [▶ 97]	Interface pointer to the source information. Default source information is created if no interface pointer is transferred.

Return value

Name	Type	Description
Create	HRESULT	Returns S_OK if a new alarm was successfully created. Returns ERROR_ALREADY_EXISTS if the alarm has already existed. Otherwise returns HRESULT as the error code

6.1.5.4 CreateEx

This method creates an alarm instance in the EventLogger.

Syntax

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

Inputs

Name	Type	Description
stEventEntry	TcEventEntry [▶ 97]	Event definition.
bWithConfirmation	BOOL	Defines whether the alarm requires mandatory confirmation.
ipSourceInfo	I_TcSourceInfo [▶ 97]	Interface pointer to the source information. Default source information is created if no interface pointer is transferred.

Return value

Name	Type	Description
CreateEx	HRESULT	Returns S_OK if a new alarm was successfully created. Returns ERROR_ALREADY_EXISTS if the alarm has already existed. Otherwise returns HRESULT as the error code.

6.1.5.5 Raise

Sets the alarm state [▶ 14] to Raised.

If the alarm requires mandatory confirmation, the confirmation state is additionally set to WaitForConfirmation.

Syntax

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

Inputs

Name	Type	Description
nTimeStamp	ULINT	0: Current time stamp is used > 0: External time stamp in 100 nanoseconds since January 1st, 1601 (UTC).

Return value

Name	Type	Description
Raise	HRESULT	Returns S_OK if the method call was successful. Returns ADS_E_INVALIDSTATE if the alarm was already in the Raised state. Otherwise returns HRESULT as the error code

6.1.5.6 SetJsonAttribute

This method sets the JSON attribute.

Syntax

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

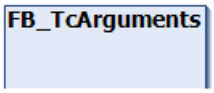
Inputs

Name	Type	Description
sJsonAttribute	STRING	JSON string

Return value

Name	Type	Description
SetJsonAttribute	HRESULT	Returns S_OK if the method call was successful, otherwise an HRESULT as the error code.

6.1.6 FB_TcArguments



Arguments of an event can be defined with this function block. The ITcArguments interface is implemented for this.

Syntax

Definition:

```
FUNCTION_BLOCK FB_TcArguments IMPLEMENTS I_TcArguments
```

Interfaces

Type	Description
I_TcArguments [▶ 81]	Defines the argument handling.

Methods

Name	Definition location	Description
AddBlob [▶ 82]	I_TcArguments [▶ 81]	Adds binary data as an argument.
AddBool [▶ 83]	I_TcArguments [▶ 81]	Adds an argument of the type BOOL.
AddByte [▶ 83]	I_TcArguments [▶ 81]	Adds an argument of the type BYTE.
AddDint [▶ 84]	I_TcArguments [▶ 81]	Adds an argument of the type DINT.
AddDWord [▶ 84]	I_TcArguments [▶ 81]	Adds an argument of the type DWORD.
AddEventReferenceId [▶ 84]	I_TcArguments [▶ 81]	Adds a reference to another event as an argument.
AddEventReferenceIdGuid [▶ 85]	I_TcArguments [▶ 81]	Adds a reference to another event as an argument.
AddInt [▶ 85]	I_TcArguments [▶ 81]	Adds an argument of the type INT.
AddLInt [▶ 86]	I_TcArguments [▶ 81]	Adds an argument of the type LINT.
AddLReal [▶ 86]	I_TcArguments [▶ 81]	Adds an argument of the type LREAL.
AddReal [▶ 87]	I_TcArguments [▶ 81]	Adds an argument of the type REAL.
AddSInt [▶ 87]	I_TcArguments [▶ 81]	Adds an argument of the type SINT.
AddString [▶ 88]	I_TcArguments [▶ 81]	Adds an argument of the type STRING.
AddUDint [▶ 88]	I_TcArguments [▶ 81]	Adds an argument of the type UDINT.
AddUInt [▶ 88]	I_TcArguments [▶ 81]	Adds an argument of the type INT.
AddULInt [▶ 89]	I_TcArguments [▶ 81]	Adds an argument of the type ULINT.
AddUSInt [▶ 89]	I_TcArguments [▶ 81]	Adds an argument of the type USINT.
AddWord [▶ 90]	I_TcArguments [▶ 81]	Adds an argument of the type WORD.
AddWString [▶ 90]	I_TcArguments [▶ 81]	Adds an argument of the type WSTRING.
Clear [▶ 91]	I_TcArguments [▶ 81]	Removes all arguments.
IsEmpty [▶ 59]	Local	Checks whether arguments have been added.

Properties

Name	Type	Access	Description
nCount	UDINT	Get	Returns the number of transferred arguments.

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT v3.1.4022.20	PC or CX (x64, x86, ARM)	Tc3_EventLogger

6.1.6.1 IsEmpty

This method checks whether arguments have been added.

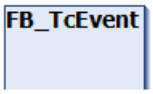
Syntax

```
METHOD IsEmpty : BOOL
```

Return value

Name	Type	Description
IsEmpty	BOOL	Returns TRUE if no arguments have been added.

6.1.7 FB_TcEvent



This function block provides only read methods and read properties for an event.

Syntax

Definition:

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

Inheritance hierarchy

FB_TcEventBase [▶ 61]

FB_TcEvent

Interfaces

Type	Description
I_TcEventBase [▶ 91]	Basic interface that defines methods and properties of an event.

Methods

Name	Definition location	Description
EqualsTo [▶ 62]	Inherited from FB_TcEventBase [▶ 61]	Compares the event with another instance.
EqualsToEventClass [▶ 63]	Inherited from FB_TcEventBase [▶ 61]	Compares the event class of the event with another event class.
EqualsToEventEntry [▶ 63]	Inherited from FB_TcEventBase [▶ 61]	Compares the event definition of the event with another event definition.
EqualsToEventEntryEx [▶ 64]	Inherited from FB_TcEventBase [▶ 61]	Compares the event definition of the event with another event definition.
GetJsonAttribute [▶ 64]	Inherited from FB_TcEventBase [▶ 61]	Returns the Json attribute.
Release [▶ 65]	Inherited from FB_TcEventBase [▶ 61]	Releases the instance created by the EventLogger again.
RequestEventClassName [▶ 65]	Inherited from FB_TcEventBase [▶ 61]	Requests the name of the event class.
RequestEventText [▶ 66]	Inherited from FB_TcEventBase [▶ 61]	Returns the text for the event.

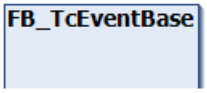
Properties

Name	Type	Access	Definition location	Description
eSeverity	TcEventSeverity [▶ 98]	Get	Inherited from FB_TcEventBase [▶ 61]	Returns the severity.
EventClass	GUID	Get	Inherited from FB_TcEventBase [▶ 61]	Returns the GUID of the event class.
ipArguments [▶ 67]	I_TcArguments [▶ 81]	Get	Inherited from FB_TcEventBase [▶ 61]	Returns the interface pointer for the arguments.
ipSourceInfo [▶ 67]	I_TcSourceInfo [▶ 97]	Get	Inherited from FB_TcEventBase [▶ 61]	The SourceInfo is created internally as the default behavior. It then contains the symbol name of the function block that instances FB_TcMessage as SourceName and the object ID of the PLC instance as SourceID. If the instance of FB_TcMessage is hidden with the attribute "hide", no symbol name can be created internally for the default behavior.
nEventId	nEventId	Get	Inherited from FB_TcEventBase [▶ 61]	Returns the ID of the event.
stEventEntry	TcEventEntry [▶ 97]	Get	Inherited from FB_TcEventBase [▶ 61]	Returns the event definition.
nTimestamp	ULINT	Get	Local	Returns the time.

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT v3.1.4022.20	PC or CX (x64, x86, ARM)	Tc3_EventLogger

6.1.8 FB_TcEventBase



This function block contains the basic implementation.

Syntax

Definition:

```
FUNCTION_BLOCK FB_TcEventBase
```

Methods

Name	Definition location	Description
EqualsTo [▶ 62]	Local	Compares the event with another instance.
EqualsToEventClass [▶ 63]	Local	Compares the event class of the event with another event class.
EqualsToEventEntry [▶ 63]	Local	Compares the event definition of the event with another event definition.
EqualsToEventEntryEx [▶ 64]	Local	Compares the event definition of the event with another event definition.
GetJsonAttribute [▶ 64]	Local	Returns the Json attribute.
Release [▶ 65]	Local	Releases the instance created by the EventLogger again.
RequestEventClassName [▶ 65]	Local	Requests the name of the event class.
RequestEventText [▶ 66]	Local	Returns the text for the event.

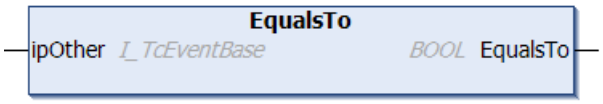
Properties

Name	Type	Access	Description
eSeverity	TcEventSeverity [▶ 98]	Get	Returns the severity.
EventClass	GUID	Get	Returns the GUID of the event class.
ipArguments [▶ 67]	I_TcArguments [▶ 81]	Get	Returns the interface pointer for the arguments.
ipSourceInfo [▶ 67]	I_TcSourceInfo [▶ 97]	Get	The SourceInfo is created internally as the default behavior. It then contains the symbol name of the function block that instances FB_TcMessage as SourceName and the object ID of the PLC instance as SourceID. If the instance of FB_TcMessage is hidden with the attribute "hide", no symbol name can be created internally for the default behavior.
nEventId	UDINT	Get	Returns the ID of the event.
nUniqueId	UDINT	Get	Returns the unique ID of the event.
stEventEntry	TcEventEntry [▶ 97]	Get	Returns the event definition.

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT v3.1.4022.20	PC or CX (x64, x86, ARM)	Tc3_EventLogger

6.1.8.1 EqualsTo



This method carries out a comparison with another event specified at the input.

Syntax

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

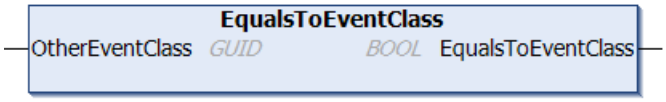
Inputs

Name	Type	Description
ipOther	I_TcEventBase [▶ 91]	Event to be compared

Return value

Name	Type	Description
EqualsTo	BOOL	Returns TRUE if the events match.

6.1.8.2 EqualsToEventClass



This method carries out a comparison with another event class specified at the input.

Syntax

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

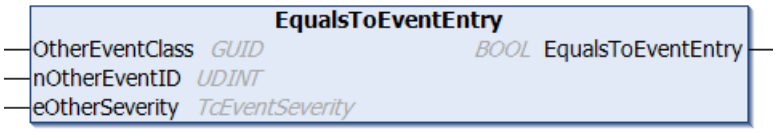
Inputs

Name	Type	Description
OtherEventClass	GUID	Event class to be compared.

Return value

Name	Type	Description
EqualsToEventClass	BOOL	Returns TRUE if the event classes match.

6.1.8.3 EqualsToEventEntry



This method carries out a comparison with another event specified at the input.

Syntax

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

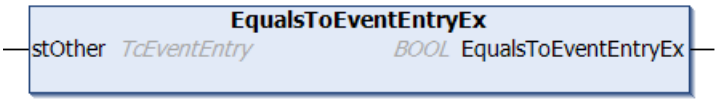
Inputs

Name	Type	Description
OtherEventClass	GUID	Event class of the event to be compared.
nOtherEventID	UDINT	Event ID of the event to be compared.
eOtherSeverity	TcEventSeverity [▶ 98]	Event severity of the event to be compared.

Return value

Name	Type	Description
EqualsToEventEntry	BOOL	Returns TRUE if the events match.

6.1.8.4 EqualsToEventEntryEx



This method carries out a comparison with another event specified at the input.

Syntax

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

Inputs

Name	Type	Description
stOther	TcEventEntry [▶ 97]	Event to be compared.

Return value

Name	Type	Description
EqualsToEventEntryEx	BOOL	Returns TRUE if the events match.

6.1.8.5 GetJsonAttribute

This method returns the JSON attribute.

Syntax

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

Inputs

Name	Type	Description
sJsonAttribute	REFERENCE TO STRING	Reference to a variable of the type String
nJsonAttribute	UDINT	Length of the String variable

Return value

Name	Type	Description
GetJsonAttribute	HRESULT	Returns S_OK if the method call was successful. Returns ERROR_BAD_LENGTH if the length of the variable is too small. Otherwise HRESULT is returned as the error code.

6.1.8.6 Release

This method releases the instance created by the EventLogger again.

Syntax

```
METHOD Release : HRESULT
```

Return value

Name	Type	Description
Release	HRESULT	Returns S_OK if the method call was successful, otherwise an HRESULT as the error code.

6.1.8.7 RequestEventClassName

This method returns the name of the event class.

Syntax

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

Inputs

Name	Type	Description
nLangId	DINT	Specifies the language ID English (en-US) = 1033 German (de-DE) = 1031 …
sResult	REFERENCE TO STRING	Reference to a variable of the type String
nResultSize	UDINT	Size of the String variable in bytes

Return value

Name	Type	Description
RequestEventClassName	BOOL	Returns TRUE as soon as the request has been terminated. Returns FALSE if the asynchronous request is still active. The method must be called until the return value is TRUE.

Outputs

Name	Type	Description
bError	BOOL	Returns FALSE if the method call was successful. Returns TRUE if an error has occurred.
hrErrorCode	HRESULT	Returns S_OK if the method call was successful. An error code is output in case of an error.

6.1.8.8 RequestEventText

This method returns the event text.

Syntax

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

Inputs

Name	Type	Description
nLangId	DINT	Specifies the language ID English (en-US) = 1033 German (de-DE) = 1031 …
sResult	REFERENCE TO STRING	Reference to a variable of the type String
nResultSize	UDINT	Size of the String variable in bytes

Return value

Name	Type	Description
RequestEventText	BOOL	Returns TRUE as soon as the request has been terminated. Returns FALSE if the asynchronous request is still active. The method must be called until the return value is TRUE.

Outputs

Name	Type	Description
bError	BOOL	Returns FALSE if the method call was successful. Returns TRUE if an error has occurred.
hrErrorCode	HRESULT	Returns S_OK if the method call was successful. An error code is output in case of an error.

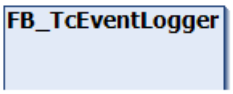
6.1.8.9 ipArguments

```
PROPERTY PUBLIC ipArguments : I_TcArguments
```

6.1.8.10 ipSourceInfo

```
PROPERTY ipSourceInfo : I_TcSourceInfo
```

6.1.9 FB_TcEventLogger



This function block represents the TwinCAT 3 EventLogger itself.

Syntax

Definition:

```
FUNCTION_BLOCK FB_TcEventLogger
```

Methods

Name	Description
ClearAlarms [▶ 68]	Clears active alarms.
ClearAllAlarms [▶ 68]	Calls Clear() for all alarms in the Raised state.
ClearLoggedEvents [▶ 69]	Clears logged events.
ConfirmAlarms [▶ 69]	
ConfirmAllAlarms [▶ 70]	Calls Confirm() for all alarms with the confirmation state WaitForConfirmation.
ExportLoggedEvents [▶ 70]	Exports logged events.
GetAlarm [▶ 71]	Returns the pointer to an existing alarm.
GetAlarmEx [▶ 71]	Returns the pointer to an existing alarm.
IsAlarmRaised [▶ 72]	Queries whether an alarm is in the Raised state.
IsAlarmRaisedEx [▶ 72]	Queries whether an alarm is in the Raised state.
SendMessage [▶ 73]	Sends a message.
SendMessage2 [▶ 73]	Sends a message.
SendMessageEx [▶ 74]	Sends a message.
SendMessageEx2 [▶ 75]	Sends a message.

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT v3.1.4022.20	PC or CX (x64, x86, ARM)	Tc3_EventLogger

6.1.9.1 ClearAlarms

Method for clearing active alarms. Returns `S_OK` if successful.

Inputs

Name	Type	Description
nTimeStamp	ULINT	Set to 0 to obtain the current time automatically. Initial: 0
bResetConfirmation	BOOL	If TRUE and the confirmation status is WaitForConfirmation, the confirmation status is set to Reset. Otherwise, the confirmation status is not changed. Initial: FALSE
ipFilter	I_TcEventFilter	Specify which alarms are to be cleared, otherwise all triggered alarms are cleared.

Return values

Name	Type	Description
ClearAlarms	HRESULT	

6.1.9.2 ClearAllAlarms

This method calls the Clear() method for all alarms in the alarm state Raised.

Syntax

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

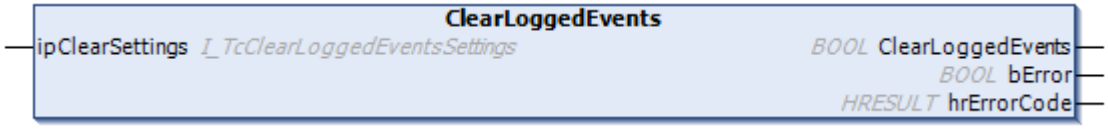
Inputs

Name	Type	Description
nTimeStamp	ULINT	0: Current time stamp is used > 0: External time stamp in 100 nanoseconds since January 1st, 1601 (UTC).
bResetConfirmation	BOOL	If TRUE and the confirmation state is WaitForConfirmation, the confirmation state is set to Reset. Otherwise the confirmation state is not changed.

Return value

Name	Type	Description
ClearAllAlarms	HRESULT	Returns S_OK if the method call was successful, otherwise an HRESULT as the error code

6.1.9.3 ClearLoggedEvents



Async method for clearing logged events. Returns TRUE if the asynchronous request is no longer assigned.

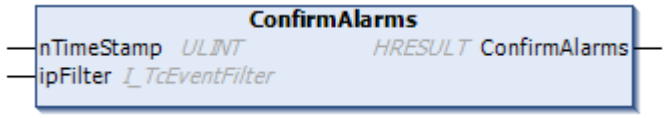
Inputs

Name	Type	Description
ipClearSettings	I_TcClearLoggedEventsSettings	Optional (otherwise the whole cache is cleared)

Return values

Name	Type	Description
ClearLoggedEvents	BOOL	
bError	BOOL	
hrErrorCode	HRESULT	

6.1.9.4 ConfirmAlarms



Inputs

Name	Type	Description
nTimeStamp	ULINT	Set to 0 to obtain the current time automatically. Initial: 0
ipFilter	I_TcEventFilter	Specify which alarms are to be confirmed, otherwise all alarms with the confirmation status WaitForConfirmation are confirmed.

Return values

Name	Type	Description
ConfirmAlarms	HRESULT	

6.1.9.5 ConfirmAllAlarms

This method calls the Confirm() method for all alarms having the confirmation state WaitForConfirmation.

Syntax

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

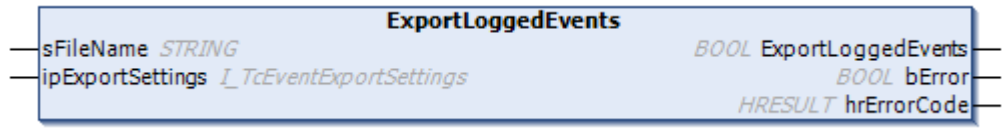
Inputs

Name	Type	Description
nTimeStamp	ULINT	0: Current time stamp is used > 0: External time stamp in 100 nanoseconds since January 1st, 1601 (UTC).

Return value

Name	Type	Description
ConfirmAllAlarms	HRESULT	Returns S_OK if the method call was successful, otherwise an HRESULT as the error code.

6.1.9.6 ExportLoggedEvents



Exports logged events asynchronously. Returns TRUE when asynchronous processing is complete.

Inputs

Name	Type	Description
sFileName	STRING	Name of the destination file
ipExportSettings	I_TcEventExportSettings	Specify which events are to be exported, otherwise all events will be exported. An instance of FB_TcEventCsvExportSettings [▶ 40] can be assigned for this purpose.

Return values

Name	Type	Description
ExportLoggedEvents	BOOL	TRUE, if the processing is completed.
bError	BOOL	TRUE when an error occurs.
hrErrorCode	HRESULT	Outputs the error information if `bError` is TRUE.

6.1.9.7 GetAlarm

Returns an interface pointer to an existing instance.

Syntax

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

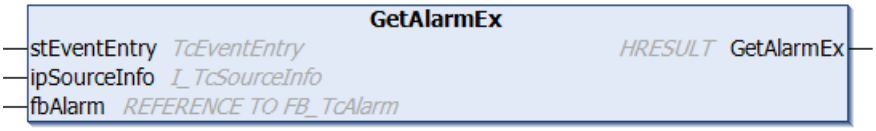
Inputs

Name	Type	Description
eventClass	GUID	GUID of the event class.
nEventId	UDINT	ID of the event
ipSourceInfo	I_TcSourceInfo [▶ 97]	Pointer to an ITcSourceInfo interface.
fbAlarm	REFERENCE TO FB_TcAlarm [▶ 51]	Pointer to an alarm.

Return value

Name	Type	Description
GetAlarm	HRESULT	Returns ADS_E_NOTFOUND if no instance was found. Returns S_OK if the method call was successful, otherwise an HRESULT as the error code.

6.1.9.8 GetAlarmEx



Returns an interface pointer to an existing instance.

Syntax

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

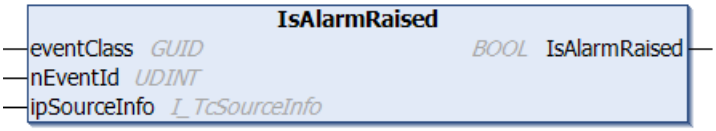
Inputs

Name	Type	Description
stEventEntry	TcEventEntry [▶ 97]	Event definition.
ipSourceInfo	I_TcSourceInfo [▶ 97]	Pointer to an ITcSourceInfo interface.
fbAlarm	REFERENCE TO FB_TcAlarm [▶ 51]	Pointer to an alarm.

Return value

Name	Type	Description
GetAlarmEx	HRESULT	Returns ADS_E_NOTFOUND if no instance was found. Returns S_OK if the method call was successful, otherwise an HRESULT as the error code.

6.1.9.9 IsAlarmRaised



This method queries whether an alarm is in the Raised state.

Syntax

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

Inputs

Name	Type	Description
eventClass	GUID	GUID of the event class.
nEventId	UDINT	ID of the event.
ipSourceInfo	I_TcSourceInfo [▶ 97]	Pointer to an ITcSourceInfo interface.

Return value

Name	Type	Description
IsAlarmRaised	BOOL	Returns S_OK if the method call was successful, otherwise an HRESULT as the error code.

6.1.9.10 IsAlarmRaisedEx

This method queries whether an alarm is in the Raised state.

Syntax

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

Inputs

Name	Type	Description
stEventEntry	UDINT	Event definition.
ipSourceInfo	I_TcSourceInfo [▶ 97]	Pointer to an ITcSourceInfo interface.

Return value

Name	Type	Description
IsAlarmRaisedEx	BOOL	Returns TRUE if the alarm is in the raised state.

6.1.9.11 SendMessage

This method sends a message.

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
```

Inputs

Name	Type	Description
eventClass	GUID	GUID of the event class.
nEventId	UDINT	ID of the event.
eSeverity	TcEventSeverity [▶ 98]	Severity of the event.
ipSourceInfo	I_TcSourceInfo [▶ 97]	Pointer to an ITcSourceInfo interface.
nTimeStamp	ULINT	0: Current time stamp is used. > 0: External time stamp in 100 nanoseconds since January 1st, 1601 (UTC).
ipArguments	I_TcArguments [▶ 81]	Pointer to the ITcArguments interface.

Return value

Name	Type	Description
SendMessage	HRESULT	Returns S_OK if the method call was successful, otherwise an HRESULT as the error code.

6.1.9.12 SendMessage2

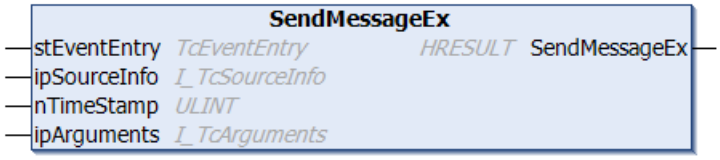
Inputs

Name	Type	Description
eventClass	GUID	
nEventId	UDINT	
eSeverity	TcEventSeverity	
ipSourceInfo	I_TcSourceInfo	Optional Initial: 0
nTimeStamp	ULINT	Set 0 to obtain the time automatically. Initial: 0
ipArguments	I_TcArguments	Optional Initial: 0
sJsonAttribute	STRING	

Return values

Name	Type	Description
SendMessage2	HRESULT	

6.1.9.13 SendMessageEx



This method sends a message.

Syntax

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

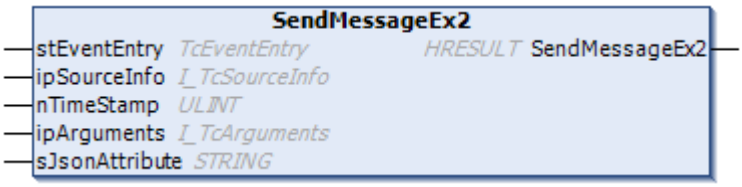
Inputs

Name	Type	Description
stEventEntry	TcEventEntry [▶ 97]	Event definition.
ipSourceInfo	I_TcSourceInfo [▶ 97]	Pointer to an ITcSourceInfo interface.
nTimeStamp	ULINT	0: Current time stamp is used > 0: External time stamp in 100 nanoseconds since January 1st, 1601 (UTC).
ipArguments	I_TcArguments [▶ 81]	Pointer to the ITcArguments interface.

Return value

Name	Type	Description
SendMessageEx	HRESULT	Returns S_OK if the method call was successful, otherwise an HRESULT as the error code.

6.1.9.14 SendMessageEx2



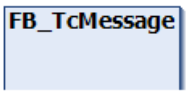
Inputs

Name	Type	Description
stEventEntry	TcEventEntry	
ipSourceInfo	I_TcSourceInfo	Optional Initial: 0
nTimeStamp	ULINT	Set to 0 to obtain the current time automatically. Initial: 0
ipArguments	I_TcArguments	Optional Initial: 0
sJsonAttribute	STRING	

Return values

Name	Type	Description
SendMessageEx2	HRESULT	

6.1.10 FB_TcMessage



This function block represents a message from the TwinCAT 3 EventLogger.

Syntax

Definition:

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

Inheritance hierarchy

FB_TcEventBase [▶ 61]

FB_TcMessage

Interfaces

Type	Description
I_TcMessage [▶ 96]	Provides methods and properties for the message handling.

Methods

Name	Definition location	Description
EqualsTo [▶ 62]	Inherited from FB_TcEventBase [▶ 61]	Compares the event with another instance.
EqualsToEventClass [▶ 63]	Inherited from FB_TcEventBase [▶ 61]	Compares the event class of the event with another event class.
EqualsToEventEntry [▶ 63]	Inherited from FB_TcEventBase [▶ 61]	Compares the event definition of the event with another event definition.
EqualsToEventEntryEx [▶ 64]	Inherited from FB_TcEventBase [▶ 61]	Compares the event definition of the event with another event definition.
GetJsonAttribute [▶ 64]	Inherited from FB_TcEventBase [▶ 61]	Returns the Json attribute.
Release [▶ 65]	Inherited from FB_TcEventBase [▶ 61]	Releases the instance created by the EventLogger again.
RequestEventClassName [▶ 65]	Inherited from FB_TcEventBase [▶ 61]	Requests the name of the event class.
RequestEventText [▶ 66]	Inherited from FB_TcEventBase [▶ 61]	Returns the text for the event.
Create [▶ 77]	Local	Creates a message instance in the EventLogger.
CreateEx [▶ 78]	Local	Creates a message instance in the EventLogger from an event definition.
SetJsonAttribute [▶ 78]	Local	Sets the Json attribute.
Send [▶ 96]	I_TcMessage [▶ 96]	Sends a message.

Properties

Name	Type	Access	Definition location	Description
eSeverity	TcEventSeverity [▶ 98]	Get	Inherited from FB_TcEventBase [▶ 61]	Returns the severity.
EventClass	GUID	Get	Inherited from FB_TcEventBase [▶ 61]	Returns the GUID of the event class.
ipArguments [▶ 67]	I_TcArguments [▶ 81]	Get	Inherited from FB_TcEventBase [▶ 61]	Returns the interface pointer for the arguments.
ipSourceInfo [▶ 67]	I_TcSourceInfo [▶ 97]	Get	Inherited from FB_TcEventBase [▶ 61]	The SourceInfo is created internally as the default behavior. It then contains the symbol name of the function block that instances FB_TcMessage as SourceName and the object ID of the PLC instance as SourceID. If the instance of FB_TcMessage is hidden with the attribute "hide", no symbol name can be created internally for the default behavior.
nEventId	nEventId	Get	Inherited from FB_TcEventBase [▶ 61]	Returns the ID of the event.
stEventEntry	TcEventEntry [▶ 97]	Get	Inherited from FB_TcEventBase [▶ 61]	Returns the event definition.
nTimeSent	ULINT	Get	Local	Returns the time of the Send.

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT v3.1.4022.20	PC or CX (x64, x86, ARM)	Tc3_EventLogger

6.1.10.1 Create

This method creates a message instance in the EventLogger.

Syntax

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

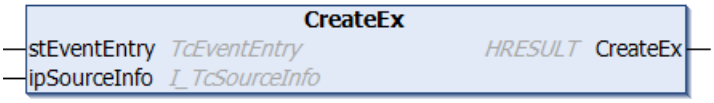
Inputs

Name	Type	Description
eventClass	GUID	GUID of the event class.
nEventId	UDINT	ID of the event.
eSeverity	TcEventSeverity [▶ 98]	Defines the severity.
ipSourceInfo	I_TcSourceInfo [▶ 97]	Pointer to an ITcSourceInfo interface.

Return value

Name	Type	Description
Create	HRESULT	Returns S_OK if the method call was successful, otherwise an HRESULT as the error code.

6.1.10.2 CreateEx



This method creates a message instance in the EventLogger from an event definition.

Syntax

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

Inputs

Name	Type	Description
stEventEntry	TcEventEntry [▶ 97]	Event definition.
ipSourceInfo	I_TcSourceInfo [▶ 97]	Interface pointer to the source information. Default source information is created if no interface pointer is transferred.

Return value

Name	Type	Description
CreateEx	HRESULT	Returns S_OK if the method call was successful, otherwise an HRESULT as the error code.

6.1.10.3 SetJsonAttribute

This method sets the JSON attribute.

Syntax

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

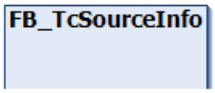
Inputs

Name	Type	Description
sJsonAttribute	STRING	JSON string

Return value

Name	Type	Description
SetJsonAttribute	HRESULT	Returns S_OK if the method call was successful, otherwise an HRESULT as the error code.

6.1.11 FB_TcSourceInfo



The source information of an event can be defined with this function block.

Syntax

Definition:

```
FUNCTION_BLOCK FB_TcSourceInfo IMPLEMENTS I_TcSourceInfo
```

Interfaces

Type	Description
I_TcSourceInfo [▶ 97]	Provides read methods and read properties of a source information.

Methods

Name	Definition location	Description
Clear [▶ 80]	Local	Resets the source information.
ExtendName [▶ 80]	Local	Appends the transferred string to the name.
ResetToDefault [▶ 81]	Local	Sets the properties to default values. sName is initialized with the symbol name of the instanced function block. nId is initialized with the object ID of the PLC instance. If the instance of FB_TcSourceInfo is hidden with the attribute "hide", no symbol name can be created internally for the default behavior.
EqualsTo [▶ 97]	I_TcSourceInfo [▶ 97]	Compares an instance with another instance.

Properties

Name	Type	Access	Definition location	Description
guid	GUID	Get	I_TcSourceInfo [▶ 97]	Returns the GUID of the source information.
guid	GUID	SET	Local	Sets the GUID as source information.
nId	UDINT	Get	I_TcSourceInfo [▶ 97]	Returns the ID of the source information.
nId	UDINT	SET	Local	Sets the ID of the source information.
sName	STRING(ParameterList.cSourceNameSize-1)	Get	I_TcSourceInfo [▶ 97]	Returns the name of the source information.
sName	STRING(ParameterList.cSourceNameSize-1)	SET	Local	Sets the name of the source information

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT v3.1.4022.20	PC or CX (x64, x86, ARM)	Tc3_EventLogger

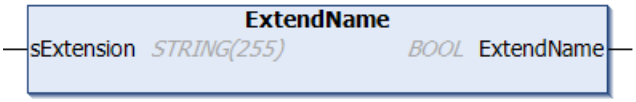
6.1.11.1 Clear

This method resets the source information.

Syntax

```
METHOD Clear
```

6.1.11.2 ExtendName



This method extends the name.

Syntax

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

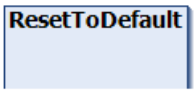
Inputs

Name	Type	Description
sExtension	STRING(255)	Text to be appended to the right.

Return value

Name	Type	Description
ExtendName	BOOL	Returns TRUE if the concatenation was successful. Returns FALSE if the resulting character string is longer than the output character string and doesn't fit in the given output buffer. The memory requirement for the resulting string is then larger than that for the output string. The string is then truncated.

6.1.11.3 ResetToDefault



This method sets the source information to default values.

Default values:

sName is initialized with the symbol name of the instanced function block.

nId is initialized with the object ID of the PLC instance.

If the instance of FB_TcSourceInfo is hidden with the attribute "hide", no symbol name can be created internally for the default behavior.

Syntax

```
METHOD ResetToDefault
```

6.2 Interfaces

6.2.1 I_TcArguments

This interface defines methods for the argument handling.

Inheritance hierarchy

__SYSTEM.IQueryInterface

I_TcArguments

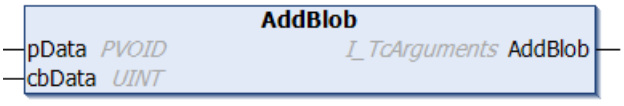
Methods

Name	Description
AddBlob [▶ 82]	Adds binary data as an argument.
AddBool [▶ 83]	Adds an argument of the type BOOL.
AddByte [▶ 83]	Adds an argument of the type BYTE.
AddDint [▶ 84]	Adds an argument of the type DINT.
AddDWord [▶ 84]	Adds an argument of the type DWORD.
AddEventReferenceId [▶ 84]	Adds a reference to another event as an argument.
AddEventReferenceIdGuid [▶ 85]	Adds a reference to another event as an argument.
AddInt [▶ 85]	Adds an argument of the type INT.
AddLInt [▶ 86]	Adds an argument of the type LINT.
AddLReal [▶ 86]	Adds an argument of the type LREAL.
AddReal [▶ 87]	Adds an argument of the type REAL.
AddSInt [▶ 87]	Adds an argument of the type SINT.
AddString [▶ 88]	Adds an argument of the type STRING.
AddUDint [▶ 88]	Adds an argument of the type UDINT.
AddUInt [▶ 88]	Adds an argument of the type INT.
AddULInt [▶ 89]	Adds an argument of the type ULINT.
AddUSInt [▶ 89]	Adds an argument of the type USINT.
AddWord [▶ 90]	Adds an argument of the type WORD.
AddWString [▶ 90]	Adds an argument of the type WSTRING.
Clear [▶ 91]	Removes all arguments.

Properties

Name	Type	Access	Description
nCount	UDINT	Get	Returns the number of transferred arguments.

6.2.1.1 AddBlob



This method adds binary data as an argument.

Syntax

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

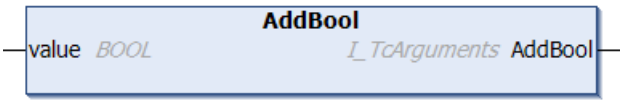
Inputs

Name	Type	Description
pData	PVOID	Pointer to the first byte of the binary data.
cbData	UINT	Length of the binary data in bytes.

Return value

Name	Type	Description
AddBlob	I_TcArguments [▶ 81]	Returns the I_TcArgument pointer again.

6.2.1.2 AddBool



This method adds an argument of the type BOOL.

Syntax

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

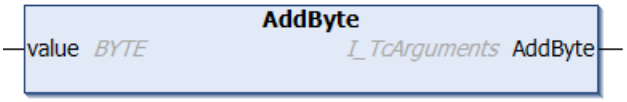
Inputs

Name	Type	Description
value	BOOL	Value to be added.

Return value

Name	Type	Description
AddBool	I_TcArguments [▶ 81]	Returns the I_TcArgument pointer again.

6.2.1.3 AddByte



This method adds an argument of the type BYTE.

Syntax

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

Inputs

Name	Type	Description
value	BYTE	Value to be added.

Return value

Name	Type	Description
AddByte	I_TcArguments [▶ 81]	Returns the I_TcArgument pointer again.

6.2.1.4 AddDint

This method adds an argument of the type DINT.

Syntax

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

Inputs

Name	Type	Description
value	DINT	Value to be added.

Return value

Name	Type	Description
AddDINT	I_TcArguments [▶ 81]	Returns the I_TcArgument pointer again.

6.2.1.5 AddDWord

This method adds an argument of the type DWORD.

Syntax

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

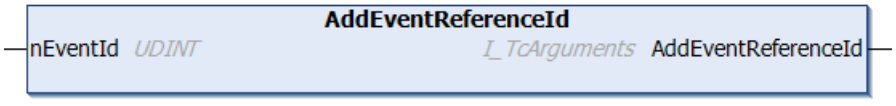
Inputs

Name	Type	Description
value	DWORD	Value to be added.

Return value

Name	Type	Description
AddDWord	I_TcArguments [▶ 81]	Returns the I_TcArgument pointer again.

6.2.1.6 AddEventReferenceId



This method adds a reference to another event as an argument.

Syntax

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

Inputs

Name	Type	Description
nEventId	UDINT	ID of the event.

Return value

Name	Type	Description
AddEventReferenceId	I_TcArguments [▶ 81]	Returns the I_TcArgument pointer again.

6.2.1.7 AddEventReferenceIdGuid

This method adds a reference to another event as an argument.

Syntax

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

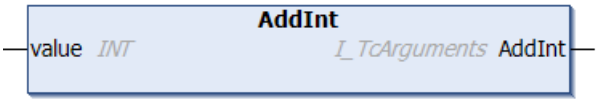
Inputs

Name	Type	Description
nEventId	UDINT	ID of the event.
EventClass	GUID	GUID of the event class.

Return value

Name	Type	Description
AddEventReferenceIdGuid	I_TcArguments [▶ 81]	Returns the I_TcArgument pointer again.

6.2.1.8 AddInt



This method adds an argument of the type INT.

Syntax

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

Inputs

Name	Type	Description
value	INT	Value to be added.

Return value

Name	Type	Description
AddInt	I_TcArguments [▶ 81]	Returns the I_TcArgument pointer again.

6.2.1.9 AddLInt

This method adds an argument of the type LINT.

Syntax

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

Inputs

Name	Type	Description
value	LINT	Value to be added.

Return value

Name	Type	Description
AddLInt	I_TcArguments [▶ 81]	Returns the I_TcArgument pointer again.

6.2.1.10 AddLReal

This method adds an argument of the type LREAL.

Syntax

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

Inputs

Name	Type	Description
value	LREAL	Value to be added.

Return value

Name	Type	Description
AddLReal	I_TcArguments [▶ 81]	Returns the I_TcArgument pointer again.

6.2.1.11 AddReal

This method adds an argument of the type REAL.

Syntax

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

Inputs

Name	Type	Description
value	REAL	Value to be added.

Return value

Name	Type	Description
AddReal	I_TcArguments [▶ 81]	Returns the I_TcArgument pointer again.

6.2.1.12 AddSInt

This method adds an argument of the type SINT.

Syntax

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

Inputs

Name	Type	Description
value	SINT	Value to be added.

Return value

Name	Type	Description
AddSInt	I_TcArguments [▶ 81]	Returns the I_TcArgument pointer again.

6.2.1.13 AddString

This method adds an argument of the type STRING.

Syntax

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

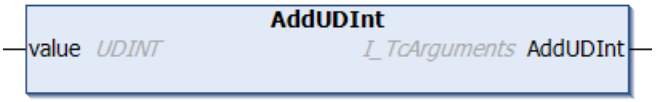
Inputs

Name	Type	Description
value	STRING	Value to be added.

Return value

Name	Type	Description
AddString	I_TcArguments [▶ 81]	Returns the I_TcArgument pointer again.

6.2.1.14 AddUDint



This method adds an argument of the type UDINT.

Syntax

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

Inputs

Name	Type	Description
value	UDINT	Value to be added.

Return value

Name	Type	Description
AddUDInt	I_TcArguments [▶ 81]	Returns the I_TcArgument pointer again.

6.2.1.15 AddUInt

This method adds an argument of the type INT.

Syntax

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

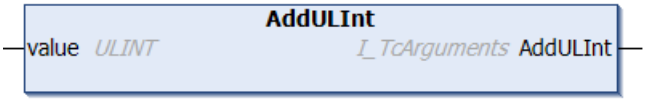
Inputs

Name	Type	Description
value	UINT	Value to be added.

Return value

Name	Type	Description
AddUInt	I_TcArguments [▶ 81]	Returns the I_TcArgument pointer again.

6.2.1.16 AddULInt



This method adds an argument of the type ULINT.

Syntax

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

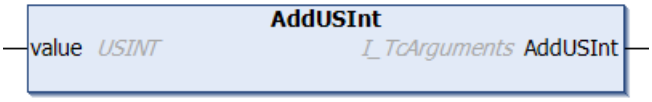
Inputs

Name	Type	Description
value	ULINT	Value to be added.

Return value

Name	Type	Description
AddULInt	I_TcArguments [▶ 81]	Returns the I_TcArgument pointer again.

6.2.1.17 AddUSInt



This method adds an argument of the type USINT.

Syntax

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

Inputs

Name	Type	Description
value	USINT	Value to be added.

Return value

Name	Type	Description
AddUSInt	I_TcArguments [▶ 81]	Returns the I_TcArgument pointer again.

6.2.1.18 AddWord

This method adds an argument of the type WORD.

Syntax

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

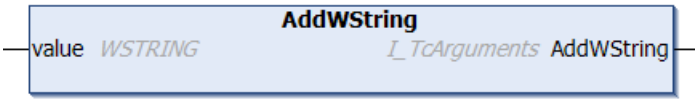
Inputs

Name	Type	Description
value	WORD	Value to be added.

Return value

Name	Type	Description
AddWord	I_TcArguments [▶ 81]	Returns the I_TcArgument pointer again.

6.2.1.19 AddWString



This method adds an argument of the type WSTRING.

Syntax

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

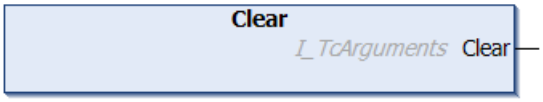
Inputs

Name	Type	Description
value	WSTRING	Value to be added.

Return value

Name	Type	Description
AddWString	I_TcArguments [▶ 81]	Returns the I_TcArgument pointer again.

6.2.1.20 Clear



This method removes all arguments.

Syntax

```
METHOD Clear : I_TcArguments
```

Return value

Name	Type	Description
Clear	I_TcArguments [▶ 81]	Returns the I_TcArgument pointer again.

6.2.2 I_TcEventBase

Methods and properties of an event are defined in this basic interface.

Methods

Name	Description
EqualsTo [▶ 92]	Compares the event with another instance.
EqualsToEventClass [▶ 92]	Compares the event class of the event with another event class.
EqualsToEventEntryEx [▶ 93]	Compares the event definition of the event with another event definition.
GetJsonAttribute [▶ 94]	Returns the Json attribute.
RequestEventClassName [▶ 94]	Requests the name of the event class.
RequestEventText [▶ 95]	Returns the text for the event.

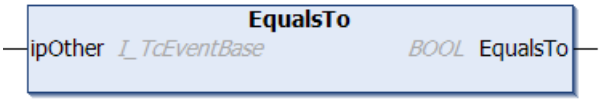
Properties

Name	Type	Access	Description
eSeverity	TcEventSeverity [▶ 98]	Get	Returns the severity.
EventClass	GUID	Get	Returns the GUID of the event class.
ipSourceInfo	I_TcSourceInfo [▶ 97]	Get	Returns a pointer to the source definition.
nEventId	UDINT	Get	Returns the ID of the event.
stEventEntry	TcEventEntry [▶ 97]	Get	Returns the event definition.

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT v3.1.4022.20	PC or CX (x64, x86, ARM)	Tc3_EventLogger

6.2.2.1 EqualsTo



This method carries out a comparison with another event specified at the input.

Syntax

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

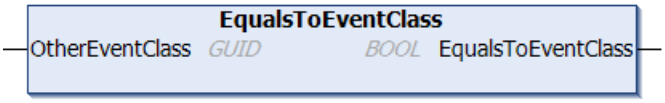
Inputs

Name	Type	Description
ipOther	I_TcEventBase [▶ 91]	Event to be compared

Return value

Name	Type	Description
EqualsTo	BOOL	Returns TRUE if the events match.

6.2.2.2 EqualsToEventClass



This method carries out a comparison with another event class specified at the input.

Syntax

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

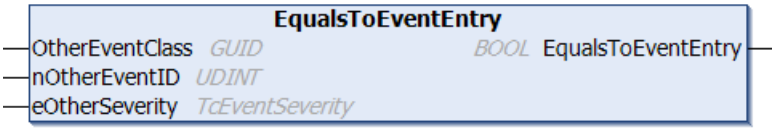
Inputs

Name	Type	Description
OtherEventClass	GUID	Event class to be compared.

Return value

Name	Type	Description
EqualsToEventClass	BOOL	Returns TRUE if the event classes match.

6.2.2.3 EqualsToEventEntry



This method carries out a comparison with another event specified at the input.

Syntax

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

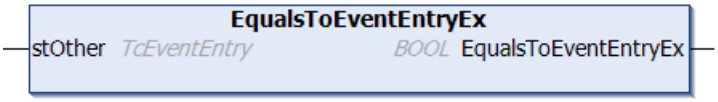
Inputs

Name	Type	Description
OtherEventClass	GUID	Event class of the event to be compared.
nOtherEventID	UDINT	Event ID of the event to be compared.
eOtherSeverity	TcEventSeverity [▶ 98]	Event severity of the event to be compared.

Return value

Name	Type	Description
EqualsToEventEntry	BOOL	Returns TRUE if the events match.

6.2.2.4 EqualsToEventEntryEx



This method carries out a comparison with another event specified at the input.

Syntax

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

Inputs

Name	Type	Description
stOther	TcEventEntry [▶ 97]	Event to be compared.

Return value

Name	Type	Description
EqualsToEventEntryEx	BOOL	Returns TRUE if the events match.

6.2.2.5 GetJsonAttribute

This method returns the JSON attribute.

Syntax

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

Inputs

Name	Type	Description
sJsonAttribute	REFERENCE TO STRING	Reference to a variable of the type String
nJsonAttribute	UDINT	Length of the String variable

Return value

Name	Type	Description
GetJsonAttribute	HRESULT	Returns S_OK if the method call was successful. Returns ERROR_BAD_LENGTH if the length of the variable is too small. Otherwise HRESULT is returned as the error code.

6.2.2.6 RequestEventClassName

This method returns the name of the event class.

Syntax

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

Inputs

Name	Type	Description
nLangId	DINT	Specifies the language ID English (en-US) = 1033 German (de-DE) = 1031 …
sResult	REFERENCE TO STRING	Reference to a variable of the type String
nResultSize	UDINT	Size of the String variable in bytes

Return value

Name	Type	Description
RequestEventClassName	BOOL	Returns TRUE as soon as the request has been terminated. Returns FALSE if the asynchronous request is still active. The method must be called until the return value is TRUE.

Outputs

Name	Type	Description
bError	BOOL	Returns FALSE if the method call was successful. Returns TRUE if an error has occurred.
hrErrorCode	HRESULT	Returns S_OK if the method call was successful. An error code is output in case of an error.

6.2.2.7 RequestEventText

This method returns the event text.

Syntax

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

Inputs

Name	Type	Description
nLangId	DINT	Specifies the language ID English (en-US) = 1033 German (de-DE) = 1031 …
sResult	REFERENCE TO STRING	Reference to a variable of the type String
nResultSize	UDINT	Size of the String variable in bytes

Return value

Name	Type	Description
RequestEventText	BOOL	Returns TRUE as soon as the request has been terminated. Returns FALSE if the asynchronous request is still active. The method must be called until the return value is TRUE.

Outputs

Name	Type	Description
bError	BOOL	Returns FALSE if the method call was successful. Returns TRUE if an error has occurred.
hrErrorCode	HRESULT	Returns S_OK if the method call was successful. An error code is output in case of an error.

6.2.3 I_TcMessage

This interface provides methods and properties for the message handling.

Inheritance hierarchy

I_TcEventBase [▶ 91]

I_TcMessage

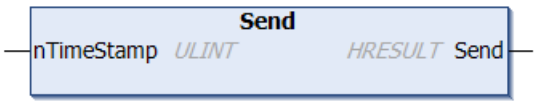
Methods

Name	Description
Send [▶ 96]	Sends a message

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT v3.1.4022.20	PC or CX (x64, x86, ARM)	Tc3_EventLogger

6.2.3.1 Send



This method sends the message.

Syntax

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

Inputs

Name	Type	Description
nTimeStamp	ULINT	0: Current time stamp is used > 0: External time stamp in 100 nanoseconds since January 1st, 1601 (UTC).

Return value

Name	Type	Description
Send	FB_ HRESULT	Returns S_OK if the method call was successful, otherwise an HRESULT as the error code

6.2.4 I_TcSourceInfo

This interface defines properties for an item of source information.

Methods

Name	Description
EqualsTo [▶ 97]	Compares an instance with source information with another instance.

Properties

Name	Type	Access	Description
guid	GUID	Get	Returns the GUID of the source information.
nId	UDINT	Get	Returns the ID of the source information.
sName	STRING(ParameterList.cSourceNameSize-1)	Get	Returns the name of the source information.

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT v3.1.4022.20	PC or CX (x64, x86, ARM)	Tc3_EventLogger

6.2.4.1 EqualsTo

This method compares an instance with source information with another instance.

Syntax

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

Inputs

Name	Type	Description
ipOther	I_TcSourceInfo [▶ 97]	Items of source information to be compared

Return value

Name	Type	Description
EqualsTo	BOOL	Returns TRUE if the items of source information match.

6.3 Data types

6.3.1 TcEventEntry

Defines an event by means of event class, event ID and severity.

Syntax

Definition:

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

Parameter

Name	Type	Description
uuidEventClass	GUID	GUID of the event class.
nEventId	UDINT	ID of the event.
eSeverity	TcEventSeverity	Event severity defines the severity of the event,

6.3.2 TcEventSeverity

Defines the severity of the event.

Syntax

Definition:

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

Parameter

	Name	Description
4	Critical	Critical
3	Error	Error
2	Warning	Warning
1	Info	Information
0	Verbose	Extended output

6.3.3 TcEventConfirmationState

Defines the confirmation state of an alarm.

Syntax

Definition:

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

Parameter

Name	Description
Confirmed	Confirmed
NotRequired	Confirmation not necessary in the current state. (Alarm not currently in the Raised state).
NotSupported	Was initialized without confirmation.
Reset	Initial state
WaitForConfirmation	Waiting for confirmation.

6.4 Global lists

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	Type	Initial value
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 [▶ 98]	TcEventSeverity.Verbose
SUCCESS_EVENT	TcEventEntry [▶ 97]	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	Type	Initial value	Description
nLangId_OnlineMonitoring	DINT	1033	Language ID for the online monitoring English (en-US) = 1033 German (de-DE) = 1031 …

6.4.3 Parameter list

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

Name	Type	Initial value	Description
cSourceNameSize	UDINT(81..10000)	256	Size in bytes for the name of the source information. A maximum of 512 bytes is recommended.

6.4.4 Global_Version

All libraries have a certain version. This version can be seen in the PLC library repository among others.
A global constant contains the library version information (of type ST_LibVersion):

Global_Version

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

To check whether the version you have is the version you need, use the function F_CmpLibVersion (defined in the Tc2_System library).

7 C++ API

7.1 Interfaces

7.1.1 ITcEvent

This interface provides general methods for ITcAlarm and ITcMessage. It is used in the callbacks of the interfaces ITcAlarmListener and ITcMessageListener.

Syntax

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

Methods

Name	Description
GetEventClass [▶ 101]	Returns the GUID of the event class.
GetEventId [▶ 101]	Returns the ID of the event.
GetSeverity [▶ 102]	Returns the Severity of the event.
GetSourceInfo [▶ 102]	Returns the SourceInfo.
GetJsonAttribute [▶ 102]	Returns the JSON attribute.
GetText [▶ 103]	Returns the text asynchronously.
GetEventClassName [▶ 103]	Returns the event class name asynchronously.

7.1.1.1 GetEventClass

Returns the GUID of the event class.

Syntax

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

Parameter

Name	Type	Description
eventClass	REFERENCE TO GUID	Reference to the GUID of the event class.

Return value

Type	Description
HRESULT	Returns S_OK if the method call was successful, otherwise an HRESULT as the error code.

7.1.1.2 GetEventId

Returns the ID of the event.

Syntax

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

Parameter

Name	Type	Description
eventId	REFERENCE TO UDINT	Reference to the ID of the event.

Return value

Type	Description
HRESULT	Returns S_OK if the method call was successful, otherwise an HRESULT as the error code.

7.1.1.3 GetSeverity

Returns the Severity of the event.

Syntax

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

Parameter

Name	Type	Description
severity	REFERENCE TO TcEventSeverity	Reference to the severity of an event.

Return value

Type	Description
HRESULT	Returns S_OK if the method call was successful, otherwise an HRESULT as the error code.

7.1.1.4 GetSourceInfo

Returns the SourceInfo.

Syntax

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

Parameter

Name	Type	Description
pipSourceInfo	POINTER TO ITcSourceInfo	Reference to the SourceInfo interface.

Return value

Type	Description
HRESULT	Returns S_OK if the method call was successful, otherwise an HRESULT as the error code.

7.1.1.5 GetJsonAttribute

Returns the JSON attribute.

Syntax

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

Parameter

Name	Type	Description
sJsonAttribute	REFERENCE TO STRING	Reference to the JSON string.
nJsonAttribute	REFERENCE TO UDINT	Reference to the length of the Json attribute.

Return value

Type	Description
HRESULT	Returns S_OK if the method call was successful, otherwise an HRESULT as the error code.

7.1.1.6 GetText

Returns the text asynchronously.

Syntax

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

Parameter

Name	Type	Description
nLangId	DINT	Language ID (LCID) of the requested language.
pipResult	POINTER TO ITcAsyncStringResult	Reference to an ITcAsyncStringResult pointer.

Return value

Type	Description
HRESULT	Returns S_OK if the method call was successful, otherwise an HRESULT as the error code.

7.1.1.7 GetEventClassName

Returns the event class name asynchronously.

Syntax

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

Parameter

Name	Type	Description
nLangId	DINT	Language ID (LCID) of the requested language.
pipResult	POINTER TO ITcAsyncStringResult	Reference to an ITcAsyncStringResult pointer.

Return value

Type	Description
HRESULT	Returns S_OK if the method call was successful, otherwise an HRESULT as the error code.

7.1.2 ITcMessage

This interface represents a message from the TwinCAT 3 EventLogger.

Syntax

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

Methods

Name	Description
SetJsonAttribute [▶ 104]	Sets the JSON attribute.
GetArguments [▶ 104]	Returns the interface pointer for the arguments.
Send [▶ 104]	Sends the message.

7.1.2.1 SetJsonAttribute

Sets the JSON attribute.

Syntax

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

Parameter

Name	Type	Description
sJsonAttribute	STRING	JSON string

Return value

Type	Description
HRESULT	Returns S_OK if the method call was successful, otherwise an HRESULT as the error code.

7.1.2.2 GetArguments

Returns the interface pointer for the arguments.

Syntax

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

Parameter

Name	Type	Description
pipArguments	POINTER to ITcArguments	Reference to the ITcArguments interface

Return value

Type	Description
HRESULT	Returns S_OK if the method call was successful, otherwise an HRESULT as the error code.

7.1.2.3 Send

Sends the message.

Syntax

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

Parameter

Name	Type	Description
timeStamp	ULINT	> 0: External time stamp in 100 nanoseconds since January 1st, 1601 (UTC).

Return value

Type	Description
HRESULT	Returns S_OK if the method call was successful, otherwise an HRESULT as the error code.

7.1.3 ITcAlarm

This interface represents an alarm from the TwinCAT 3 EventLogger.

Syntax

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

Methods

Name	Description
SetJsonAttribute [▶ 105]	Sets the JSON attribute.
GetArguments [▶ 106]	Returns the interface pointer for the arguments.
GetIsRaised [▶ 106]	Returns TRUE if the alarm is in the raised state.
Raise [▶ 106]	Sets the alarm state to Raised.
Clear [▶ 107]	Sets the alarm state to Not Raised.
GetConfirmationState [▶ 107]	Returns the confirmation state.
Confirm [▶ 107]	Sets the alarm state to Confirmed.

7.1.3.1 SetJsonAttribute

Sets the Json attribute.

Syntax

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

Parameter

Name	Type	Description
sJsonAttribute	STRING	JSON string

Return value

Type	Description
HRESULT	Returns S_OK if the method call was successful, otherwise an HRESULT as the error code.

7.1.3.2 GetArguments

Returns the interface pointer for the arguments.

Syntax

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

Parameter

Name	Type	Description
pipArguments	POINTER to ITcArguments	Reference to the ITcArguments interface

Return value

Type	Description
HRESULT	Returns S_OK if the method call was successful, otherwise an HRESULT as the error code.

7.1.3.3 GetIsRaised

Returns TRUE in the parameter bIsRaised if the alarm is in the Raised state.

Syntax

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

Parameter

Name	Type	Description
bIsRaised	REFERENCE TO BOOL32	Reference to the state. Returns TRUE if the alarm is in the raised state.

Return value

Type	Description
HRESULT	Returns S_OK if the method call was successful, otherwise an HRESULT as the error code.

7.1.3.4 Raise

Sets the alarm state [▶ 14] to Raised.

If the alarm requires mandatory confirmation, the confirmation state is additionally set to WaitForConfirmation.

Syntax

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

Parameter

Name	Type	Description
timeStamp	ULINT	> 0: External time stamp in 100 nanoseconds since January 1st, 1601 (UTC).

Return value

Type	Description
HRESULT	Returns S_OK if the method call was successful, otherwise an HRESULT as the error code.

7.1.3.5 Clear

Sets the alarm state [▶ 14] to Not Raised.

Syntax

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

Parameter

Name	Type	Description
timeStamp	ULINT	> 0: External time stamp in 100 nanoseconds since January 1st, 1601 (UTC).
bResetConfirmation	BOOL32	If TRUE and the confirmation state is WaitForConfirmation, the confirmation state is set to Reset. Otherwise the confirmation state is not changed.

Return value

Type	Description
HRESULT	Returns S_OK if the method call was successful, otherwise an HRESULT as the error code.

7.1.3.6 GetConfirmationState

Returns the confirmation state [▶ 14].

Syntax

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

Parameter

Name	Type	Description
state	REFERENCE TO TcEventConfirmationState	Returns the confirmation state.

Return value

Type	Description
HRESULT	Returns S_OK if the method call was successful, otherwise an HRESULT as the error code.

7.1.3.7 Confirm

Sets the confirmation state [▶ 14] of WaitingForConfirmation to Confirmed.

Syntax

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

Parameter

Name	Type	Description
timeStamp	ULINT	> 0: External time stamp in 100 nanoseconds since January 1st, 1601 (UTC).

Return value

Type	Description
HRESULT	Returns S_OK if the method call was successful, otherwise an HRESULT as the error code.

7.1.4 ITcEventLogger

This interface represents the TwinCAT 3 EventLogger itself.

Syntax

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

Required includes:

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

Methods

Name	Description
CreateMessage [▶ 108]	Creates an instance that implements ITcMessage.
CreateAlarm [▶ 109]	Creates an instance that implements ITcAlarm.
GetAlarm [▶ 109]	Returns the pointer to an existing alarm.
IsAlarmRaised [▶ 110]	Queries whether an alarm is in the Raised state.
ConfirmAllAlarms [▶ 110]	Calls Confirm() for all alarms with the confirmation state WaitForConfirmation.
ClearAllAlarms [▶ 111]	Calls Clear() for all alarms in the Raised state.
SendTcMessage [▶ 111]	Sends a message.
AddMessageListener [▶ 112]	Adds a message listener.
RemoveMessageListener [▶ 112]	Removes a message listener.
NotifyMessageListener [▶ 113]	Processes a queue for a message listener.
AddAlarmListener [▶ 113]	Adds an alarm listener.
RemoveAlarmListener [▶ 113]	Removes an alarm listener.
NotifyAlarmListener [▶ 114]	Processes a queue for an alarm listener.
GetEventText [▶ 114]	Returns a text for an event.
GetEventClassName [▶ 115]	Returns the class name for an event.
CreateArguments [▶ 115]	Creates an instance that implements ITcArguments.

7.1.4.1 CreateMessage

Creates an instance that implements ITcMessage.

Syntax

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

Parameter

Name	Type	Description
eventClass	REFERENCE TO GUID	Reference to the GUID of the event class.
eventId	REFERENCE TO UDINT	Reference to the ID of the event.
severity	REFERENCE TO TcEventSeverity	Reference to the severity of an event.
ipSourceInfo	ITcSourceInfo	Pointer to the ITcSourceInfo interface.
pipMessage	ITcMessage [▶ 104]	Pointer to an ITcMessage pointer.

Return value

Type	Description
HRESULT	Returns S_OK if the method call was successful, otherwise an HRESULT as the error code.

7.1.4.2 CreateAlarm

Creates an instance that implements ITcAlarm.

Syntax

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

Parameter

Name	Type	Description
eventClass	REFERENCE TO GUID	Reference to the GUID of the event class.
eventId	REFERENCE TO UDINT	Reference to the ID of the event.
severity	REFERENCE TO TcEventSeverity	Reference to the severity of an event.
bWithConfirmation	BOOL32	Defines whether the alarm requires mandatory confirmation.
ipSourceInfo	ITcSourceInfo	Pointer to the ITcSourceInfo interface.
pipAlarm	ITcAlarm [▶ 105]	Pointer to an ITcAlarm pointer.

Return value

Type	Description
HRESULT	Returns S_OK if a new alarm was successfully created. Returns ERROR_ALREADY_EXISTS if the alarm already exists. In case of error an HRESULT is returned as the error code.

7.1.4.3 GetAlarm

Returns an interface pointer to an existing instance.

Syntax

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

Parameter

Name	Type	Description
eventClass	REFERENCE TO GUID	Reference to the GUID of the event class.
eventId	REFERENCE TO UDINT	Reference to the ID of the event.
ipSourceInfo	ITcSourceInfo	Pointer to the ITcSourceInfo interface.
pipAlarm	ITcAlarm [▶ 105]	Pointer to an ITcAlarm pointer.

Return value

Type	Description
HRESULT	Returns ADS_E_NOTFOUND if no instance was found. Returns S_OK if everything was successful, otherwise an HRESULT as the error code.

7.1.4.4 IsAlarmRaised

Queries whether an alarm is in the Raised state.

Syntax

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

Parameter

Name	Type	Description
eventClass	REFERENCE TO GUID	Reference to the GUID of the event class.
eventId	REFERENCE TO UDINT	Reference to the ID of the event.
bIsRaised	REFERENCE TO BOOL32	Reference to the state. Returns TRUE if the alarm is in the raised state.
ipSourceInfo	ITcSourceInfo	Pointer to the ITcSourceInfo interface.

Return value

Type	Description
HRESULT	Returns S_OK if the method call was successful, otherwise an HRESULT as the error code.

7.1.4.5 ConfirmAllAlarms

Calls Confirm() for all alarms with the confirmation state WaitForConfirmation.

Syntax

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

Parameter

Name	Type	Description
timeStamp	ULINT	> 0: External time stamp in 100 nanoseconds since January 1st, 1601 (UTC).

Return value

Type	Description
HRESULT	Returns S_OK if the method call was successful, otherwise an HRESULT as the error code.

7.1.4.6 ClearAllAlarms

Calls Clear() for all alarms in the Raised state.

Syntax

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

Parameter

Name	Type	Description
timeStamp	ULINT	> 0: External time stamp in 100 nanoseconds since January 1st, 1601 (UTC).
bResetConfirmation	BOOL32	If TRUE and the confirmation state is WaitForConfirmation, the confirmation state is set to Reset. Otherwise the confirmation state is not changed.

Return value

Type	Description
HRESULT	Returns S_OK if the method call was successful, otherwise an HRESULT as the error code.

7.1.4.7 SendTcMessage

Sends a message.

Syntax

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

Parameter

Name	Type	Description
eventClass	REFERENCE TO GUID	Reference to the GUID of the event class.
eventId	REFERENCE TO UDINT	Reference to the ID of the event.
severity	REFERENCE TO TcEventSeverity	Reference to the severity of an event.
ipSourceInfo	ITcSourceInfo	Pointer to the ITcSourceInfo interface.
timeStamp	ULINT	> 0: External time stamp in 100 nanoseconds since January 1st, 1601 (UTC).
ipSerializedArguments	ITcArguments	Pointer to the ITcArguments interface.

Return value

Type	Description
HRESULT	Returns S_OK if the method call was successful, otherwise an HRESULT as the error code.

7.1.4.8 AddMessageListener

Adds a message listener.

Syntax

```
virtual HRESULT TCOMAPI AddMessageListener (ITcMessageListener ipListener, ITcEventFilterConfig pipFilterConfig)
```

Parameter

Name	Type	Description
ipListener	ITcMessageListener	Pointer to the ITcMessageListener interface.
pipFilterConfig	ITcEventFilterConfig	Pointer to an ITcEventFilterConfig pointer.

Return value

Type	Description
HRESULT	Returns S_OK if the method call was successful, otherwise an HRESULT as the error code.

7.1.4.9 RemoveMessageListener

Removes a message listener.

Syntax

```
virtual HRESULT TCOMAPI RemoveMessageListener (ITcMessageListener ipListener)
```

Parameter

Name	Type	Description
ipListener	ITcMessageListener	Pointer to the ITcMessageListener interface.

Return value

Type	Description
HRESULT	Returns S_OK if the method call was successful, otherwise an HRESULT as the error code.

7.1.4.10 NotifyMessageListener

Processes a queue for the message listener.

Syntax

```
virtual HRESULT TCOMAPI NotifyMessageListener (ITcMessageListener ipListener)
```

Parameter

Name	Type	Description
ipListener	ITcMessageListener	Pointer to the ITcMessageListener interface.

Return value

Type	Description
HRESULT	Returns S_OK if the method call was successful, otherwise an HRESULT as the error code.

7.1.4.11 AddAlarmListener

Adds an alarm listener.

Syntax

```
virtual HRESULT TCOMAPI AddAlarmListener (ITcMessageListener ipListener, ITcEventFilterConfig pipFil
terConfig)
```

Parameter

Name	Type	Description
ipListener	ITcMessageListener	Pointer to the ITcMessageListener interface.
pipFilterConfig	ITcEventFilterConfig	Pointer to an ITcEventFilterConfig pointer.

Return value

Type	Description
HRESULT	Returns S_OK if the method call was successful, otherwise an HRESULT as the error code.

7.1.4.12 RemoveAlarmListener

Removes an alarm listener.

Syntax

```
virtual HRESULT TCOMAPI RemoveAlarmListener (ITcMessageListener ipListener)
```

Parameter

Name	Type	Description
ipListener	ITcMessageListener	Pointer to the ITcMessageListener interface.

Return value

Type	Description
HRESULT	Returns S_OK if the method call was successful, otherwise an HRESULT as the error code.

7.1.4.13 NotifyAlarmListener

Processes a queue for an alarm listener.

Syntax

```
virtual HRESULT TCOMAPI NotifyMessageListener (ITcMessageListener ipListener)
```

Parameter

Name	Type	Description
ipListener	ITcMessageListener	Pointer to the ITcMessageListener interface.

Return value

Type	Description
HRESULT	Returns S_OK if the method call was successful, otherwise an HRESULT as the error code.

7.1.4.14 GetEventText

Returns a text for an event.

Syntax

```
virtual HRESULT TCOMAPI GetEventText (GUID eventClass, UDINT eventId, ITcSourceInfo ipSourceInfo, IT
cArguments ipArguments, DINT nLangId, ITcAsyncStringResult pipResult)
```

Parameter

Name	Type	Description
eventClass	REFERENCE TO GUID	Reference to the GUID of the event class.
eventId	REFERENCE TO UDINT	Reference to the ID of the event.
ipSourceInfo	ITcSourceInfo	Pointer to the ITcSourceInfo interface.
ipArguments	ITcArguments	Pointer to the ITcArguments interface.
nLangId	DINT	Language ID (LCID) of the requested language.
pipResult	POINTER TO ITcAsyncStringResult	Reference to an ITcAsyncStringResult pointer.

Return value

Type	Description
HRESULT	Returns S_OK if the method call was successful, otherwise an HRESULT as the error code.

7.1.4.15 GetEventClassName

Returns the class name for an event.

Syntax

```
virtual HRESULT TCOMAPI GetEventClassName (GUID eventClass, DINT nLangId, ITcAsyncStringResult pipResult)
```

Parameter

Name	Type	Description
eventClass	REFERENCE TO GUID	Reference to the GUID of the event class.
nLangId	DINT	Language ID (LCID) of the requested language.
pipResult	POINTER TO ITcAsyncStringResult	Reference to an ITcAsyncStringResult pointer.

Return value

Type	Description
HRESULT	Returns S_OK if the method call was successful, otherwise an HRESULT as the error code.

7.1.4.16 CreateArguments

Creates an instance that implements ITcArguments.

Syntax

```
virtual HRESULT TCOMAPI CreateArguments (ITcArguments ipArguments)
```

Parameter

Name	Type	Description
ipArguments	ITcArguments	Pointer to the ITcArguments interface.

Return value

Type	Description
HRESULT	Returns S_OK if the method call was successful, otherwise an HRESULT as the error code.

7.2 Data types

7.2.1 TcEventEntry

Defines an event by means of event class, event ID and severity.

Syntax

Definition:

```
typedef struct
{
    GUID            uuidEventClass;
    UDINT           nEventId;
    TcEventSeverity  eSeverity;
}TcEventEntry;
```

Parameter

Name	Type	Description
uuidEventClass	GUID	GUID of the event class.
nEventId	UDINT	ID of the event.
eSeverity	TcEventSeverity	Event severity defines the severity of the event,

7.2.2 TcEventSeverity

Defines the severity of the event.

Syntax

Definition:

```
typedef enum
{
    Verbose   = 0,
    Info      = 1,
    Warning   = 2,
    Error     = 3,
    Critical  = 4
}TcEventSeverity;
```

7.2.3 TcEventConfirmationState

Defines the confirmation state of an alarm.

Syntax

Definition:

```
typedef enum
{
   NotSupported        = 0,
   NotRequired         = 1,
   WaitForConfirmation = 2,
   Confirmed           = 3,
   Reset               = 4
}TcEventConfirmationState;
```

8 User mode API

The EventLogger provides an interface to both send messages and receive sent events from user mode programs.

Beckhoff.TwinCAT.TcEventLoggerAdsProxy.Net from NuGet.org

The API is available on NuGet.org via the package Beckhoff.TwinCAT.TcEventLoggerAdsProxy.Net for implementation into projects. To get started easily, you will find there sample code in the README file, which only needs to be copied into a .NET project.

COM based interface is replaced

The COM based interface, which is described here before, is supported up to TwinCAT 3.1 4024. Due to the technology used, it cannot be offered under TwinCAT/BSD.
The API provided on NuGet.org is the successor and provides easy porting for customer applications through an equivalent API.

8.1 Classes

8.1.1 TcEventLogger

This class represents the connection to a TwinCAT 3 Eventlogger.

Syntax

```
public class: ITcEventLogger2, _ITcEventLoggerEvents
```

Constructor

Initializes a new instance of the TcEventLogger class.

```
public TcEventLoggerClass();
```

Call:

```
TcEventLogger logger : new TcEventLogger();
```

Interfaces

Name	Description
ITcEventLogger2 [▶ 141]	Interface for sending commands to the EventLogger.
_ITcEventLoggerEvents [▶ 126]	This interface provides the messages for occurring events.

Methods

Name	Modifier	Return type	Definition location	Description
ClearAllAlarms [▶ 142]	public virtual	void	ITcEventLogger/ ITcEventLogger2 [▶ 141]	Calls Clear() for all alarms in the "Raised" state.
ClearLoggedEvents [▶ 142]	public virtual	void	ITcEventLogger/ ITcEventLogger2 [▶ 141]	Calls Clear() for all events that are currently in the "Raised" state.
ConfirmAllAlarms [▶ 142]	public virtual	void	ITcEventLogger/ ITcEventLogger2 [▶ 141]	Calls Confirm() for all alarms with the confirmation state "WaitForConfirmation".
Connect [▶ 143]	public virtual	void	ITcEventLogger/ ITcEventLogger2 [▶ 141]	Connects to the EventLogger of the TwinCAT system on a given system.
Disconnect [▶ 143]	public virtual	void	ITcEventLogger/ ITcEventLogger2 [▶ 141]	Cancels the connection to an EventLogger of the TwinCAT system.
GetEventClassName [▶ 143]	public virtual	string	ITcEventLogger/ ITcEventLogger2 [▶ 141]	Returns the class name for an event.
GetLoggedEvents [▶ 143]	public virtual	TcEventLoggerAdsProxyLib.TcLoggedEventCollection	ITcEventLogger/ ITcEventLogger2 [▶ 141]	Queries the events located in the cache.
ITcEventLogger2_SendTcMessage [▶ 121]	public virtual	void	ITcEventLogger2 [▶ 141]	Sends a message.

events

Name	Return type	Definition location	Description
AlarmCleared	_ITcEventLoggerEvents_AlarmClearedEventHandler	_ITcEventLoggerEvents [▶ 126]	Called if an alarm changes to the "Not Raised" state.
AlarmConfirmed	_ITcEventLoggerEvents_AlarmConfirmedEventHandler	_ITcEventLoggerEvents [▶ 126]	Called if an alarm changes to the "Confirmed" state.
AlarmRaised	_ITcEventLoggerEvents_AlarmRaisedEventHandler	_ITcEventLoggerEvents [▶ 126]	Called if an alarm changes to the "Raised" state.
MessageSent	_ITcEventLoggerEvents_MessageSentEventHandler	_ITcEventLoggerEvents [▶ 126]	Called when a message has been sent.

Properties

Name	Modifier	Type	Access	Definition location	Description
ActiveAlarms [▶ 144]	public virtual	TcAlarmCollection	get	ITcEventLogger/ ITcEventLogger2 [▶ 141]	Returns a collection with all currently active alarms.
IsConnected [▶ 144]	public virtual	bool	get	ITcEventLogger/ ITcEventLogger2 [▶ 141]	Represents the connection that was established by means of Connect. Should be checked regularly.

8.1.1.1 ITcEventLogger_ClearAllAlarms

This method sets all alarms that are in the "Raised" state to "Not Raised".

Syntax

```
public virtual void ITcEventLogger_ClearAllAlarms([bool bResetConfirmation = True])
```

Parameter

Name	Type	Description
bResetConfirmation	bool	Determines whether the confirmations for alarms should be triggered.

8.1.1.2 ITcEventLogger_ClearLoggedEvents

This method clears the events cache.
The current alarm states are not changed by this.

Syntax

```
public virtual void ITcEventLogger_ClearLoggedEvents()
```

8.1.1.3 ITcEventLogger_ConfirmAllAlarms

This method confirms all alarms that have to be confirmed, i.e. those that are in the "WaitForConfirmation" state.

Syntax

```
public virtual void ITcEventLogger_ConfirmAllAlarms()
```

8.1.1.4 ITcEventLogger_Connect

This method connects the object with an EventLogger on a runtime system on the basis of the AmsNetId.

Syntax

```
public virtual void ITcEventLogger_Connect([string Address = localhost])
```

Parameter

Name	Type	Description
Address	string	Defines the AmsNetId.

8.1.1.5 ITcEventLogger_Disconnect

This method disconnects the object from the EventLogger.

Syntax

```
public virtual void ITcEventLogger_Disconnect()
```

8.1.1.6 ITcEventLogger_GetEventClassName

This method returns the EventClass name that matches a given EventClass GUID.

Syntax

```
public virtual string ITcEventLogger_GetEventClassName(System.Guid EventClass, int nLangId)
```

Parameter

Name	Type	Description
EventClass	System.Guid	GUID of the event class.
langId	int	LangID in which the name is to be returned.

Return value

Name	Type	Description
ITcEventLogger_GetEventClassName	string	Name of the EventClass.

8.1.1.7 ITcEventLogger_GetLoggedEvents

This method returns a collection of stored latest events.

Syntax

```
public virtual TcLoggedEventCollection ITcEventLogger_GetLoggedEvents(uint nMaxEntiries)
```

Parameter

Name	Type	Description
nMaxEntiries	uint	Maximum number of events to be returned.

Return value

Name	Type	Description
ITcEventLogger_GetLoggedEvents	TcLoggedEventCollection	A collection of the latest events.

8.1.1.8 ITcEventLogger_GetText

This method returns the text of an event.

Syntax

```
public virtual string ITcEventLogger_GetText(System.Guid EventClass, uint EventId, uint objectId,
TcEventLoggerAdsProxyLib.TcEventArgumentsInfo pArgInfo, System.IntPtr pArgData, int nLangId)
```

Parameter

Name	Type	Description
EventClass	System.Guid	GUID of the event class.
EventId	uint	ID of the event
objectId	uint	
pArgInfo	TcEventLoggerAdsProxyLib.TcEventArgumentsInfo	
pArgData	System.IntPtr	
langId	int	LangID in which the name is to be returned.

Return value

Name	Type	Description
ITcEventLogger_GetText	string	Text of the event.

8.1.1.9 ITcEventLogger_ActiveAlarms

This property box returns a collection of the currently active alarms (active state is "Raised" or "WaitingForConfirmation").

Syntax

```
public virtual TcEventLoggerAdsProxyLib.TcAlarmCollection ITcEventLogger_ActiveAlarms
```

8.1.1.10 ITcEventLogger_IsConnected

This property indicates whether the TcEventLogger object is currently connected to a target system. This should be checked regularly in order to be able to react to a loss of connection.

Syntax

```
public virtual bool ITcEventLogger_IsConnected
```

8.1.1.11 ITcEventLogger2_SendTcMessage

This method sends a message.

Syntax

```
public virtual void SendTcMessage(System.Guid EventClass, uint EventId,
TcEventLoggerAdsProxyLib.SeverityLevelEnum severity, string JsonAttribute,
TcEventLoggerAdsProxyLib.TcSourceInfo pSourceInfo, TcEventLoggerAdsProxyLib.TcArguments pArguments)
```

Parameter

Name	Type	Description
EventClass	System.Guid	GUID of the event classes
EventId	uint	ID of the event
Severity	TcEventLoggerAdsProxyLib.SeverityLevelEnum [▶ 155]	Severity of the event
JsonAttribute	string	JSON attribute
pSourceInfo	TcEventLoggerAdsProxyLib.TcSourceInfo	Reference to the source information
pArguments	TcArguments	Reference to the arguments.

8.1.2 TcArguments

Arguments of an event can be defined with this class. The ITcArguments interface is implemented for this.

Syntax

```
public class: ITcArguments
```

Constructor

Initializes a new instance of the class TcArguments.

```
public TcArguments();
```

Call:

```
TcArguments args : new TcArguments();
```

Interfaces

Name	Description
ITcArguments	Interface for describing the arguments.

Methods

Name	Modifier	Return type	Definition location	Description
Add [▶ 122]	public virtual	void	ITcEventLogger/ ITcArguments	Adds an argument.
AddV [▶ 123]	public virtual	void	ITcEventLogger/ ITcArguments	Adds an array of arguments.
Clear [▶ 123]	public virtual	void	ITcEventLogger/ ITcArguments	Deletes all arguments.
GetEnumerator [▶ 123]	public virtual	System.Collections.IEnumerator	ITcEventLogger/ ITcArguments	Returns an enumeration of the arguments.
Remove [▶ 123]	public virtual	void	ITcEventLogger/ ITcArguments	Deletes an argument.
Set [▶ 136]	public virtual	void	ITcEventLogger/ ITcArguments	Sets an argument.

Properties

Name	Modifier	Type	Access	Definition location	Description
Count [▶ 124]	public virtual	int	get	ITcEventLogger/ ITcArguments	number of arguments.
this [uint]	public virtual	ITcLoggedEvent	get	ITcEventLogger/ ITcArguments	The event to which the arguments relate.

8.1.2.1 Add

This method adds an argument.

Syntax

```
public virtual void Add(object Item)
```

Parameter

Name	Type	Description
Item	object	The argument to be added.

8.1.2.2 AddV

This method adds an array of arguments.

Syntax

```
public virtual void AddV(params object[] args)
```

Parameter

Name	Type	Description
args	params object[]	Array of the arguments to be added.

8.1.2.3 Clear

This method deletes all arguments.

Syntax

```
public virtual void Clear()
```

8.1.2.4 GetEnumerator

This method returns an enumeration of the arguments.

Syntax

```
public virtual System.Collections.IEnumerator GetEnumerator()
```

Return value

Name	Type	Description
GetEnumerator	System.Collections.IEnumerator	Enumeration of the arguments.

8.1.2.5 Remove

This method deletes an argument.

Syntax

```
public virtual void Remove(uint Index)
```

Parameter

Name	Type	Description
Index	uint	Index of the arguments to be deleted.

8.1.2.6 Set

This method sets an argument.

Syntax

```
public virtual void Set(uint Index, object Item)
```

Parameter

Name	Type	Description
Index	uint	Index of the argument to be set.
Item	object	The new argument.

8.1.2.7 Count

This property returns the number.

Syntax

```
public virtual int Count
```

8.1.2.8 this [uint]

This property is the event to which the arguments refer.

Syntax

```
public virtual TcEventLoggerAdsProxyLib.TcArgumentEntry this[uint Index]
```

8.1.3 TcSourceInfo

This class describes the source information of an event.

Syntax

```
public class: ITcSourceInfo
```

Constructor

Initializes a new instance of the class TcSourceInfo.

```
public TcArguments();
```

Call:

```
TcSourceInfo souceInfo: new SourceInfo();
```

Methods

Name	Modifier	Return type	Definition location	Description
IsSourceInfoTypeSupported [▶ 125]	public virtual	bool	ITcEventLogger/ ITcSourceInfo	Query option for the type of source information.

Properties

Name	Modifier	Type	Access	Definition location	Description
Count [▶ 125]	public virtual	int	get	ITcEventLogger/ ITcSourceInfo	Number of SourceInfo types.
Guid [▶ 125]	public virtual	System.Guid	get, set	ITcEventLogger/ ITcSourceInfo	GUID of the source.
Id [▶ 125]	public virtual	uint	get, set	ITcEventLogger/ ITcSourceInfo	Id of the source.
Name [▶ 126]	public virtual	string	get, set	ITcEventLogger/ ITcSourceInfo	Name of the source.

8.1.3.1 GetData

This method returns the data for a source.

Syntax

```
public virtual void GetData(uint Index, out TcEventLoggerAdsProxyLib.TcSourceInfoTypeEnum pInfoType,
 System.IntPtr pData, out uint cbData)
```

Parameter

Name	Type	Description
Index	uint	Index of the source.
pInfoType	TcEventLoggerAdsProxyLib.TcSourceInfoTypeEnum [▶ 156]	Reference to the type information.
pData	System.IntPtr	Reference to the data.
cbData	uint	Length of the data.

8.1.3.2 IsSourceInfoTypeSupported

This method can be used to check whether the source information type has been defined.

Syntax

```
public virtual bool IsSourceInfoTypeSupported(TcSourceInfoTypeEnum infoType)
```

Parameter

Name	Type	Description
infoType	TcSourceInfoTypeEnum [▶ 156]	Query type of source

Return value

Name	Type	Description
IsSourceInfoTypeSupported	bool	Information as to whether supported.

8.1.3.3 Count

This property returns the number.

Syntax

```
public virtual int Count
```

8.1.3.4 Guid

This property returns the GUID of the source.

Syntax

```
public virtual System.Guid Guid
```

8.1.3.5 Id

This property returns the Id.

Syntax

```
public virtual uint Id
```

8.1.3.6 Name

This property returns the name.

Syntax

```
public virtual string Name
```

8.2 Interfaces

8.2.1 _ITcEventLoggerEvents

This interface provides the messages for occurring events.

Syntax

```
public interface _ITcEventLoggerEvents
```

Methods

Name	Modifier	Return type	Description
AlarmCleared [▶ 126]	public virtual	void	Called if an alarm changes to the "Not Raised" state.
AlarmConfirmed [▶ 126]	public virtual	void	Called if an alarm changes to the "Confirmed" state.
AlarmRaised [▶ 127]	public virtual	void	Called if an alarm changes to the "Raised" state.
MessageSend [▶ 127]	public virtual	void	Called when a message has been sent.

8.2.1.1 AlarmCleared

Called if an alarm changes to the "Not Raised" state.

Syntax

```
public virtual void AlarmCleared(TcAlarm evtObj, bool bRemove)
```

Parameter

Name	Type	Description
evtObj	TcAlarm	The alarm
bRemove	bool	TRUE = the alarm is in the "Not Raised" state and not in the "Wait for Confirmation" state. FALSE = the alarm is still waiting for a confirmation.

8.2.1.2 AlarmConfirmed

Called if an alarm changes to the "Confirmed" state.

Syntax

```
public virtual void AlarmConfirmed(TcAlarm evtObj, bool bRemove)
```

Parameter

Name	Type	Description
evtObj	TcAlarm	The alarm
bRemove	bool	TRUE if the alarm is not in the "Raised" state. FALSE if the alarm is in the "Raised" state.

8.2.1.3 AlarmRaised

Called if an alarm changes to the "Raised" state.

Syntax

```
public virtual void AlarmRaised(TcAlarm evtObj)
```

Parameter

Name	Type	Description
evtObj	TcAlarm	The alarm

8.2.1.4 MessageSend

Called when a message has been sent.

Syntax

```
void MessageSent(TcMessage evtObj)
```

Parameter

Name	Type	Description
evtObj	TcMessage	The message

8.2.2 ITcAlarm3

This interface represents an alarm.

Syntax

```
public interface ITcAlarm3
```

Methods

Name	Modifier	Return type	Description
Confirm [▶ 128]	public virtual	void	Confirms the alarm.
GetCauseRemedy [▶ 130]	Public virtual	TcCauseRemedyCollection	Returns information on the cause and remedy (if these are defined).
GetDetails [▶ 130]	Public virtual	TcDetailCollection	Returns the details.
GetEventClassName [▶ 129]	public virtual	string	Returns the event class name.
GetText [▶ 129]	public virtual	string	Returns the event text, including the arguments.
IsSourceInfoTypeSupported [▶ 130]	public virtual	bool	Query option for the type of source information.

Properties

Name	Modifier	Type	Access	Description
ConfirmationState [▶ 130]	public virtual	ConfirmationStateEnum	get	Returns the Confirmation State [▶ 155].
EventClass [▶ 131]	public virtual	System.Guid	get	Returns the EventClass GUID.
EventId [▶ 131]	public virtual	uint	get	Returns the event ID.
EventType [▶ 131]	public virtual	EventTypeEnum	get	Returns the type of the event [▶ 155].
FileTimeCleared [▶ 131]	public virtual	long	get	Timestamp when the alarm was changed to the Cleared state.
FileTimeConfirmed [▶ 131]	public virtual	long	get	Timestamp when the alarm was confirmed.
FileTimeRaised [▶ 131]	public virtual	long	get	Timestamp when the alarm was changed to the Raised state.
IsRaised [▶ 131]	public virtual	bool	get	Indicates whether the alarm has been changed to the Raised state.
JsonAttribute [▶ 132]	public virtual	string	get	The JSON attribute.
SeverityLevel [▶ 132]	public virtual	SeverityLevelEnum	get	The Severity Level [▶ 155].
SourceGuid [▶ 132]	public virtual	System.Guid	get	Returns the GUID of the source.
SourceId [▶ 132]	public virtual	uint	get	Returns the ID of the source.
SourceName [▶ 132]	public virtual	string	get	Returns the name of the source.
TimeCleared [▶ 132]	public virtual	System.DateTime	get	Timestamp when the alarm was changed to the Cleared state.
TimeConfirmed [▶ 132]	public virtual	System.DateTime	get	Timestamp when the alarm was confirmed.
TimeRaised [▶ 132]	public virtual	Sytem.DateTime	get	Timestamp when the alarm was changed to the Raised state.

8.2.2.1 Confirm

This method confirms the alarm.

Syntax

```
public virtual void Confirm()
```

8.2.2.2 GetArgumentData

This method returns the data of the arguments.

Syntax

```
public virtual void GetArgumentData(ref System.IntPtr ppArgData, uint size)
```

Parameter

Name	Type	Description
ppArgData	ref System.IntPtr	Reference to the data.
size	Uint	Size

8.2.2.3 GetArgumentInfo

This method returns the (type) information for the arguments.

Syntax

```
public virtual void GetArgumentInfo(ref TcEventArgumentsInfo pArgInfo)
```

Parameter

Name	Type	Description
pArgInfo	ref TcEventArgumentsInfo	Reference to the information provided.

8.2.2.4 GetEventClassName

This method returns the EventClass name.

Syntax

```
public virtual string GetEventClassName(int langId)
```

Parameter

Name	Type	Description
langId	int	LangID of the language

Return value

Name	Type	Description
GetEventClassName	string	Names of the EventClass

8.2.2.5 GetText

This method returns the event text, including the arguments.

Syntax

```
public virtual string GetText(int langId)
```

Parameter

Name	Type	Description
langId	int	LangID of the language

Return value

Name	Type	Description
GetText	string	Event text

8.2.2.6 IsSourceInfoTypeSupported

This method can be used to check whether the source information type has been defined.

Syntax

```
public virtual bool IsSourceInfoTypeSupported(TcSourceInfoTypeEnum infoType)
```

Parameter

Name	Type	Description
infoType	TcSourceInfoTypeEnum [▶ 156]	Query type of source

Return value

Name	Type	Description
IsSourceInfoTypeSupported	bool	Information as to whether supported.

8.2.2.7 GetCauseRemedy

This method returns the cause/remedy information, if this has been defined.

Syntax

```
public virtual TcCauseRemedyCollection GetCauseRemedy(int langId)
```

Parameter

Name	Type	Description
langId	int	LangID of the language.

Return value

Name	Type	Description
GetCauseRemedy	TcCauseRemedyCollection	Collection of the cause/remedy information.

8.2.2.8 ConfirmationState

This property returns the Confirmation State.

Syntax

```
Public virtual ConfirmationStateEnum ConfirmationState
```

8.2.2.9 GetDetails

This method returns the details.

Syntax

```
public virtual TcDetailCollection GetDetails(int langId)
```

Parameter

Name	Type	Description
langId	int	LangID of the language

Return value

Name	Type	Description
GetDetails	TcDetailCollection	Collection of the details

8.2.2.10 EventClass

This property returns the EventClass GUID.

Syntax

```
public virtual System.Guid EventClass
```

8.2.2.11 EventId

This property returns the event ID.

Syntax

```
public virtual uint EventId
```

8.2.2.12 EventType

This property returns the type of the event.

Syntax

```
public virtual EventTypeEnum EventType
```

8.2.2.13 FileTimeCleared

This property returns the timestamp when the alarm was changed to the Cleared state.

Syntax

```
public virtual long FileTimeCleared
```

8.2.2.14 FileTimeConfirmed

This property returns the timestamp when the alarm was changed to the Confirmed state.

Syntax

```
public virtual long FileTimeConfirmed
```

8.2.2.15 FileTimeRaised

This property returns the timestamp when the alarm was changed to the Raised state.

Syntax

```
public virtual long FileTimeRaised
```

8.2.2.16 IsRaised

This property indicates whether the alarm has been changed to the Raised state.

Syntax

```
public virtual bool IsRaised
```

8.2.2.17 JsonAttribute

This property returns the JSON attribute.

Syntax

```
public virtual string JsonAttribute
```

8.2.2.18 SeverityLevel

This property returns the Severity level.

Syntax

```
public virtual TcEventLoggerAdsProxyLib.SeverityLevelEnum SeverityLevel
```

8.2.2.19 SourceGuid

This property returns the GUID of the source.

Syntax

```
public virtual System.Guid SourceGuid
```

8.2.2.20 SourceId

This property returns the ID of the source.

Syntax

```
public virtual uint SourceId
```

8.2.2.21 SourceName

This property returns the name of the source.

Syntax

```
public virtual string SourceName
```

8.2.2.22 TimeCleared

This property returns the timestamp when the alarm was changed to the Cleared state.

Syntax

```
public virtual System.DateTime TimeCleared
```

8.2.2.23 TimeConfirmed

This property returns the timestamp when the alarm was changed to the Confirmed state.

Syntax

```
public virtual System.DateTime TimeConfirmed
```

8.2.2.24 TimeRaised

This property returns the timestamp when the alarm was changed to the Raised state.

Syntax

```
public virtual System.DateTime TimeRaised
```

8.2.3 ITcArgumentEntry

This interface describes an argument.

Syntax

```
public interface ITcArgumentEntry
```

Methods

Name	Modifier	Return type	Description
Get [▶ 133]	public virtual	dynamic	Returns the value as a dynamic type.
GetBoolean [▶ 133]	public virtual	bool	Returns the value as Boolean.
GetDouble [▶ 134]	public virtual	void	Returns the value as Double.
GetFloat [▶ 134]	public virtual	float	Returns the value as Float.
GetInt16 [▶ 134]	public virtual	short	Returns the value as Short.
GetInt32 [▶ 135]	public virtual	int	Returns the value as Int.
GetInt64 [▶ 135]	public virtual	long	Returns the value as Long.
GetInt8 [▶ 135]	public virtual	sbyte	Returns the value as SByte.
GetString [▶ 135]	public virtual	string	Returns the value as String.
GetUInt16 [▶ 135]	public virtual	ushort	Returns the value as UShort.
GetUInt32 [▶ 136]	public virtual	uint	Returns the value as UInt.
GetUInt64 [▶ 136]	public virtual	ulong	Returns the value as ULong.
GetUInt8 [▶ 136]	public virtual	byte	Returns the value as Byte.
Set [▶ 136]	public virtual	void	Sets the value.

8.2.3.1 Get

This method returns the value as a dynamic type.

Syntax

```
public virtual Object Get()
```

Return value

Name	Type	Description
Get	Object	Return is an object of the type corresponding to the argument.

8.2.3.2 GetBoolean

This method returns the value as Boolean.

Syntax

```
public virtual bool GetBoolean()
```

Return value

Name	Type	Description
GetBoolean	bool	Return

8.2.3.3 GetData

This method returns the value as a reference to data.

Syntax

```
public virtual void GetData(out TcEventArgumentTypeEnum pInfoType, System.IntPtr pData, out uint cbData)
```

Parameter

Name	Type	Description
pInfoType	TcEventArgumentTypeEnum [▶ 156]	Reference to the type of data.
pData	System.IntPtr	Reference to the data.
cbData	uint	Length of the data.

8.2.3.4 GetDouble

This method returns the value as Double.

Syntax

```
public virtual double GetDouble
```

Return value

Name	Type	Description
GetDouble	double	Return

8.2.3.5 GetFloat

This method returns the value as Float.

Syntax

```
public virtual float GetFloat()
```

Return value

Name	Type	Description
GetFloat	float	Return

8.2.3.6 GetInt16

This method returns the value as Short.

Syntax

```
public virtual short GetInt16()
```

Return value

Name	Type	Description
GetInt16	short	Return

8.2.3.7 GetInt32

This method returns the value as Int.

Syntax

```
public virtual int GetInt32()
```

Return value

Name	Type	Description
GetInt32	int	Return

8.2.3.8 GetInt64

This method returns the value as Long.

Syntax

```
public virtual long GetInt64()
```

Return value

Name	Type	Description
GetInt64	long	Return

8.2.3.9 GetInt8

This method returns the value as SByte.

Syntax

```
public virtual sbyte GetInt8()
```

Return value

Name	Type	Description
GetInt8	sbyte	Return

8.2.3.10 GetString

This method returns the value as String.

Syntax

```
public virtual string GetString()
```

Return value

Name	Type	Description
GetString	string	Return

8.2.3.11 GetUInt16

This method returns the value as UShort.

Syntax

```
public virtual ushort GetUInt16()
```

Return value

Name	Type	Description
GetUInt16	ushort	Return

8.2.3.12 GetUInt32

This method returns the value as UInt.

Syntax

```
public virtual uint GetUInt32()
```

Return value

Name	Type	Description
GetUInt32	uint	Return

8.2.3.13 GetUInt64

This method returns the value as ULong.

Syntax

```
public virtual ulong GetUInt64()
```

Return value

Name	Type	Description
GetUInt64	ulong	Return

8.2.3.14 GetUInt8

This method returns the value as Byte.

Syntax

```
public virtual byte GetUInt8()
```

Return value

Name	Type	Description
GetUInt8	byte	Return

8.2.3.15 Set

This method sets the value.

Syntax

```
public virtual void Set(object Item)
```

Parameter

Name	Type	Description
Item	object	New argument. An object of the type corresponding to the argument.

8.2.4 ITcCauseRemedy

This interface describes the cause/remedy information for an event.

Syntax

```
public interface ITcCauseRemedy
```

Properties

Name	Modifier	Type	Access	Description
Cause [▶ 137]	public virtual	string	get	The cause.
Id [▶ 137]	public virtual	uint	get	The ID.
Remedy [▶ 137]	public virtual	string	get	The remedy.

8.2.4.1 Cause

This property returns the cause.

Syntax

```
public virtual string Cause
```

8.2.4.2 Id

This property returns the Id.

Syntax

```
public virtual uint Id
```

8.2.4.3 Remedy

This property returns the remedy.

Syntax

```
public virtual string Remedy
```

8.2.5 ITcDetail

This interface describes the details of an event.

Syntax

```
public interface ITcDetail
```

Properties

Name	Modifier	Type	Access	Description
Comment [▶ 137]	Public virtual	string	get	The comment
Name [▶ 138]	public virtual	string	get	The name
text [▶ 138]	Public virtual	string	get	The text

8.2.5.1 Comment

This property returns the comment on a detail.

Syntax

```
public virtual string Comment
```

8.2.5.2 Name

This property returns the name.

Syntax

```
public virtual string Name
```

8.2.5.3 text

This property returns the text for a detail.

Syntax

```
public virtual string text
```

8.2.6 ITcEvent

This interface describes an event. This event can be either a message or an alarm.

Syntax

```
public interface ITcEvent
```

Methods

Name	Modifier	Return type	Description
GetArgumentData [▶ 139]	public virtual	void	Returns the data of the arguments.
GetArgumentInfo [▶ 139]	public virtual	void	Returns the (type) information for the arguments.
GetEventClassName [▶ 139]	public virtual	string	Returns the event class name.
GetText [▶ 140]	public virtual	string	Returns the event text, including the arguments.
IsSourceInfoTypeSupported [▶ 140]	public virtual	bool	Query option for the type of source information.

Properties

Name	Modifier	Type	Access	Description
EventClass [▶ 140]	public virtual	System.Guid	get	Returns the EventClass GUID.
EventId [▶ 140]	public virtual	uint	get	Returns the event ID.
EventType [▶ 141]	public virtual	EventTypeEnum	get	Returns the type of the event [▶ 155].
FileTimeRaised [▶ 141]	public virtual	long	get	Timestamp when the event was sent.
SeverityLevel [▶ 141]	public virtual	SeverityLevelEnum	get	The Severity Level [▶ 155].
SourceGuid [▶ 141]	public virtual	System.Guid	get	Returns the GUID of the source.
SourceId [▶ 141]	public virtual	uint	get	Returns the ID of the source.
SourceName [▶ 141]	public virtual	string	get	Returns the name of the source.
TimeRaised [▶ 141]	public virtual	System.DateTime	get	Timestamp when the event was sent.

8.2.6.1 GetArgumentData

This method returns the data of the arguments.

Syntax

```
public virtual void GetArgumentData(ref System.IntPtr ppArgData, uint size)
```

Parameter

Name	Type	Description
ppArgData	ref System.IntPtr	Reference to the data.
size	Uint	Size

8.2.6.2 GetArgumentInfo

This method returns the (type) information for the arguments.

Syntax

```
public virtual void GetArgumentInfo(ref TcEventArgumentsInfo pArgInfo)
```

Parameter

Name	Type	Description
pArgInfo	ref TcEventArgumentsInfo	Reference to the information provided.

8.2.6.3 GetEventClassName

This method returns the EventClass name.

Syntax

```
public virtual string GetEventClassName(int langId)
```

Parameter

Name	Type	Description
langId	int	LangID of the language

Return value

Name	Type	Description
GetEventClassName	string	Names of the EventClass

8.2.6.4 GetText

This method returns the event text, including the arguments.

Syntax

```
public virtual string GetText(int langId)
```

Parameter

Name	Type	Description
langId	int	LangID of the language

Return value

Name	Type	Description
GetText	string	Event text

8.2.6.5 IsSourceInfoTypeSupported

This method can be used to check whether the source information type has been defined.

Syntax

```
public virtual bool IsSourceInfoTypeSupported(TcSourceInfoTypeEnum infoType)
```

Parameter

Name	Type	Description
infoType	TcSourceInfoTypeEnum [▶ 156]	Query type of source

Return value

Name	Type	Description
IsSourceInfoTypeSupported	bool	Information as to whether supported.

8.2.6.6 EventClass

This property returns the EventClass GUID.

Syntax

```
public virtual System.Guid EventClass
```

8.2.6.7 EventId

This property returns the event ID.

Syntax

```
public virtual uint EventId
```

8.2.6.8 EventType

This property returns the type of the event.

Syntax

```
public virtual EventTypeEnum EventType
```

8.2.6.9 FileTimeRaised

This property returns the timestamp when the alarm was changed to the Raised state.

Syntax

```
public virtual long FileTimeRaised
```

8.2.6.10 SeverityLevel

This property returns the Severity level.

Syntax

```
public virtual TcEventLoggerAdsProxyLib.SeverityLevelEnum SeverityLevel
```

8.2.6.11 SourceGuid

This property returns the GUID of the source.

Syntax

```
public virtual System.Guid SourceGuid
```

8.2.6.12 SourceId

This property returns the ID of the source.

Syntax

```
public virtual uint SourceId
```

8.2.6.13 SourceName

This property returns the name of the source.

Syntax

```
public virtual string SourceName
```

8.2.6.14 TimeRaised

This property returns the timestamp when the alarm was changed to the Raised state.

Syntax

```
public virtual System.DateTime TimeRaised
```

8.2.7 ITcEventLogger2

Interface for sending commands to the EventLogger.

Syntax

```
public interface ITcEventLogger2
```

Methods

Name	Modifier	Return type	Description
ClearAllAlarms [▶ 142]	public virtual	void	Calls Clear() for all alarms in the "Raised" state.
ClearLoggedEvents [▶ 142]	public virtual	void	Clears the cache.
ConfirmAllAlarms [▶ 142]	public virtual	void	Calls Confirm() for all alarms with the confirmation state "WaitForConfirmation".
Connect [▶ 143]	public virtual	void	Connects to the EventLogger of the TwinCAT system on a given system.
Disconnect [▶ 143]	public virtual	void	Cancels the connection to an EventLogger of the TwinCAT system.
GetEventClassName [▶ 143]	public virtual	string	Returns the class name for an event.
GetLoggedEvents [▶ 143]	public virtual	TcLoggedEventCollection	Queries the events located in the cache.

Properties

Name	Modifier	Type	Access	Description
ActiveAlarms [▶ 144]	public virtual	TcAlarmCollection	get	Returns a collection with all currently active alarms.
IsConnected [▶ 144]	public virtual	bool	get	Represents the connection that was established by means of Connect. Should be checked regularly.

8.2.7.1 ClearAllAlarms

Calls Clear() for all alarms in the "Raised" state.

Syntax

```
public virtual void ClearAllAlarms([bool bResetConfirmation = True])
```

Parameter

Name	Type	Description
bResetConfirmation	bool	Indicates whether the alarms should also be changed to the "Confirmed" state.

8.2.7.2 ClearLoggedEvents

Clears the cache.

Syntax

```
Public virtual void ClearLoggedEvents()
```

8.2.7.3 ConfirmAllAlarms

Calls Confirm() for all alarms with the confirmation state "WaitForConfirmation".

Syntax

```
public virtual void ConfirmAllAlarms()
```

8.2.7.4 Connect

Connects to the EventLogger of the TwinCAT system on a given system.

Syntax

```
public virtual void Connect([string Address = localhost])
```

Parameter

Name	Type	Description
Adress	string	AMS Net ID

8.2.7.5 Disconnect

Cancels the connection to an EventLogger of the TwinCAT system.

Syntax

```
public virtual void Disconnect()
```

8.2.7.6 GetEventClassName

Returns the class name for an event.

Syntax

```
public virtual string GetEventClassName(System.Guid EventClass, int nLangId)
```

Parameter

Name	Type	Description
langId	int	LangID for the language
EventClass	System.Guid	GUID of the event class

Return value

Name	Type	Description
GetEventClassName	string	

8.2.7.7 GetLoggedEvents

Queries the events located in the cache.

Syntax

```
public virtual TcEventLoggerAdsProxyLib.TcLoggedEventCollection GetLoggedEvents(uint nMaxEntiries)
```

Parameter

Name	Type	Description
nMaxEntiries	uint	Number of events to be returned.

Return value

Name	Type	Description
GetLoggedEvents	TcEventLoggerAdsProxyLib.TcLoggedEventCollection	

8.2.7.8 ActiveAlarms

Returns a collection with all currently active alarms.

Syntax

```
public virtual TcEventLoggerAdsProxyLib.TcAlarmCollection ActiveAlarms
```

8.2.7.9 IsConnected

Represents the connection that was established by means of Connect. Should be checked regularly.

Syntax

```
public virtual bool IsConnected
```

8.2.7.10 ITcEventLogger2_SendTcMessage

This method sends a message.

Syntax

```
public virtual void SendTcMessage(System.Guid EventClass, uint EventId,
TcEventLoggerAdsProxyLib.SeverityLevelEnum severity, string JsonAttribute,
TcEventLoggerAdsProxyLib.TcSourceInfo pSourceInfo, TcEventLoggerAdsProxyLib.TcArguments pArguments)
```

Parameter

Name	Type	Description
EventClass	System.Guid	GUID of the event classes
EventId	uint	ID of the event
Severity	TcEventLoggerAdsProxyLib.SeverityLevelEnum [▶ 155]	Severity of the event
JsonAttribute	string	JSON attribute
pSourceInfo	TcEventLoggerAdsProxyLib.TcSourceInfo	Reference to the source information
pArguments	TcArguments	Reference to the arguments.

8.2.8 ITcLoggedEvent4

This interface describes a stored event. This event can be either a message or an alarm.

Syntax

```
public interface ITcLoggedEvent4
```

Methods

Name	Modifier	Return type	Description
GetArgumentData [▶ 146]	public virtual	void	Returns the data of the arguments.
GetArgumentInfo [▶ 147]	public virtual	void	Returns the (type) information for the arguments.
GetCauseRemedy [▶ 147]	public virtual	TcCauseRemedyCollection	Returns the cause/ remedy information.
GetDetails [▶ 147]	public virtual	TcDetailCollection	Returns the details.
GetEventClassName [▶ 148]	public virtual	string	Returns the event class name.
GetText [▶ 148]	public virtual	string	Returns the event text, including the arguments.
IsSourceInfoTypeSupported [▶ 148]	public virtual	bool	Query option for the type of source information.

Properties

Name	Modifier	Type	Access	Description
EventClass [▶ 148]	public virtual	System.Guid	get	Returns the EventClass GUID.
EventId [▶ 149]	public virtual	uint	get	Returns the event ID.
EventType [▶ 149]	public virtual	EventTypeEnum	get	Returns the type of the event [▶ 155].
FileTimeCleared [▶ 149]	public virtual	long	get	Only alarm: Timestamp when the alarm was changed to the Cleared state.
FileTimeConfirmed [▶ 149]	public virtual	long	get	Only alarm: Timestamp when the alarm was confirmed.
FileTimeRaised [▶ 149]	public virtual	long	get	Timestamp when the event was sent / the alarm was changed to the Raised state.
JsonAttribute [▶ 149]	public virtual	string	get	The JSON attribute.
SeverityLevel [▶ 149]	public virtual	SeverityLevelEnum	get	The Severity Level [▶ 155].
SourceGuid [▶ 149]	public virtual	System.Guid	get	Returns the GUID of the source.
SourceId [▶ 150]	public virtual	uint	get	Returns the ID of the source.
SourceName [▶ 150]	public virtual	string	get	Returns the name of the source.
TimeCleared [▶ 150]	public virtual	System.DateTime	get	Only alarm: Timestamp when the alarm was changed to the Cleared state.
TimeConfirmed [▶ 150]	public virtual	System.DateTime	get	Only alarm: Timestamp when the alarm was confirmed.
TimeRaised [▶ 150]	public virtual	System.DateTime	get	Timestamp when the event was sent / the alarm was changed to the Raised state.
WithConfirmation [▶ 150]	public virtual	bool	get	Only alarm: A confirmation is necessary.

8.2.8.1 GetArgumentData

This method returns the data of the arguments.

Syntax

```
public virtual void GetArgumentData(ref System.IntPtr ppArgData, uint size)
```

Parameter

Name	Type	Description
ppArgData	ref System.IntPtr	Reference to the data.
size	Uint	Size

8.2.8.2 GetArgumentInfo

This method returns the (type) information for the arguments.

Syntax

```
public virtual void GetArgumentInfo(ref TcEventArgumentsInfo pArgInfo)
```

Parameter

Name	Type	Description
pArgInfo	ref TcEventArgumentsInfo	Reference to the information provided.

8.2.8.3 GetCauseRemedy

This method returns the cause/remedy information, if this has been defined.

Syntax

```
public virtual TcCauseRemedyCollection GetCauseRemedy(int langId)
```

Parameter

Name	Type	Description
langId	int	LangID of the language.

Return value

Name	Type	Description
GetCauseRemedy	TcCauseRemedyCollection	Collection of the cause/remedy information.

8.2.8.4 GetDetails

This method returns the details.

Syntax

```
public virtual TcDetailCollection GetDetails(int langId)
```

Parameter

Name	Type	Description
langId	int	LangID of the language

Return value

Name	Type	Description
GetDetails	TcDetailCollection	Collection of the details

8.2.8.5 GetEventClassName

This method returns the EventClass name.

Syntax

```
public virtual string GetEventClassName(int langId)
```

Parameter

Name	Type	Description
langId	int	LangID of the language

Return value

Name	Type	Description
GetEventClassName	string	Names of the EventClass

8.2.8.6 GetText

This method returns the event text, including the arguments.

Syntax

```
public virtual string GetText(int langId)
```

Parameter

Name	Type	Description
langId	int	LangID of the language

Return value

Name	Type	Description
GetText	string	Event text

8.2.8.7 IsSourceInfoTypeSupported

This method can be used to check whether the source information type has been defined.

Syntax

```
public virtual bool IsSourceInfoTypeSupported(TcSourceInfoTypeEnum infoType)
```

Parameter

Name	Type	Description
infoType	TcSourceInfoTypeEnum [▶ 156]	Query type of source

Return value

Name	Type	Description
IsSourceInfoTypeSupported	bool	Information as to whether supported.

8.2.8.8 EventClass

This property returns the EventClass GUID.

Syntax

```
public virtual System.Guid EventClass
```

8.2.8.9 EventId

This property returns the event ID.

Syntax

```
public virtual uint EventId
```

8.2.8.10 EventType

This property returns the type of the event.

Syntax

```
public virtual EventTypeEnum EventType
```

8.2.8.11 FileTimeCleared

This property returns the timestamp when the alarm was changed to the Cleared state.

Syntax

```
public virtual long FileTimeCleared
```

8.2.8.12 FileTimeConfirmed

This property returns the timestamp when the alarm was changed to the Confirmed state.

Syntax

```
public virtual long FileTimeConfirmed
```

8.2.8.13 FileTimeRaised

This property returns the timestamp when the alarm was changed to the Raised state.

Syntax

```
public virtual long FileTimeRaised
```

8.2.8.14 JsonAttribute

This property returns the JSON attribute.

Syntax

```
public virtual string JsonAttribute
```

8.2.8.15 SeverityLevel

This property returns the Severity level.

Syntax

```
public virtual TcEventLoggerAdsProxyLib.SeverityLevelEnum SeverityLevel
```

8.2.8.16 SourceGuid

This property returns the GUID of the source.

Syntax

```
public virtual System.Guid SourceGuid
```

8.2.8.17 SourceId

This property returns the ID of the source.

Syntax

```
public virtual uint SourceId
```

8.2.8.18 SourceName

This property returns the name of the source.

Syntax

```
public virtual string SourceName
```

8.2.8.19 TimeCleared

This property returns the timestamp when the alarm was changed to the Cleared state.

Syntax

```
public virtual System.DateTime TimeCleared
```

8.2.8.20 TimeConfirmed

This property returns the timestamp when the alarm was changed to the Confirmed state.

Syntax

```
public virtual System.DateTime TimeConfirmed
```

8.2.8.21 TimeRaised

This property returns the timestamp when the alarm was changed to the Raised state.

Syntax

```
public virtual System.DateTime TimeRaised
```

8.2.8.22 WithConfirmation

This property describes whether a confirmation is necessary.

Syntax

```
public virtual bool WithConfirmation
```

8.2.9 ITcMessage3

This interface represents a message.

Syntax

```
public interface ITcMessage
```

Methods

Name	Modifier	Return type	Description
GetArgumentData [▶ 151]	public virtual	void	Returns the data of the arguments.
GetArgumentInfo [▶ 152]	public virtual	void	Returns the (type) information for the arguments.
GetCauseRemedy [▶ 152]	public virtual	TcCauseRemedyCollection	Returns the cause/ remedy information.
GetDetails [▶ 152]	public virtual	TcDetailCollection	Returns the details.
GetEventClassName [▶ 152]	public virtual	string	Returns the EventClass name
GetText [▶ 153]	public virtual	string	Returns the event text, including the arguments.
IsSourceInfoTypeSupported [▶ 153]	public virtual	bool	Query option for the type of source information.

Properties

Name	Modifier	Type	Access	Description
EventClass [▶ 153]	public virtual	System.Guid	get	Returns the EventClass GUID.
EventId [▶ 153]	public virtual	uint	get	Returns the event ID.
EventType [▶ 154]	public virtual	EventTypeEnum	get	Returns the type of the event [▶ 155].
FileTimeRaised [▶ 154]	public virtual	long	get	Timestamp when the message was sent.
JsonAttribute [▶ 154]	public virtual	string	get	The JSON attribute.
SeverityLevel [▶ 154]	public virtual	SeverityLevelEnum	get	The Severity Level [▶ 155].
SourceGuid [▶ 154]	public virtual	System.Guid	get	Returns the GUID of the source.
SourceId [▶ 154]	public virtual	uint	get	Returns the ID of the source.
SourceName [▶ 154]	public virtual	string	get	Returns the name of the source.
TimeRaised [▶ 154]	public virtual	System.DateTime	get	Timestamp when the message was sent.

8.2.9.1 GetArgumentData

This method returns the data of the arguments.

Syntax

```
public virtual void GetArgumentData(ref System.IntPtr ppArgData, uint size)
```

Parameter

Name	Type	Description
ppArgData	ref System.IntPtr	Reference to the data.
size	Uint	Size

8.2.9.2 GetArgumentInfo

This method returns the (type) information for the arguments.

Syntax

```
public virtual void GetArgumentInfo(ref TcEventArgumentsInfo pArgInfo)
```

Parameter

Name	Type	Description
pArgInfo	ref TcEventArgumentsInfo	Reference to the information provided.

8.2.9.3 GetCauseRemedy

This method returns the cause/remedy information, if this has been defined.

Syntax

```
public virtual TcCauseRemedyCollection GetCauseRemedy(int langId)
```

Parameter

Name	Type	Description
langId	int	LangID of the language.

Return value

Name	Type	Description
GetCauseRemedy	TcCauseRemedyCollection	Collection of the cause/remedy information.

8.2.9.4 GetDetails

This method returns the details.

Syntax

```
public virtual TcDetailCollection GetDetails(int langId)
```

Parameter

Name	Type	Description
langId	int	LangID of the language

Return value

Name	Type	Description
GetDetails	TcDetailCollection	Collection of the details

8.2.9.5 GetEventClassName

This method returns the EventClass name.

Syntax

```
public virtual string GetEventClassName(int langId)
```

Parameter

Name	Type	Description
langId	int	LangID of the language

Return value

Name	Type	Description
GetEventClassName	string	Names of the EventClass

8.2.9.6 GetText

This method returns the event text, including the arguments.

Syntax

```
public virtual string GetText(int langId)
```

Parameter

Name	Type	Description
langId	int	LangID of the language

Return value

Name	Type	Description
GetText	string	Event text

8.2.9.7 IsSourceInfoTypeSupported

This method can be used to check whether the source information type has been defined.

Syntax

```
public virtual bool IsSourceInfoTypeSupported(TcSourceInfoTypeEnum infoType)
```

Parameter

Name	Type	Description
infoType	TcSourceInfoTypeEnum [▶ 156]	Query type of source

Return value

Name	Type	Description
IsSourceInfoTypeSupported	bool	Information as to whether supported.

8.2.9.8 EventClass

This property returns the EventClass GUID.

Syntax

```
public virtual System.Guid EventClass
```

8.2.9.9 EventId

This property returns the event ID.

Syntax

```
public virtual uint EventId
```

8.2.9.10 EventType

This property returns the type of the event.

Syntax

```
public virtual EventTypeEnum EventType
```

8.2.9.11 FileTimeRaised

This property returns the timestamp when the alarm was changed to the Raised state.

Syntax

```
public virtual long FileTimeRaised
```

8.2.9.12 JsonAttribute

This property returns the JSON attribute.

Syntax

```
public virtual string JsonAttribute
```

8.2.9.13 SeverityLevel

This property returns the Severity level.

Syntax

```
public virtual TcEventLoggerAdsProxyLib.SeverityLevelEnum SeverityLevel
```

8.2.9.14 SourceGuid

This property returns the GUID of the source.

Syntax

```
public virtual System.Guid SourceGuid
```

8.2.9.15 SourceId

This property returns the ID of the source.

Syntax

```
public virtual uint SourceId
```

8.2.9.16 SourceName

This property returns the name of the source.

Syntax

```
public virtual string SourceName
```

8.2.9.17 TimeRaised

This property returns the timestamp when the alarm was changed to the Raised state.

Syntax

```
public virtual System.DateTime TimeRaised
```

8.3 Data types

8.3.1 ConfirmationStateEnum

Defines the confirmation state of an alarm.

Syntax

```
public enum ConfirmationStateEnum
{
    Confirmed,
    NotRequired,
    NotSupported,
    Reset,
    WaitForConfirmation
}
```

Parameter

Name	Description
Confirmed	Confirmed
NotRequired	Confirmation not necessary in the current state. (Alarm not currently in the Raised state).
NotSupported	Was initialized without confirmation.
Reset	Initial state
WaitForConfirmation	Waiting for confirmation.

8.3.2 EventTypeEnum

Type definition whether a TcEvent is of the type alarm or message.

Syntax

```
public enum EventTypeEnum
{
    Alarm,
    Message
}
```

Parameter

Name	Description
Alarm	The TcEvent is of the type TcAlarm.
Message	The TcEvent is of the type TcMessage.

8.3.3 SeverityLevelEnum

This enumeration defines the "severity" of the event. It is an ordered list.

Syntax

```
public enum SeverityLevelEnum
{
    Critical,
    Error,
    Warning,
    Info,
    Verbose
}
```

Parameter

	Name	Description
4	Critical	Critical
3	Error	Error
2	Warning	Warning
1	Info	Information
0	Verbose	Extended output

8.3.4 TcEventArgumentTypeEnum

Type definition of what type a TcArgument is.

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 of which entry in a TcSourceInfo is identified.

Syntax

```
public enum TcSourceInfoTypeEnum
{
    SourceGuid,
    SourceId,
    SourceName
}
```

Parameter

Name	Description
SourceGuid	The source GUID of the TcSourceInfo.
SourceId	The source ID of the TcSourceInfo. For example, the TcCOM object ID.
SourceName	The SourceName of the TcSourceInfo. For example, the instance path within a PLC.

9 Example

Samples of how to use the EventLogger are provided at this point.

The samples PLC [▶ 157] as well as C++ [▶ 161] refer to the real-time programming interfaces of TwinCAT.

A .NET interface [▶ 166] is available for user mode programs.

9.1 PLC

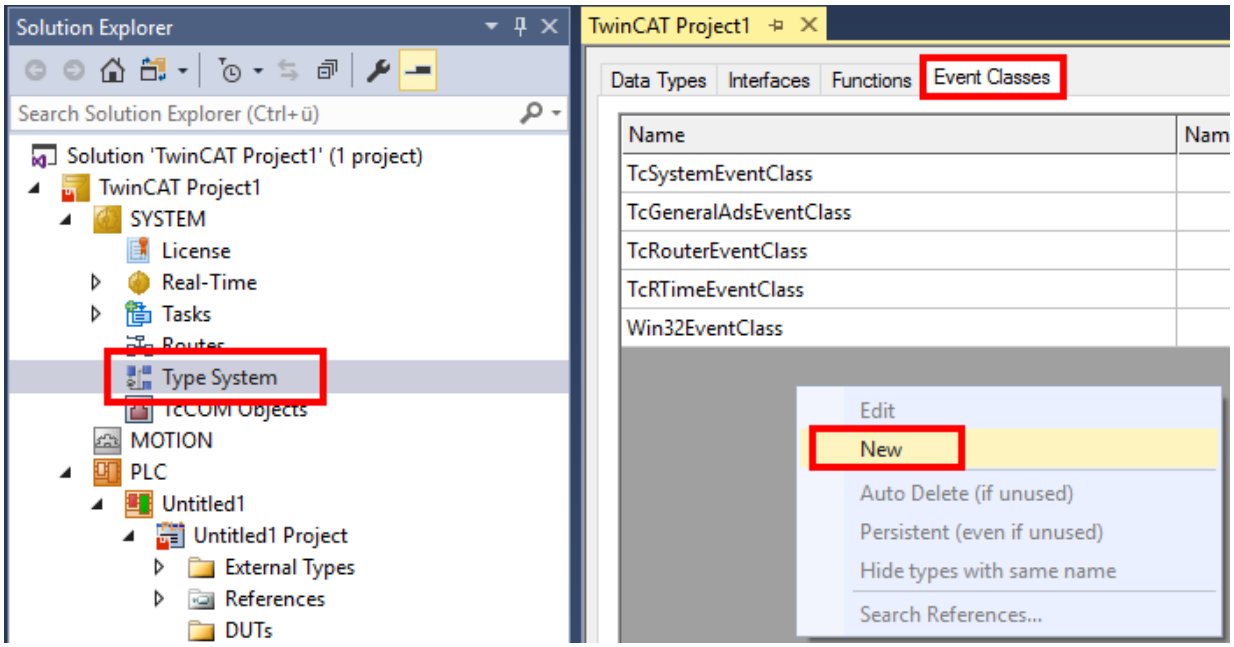
9.1.1 Tutorial

This tutorial illustrates the work steps from an empty TwinCAT project to a dispatched message. It depicts the properties of the TwinCAT 3 EventLogger described in the Technical Introduction [▶ 13] section in the work sequence.

Creating an event class in the TwinCAT type system

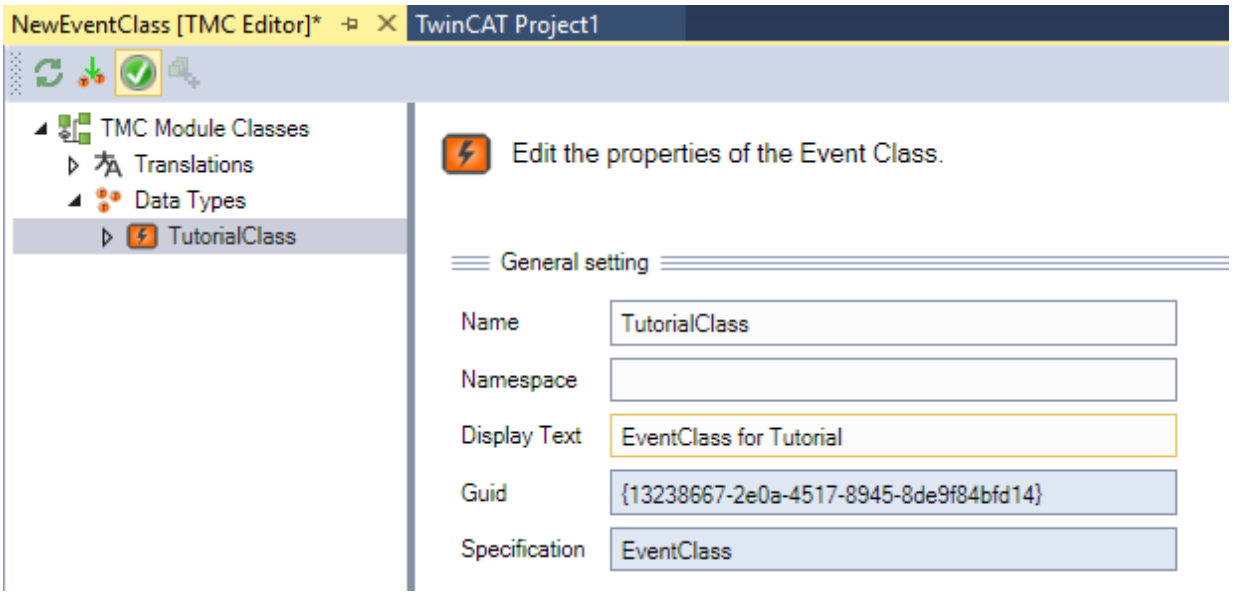
✓ A standard TwinCAT PLC project exists.

1. Double-click on **Type System** in the SYSTEM subtree and select the **Event Classes** tab in the editor which then opens. Open the context menu and select the **New** command.

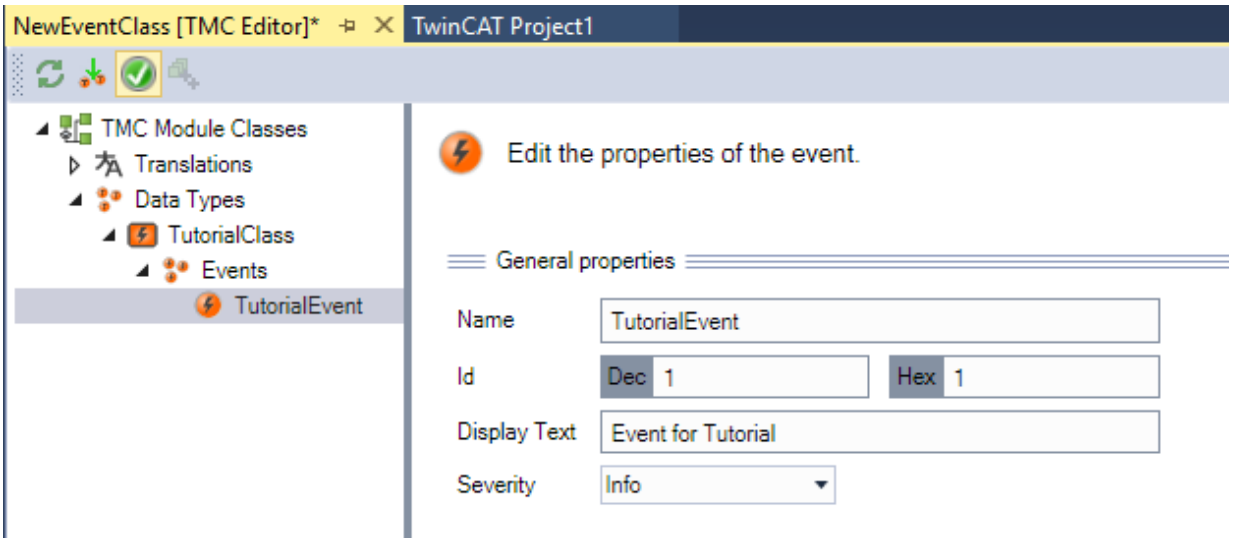


⇨ The TMC editor opens.

2. Give the event class a name and enter a display text.



3. An event is already created below the event class. Give the event a name and enter a display text and the severity.

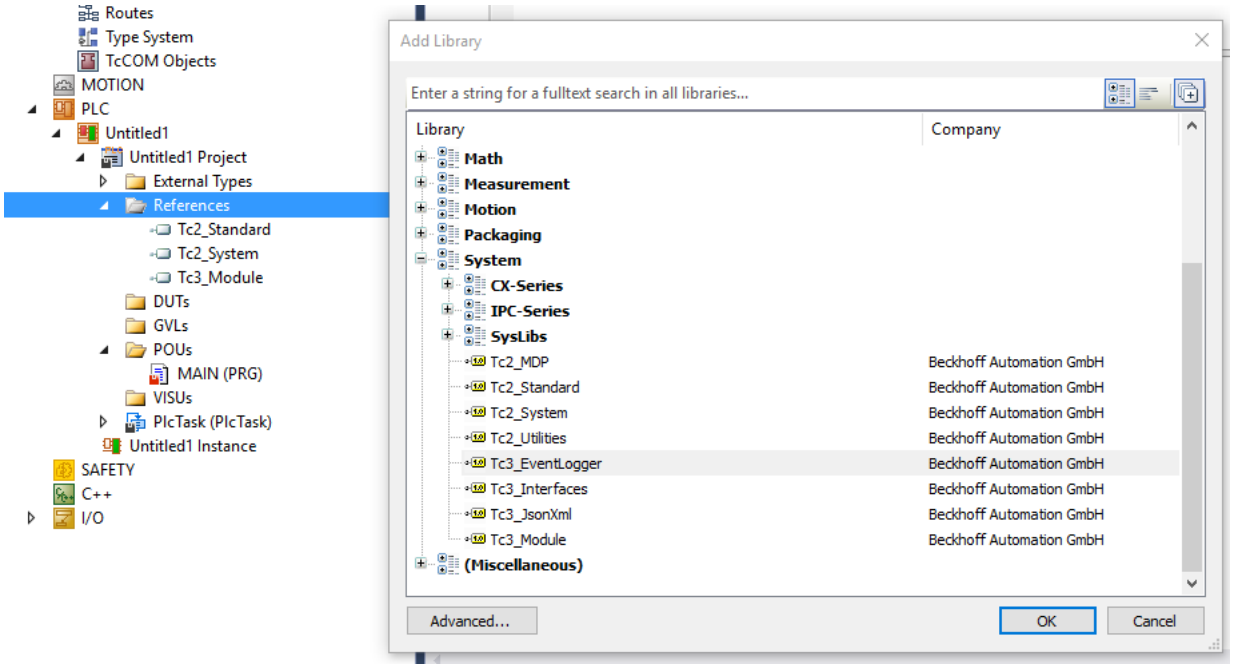


4. Save and, if applicable, close the event class.
   ⇨ The source code is provided in the PLC and is accessible under the TC_EVENTS symbol.

Adding the TC3_EventLogger library

5. Select the **Add Library** command in the context menu of the **References** object.
   ⇨ The **Add Library** dialog opens.

6. Select the library and confirm the dialog.



⇨ The library is added to the PLC project.

Creating a PLC program

1. Open the MAIN program of the PLC project in the editor by double-clicking.
2. Declare and initialize the variables bInit and bSend and declare an instance of the function block FB_TcMessage:

```
PROGRAM MAIN
VAR
    bInit : BOOL := TRUE;
    bSend : BOOL := TRUE;

    fbMsg : FB_TcMessage;
END_VAR
```

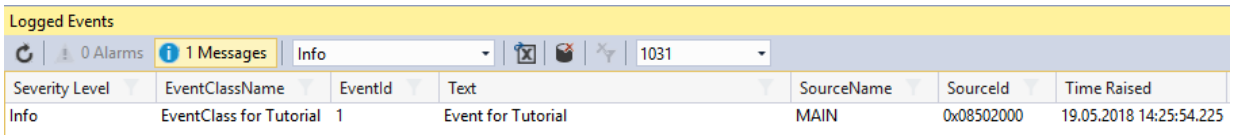
3. Implement the send procedure as shown in the code. The message is initialized once by means of the CreateEx method. Since the initialization requires dynamic resources it should not take place cyclically. The initialized message is subsequently sent using the Send method.

```
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. Create the PLC project and start the PLC.

⇨ The result is shown in the LoggedEvents window in the TwinCAT 3 Engineering.



9.1.2 Example ResultMessage

This example illustrates the use of the TwinCAT 3 EventLogger with function blocks. Firstly, it demonstrates how an output on a function block can be utilized in order to use the event information as an extended return. Secondly, it demonstrates how to carry out the parameterization so that the messages are only output via the TwinCAT 3 EventLogger in certain cases.

Download: https://infosys.beckhoff.com/content/1033/tc3_eventlogger/Resources/5288319115/.zip

The example consists of two function blocks:

- FB_MathCalculation: This function block offers two methods and two properties that always output messages at the output ipResultMessage and additionally send them via the EventLogger if a trace level is exceeded.
  - Addition() method: adds two numbers and sends a message in case of overflow
  - Divison() method: divides two numbers after checking. Sends a message in case of division by 0.
  - Property bTraceLevelDefault: indicates whether the trace level is to be observed locally on the function block or whether to use a trace level library, which exists in the GVL in the example.
  - Property eTraceLevel: the methods only send the message via the EventLogger if the severity is greater than or equal to this property.
- FB_Control: this function block shows the use of the FB_MathCalculation function block within another function block. The Execute method of the FB_Control thereby uses the FB_MathCalculation.Divison() and handles the message further itself as error code.

9.1.3 Example Listener

This sample illustrates the use of the TwinCAT 3 EventLogger in relation to messages and alarms. At the same time the reception of messages is shown in a second project.

Download: https://infosys.beckhoff.com/content/1033/tc3_eventlogger/Resources/5288316939/.zip

Publisher project

Single BOOL variables are used as triggers in the Publisher project:

- bSendMessage to send a message.
- bRaiseAlarm to set an alarm.
- bClearAlarm to cancel an alarm.
- bConfirmAlarm to confirm an alarm.

In addition there is an option to set the JSON attribute in order to send it with both messages.

Listener project

The Listener project contains a function block, FB_Listener, which extends the FB_ListenerBase function block contained in the Tc3_EventLogger. The function block implements the functions for receiving the messages:

- OnMessageSent: when a message has been sent the EventLogger will call this method as a callback. The method counts the number of messages.
- OnAlarmRaised/OnAlarmCleared/OnAlarmConfirmed: if the alarm changes its state the EventLogger will call this method as a callback. The methods count the number of state changes.
- To initiate receiving of messages, an Execute method is implemented at the function block.
- The text of the last received message can be retrieved.
- The function block FB_ListenerTest uses the FB_Listener. In doing so it registers the event class to be received once. Another event class that exists is not received to demonstrate the filter functionality.

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT v3.1.4024.17	PC or CX (x64, x86, ARM)	Tc3_EventLogger (>= v3.1.27.0)

9.1.4 Example filter

This sample illustrates the application of the TwinCAT 3 EventLogger for receiving messages. The focus is on the filter functions in order to process the right messages in a targeted manner.

Download: https://infosys.beckhoff.com/content/1033/tc3_eventlogger/Resources/10400437387/.zip

The sample has four components:

- A number of different messages are sent, demonstrating the selection of messages in different filters.
- One component shows how to discard from the cache messages that are specified by a filter.
- Another component illustrates the export of cached messages to a CSV file. Here, too, the filters are used to program which messages are to be selected.
- Another component illustrates the general principle of receiving messages sent in real-time and receiving of EtherCAT emergency messages.

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT v3.1.4024.17	PC or CX (x64, x86, ARM)	Tc3_EventLogger (>= v3.1.27.0)

9.2 C++

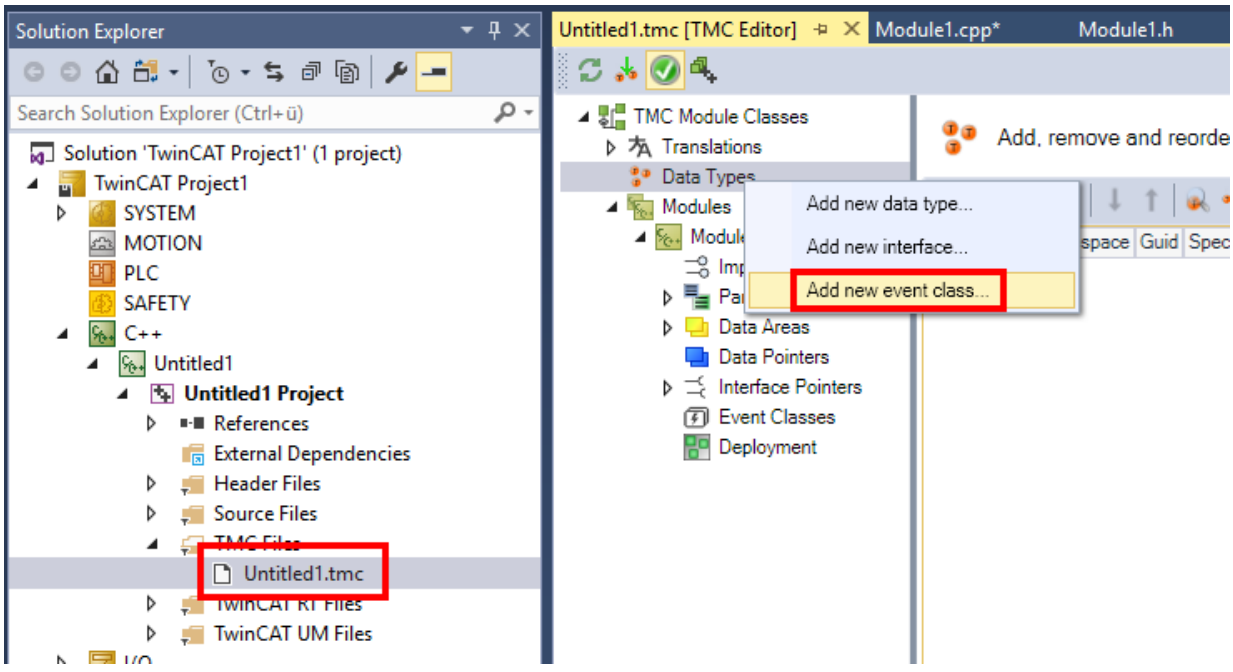
9.2.1 Tutorial

This tutorial illustrates the work steps from an empty TwinCAT project to a dispatched message. It depicts the properties of the TwinCAT 3 EventLogger described in the Technical Introduction [▶ 13] section in the work sequence.

Creating an event class in the data types of the TMC file in the C++ project

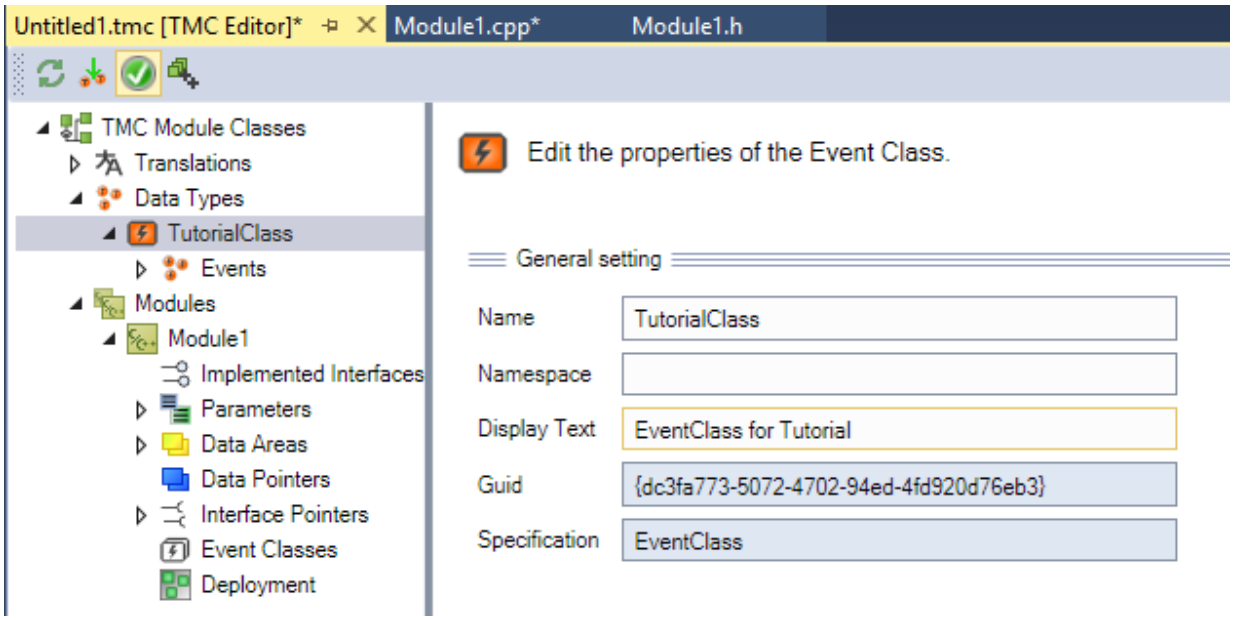
✓ A new TwinCAT C++ project exists with the module from the wizard "TwinCAT Module Class with Cyclic IO".

1. Click twice on the TMC file in the C++ project to open the TMC editor. Select the command **Add new event class…** from the context menu of **Data Types.**

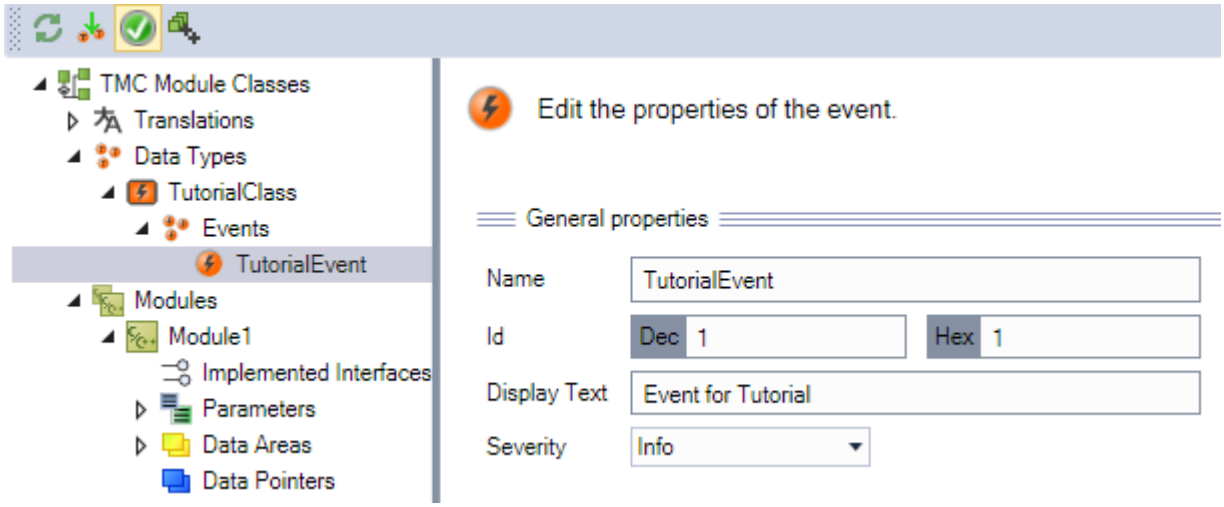


⇨ The TMC editor opens the event class.

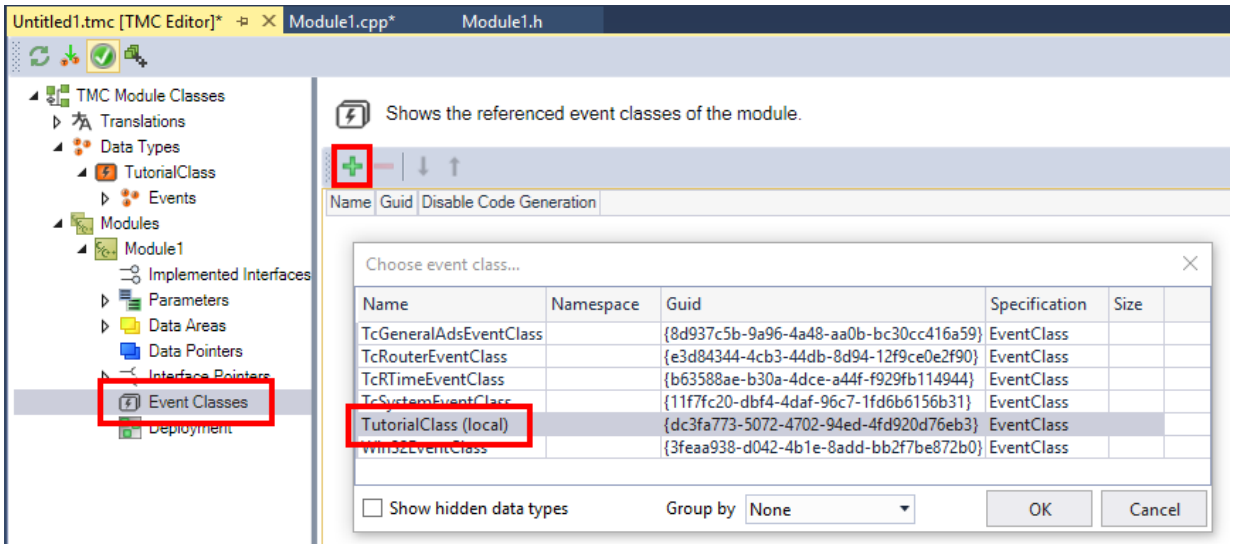
2. Give the event class a name and optionally enter a display text.



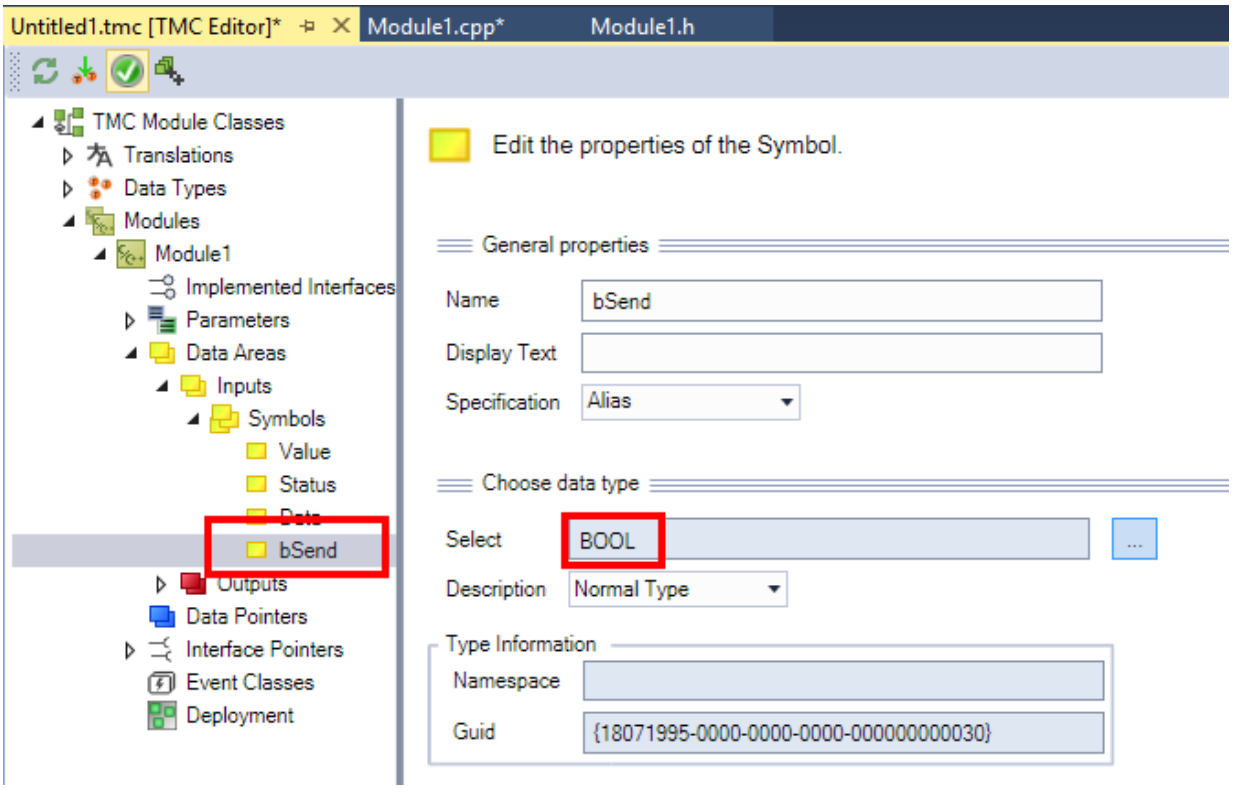
3. An event is already created below the event class. Give the event a name and enter a display text and the severity.



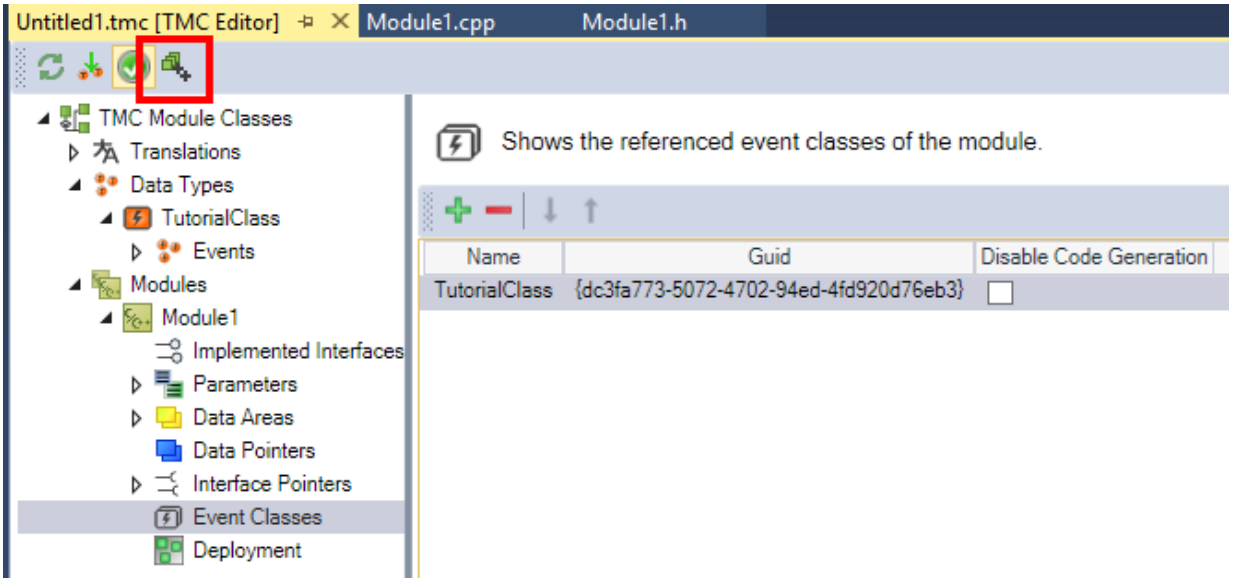
4. Use the event class in the previously created module.



5. In addition, create an input bSend in order to send the event on a rising edge.



6. Generate the source code of the module.



Creating a C++ program

7. Add headers to the Untitled1Interfaces.h in the C++ program:

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

8. You need the following declarations in the Module1.h:

```
UINT m_counter;
ITcEventLoggerPtr m_spEventLogger;
ITcMessagePtr m_spMessage;
BOOL m_OldSend;
```

9. Initialize the following values in the constructor of the module in Module1.cpp:

```
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. In addition, carry out the following initialization in the method SetObjStateSO():

```
// 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. If the call of the method AddModuleToCaller() fails, carry out a deinitialization:

```
// Cleanup if transition failed at some stage
if ( FAILED(hr) )
{
    RemoveModuleFromCaller();
    m_spEventLogger = NULL;
    m_spMessage = NULL;
}
```

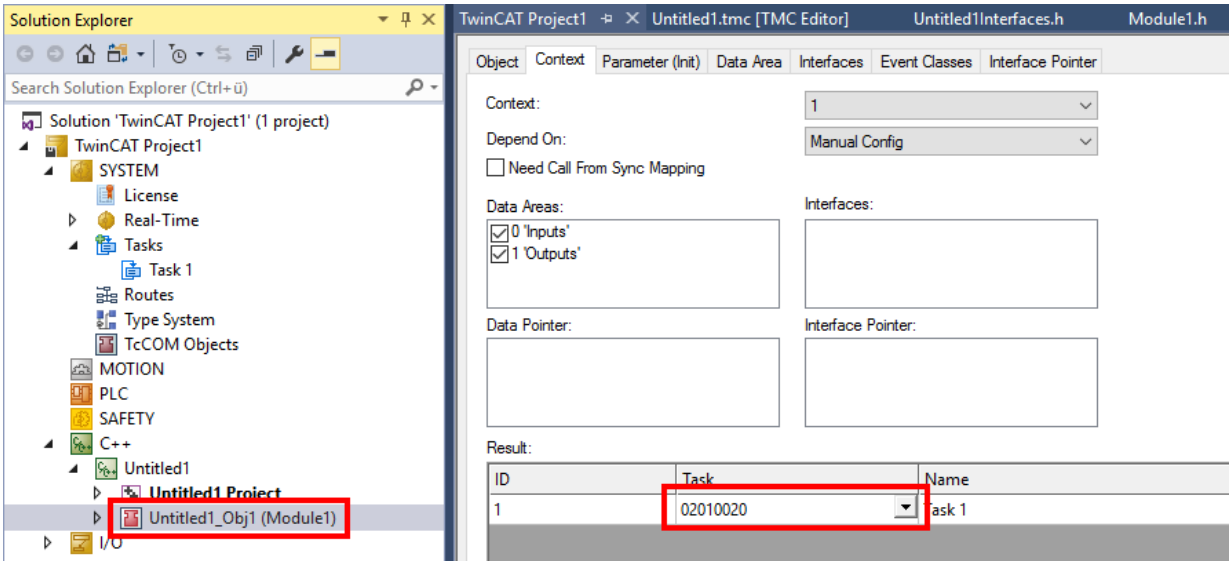
12. Carry out a deinitialization in the method SetObjStateOS():

```
// TODO: Add any additional deinitialization
m_spEventLogger = NULL;
m_spMessage = NULL;
```

13. Extend the cyclic code of the module by the sending of the message:

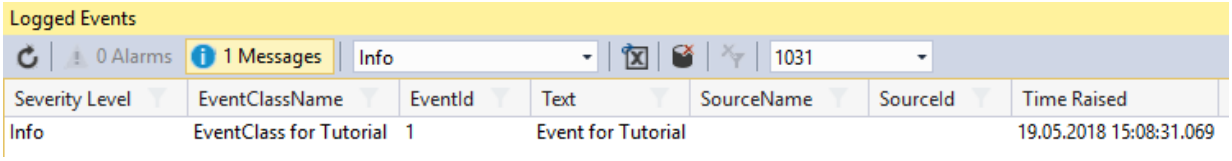
```
// TODO: Replace the sample with your cyclic code
if (m_Inputs.bSend && ! m_OldSend) // raising edge
{
   m_spMessage->Send(0);
}
m_OldSend = m_Inputs.bSend;
```

14. Create a module instance and link it with a task.



15. Create the C++ project and start TwinCAT.

⇨ The result is shown in the LoggedEvents window in the TwinCAT 3 Engineering.



9.2.2 Example: Start-Stop

This example illustrates the use of the TwinCAT 3 EventLogger when starting and stopping TwinCAT. A message is used with arguments that map the different states of the C++ module.

Download: https://infosys.beckhoff.com/content/1033/tc3_eventlogger/Resources/5288321291/.zip

The message is used in the methods of the state machines:

- SetObjStatePS(): the EventLogger is being started by TwinCAT. Use in this state is thus impossible.
- SetObjStateSO(): here, the reference to the EventLogger is procured through TcQuerySmartObjectInterface and the messages initiated by CreateMessage(). A corresponding message is dispatched. A message is dispatched with AddModuleToCaller(). A message is also dispatched here at the end of the transition.
- CycleUpdate(): in this example no messages are dispatched. The behavior of the module towards the OP state is thus equivalent to that of a "Cyclic IO" module.
- SetObjStateOS(): a message is dispatched with RemoveModuleFromCaller(). A message is also dispatched here at the end of the transition. Thereafter the references to the message and to the EventLogger are set to ZERO, which is also notified by means of a message.
- SetObjStateSP(): The EventLogger is shut down by TwinCAT. Use in this state is thus impossible.

The following messages result:

Logged Events

0 Alarms | 6 Messages | Verbose | 1031

Severity Level	EventClassName	EventId	Text	Time Raised
Verbose	TcComMessages	5	SetObjStateOS Done	29.05.2018 15:25:35.736
Verbose	TcComMessages	2	Eventlogger deinitialized	29.05.2018 15:25:35.736
Verbose	TcComMessages	8	RemoveModuleFromCaller Done	29.05.2018 15:25:35.736
Verbose	TcComMessages	4	SetObjStateSO Done	29.05.2018 15:25:32.495
Verbose	TcComMessages	7	AddModuleToCaller Done	29.05.2018 15:25:32.495
Verbose	TcComMessages	1	Eventlogger initialized	29.05.2018 15:25:32.495

9.2.3 Example Listener

This example illustrates the use of the TwinCAT 3 EventLogger in relation to messages and alarms.

It consists of a module that can send a message and an alarm, and a module that receives these messages.

Download: https://infosys.beckhoff.com/content/1033/tc3_eventlogger/Resources/5288323467/.zip

Publisher module

Single BOOL variables are used in the Publisher module as triggers on the input data area:

- bSendMessage to send a message
- bRaiseAlarm to set an alarm
- bClearAlarm to cancel an alarm
- bConfirmAlarm to confirm an alarm

In addition there is an option to set and to remove the JSON attribute in order to send it with both messages.

Listener module

The Listener module implements both the ITCMessageListener and the ITcAlarmListener interface. The methods specified as a result of this are called by the EventLogger as callback.

- OnMessageSent: when a message has been sent the EventLogger will call this method as a callback. The method counts the number of messages.
- OnAlarmRaised/OnAlarmCleared/OnAlarmConfirmed: if the alarm changes its state the EventLogger will call this method as a callback. The methods count the number of state changes.

The Listener module thereby registers itself with the EventLogger for the corresponding event classes on starting in the method SetObjStateSO().

9.3 User mode API

Beckhoff.TwinCAT.TcEventLoggerAdsProxy.Net from NuGet.org

The API is available on NuGet.org via the package Beckhoff.TwinCAT.TcEventLoggerAdsProxy.Net for implementation into projects. To get started easily, you will find there sample code in the README file, which only needs to be copied into a .NET project.

COM based interface is replaced

The COM based interface, which is described here before, is supported up to TwinCAT 3.1 4024. Due to the technology used, it cannot be offered under TwinCAT/BSD.
The API provided on NuGet.org is the successor and provides easy porting for customer applications through an equivalent API.

Also see about this

User mode API [▶ 117]

10 Appendix

10.1 ADS Return Codes

Grouping of error codes:
Global error codes: ADS Return Codes [▶ 167]... (0x9811_0000 ...)
Router error codes: ADS Return Codes [▶ 167]... (0x9811_0500 ...)
General ADS errors: ADS Return Codes [▶ 168]... (0x9811_0700 ...)
RTime error codes: ADS Return Codes [▶ 170]... (0x9811_1000 ...)

Global error codes

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

Router error codes

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

General ADS error codes

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

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

RTime error codes

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

Specific positive HRESULT Return Codes:

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

TCP Winsock error codes

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

10.2 Support and Service

Beckhoff and their partners around the world offer comprehensive support and service, making available fast and competent assistance with all questions related to Beckhoff products and system solutions.

Beckhoff's branch offices and representatives

Please contact your Beckhoff branch office or representative for local support and service on Beckhoff products!

The addresses of Beckhoff's branch offices and representatives round the world can be found on her internet pages: https://www.beckhoff.com

You will also find further documentation for Beckhoff components there.

Beckhoff Support

Support offers you comprehensive technical assistance, helping you not only with the application of individual Beckhoff products, but also with other, wide-ranging services:

- support
- design, programming and commissioning of complex automation systems
- and extensive training program for Beckhoff system components

Hotline:	+49 5246 963 157
Fax:	+49 5246 963 9157
e-mail:	support@beckhoff.com

Beckhoff Service

The Beckhoff Service Center supports you in all matters of after-sales service:

- on-site service
- repair service
- spare parts service
- hotline service

Hotline:	+49 5246 963 460
Fax:	+49 5246 963 479
e-mail:	service@beckhoff.com

Beckhoff Headquarters

Beckhoff Automation GmbH & Co. KG

Huelshorstweg 20
33415 Verl
Germany

Phone: +49 5246 963 0
Fax: +49 5246 963 198
e-mail: info@beckhoff.com
web: https://www.beckhoff.com

More Information:
www.beckhoff.com/te1000

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