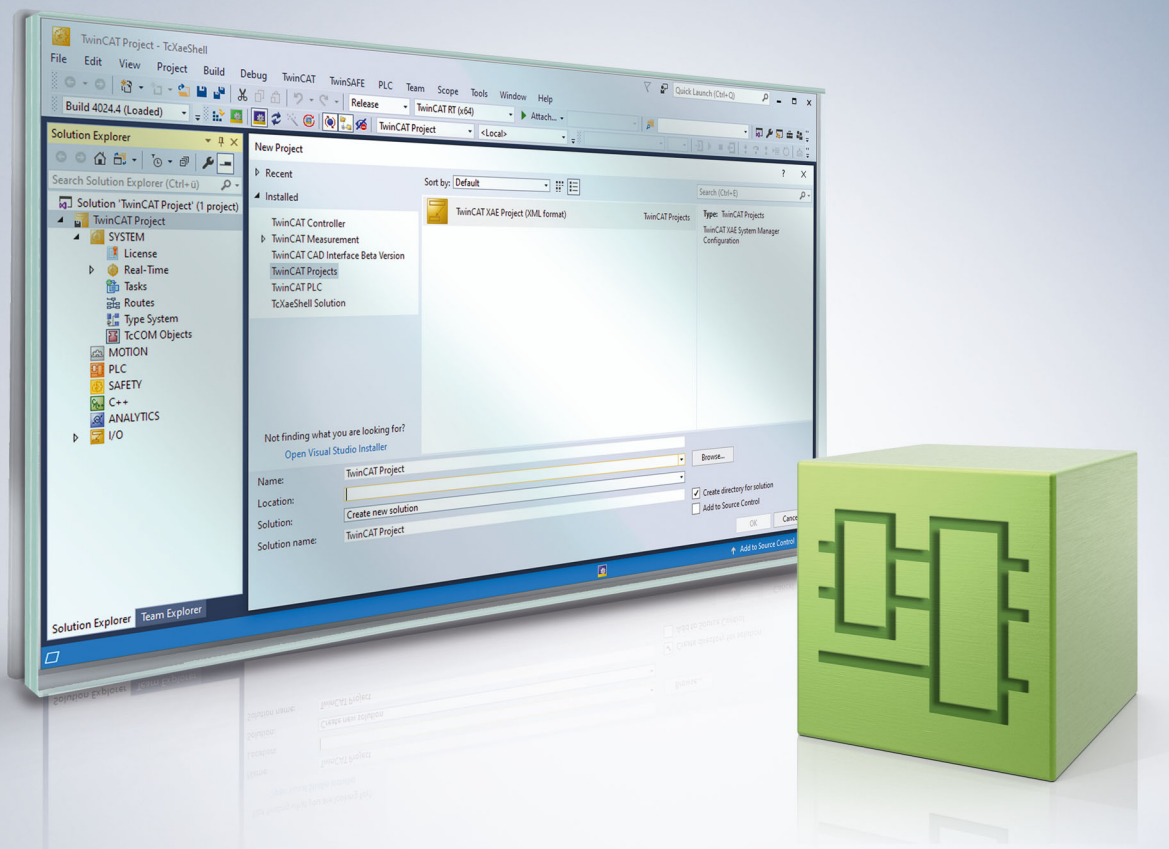
BECKHOFF New Automation Technology

Manual | EN

TE1000

TwinCAT 3 | PLC Library: Tc3_EventLogger



2022-12-06 | Version: 1.5.1

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.

NOTE

Damage to the environment or devices

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

Tip or pointer

This symbol indicates information that contributes to better understanding.

1.3 Notes on information security

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

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

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 Using outdated or unsupported product versions can increase the risk of cyber threats.

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

2 Overview

The PLC library contains functions and function blocks for using the TwinCAT 3 EventLogger.

See also: Dokumentation TwinCAT 3 EventLogger

Function blocks for using the TC3 EventLogger

FB_ListenerBase2 [▶ 26]	Basic implementation of an event listener.
FB_TcAlarm [▶ 31]	Represents an alarm of the TwinCAT 3 EventLogger.
FB_TcArguments [▶ 37]	Defines arguments of an event.
FB_TcEventLogger [▶ 47]	Represents the TwinCAT 3 EventLogger itself.
FB_TcMessage [▶ 56]	Represents a message from the TwinCAT 3 EventLogger.
FB_TcSourceInfo [▶ 60]	Defines the source information of an event.

Asynchronous text requests

F_GetEventClassName [▶ 16]	Triggers the asynchronous request for the name of an event class.
F_GetEventText [▶ 17]	Triggers the asynchronous request for an event text.
FB_AsyncStrResult [▶ 11]	Enables the asynchronous request for a text.
FB_RequestEventClassName [▶ 12]	Enables the asynchronous request for the name of an event class.
FB_RequestEventText [▶ 14]	Enables the asynchronous request for an event text in the desired language.

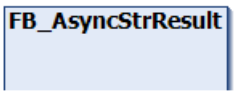
EventEntry conversion

AdsErr_TO_TcEventEntry [▶ 23]	Converts a standard ADS error into a TcEventEntry.
HRESULTAdsErr_TO_TcEventEntry [▶ 24]	Converts a standard ADS error (HRESULT) into a TcEventEntry.
TcEventEntry_TO_AdsErr [▶ 25]	Converts a TcEventEntry into a standard ADS error.
TcEventEntry_TO_HRESULTAdsErr [▶ 25]	Converts a TcEventEntry into a standard ADS error (HRESULT).

3 Functions and function blocks

3.1 Asynchronous text requests

3.1.1 FB_AsyncStrResult



This function block enables the asynchronous request for a text.

Syntax

Definition:

```
FUNCTION_BLOCK FB_AsyncStrResult
```

Methods

Name	Description
GetString [▶ 11]	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

3.1.1.1 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

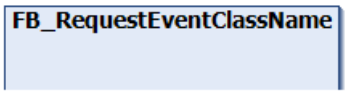
Name	Type	Description
GetString	BOOL	Returns TRUE if the text could be assigned. Returns FALSE if the text could not be completely assigned because the specified buffer variable is too small.

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
```

3.1.2 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 [▶ 13]	As soon as bBusy is FALSE and provided no error has occurred (bError = FALSE), the requested text can be fetched with this method.
Request [▶ 13]	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

3.1.2.1 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

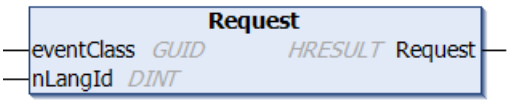
Name	Type	Description
GetString	BOOL	Returns TRUE if the text could be assigned. Returns FALSE if the text could not be completely assigned because the specified buffer variable is too small.

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
```

3.1.2.2 Request



Calling this method triggers the asynchronous text request.

Syntax

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

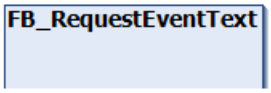
Inputs

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.

3.1.3 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 [▶ 14]	As soon as bBusy is FALSE and provided no error has occurred (bError = FALSE), the requested text can be fetched with this method.
Request [▶ 15]	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

3.1.3.1 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
```

3.1.3.2 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
```

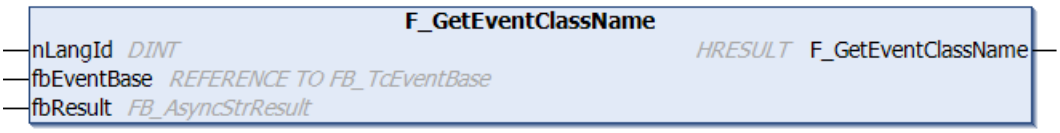
Inputs

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 [▶ 63]	Optional specification of arguments.

Return value

Name	Type	Description
Request	HRESULT	Returns possible error information.

3.1.4 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 [▶ 41]	Specification of an event/alarm/message object.

Inputs/outputs

Name	Type	Description
fbResult	FB_AsyncStrResult [▶ 11]	Specification of a function block instance in order to track an asynchronous text request.

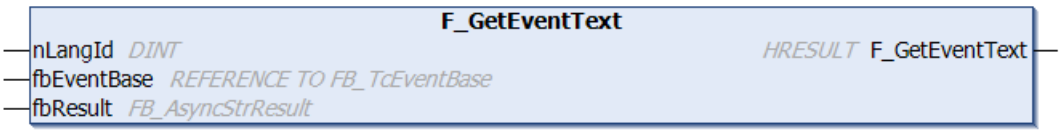
Return value

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

3.1.5 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 [▶ 41]	Specification of an event/alarm/message object.

Inputs/outputs

Name	Type	Description
fbResult	FB_AsyncStrResult [▶ 11]	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.

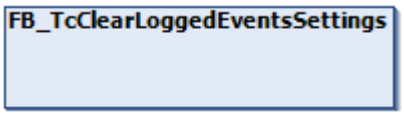
Requirements

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

3.2 Filter

The filter functionality is used in different places. A sample that describes the possible uses is covered in the Example filter [▶ 87].

3.2.1 FB_TcClearLoggedEventsSettings



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

Syntax

Definition:

```
FUNCTION_BLOCK FB_TcClearLoggedEventsSettings IMPLEMENTS I_TcClearLoggedEventsSettings
```

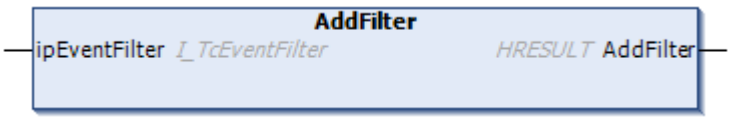
Methods

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)

3.2.1.1 AddFilter



Method for adding a filter.

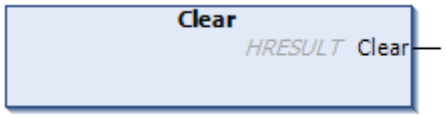
Inputs

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.

3.2.1.2 Clear



Method for clearing the settings.

Return values

Name	Type	Description
Clear	HRESULT	Returns S_OK if successful.

3.2.1.3 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.

3.2.1.4 SetSorting

Sets the sort order for the query.

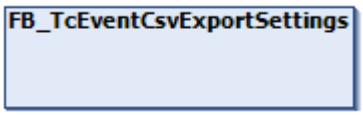
Inputs

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.

3.2.2 FB_TcEventCsvExportSettings



Provides the functionality to specify the csv export.

Syntax

Definition:

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

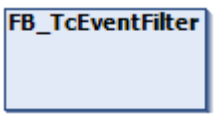
Methods

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)

3.2.3 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 [▶ 87] 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.

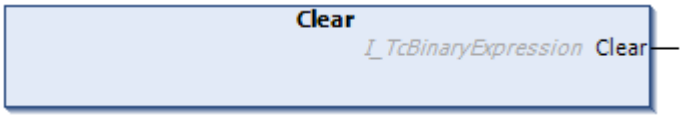
Properties

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)

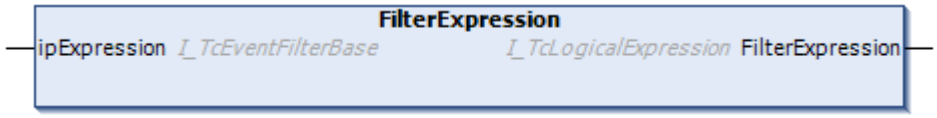
3.2.3.1 Clear



Return values

Name	Type	Description
Clear	I_TcBinaryExpression	

3.2.3.2 FilterExpression



Inputs

Name	Type	Description
ipExpression	I_TcEventFilterBase	

Return values

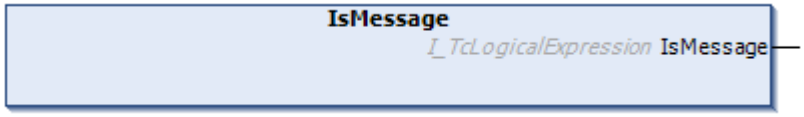
Name	Type	Description
FilterExpression	I_TcLogicalExpression	

3.2.3.3 IsAlarm

Return values

Name	Type	Description
IsAlarm	I_TcLogicalExpression	

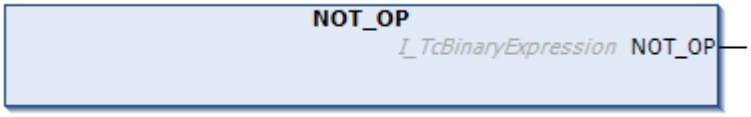
3.2.3.4 IsMessage



Return values

Name	Type	Description
IsMessage	I_TcLogicalExpression	

3.2.3.5 NOT_OP

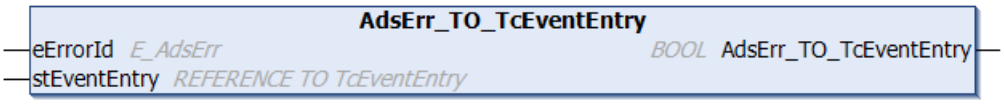


Return values

Name	Type	Description
NOT_OP	I_TcBinaryExpression	

3.3 EventEntry conversion

3.3.1 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 [▶ 80]	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

3.3.2 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 [▶ 80]	Outputs the resulting event definition.

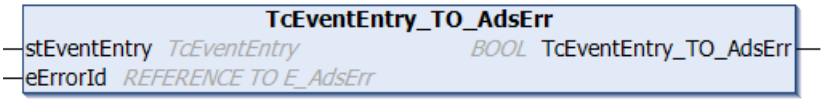
Return value

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

3.3.3 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 [▶ 80]	Event definition to be converted.
eErrorId	REFERENCE TO E_AdsErr	Outputs the resulting error code.

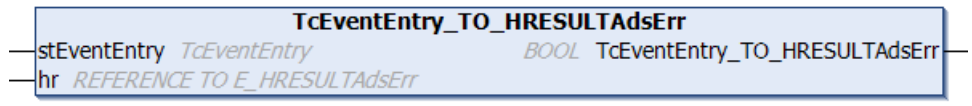
Return value

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

3.3.4 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 [▶ 80]	Event definition to be converted.
hr	REFERENCE TO E_HRESULTAdsErr	Outputs the resulting error code.

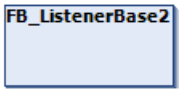
Return value

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

3.4 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 [▶ 27]	Local	Must be called cyclically so that the event queue can be processed.
Subscribe [▶ 29]	Local	Subscribes messages.
Unsubscribe [▶ 30]	Local	Unsubscribes messages.

Event-driven methods (callback methods)

Name	Definition location	Description
OnAlarmCleared [▶ 27]	I_Listener2	Called when the state of an alarm changes from "Raised" to "Clear".
OnAlarmConfirmed [▶ 28]	I_Listener2	Called when an alarm has been confirmed.
OnAlarmDisposed [▶ 28]	I_Listener2	Called when an alarm instance has been released again.
OnAlarmRaised [▶ 29]	I_Listener2	Called when the state of an alarm changes from "Clear" to "Raised".
OnMessageSent [▶ 29]	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)

3.4.1 Execute

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

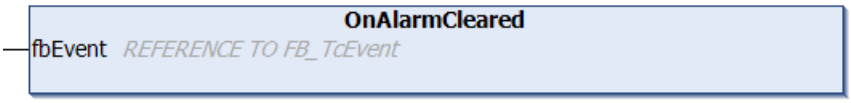
Syntax

```
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.

3.4.2 OnAlarmCleared



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

Syntax

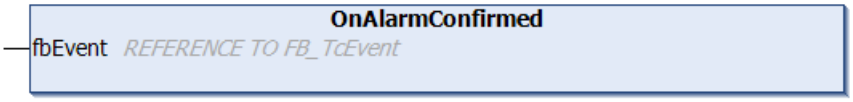
```
METHOD OnAlarmCleared : HRESULT
VAR_INPUT
    fbEvent : REFERENCE TO FB_TcEvent;
END_VAR
```

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 [▶ 39]	Reference to the alarm that has occurred. This reference must not be copied, e.g. through assignment.

3.4.3 OnAlarmConfirmed



This method is called when an alarm has been confirmed.

Syntax

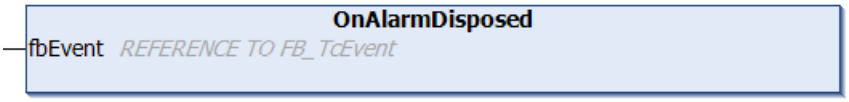
```
METHOD OnAlarmConfirmed : HRESULT
VAR_INPUT
    fbEvent : REFERENCE TO FB_TcEvent;
END_VAR
```

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 [▶ 39]	Reference to the alarm that has occurred. This reference must not be copied, e.g. through assignment.

3.4.4 OnAlarmDisposed



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

Syntax

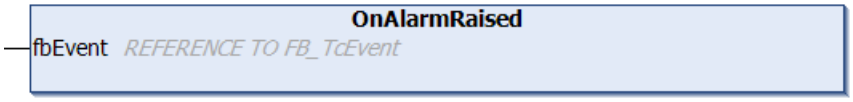
```
METHOD OnAlarmConfirmed : HRESULT
VAR_INPUT
    fbEvent : REFERENCE TO FB_TcEvent;
END_VAR
```

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 [▶ 39]	Reference to the alarm that has occurred. This reference must not be copied, e.g. through assignment.

3.4.5 OnAlarmRaised



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

Syntax

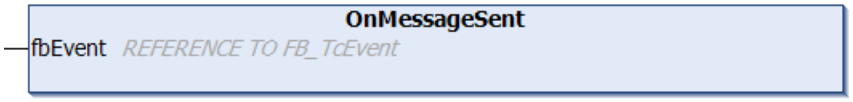
```
METHOD OnAlarmRaised : HRESULT
VAR_INPUT
    fbEvent : REFERENCE TO FB_TcEvent;
END_VAR
```

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 [▶ 39]	Reference to the alarm that has occurred. This reference must not be copied, e.g. through assignment.

3.4.6 OnMessageSent



This method is called when a message has been sent.

Syntax

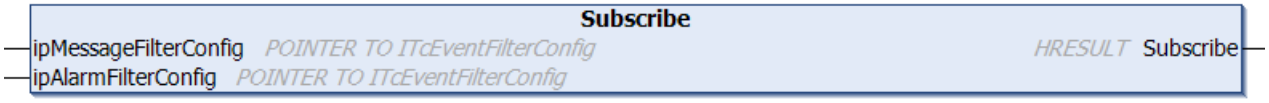
```
METHOD OnMessageSent : HRESULT
VAR_INPUT
    fbEvent : REFERENCE TO FB_TcEvent;
END_VAR
```

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 [▶ 39]	Reference to the event that has occurred. This reference must not be copied, e.g. through assignment.

3.4.7 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
```

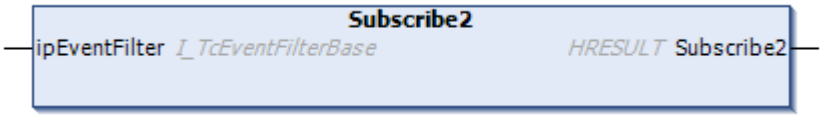
Inputs

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.

3.4.8 Subscribe2



Syntax

```
METHOD Subscribe2 : HRESULT
```

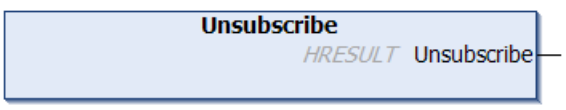
Input

Name	Type	Description
ipEventFilter	I_TcEventFilterBase	Pointer to an instance of FB_TcEventFilter [▶ 20], if a filter is to be activated.

Return value

Name	Type	Description
Subscribe2	HRESULT	Delivers if successful S_OK.

3.4.9 Unsubscribe



The listener is unsubcribed with this method.

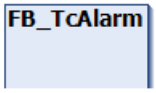
Syntax

```
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.

3.5 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 [▶ 41]

FB_TcAlarm

Methods

Name	Definition location	Description
EqualsTo [▶ 42]	Inherited from FB_TcEventBase [▶ 41]	Compares the event with another instance.
EqualsToEventClass [▶ 43]	Inherited from FB_TcEventBase [▶ 41]	Compares the event class of the event with another event class.
EqualsToEventEntry [▶ 43]	Inherited from FB_TcEventBase [▶ 41]	Compares the event class, the event ID and the severity of the event with those of another event.
EqualsToEventEntryEx [▶ 44]	Inherited from FB_TcEventBase [▶ 41]	Compares the event definition of the event with another event definition.
GetJsonAttribute [▶ 44]	Inherited from FB_TcEventBase [▶ 41]	Returns the Json attribute.
Release [▶ 45]	Inherited from FB_TcEventBase [▶ 41]	Releases the instance created by the EventLogger again.
RequestEventClassName [▶ 45]	Inherited from FB_TcEventBase [▶ 41]	Requests the name of the event class.
RequestEventText [▶ 46]	Inherited from FB_TcEventBase [▶ 41]	Returns the text for the event.
Clear [▶ 33]	Local	Sets the alarm state to "Not Raised".
Confirm [▶ 34]	Local	Confirms the alarm.
Create [▶ 35]	Local	Creates an alarm instance in the EventLogger.
CreateEx [▶ 35]	Local	Creates an alarm instance in the EventLogger from an event definition.
Raise [▶ 36]	Local	Sets the alarm state to "Raised".
SetJsonAttribute [▶ 37]	Local	Sets the Json attribute.

Properties

Name	Type	Access	Definition location	Description
eSeverity	TcEventSeverity [▶ 80]	Get	Inherited from FB_TcEventBase [▶ 41]	Returns the severity.
EventClass	GUID	Get	Inherited from FB_TcEventBase [▶ 41]	Returns the GUID of the event class.
ipArguments [▶ 47]	I_TcArguments [▶ 63]	Get	Inherited from FB_TcEventBase [▶ 41]	Returns the interface pointer for the arguments.
ipSourceInfo [▶ 47]	I_TcSourceInfo [▶ 78]	Get	Inherited from FB_TcEventBase [▶ 41]	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 [▶ 41]	Returns the ID of the event.
stEventEntry	TcEventEntry [▶ 80]	Get	Inherited from FB_TcEventBase [▶ 41]	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 [▶ 80]	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

3.5.1 Clear

This method sets the alarm state 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.

3.5.2 Confirm

Sets the confirmation state 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.

3.5.3 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
```

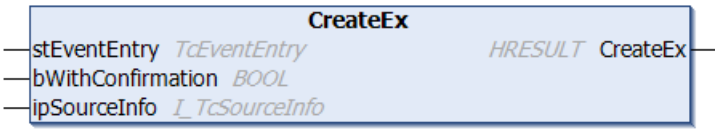
Inputs

Name	Type	Description
eventClass	GUID	GUID of the event class.
nEventId	UDINT	ID of the event.
eSeverity	TcEventSeverity [▶ 80]	Severity of the event.
bWithConfirmation	BOOL	Defines whether the alarm requires mandatory confirmation.
ipSourceInfo	I_TcSourceInfo [▶ 78]	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

3.5.4 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 [▶ 80]	Event definition.
bWithConfirmation	BOOL	Defines whether the alarm requires mandatory confirmation.
ipSourceInfo	I_TcSourceInfo [▶ 78]	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.

3.5.5 Raise

Sets the alarm state 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

3.5.6 SetJsonAttribute

This method sets the JSON attribute.

Syntax

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

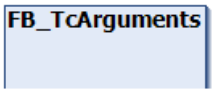
Inputs

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.

3.6 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 [▶ 63]	Defines the argument handling.

Methods

Name	Definition location	Description
AddBlob [▶ 63]	I_TcArguments [▶ 63]	Adds binary data as an argument.
AddBool [▶ 64]	I_TcArguments [▶ 63]	Adds an argument of the type BOOL.
AddByte [▶ 64]	I_TcArguments [▶ 63]	Adds an argument of the type BYTE.
AddDint [▶ 65]	I_TcArguments [▶ 63]	Adds an argument of the type DINT.
AddDWord [▶ 65]	I_TcArguments [▶ 63]	Adds an argument of the type DWORD.
AddEventReferenceId [▶ 66]	I_TcArguments [▶ 63]	Adds a reference to another event as an argument.
AddEventReferenceIdGuid [▶ 66]	I_TcArguments [▶ 63]	Adds a reference to another event as an argument.
AddInt [▶ 67]	I_TcArguments [▶ 63]	Adds an argument of the type INT.
AddLInt [▶ 67]	I_TcArguments [▶ 63]	Adds an argument of the type LINT.
AddLReal [▶ 68]	I_TcArguments [▶ 63]	Adds an argument of the type LREAL.
AddReal [▶ 68]	I_TcArguments [▶ 63]	Adds an argument of the type REAL.
AddSInt [▶ 69]	I_TcArguments [▶ 63]	Adds an argument of the type SINT.
AddString [▶ 69]	I_TcArguments [▶ 63]	Adds an argument of the type STRING.
AddUDint [▶ 70]	I_TcArguments [▶ 63]	Adds an argument of the type UDINT.
AddUInt [▶ 70]	I_TcArguments [▶ 63]	Adds an argument of the type INT.
AddULInt [▶ 70]	I_TcArguments [▶ 63]	Adds an argument of the type ULINT.
AddUSInt [▶ 71]	I_TcArguments [▶ 63]	Adds an argument of the type USINT.
AddWord [▶ 71]	I_TcArguments [▶ 63]	Adds an argument of the type WORD.
AddWString [▶ 72]	I_TcArguments [▶ 63]	Adds an argument of the type WSTRING.
Clear [▶ 72]	I_TcArguments [▶ 63]	Removes all arguments.
IsEmpty [▶ 39]	Local	Checks whether arguments have been added.

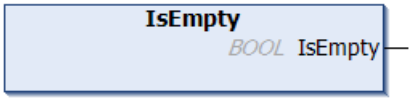
Properties

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

3.6.1 IsEmpty



This method checks whether arguments have been added.

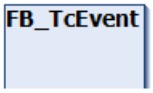
Syntax

```
METHOD IsEmpty : BOOL
```

Return value

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

3.7 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 [▶ 41]

FB_TcEvent

Interfaces

Type	Description
I_TcEventBase [▶ 73]	Basic interface that defines methods and properties of an event.

Methods

Name	Definition location	Description
EqualsTo [▶ 42]	Inherited from FB_TcEventBase [▶ 41]	Compares the event with another instance.
EqualsToEventClass [▶ 43]	Inherited from FB_TcEventBase [▶ 41]	Compares the event class of the event with another event class.
EqualsToEventEntry [▶ 43]	Inherited from FB_TcEventBase [▶ 41]	Compares the event definition of the event with another event definition.
EqualsToEventEntryEx [▶ 44]	Inherited from FB_TcEventBase [▶ 41]	Compares the event definition of the event with another event definition.
GetJsonAttribute [▶ 44]	Inherited from FB_TcEventBase [▶ 41]	Returns the Json attribute.
Release [▶ 45]	Inherited from FB_TcEventBase [▶ 41]	Releases the instance created by the EventLogger again.
RequestEventClassName [▶ 45]	Inherited from FB_TcEventBase [▶ 41]	Requests the name of the event class.
RequestEventText [▶ 46]	Inherited from FB_TcEventBase [▶ 41]	Returns the text for the event.

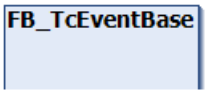
Properties

Name	Type	Access	Definition location	Description
eSeverity	TcEventSeverity [▶ 80]	Get	Inherited from FB_TcEventBase [▶ 41]	Returns the severity.
EventClass	GUID	Get	Inherited from FB_TcEventBase [▶ 41]	Returns the GUID of the event class.
ipArguments [▶ 47]	I_TcArguments [▶ 63]	Get	Inherited from FB_TcEventBase [▶ 41]	Returns the interface pointer for the arguments.
ipSourceInfo [▶ 47]	I_TcSourceInfo [▶ 78]	Get	Inherited from FB_TcEventBase [▶ 41]	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 [▶ 41]	Returns the ID of the event.
stEventEntry	TcEventEntry [▶ 80]	Get	Inherited from FB_TcEventBase [▶ 41]	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

3.8 FB_TcEventBase



This function block contains the basic implementation.

Syntax

Definition:

```
FUNCTION_BLOCK FB_TcEventBase
```

Methods

Name	Definition location	Description
EqualsTo [▶ 42]	Local	Compares the event with another instance.
EqualsToEventClass [▶ 43]	Local	Compares the event class of the event with another event class.
EqualsToEventEntry [▶ 43]	Local	Compares the event definition of the event with another event definition.
EqualsToEventEntryEx [▶ 44]	Local	Compares the event definition of the event with another event definition.
GetJsonAttribute [▶ 44]	Local	Returns the Json attribute.
Release [▶ 45]	Local	Releases the instance created by the EventLogger again.
RequestEventClassName [▶ 45]	Local	Requests the name of the event class.
RequestEventText [▶ 46]	Local	Returns the text for the event.

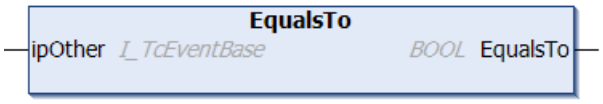
Properties

Name	Type	Access	Description
eSeverity	TcEventSeverity [▶ 80]	Get	Returns the severity.
EventClass	GUID	Get	Returns the GUID of the event class.
ipArguments [▶ 47]	I_TcArguments [▶ 63]	Get	Returns the interface pointer for the arguments.
ipSourceInfo [▶ 47]	I_TcSourceInfo [▶ 78]	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 [▶ 80]	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

3.8.1 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
```

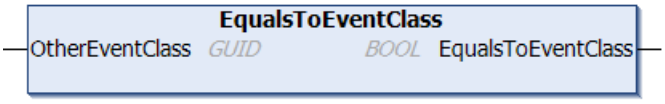
Inputs

Name	Type	Description
ipOther	I_TcEventBase [▶ 73]	Event to be compared

Return value

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

3.8.2 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
```

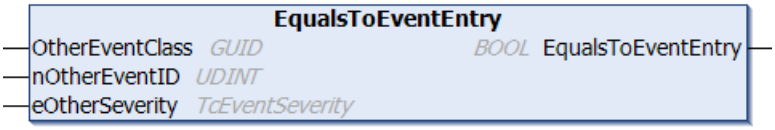
Inputs

Name	Type	Description
OtherEventClass	GUID	Event class to be compared.

Return value

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

3.8.3 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
```

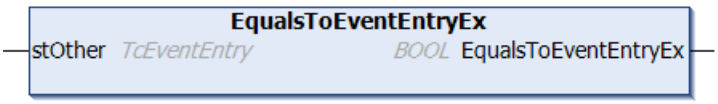
Inputs

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 [▶ 80]	Event severity of the event to be compared.

Return value

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

3.8.4 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 [▶ 80]	Event to be compared.

Return value

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

3.8.5 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.

3.8.6 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.

3.8.7 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.

3.8.8 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.

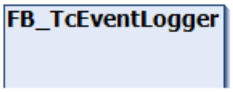
3.8.9 ipArguments

```
PROPERTY PUBLIC ipArguments : I_TcArguments
```

3.8.10 ipSourceInfo

```
PROPERTY ipSourceInfo : I_TcSourceInfo
```

3.9 FB_TcEventLogger



This function block represents the TwinCAT 3 EventLogger itself.

Syntax

Definition:

```
FUNCTION_BLOCK FB_TcEventLogger
```

Methods

Name	Description
ClearAlarms [▶ 48]	Clears active alarms.
ClearAllAlarms [▶ 48]	Calls Clear() for all alarms in the Raised state.
ClearLoggedEvents [▶ 49]	Clears logged events.
ConfirmAlarms [▶ 49]	
ConfirmAllAlarms [▶ 50]	Calls Confirm() for all alarms with the confirmation state WaitForConfirmation.
ExportLoggedEvents [▶ 50]	Exports logged events.
GetAlarm [▶ 51]	Returns the pointer to an existing alarm.
GetAlarmEx [▶ 51]	Returns the pointer to an existing alarm.
IsAlarmRaised [▶ 52]	Queries whether an alarm is in the Raised state.
IsAlarmRaisedEx [▶ 53]	Queries whether an alarm is in the Raised state.
SendMessage [▶ 53]	Sends a message.
SendMessage2 [▶ 54]	Sends a message.
SendMessageEx [▶ 55]	Sends a message.
SendMessageEx2 [▶ 55]	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

3.9.1 ClearAlarms

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

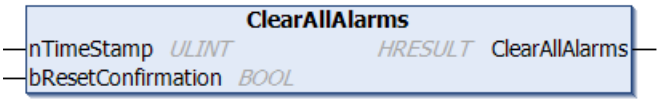
Inputs

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	

3.9.2 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
```

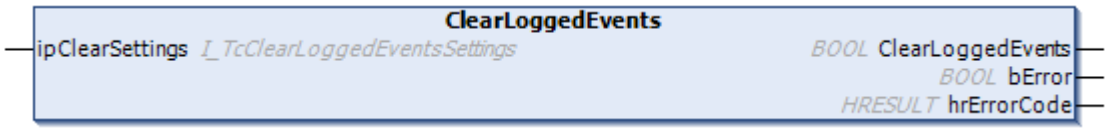
Inputs

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

3.9.3 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	

3.9.4 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	

3.9.5 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
```

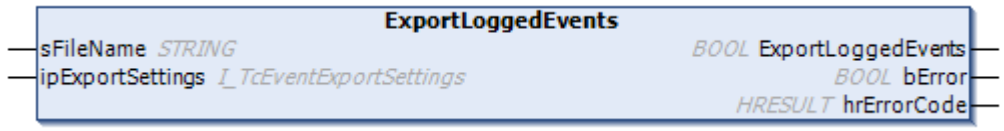
Inputs

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.

3.9.6 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 [▶ 20] 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.

3.9.7 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
```

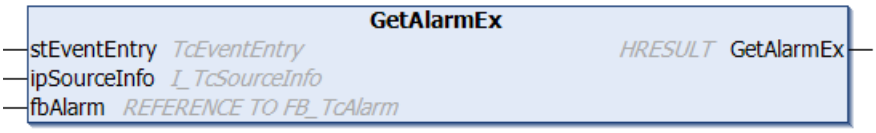
Inputs

Name	Type	Description
eventClass	GUID	GUID of the event class.
nEventId	UDINT	ID of the event
ipSourceInfo	I_TcSourceInfo [▶ 78]	Pointer to an ITcSourceInfo interface.
fbAlarm	REFERENCE TO FB_TcAlarm [▶ 31]	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.

3.9.8 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
```

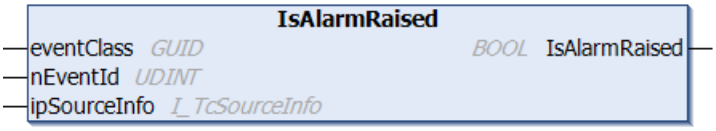
Inputs

Name	Type	Description
stEventEntry	TcEventEntry [▶ 80]	Event definition.
ipSourceInfo	I_TcSourceInfo [▶ 78]	Pointer to an ITcSourceInfo interface.
fbAlarm	REFERENCE TO FB_TcAlarm [▶ 31]	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.

3.9.9 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 [▶ 78]	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.

3.9.10 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
```

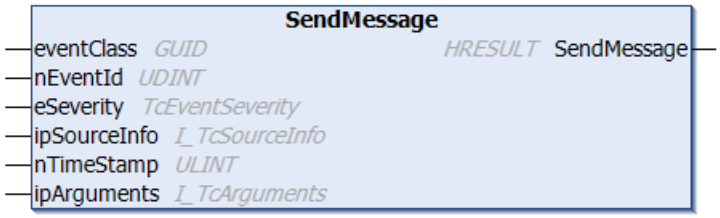
Inputs

Name	Type	Description
stEventEntry	UDINT	Event definition.
ipSourceInfo	I_TcSourceInfo [▶ 78]	Pointer to an ITcSourceInfo interface.

Return value

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

3.9.11 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 [▶ 80]	Severity of the event.
ipSourceInfo	I_TcSourceInfo [▶ 78]	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 [▶ 63]	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.

3.9.12 SendMessage2

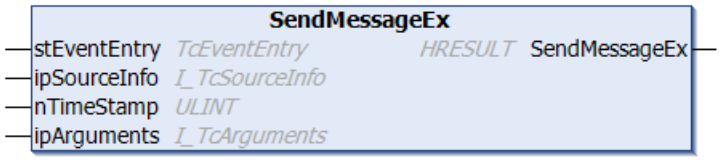
Inputs

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	

3.9.13 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
```

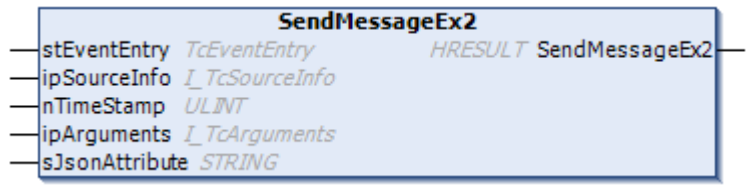
Inputs

Name	Type	Description
stEventEntry	TcEventEntry [▶ 80]	Event definition.
ipSourceInfo	I_TcSourceInfo [▶ 78]	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 [▶ 63]	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.

3.9.14 SendMessageEx2



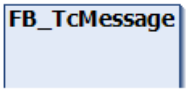
Inputs

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	

3.10 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 [▶ 41]

FB_TcMessage

Interfaces

Type	Description
I_TcMessage [▶ 77]	Provides methods and properties for the message handling.

Methods

Name	Definition location	Description
EqualsTo [▶ 42]	Inherited from FB_TcEventBase [▶ 41]	Compares the event with another instance.
EqualsToEventClass [▶ 43]	Inherited from FB_TcEventBase [▶ 41]	Compares the event class of the event with another event class.
EqualsToEventEntry [▶ 43]	Inherited from FB_TcEventBase [▶ 41]	Compares the event definition of the event with another event definition.
EqualsToEventEntryEx [▶ 44]	Inherited from FB_TcEventBase [▶ 41]	Compares the event definition of the event with another event definition.
GetJsonAttribute [▶ 44]	Inherited from FB_TcEventBase [▶ 41]	Returns the Json attribute.
Release [▶ 45]	Inherited from FB_TcEventBase [▶ 41]	Releases the instance created by the EventLogger again.
RequestEventClassName [▶ 45]	Inherited from FB_TcEventBase [▶ 41]	Requests the name of the event class.
RequestEventText [▶ 46]	Inherited from FB_TcEventBase [▶ 41]	Returns the text for the event.
Create [▶ 58]	Local	Creates a message instance in the EventLogger.
CreateEx [▶ 59]	Local	Creates a message instance in the EventLogger from an event definition.
SetJsonAttribute [▶ 59]	Local	Sets the Json attribute.
Send [▶ 78]	I_TcMessage [▶ 77]	Sends a message.

Properties

Name	Type	Access	Definition location	Description
eSeverity	TcEventSeverity [▶ 80]	Get	Inherited from FB_TcEventBase [▶ 41]	Returns the severity.
EventClass	GUID	Get	Inherited from FB_TcEventBase [▶ 41]	Returns the GUID of the event class.
ipArguments [▶ 47]	I_TcArguments [▶ 63]	Get	Inherited from FB_TcEventBase [▶ 41]	Returns the interface pointer for the arguments.
ipSourceInfo [▶ 47]	I_TcSourceInfo [▶ 78]	Get	Inherited from FB_TcEventBase [▶ 41]	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 [▶ 41]	Returns the ID of the event.
stEventEntry	TcEventEntry [▶ 80]	Get	Inherited from FB_TcEventBase [▶ 41]	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

3.10.1 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
```

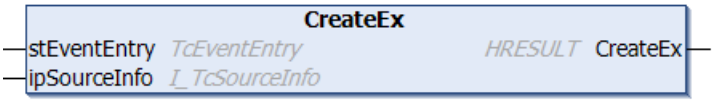
Inputs

Name	Type	Description
eventClass	GUID	GUID of the event class.
nEventId	UDINT	ID of the event.
eSeverity	TcEventSeverity [▶ 80]	Defines the severity.
ipSourceInfo	I_TcSourceInfo [▶ 78]	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.

3.10.2 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 [▶ 80]	Event definition.
ipSourceInfo	I_TcSourceInfo [▶ 78]	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.

3.10.3 SetJsonAttribute

This method sets the JSON attribute.

Syntax

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

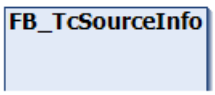
Inputs

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.

3.11 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 [▶ 78]	Provides read methods and read properties of a source information.

Methods

Name	Definition location	Description
Clear [▶ 61]	Local	Resets the source information.
ExtendName [▶ 61]	Local	Appends the transferred string to the name.
ResetToDefault [▶ 62]	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 [▶ 79]	I_TcSourceInfo [▶ 78]	Compares an instance with another instance.

Properties

Name	Type	Access	Definition location	Description
guid	GUID	Get	I_TcSourceInfo [▶ 78]	Returns the GUID of the source information.
guid	GUID	SET	Local	Sets the GUID as source information.
nId	UDINT	Get	I_TcSourceInfo [▶ 78]	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 [▶ 78]	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

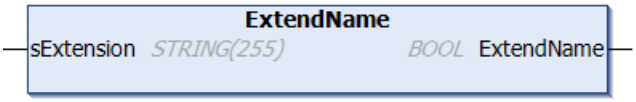
3.11.1 Clear

This method resets the source information.

Syntax

```
METHOD Clear
```

3.11.2 ExtendName



This method extends the name.

Syntax

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

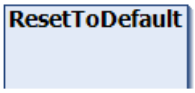
Inputs

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.

3.11.3 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
```

4 Interfaces

4.1 I_TcArguments

This interface defines methods for the argument handling.

Inheritance hierarchy

__SYSTEM.IQueryInterface

I_TcArguments

Methods

Name	Description
AddBlob [▶ 63]	Adds binary data as an argument.
AddBool [▶ 64]	Adds an argument of the type BOOL.
AddByte [▶ 64]	Adds an argument of the type BYTE.
AddDint [▶ 65]	Adds an argument of the type DINT.
AddDWord [▶ 65]	Adds an argument of the type DWORD.
AddEventReferenceId [▶ 66]	Adds a reference to another event as an argument.
AddEventReferenceIdGuid [▶ 66]	Adds a reference to another event as an argument.
AddInt [▶ 67]	Adds an argument of the type INT.
AddLInt [▶ 67]	Adds an argument of the type LINT.
AddLReal [▶ 68]	Adds an argument of the type LREAL.
AddReal [▶ 68]	Adds an argument of the type REAL.
AddSInt [▶ 69]	Adds an argument of the type SINT.
AddString [▶ 69]	Adds an argument of the type STRING.
AddUDint [▶ 70]	Adds an argument of the type UDINT.
AddUInt [▶ 70]	Adds an argument of the type INT.
AddULInt [▶ 70]	Adds an argument of the type ULINT.
AddUSInt [▶ 71]	Adds an argument of the type USINT.
AddWord [▶ 71]	Adds an argument of the type WORD.
AddWString [▶ 72]	Adds an argument of the type WSTRING.
Clear [▶ 72]	Removes all arguments.

Properties

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

4.1.1 AddBlob

This method adds binary data as an argument.

Syntax

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

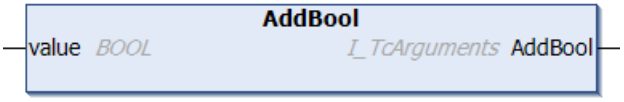
Inputs

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 [▶ 63]	Returns the I_TcArgument pointer again.

4.1.2 AddBool



This method adds an argument of the type BOOL.

Syntax

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

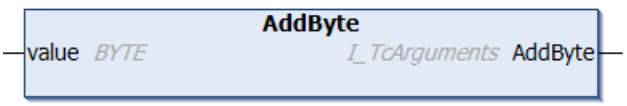
Inputs

Name	Type	Description
value	BOOL	Value to be added.

Return value

Name	Type	Description
AddBool	I_TcArguments [▶ 63]	Returns the I_TcArgument pointer again.

4.1.3 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 [▶ 63]	Returns the I_TcArgument pointer again.

4.1.4 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 [▶ 63]	Returns the I_TcArgument pointer again.

4.1.5 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 [▶ 63]	Returns the I_TcArgument pointer again.

4.1.6 AddEventReferenceId

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

Syntax

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

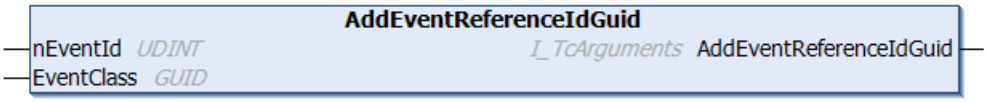
Inputs

Name	Type	Description
nEventId	UDINT	ID of the event.

Return value

Name	Type	Description
AddEventReferenceId	I_TcArguments [▶ 63]	Returns the I_TcArgument pointer again.

4.1.7 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
```

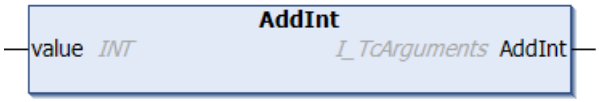
Inputs

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

Return value

Name	Type	Description
AddEventReferenceIdGuid	I_TcArguments [▶ 63]	Returns the I_TcArgument pointer again.

4.1.8 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 [▶ 63]	Returns the I_TcArgument pointer again.

4.1.9 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 [▶ 63]	Returns the I_TcArgument pointer again.

4.1.10 AddLReal

This method adds an argument of the type LREAL.

Syntax

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

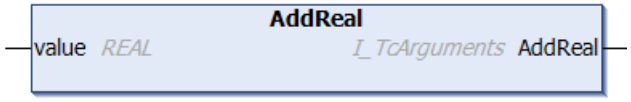
Inputs

Name	Type	Description
value	LREAL	Value to be added.

Return value

Name	Type	Description
AddLReal	I_TcArguments [▶ 63]	Returns the I_TcArgument pointer again.

4.1.11 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 [▶ 63]	Returns the I_TcArgument pointer again.

4.1.12 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 [▶ 63]	Returns the I_TcArgument pointer again.

4.1.13 AddString

This method adds an argument of the type STRING.

Syntax

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

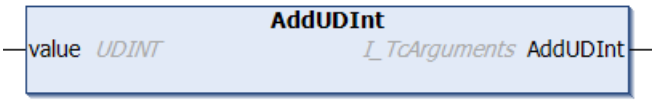
Inputs

Name	Type	Description
value	STRING	Value to be added.

Return value

Name	Type	Description
AddString	I_TcArguments [▶ 63]	Returns the I_TcArgument pointer again.

4.1.14 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 [▶ 63]	Returns the I_TcArgument pointer again.

4.1.15 AddUInt

This method adds an argument of the type INT.

Syntax

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

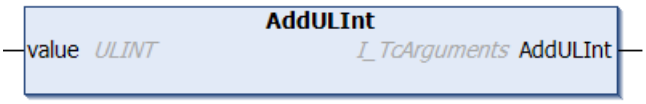
Inputs

Name	Type	Description
value	UINT	Value to be added.

Return value

Name	Type	Description
AddUInt	I_TcArguments [▶ 63]	Returns the I_TcArgument pointer again.

4.1.16 AddULInt



This method adds an argument of the type ULINT.

Syntax

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

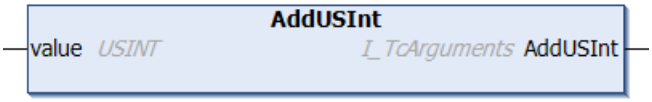
Inputs

Name	Type	Description
value	ULINT	Value to be added.

Return value

Name	Type	Description
AddULInt	I_TcArguments [▶ 63]	Returns the I_TcArgument pointer again.

4.1.17 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 [▶ 63]	Returns the I_TcArgument pointer again.

4.1.18 AddWord

This method adds an argument of the type WORD.

Syntax

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

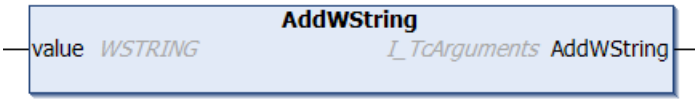
Inputs

Name	Type	Description
value	WORD	Value to be added.

Return value

Name	Type	Description
AddWord	I_TcArguments [▶ 63]	Returns the I_TcArgument pointer again.

4.1.19 AddWString



This method adds an argument of the type WSTRING.

Syntax

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

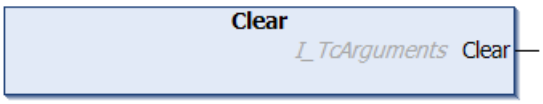
Inputs

Name	Type	Description
value	WSTRING	Value to be added.

Return value

Name	Type	Description
AddWString	I_TcArguments [▶ 63]	Returns the I_TcArgument pointer again.

4.1.20 Clear



This method removes all arguments.

Syntax

```
METHOD Clear : I_TcArguments
```

Return value

Name	Type	Description
Clear	I_TcArguments [▶ 63]	Returns the I_TcArgument pointer again.

4.2 I_TcEventBase

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

Methods

Name	Description
EqualsTo [▶ 73]	Compares the event with another instance.
EqualsToEventClass [▶ 74]	Compares the event class of the event with another event class.
EqualsToEventEntryEx [▶ 75]	Compares the event definition of the event with another event definition.
GetJsonAttribute [▶ 75]	Returns the Json attribute.
RequestEventClassName [▶ 76]	Requests the name of the event class.
RequestEventText [▶ 77]	Returns the text for the event.

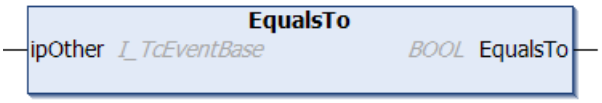
Properties

Name	Type	Access	Description
eSeverity	TcEventSeverity [▶ 80]	Get	Returns the severity.
EventClass	GUID	Get	Returns the GUID of the event class.
ipSourceInfo	I_TcSourceInfo [▶ 78]	Get	Returns a pointer to the source definition.
nEventId	UDINT	Get	Returns the ID of the event.
stEventEntry	TcEventEntry [▶ 80]	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

4.2.1 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
```

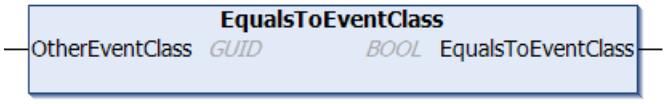
Inputs

Name	Type	Description
ipOther	I_TcEventBase [▶ 73]	Event to be compared

Return value

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

4.2.2 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
```

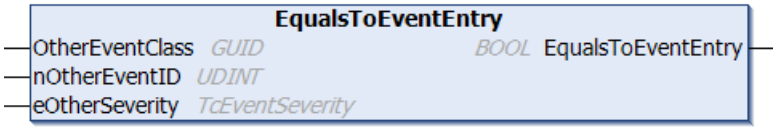
Inputs

Name	Type	Description
OtherEventClass	GUID	Event class to be compared.

Return value

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

4.2.3 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
```

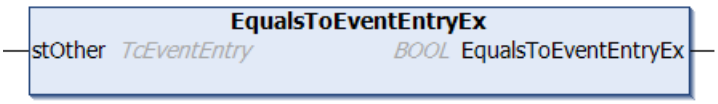
Inputs

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 [▶ 80]	Event severity of the event to be compared.

Return value

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

4.2.4 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 [▶ 80]	Event to be compared.

Return value

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

4.2.5 GetJsonAttribute

This method returns the JSON attribute.

Syntax

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

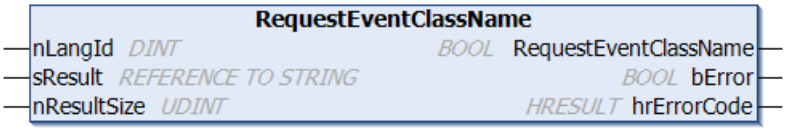
Inputs

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.

4.2.6 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.

4.2.7 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.

4.3 I_TcMessage

This interface provides methods and properties for the message handling.

Inheritance hierarchy

I_TcEventBase [▶ 73]

I_TcMessage

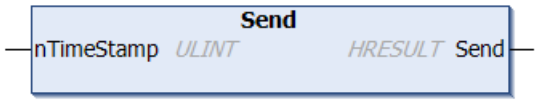
Methods

Name	Description
Send [▶ 78]	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

4.3.1 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

4.4 I_TcSourceInfo

This interface defines properties for an item of source information.

Methods

Name	Description
EqualsTo [▶ 79]	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

4.4.1 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 [▶ 78]	Items of source information to be compared

Return value

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

5 Data types

5.1 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,

5.2 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

5.3 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 Global lists

6.1 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 [▶ 80]	TcEventSeverity.Verbose
SUCCESS_EVENT	TcEventEntry [▶ 80]	STRUCT(uuidEventClass := EMPTY_EVENT_CLASS, nEventID := EMPTY_EVENT_ID, eSeverity := EMPTY_SEVERITY)

6.2 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 …

6.3 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 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 Examples

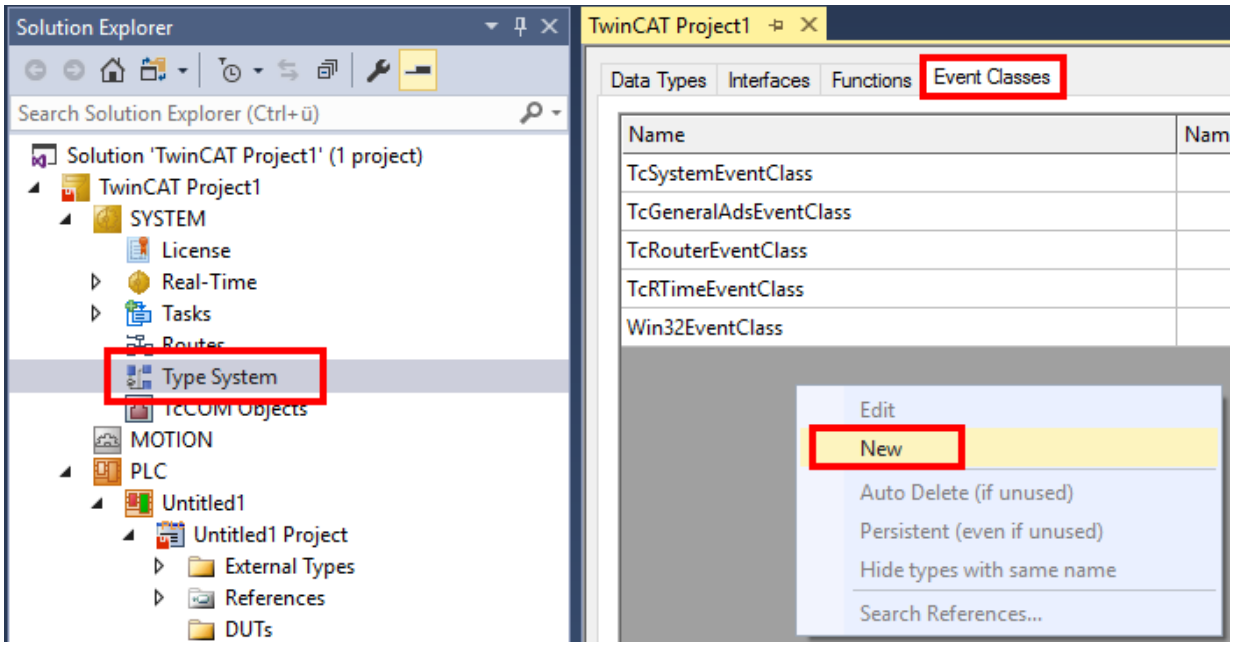
7.1 Tutorial

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 section in the work sequence.

Creating an event class in the TwinCAT type system

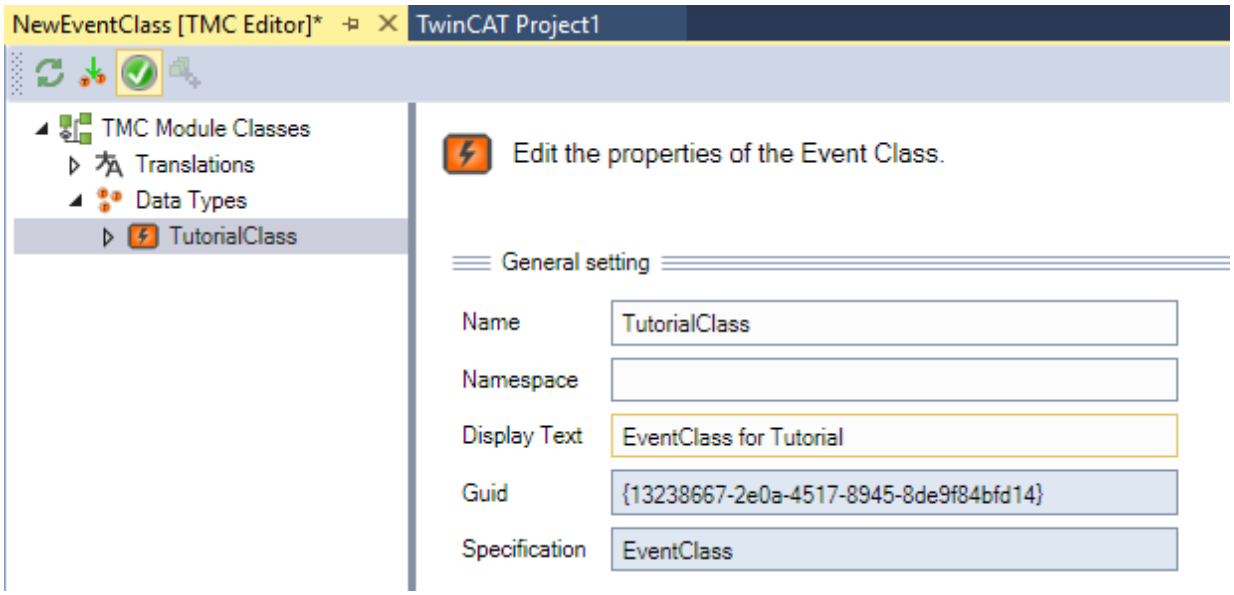
✓ A standard TwinCAT PLC project exists.

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.

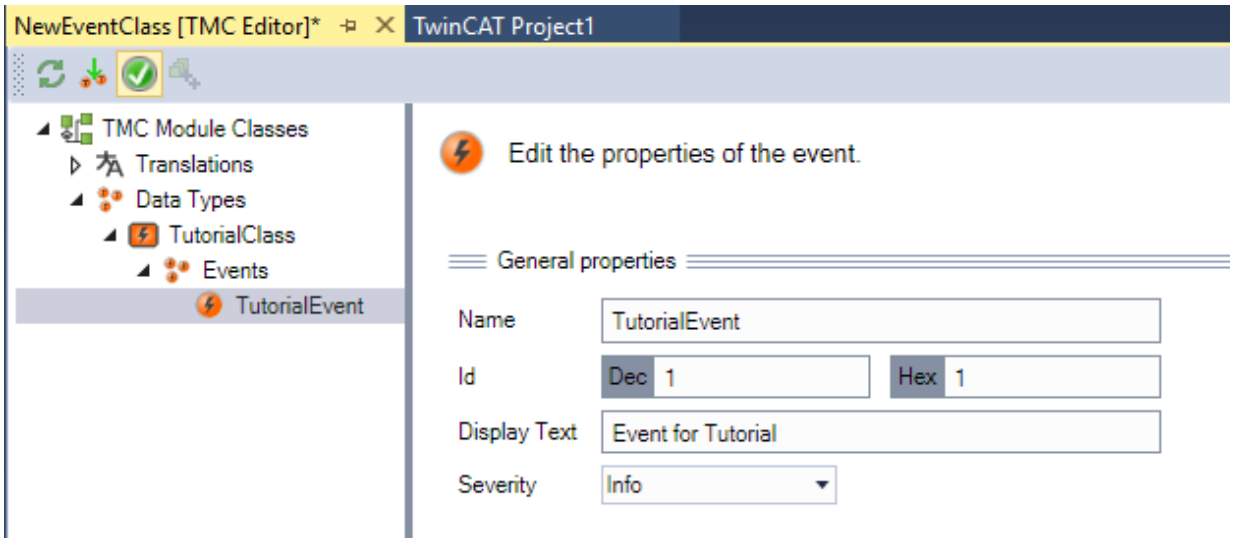


⇨ The TMC editor opens.

2. Give the event class a name and enter a display text.



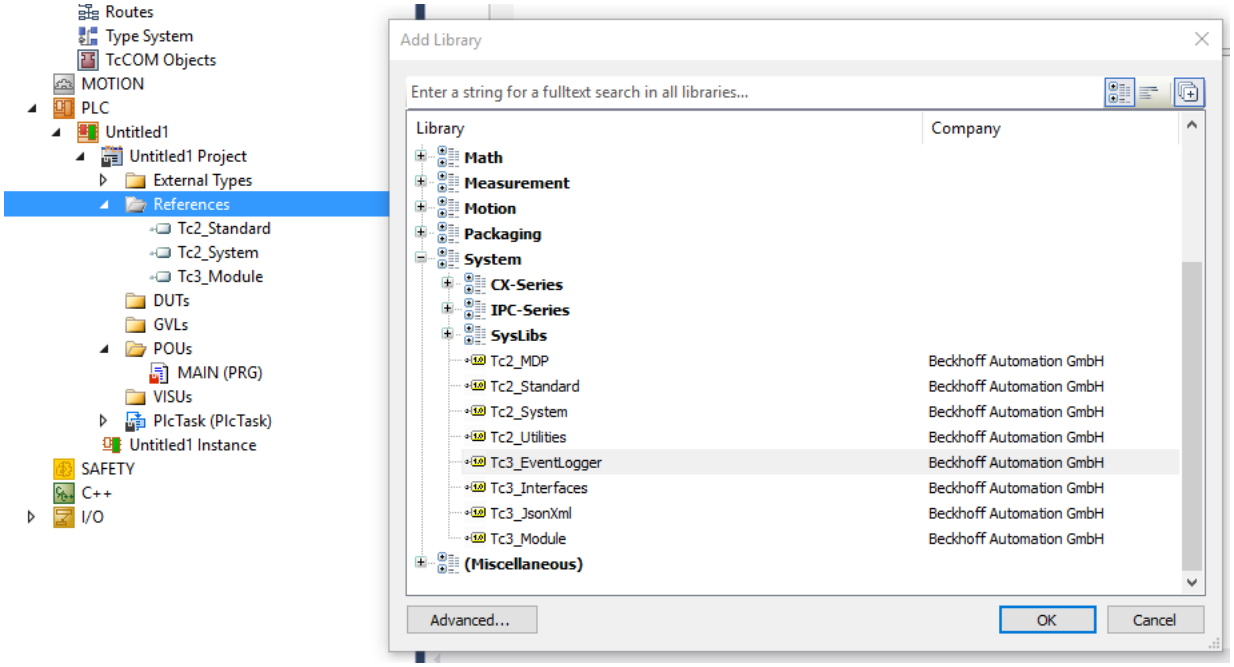
3. An event is already created below the event class. Give the event a name and enter a display text and the severity.



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
```

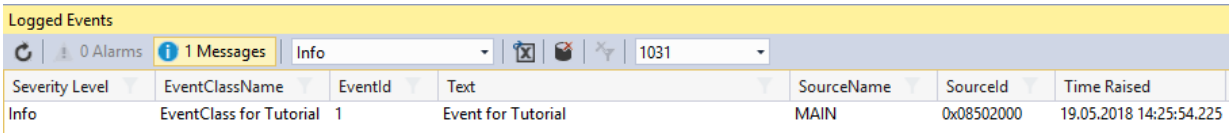
3. Implement the send procedure as shown in the code. The message is initialized once by means of the CreateEx method. Since the initialization requires dynamic resources it should not take place cyclically. The initialized message is subsequently sent using the Send method.

```
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.



7.2 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/tcplclib_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.

7.3 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/tcplclib_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)

7.4 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/tcplclib_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)

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