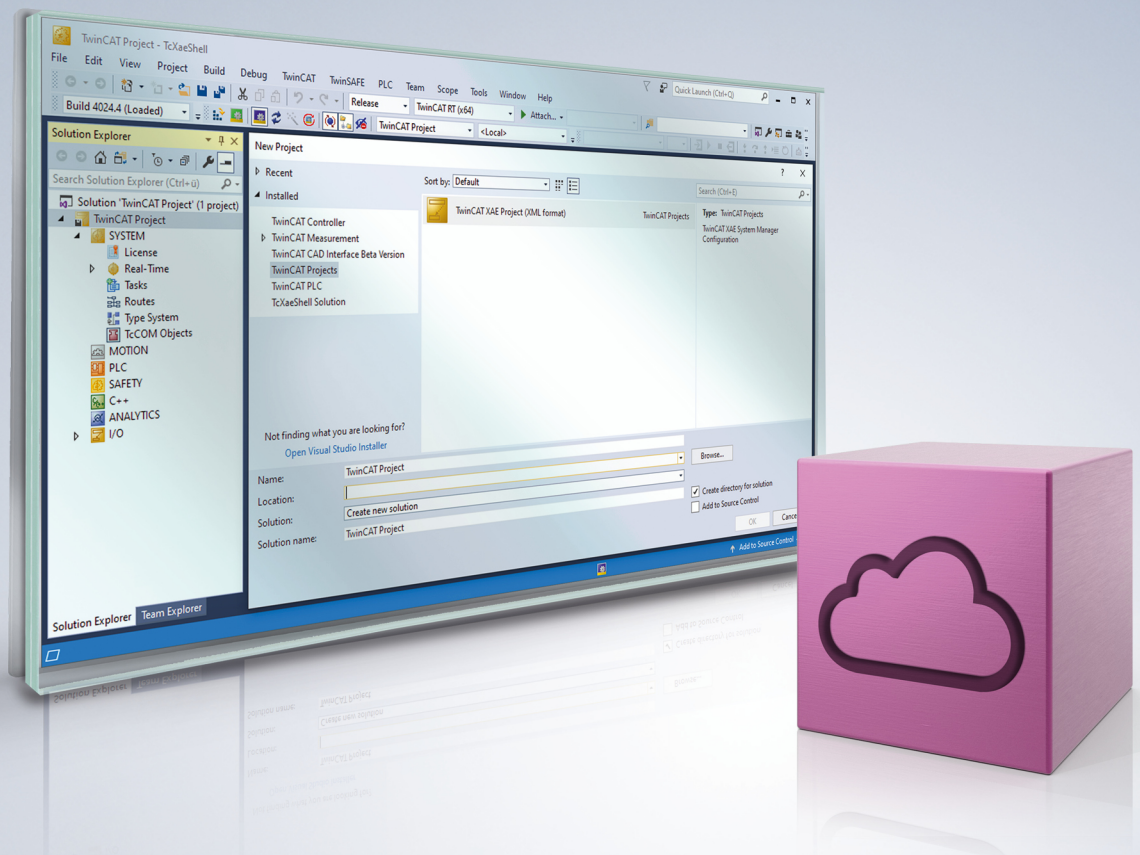
BECKHOFF New Automation Technology

Handbuch | DE

TF6710

TwinCAT 3 | IoT Functions



19.08.2024 | Version: 1.1.1

Inhaltsverzeichnis

1 Vorwort

1.1 Hinweise zur Dokumentation

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

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

Disclaimer

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

Marken

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

Patente

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

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

Copyright

1.2 Zu Ihrer Sicherheit

Sicherheitsbestimmungen

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

Haftungsausschluss

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

Qualifikation des Personals

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

Signalwörter

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

Warnungen vor Personenschäden

⚠ GEFAHR

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

⚠ WARNUNG

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

⚠ VORSICHT

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

Warnung vor Umwelt- oder Sachschäden

HINWEIS

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

Information zum Umgang mit dem Produkt

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

1.3 Hinweise zur Informationssicherheit

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

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

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

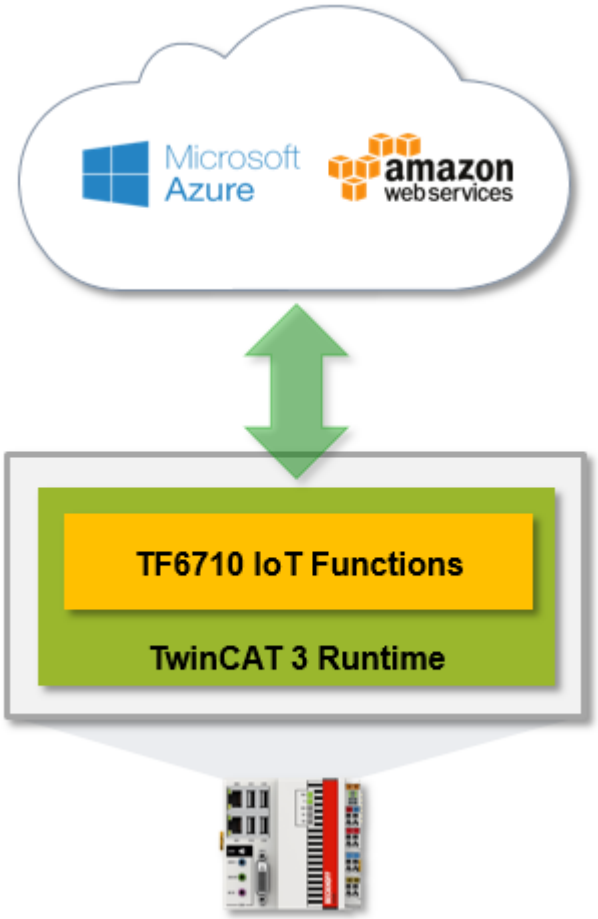
Um stets über Hinweise zur Informationssicherheit zu Produkten von Beckhoff informiert zu sein, abonnieren Sie den RSS Feed unter https://www.beckhoff.de/secinfo.

1.4 Ausgabestände der Dokumentation

Version	Änderung
1.1.x	Überarbeitung

2 Übersicht

TC3 IoT Functions ist ein Produkt für die TwinCAT 3 Runtime, das die bidirektionale Datenkommunikation mit Cloud-Diensten aus dem Maschinenprogramm heraus ermöglicht. Der Transportmechanismus hängt vom jeweiligen Cloud-Dienst ab. Dies kann MQTT, AMQP, ADS oder OPC UA sein.



3 Installation

3.1 Systemvoraussetzungen

TC3 IoT Functions benötigt den TC3 IoT Data Agent, der im Hintergrund läuft. Weitere Informationen finden Sie in der technischen Einführung [▶ 16], den Setup-Szenarien [▶ 9] und den Systemanforderungen für den TC3 IoT Data Agent.

Technische Daten	Beschreibung
Betriebssystem	Windows 7/10, Windows Embedded Standard 7, Windows Embedded Compact 7
Zielplattform	PC-Architektur (x86, x64, ARM)
.NET Framework	nicht erforderlich
TwinCAT Version [1]	TwinCAT 3 Build 4022.20 (oder höher)
TwinCAT-Installationslevel	TwinCAT 3 XAE, XAR
Erforderliche TwinCAT-Lizenz [2]	TF6710 TC3 IoT Functions
Erforderliches Setup	TF6720 TC3 IoT Data Agent Treiber und SPS-Bibliothek für TC3 IoT Functions sind automatisch in der TwinCAT-Basisinstallation enthalten.

[1] Version der TwinCAT 3 Runtime, auf der TC3 IoT Functions ausgeführt werden kann

[2] Obwohl TC3 IoT Functions (TF6710) in einer technischen Abhängigkeit zu TC3 IoT Data Agent (TF6720) steht, ist eine Lizenz für TF6720 nicht erforderlich.

3.2 Setup-Szenarien

TC3 IoT Functions und TC3 IoT Data Agent können entweder auf demselben Computer oder getrennt voneinander auf verschiedenen Geräten installiert werden.

TC3 IoT Functions und TC3 IoT Data Agent auf der Steuerung

In diesem Szenario laufen TC3 IoT Functions und TC3 IoT Data Agent auf der (gleichen) Steuerung. Dies ist das Standard-Setup-Szenario und es müssen keine weiteren Einstellungen vorgenommen werden. Für die Verwendung der TC3 IoT Functions SPS-Bibliothek ist eine TF6710-Lizenz auf der Steuerung erforderlich.

TC3 IoT Functions auf der Steuerung und TC3 IoT Data Agent auf einem Gateway-Gerät

In diesem Szenario laufen die TC3 IoT Functions auf dem Controller und der TC3 IoT Data Agent ist auf einem Gateway-Gerät installiert. Die Kommunikation zwischen TC3 IoT Functions und TC3 IoT Data Agent basiert auf ADS. Die SPS-Funktionsbausteine von TC3 IoT Functions müssen auf das Gateway-Gerät verweisen, auf dem der TC3 IoT Data Agent installiert ist (siehe Reference Data Agent [▶ 17]). Für das Gateway-Gerät sind keine zusätzlichen Lizenzen erforderlich. Eine TF6710-Lizenz ist nur für Geräte erforderlich, die die TC3 IoT Functions SPS-Bibliothek verwenden.

TC3 IoT Functions und TC3 IoT Data Agent auf einem Gateway-Gerät

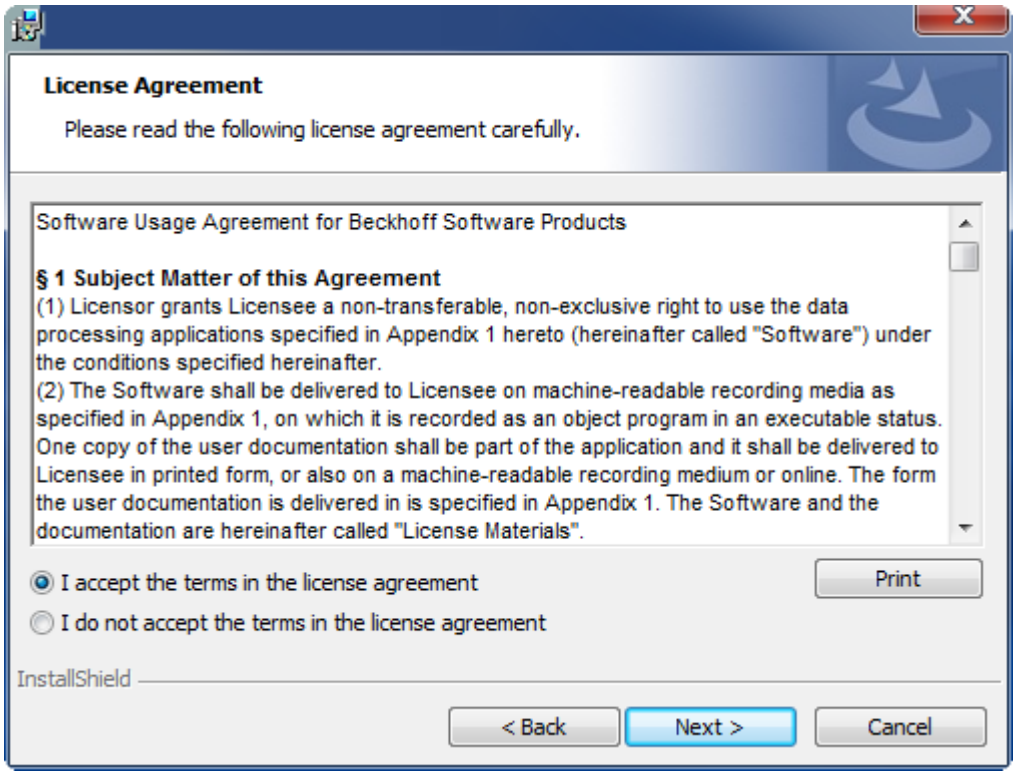
In diesem Szenario werden TC3 IoT Functions und TC3 IoT Data Agent auf einem Gateway-Gerät installiert und ausgeführt, z.B. in einem Edge-Szenario. Die Dateneingabe in das Edge Device kann beliebig erfolgen, von TCP-basierten Protokollen wie OPC UA bis hin zu völlig anderen Arten der Datenkommunikation. Eine TF6710-Lizenz ist auf dem Gateway-Gerät erforderlich, um die TC3 IoT Functions SPS-Bibliothek zu verwenden.

3.3 Installation

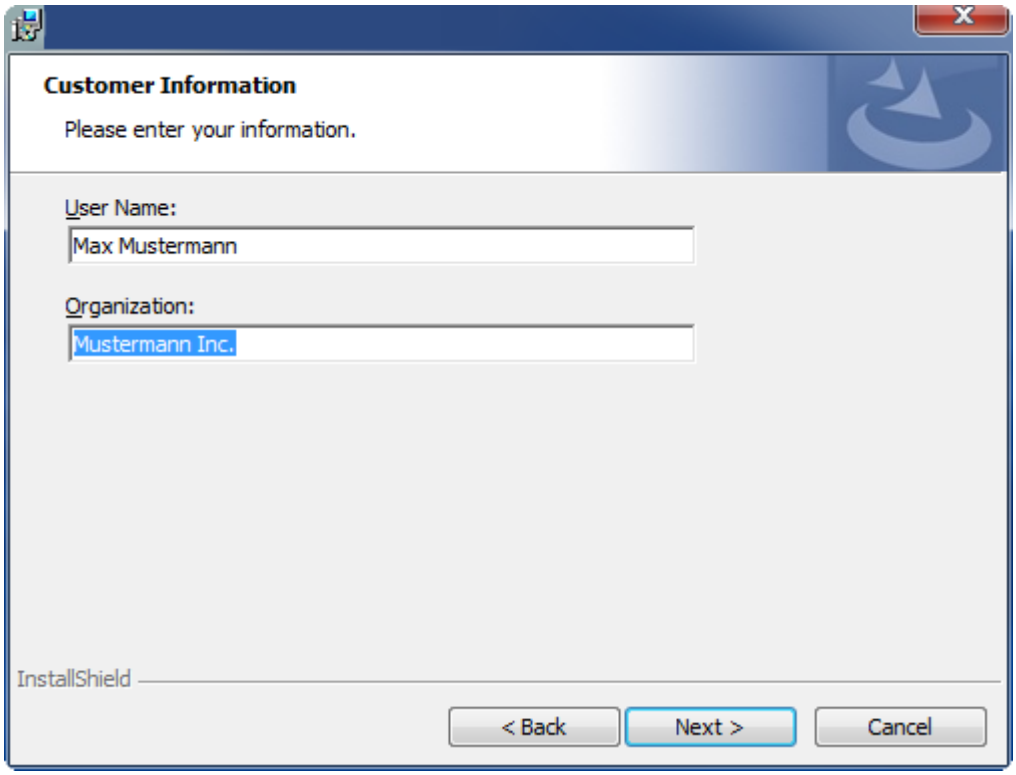
Nachfolgend wird beschrieben, wie die TwinCAT 3 Function für Windows-basierte Betriebssysteme installiert wird.

✓ Die Setup-Datei der TwinCAT 3 Function wurde von der Beckhoff-Homepage heruntergeladen.

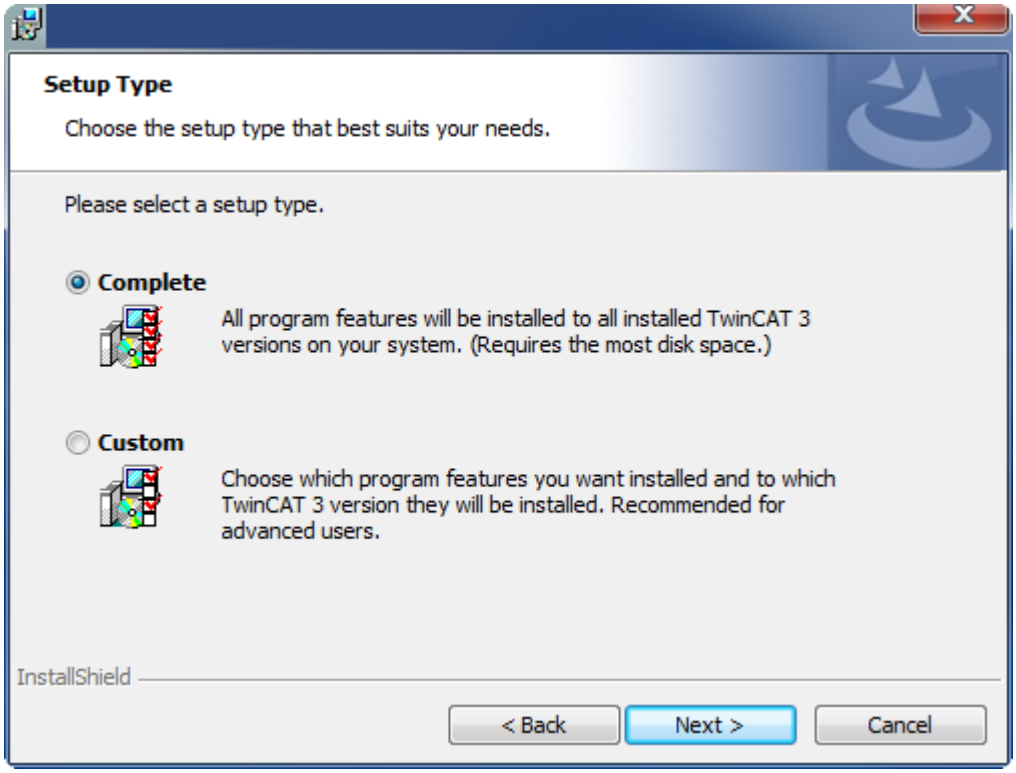
1. Führen Sie die Setup-Datei als Administrator aus. Wählen Sie dazu im Kontextmenü der Datei den Befehl **Als Administrator ausführen**.
   ⇨ Der Installationsdialog öffnet sich.
2. Akzeptieren Sie die Endbenutzerbedingungen und klicken Sie auf **Next**.



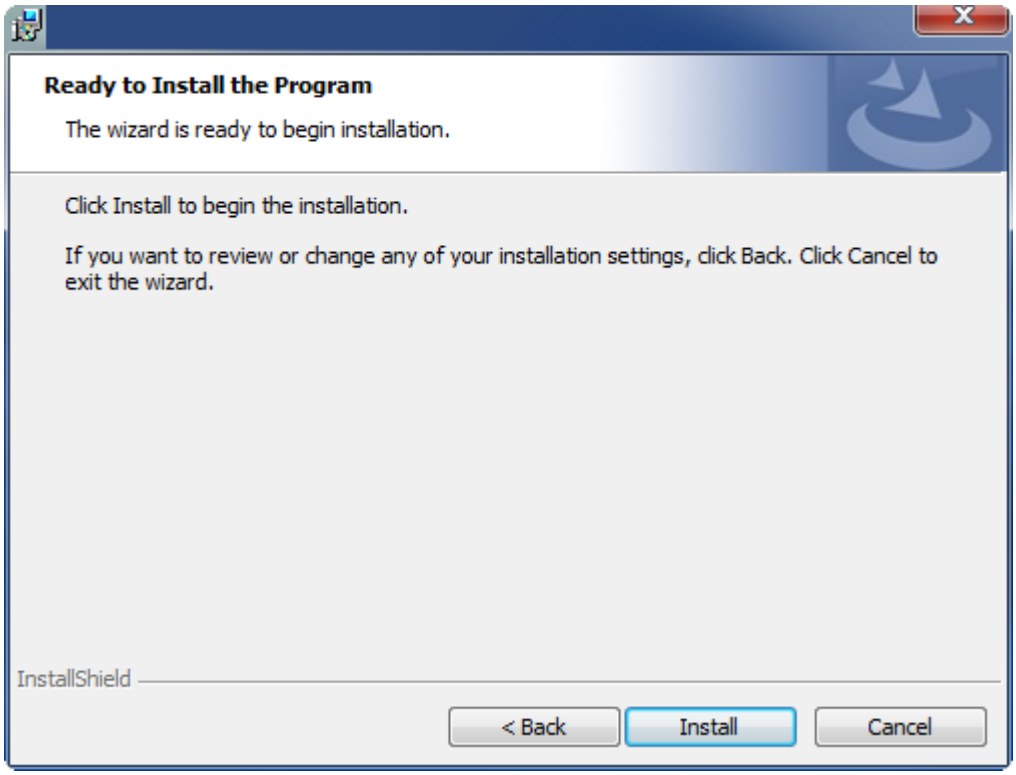
3. Geben Sie Ihre Benutzerdaten ein.



4. Wenn Sie die TwinCAT 3 Function vollständig installieren möchten, wählen Sie **Complete** als Installationstyp. Wenn Sie die Komponenten der TwinCAT 3 Function separat installieren möchten, wählen Sie **Custom**.

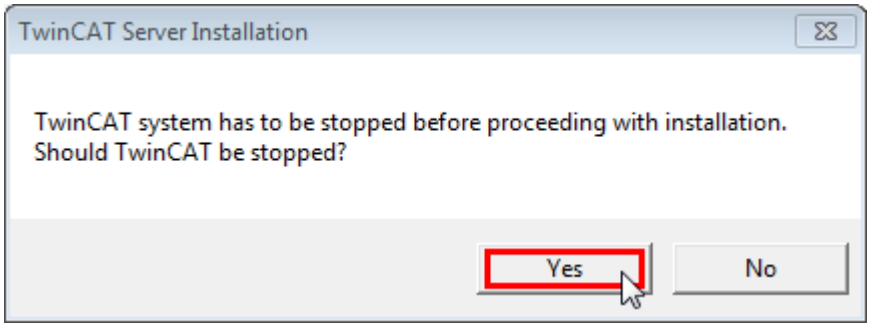


5. Wählen Sie **Next** und anschließend **Install**, um die Installation zu beginnen.

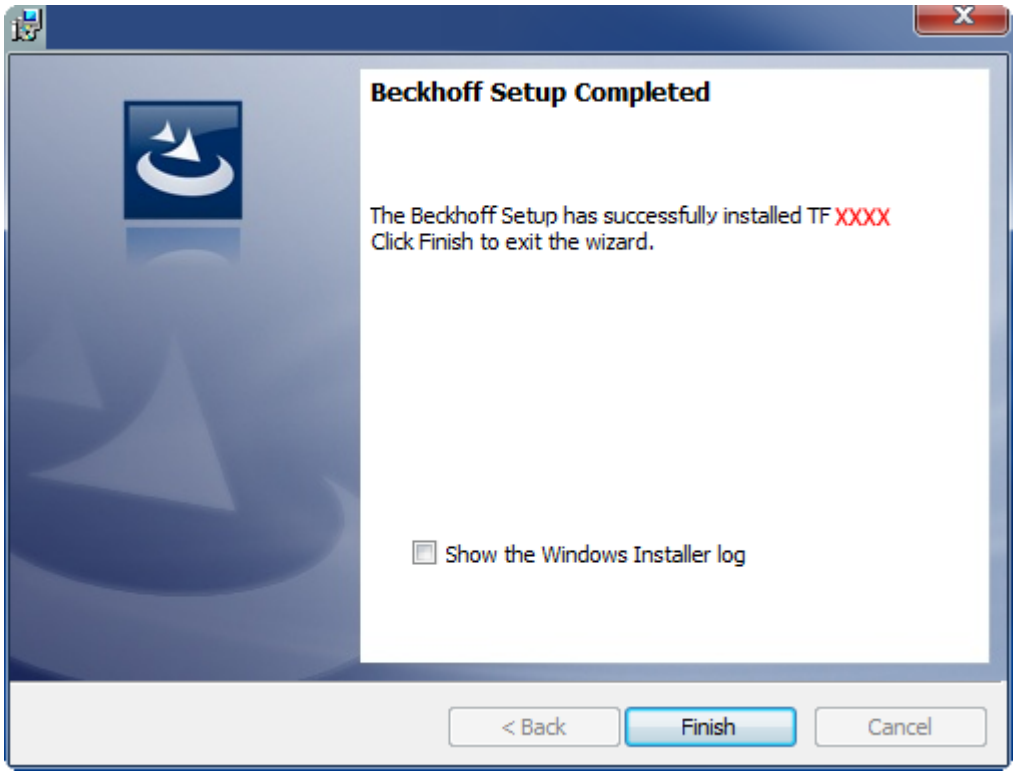


⇨ Ein Dialog weist Sie darauf hin, dass das TwinCAT-System für die weitere Installation gestoppt werden muss.

6. Bestätigen Sie den Dialog mit **Yes.**



7. Wählen Sie **Finish**, um das Setup zu beenden.



⇨ Die TwinCAT 3 Function wurde erfolgreich installiert.

3.4 Installation unter TwinCAT 4026

TwinCAT Package Manager

Wenn Sie TwinCAT 3.1 Build 4026 (und höher) auf dem Betriebssystem Microsoft Windows verwenden, können Sie diese Function über den TwinCAT Package Manager installieren, siehe Dokumentation zur Installation.

Normalerweise installieren Sie die Function über den entsprechenden Workload; dennoch können Sie die im Workload enthaltenen Pakete auch einzeln installieren. Diese Dokumentation beschreibt im Folgenden kurz den Installationsvorgang über den Workload.

Kommandozeilenprogramm TcPkg

Über das TcPkg **C**ommand **L**ine **I**nterface (CLI) können Sie sich die verfügbaren Workloads auf dem System anzeigen lassen:

```
tcpkg list -t workload
```

Über das folgende Kommando können Sie den Workload der TF6710 TC3 IoT Functions-Function installieren.

```
tcpkg install TF6710.IoTFunctions.XAE
```

TwinCAT Package Manager UI

Über das **U**ser **I**nterface (UI) können Sie sich alle verfügbaren Workloads anzeigen lassen und diese bei Bedarf installieren.
Folgen Sie hierzu den entsprechenden Anweisungen in der Oberfläche.

3.5 Lizenzierung

Die TwinCAT 3 Function ist als Vollversion oder als 7-Tage-Testversion freischaltbar. Beide Lizenztypen sind über die TwinCAT-3-Entwicklungsumgebung (XAE) aktivierbar.

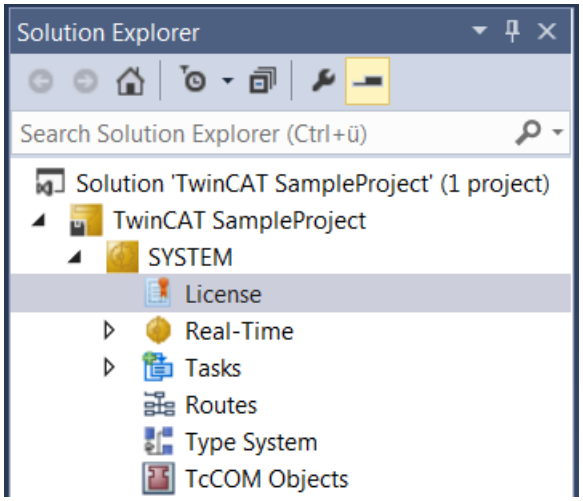
Lizenzierung der Vollversion einer TwinCAT 3 Function

Die Beschreibung der Lizenzierung einer Vollversion finden Sie im Beckhoff Information System in der Dokumentation „TwinCAT-3-Lizenzierung“.

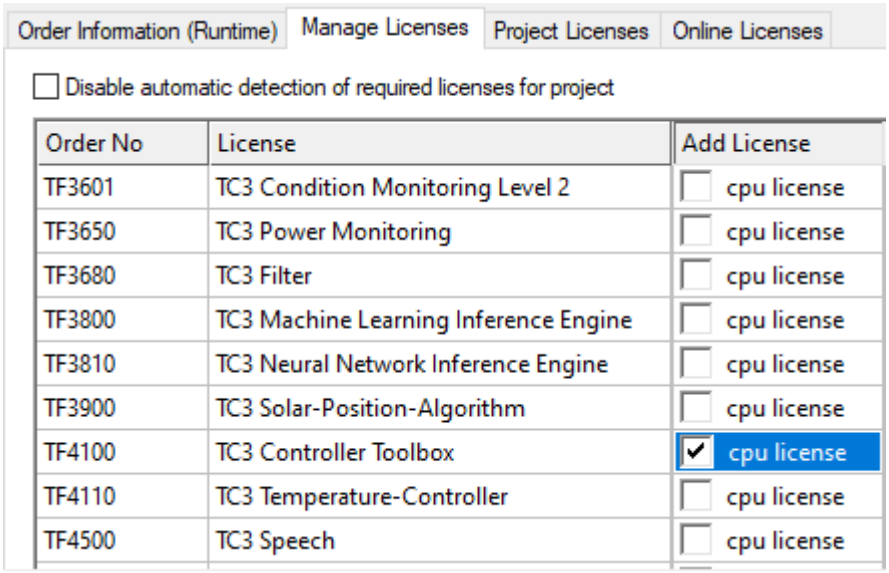
Lizenzierung der 7-Tage-Testversion einer TwinCAT 3 Function

Eine 7-Tage-Testversion kann nicht für einen TwinCAT-3-Lizenz-Dongle freigeschaltet werden.

1. Starten Sie die TwinCAT-3-Entwicklungsumgebung (XAE).
2. Öffnen Sie ein bestehendes TwinCAT-3-Projekt oder legen Sie ein neues Projekt an.
3. Wenn Sie die Lizenz für ein Remote-Gerät aktivieren wollen, stellen Sie das gewünschte Zielsystem ein. Wählen Sie dazu in der Symbolleiste in der Drop-down-Liste **Choose Target System** das Zielsystem aus.
   ⇨ Die Lizenzierungseinstellungen beziehen sich immer auf das eingestellte Zielsystem. Mit der Aktivierung des Projekts auf dem Zielsystem werden automatisch auch die zugehörigen TwinCAT-3-Lizenzen auf dieses System kopiert.
4. Klicken Sie im **Solution Explorer** im Teilbaum **SYSTEM** doppelt auf **License.**

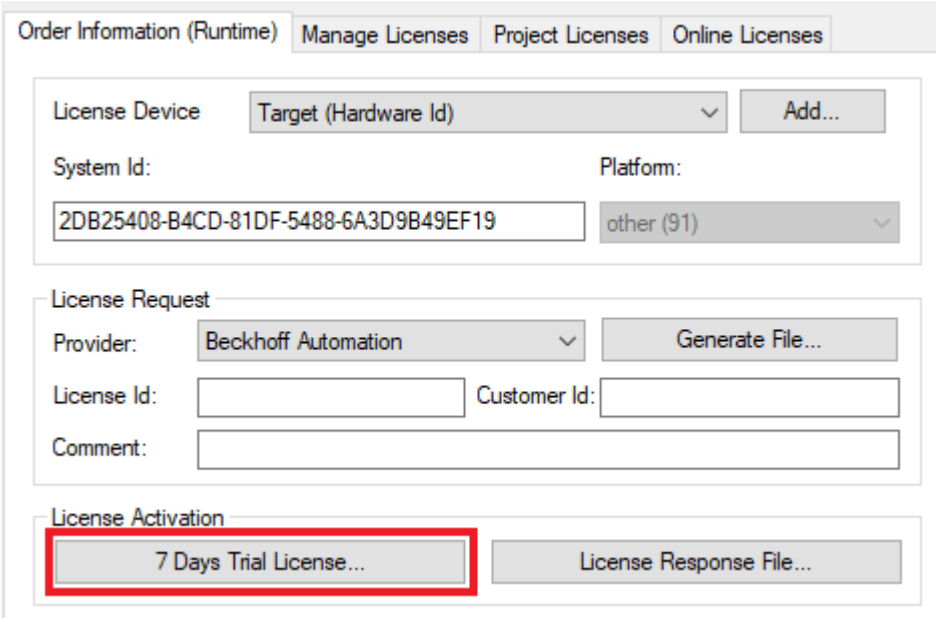


   ⇨ Der TwinCAT-3-Lizenzmanager öffnet sich.
5. Öffnen Sie die Registerkarte **Manage Licenses.** Aktivieren Sie in der Spalte **Add License** das Auswahlkästchen für die Lizenz, die Sie Ihrem Projekt hinzufügen möchten (z. B. „TF4100 TC3 Controller Toolbox“).

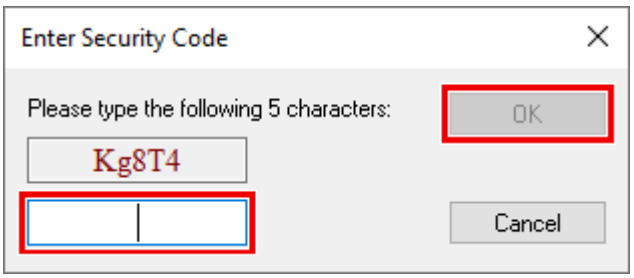


Order No	License	Add License
TF3601	TC3 Condition Monitoring Level 2	cpu license
TF3650	TC3 Power Monitoring	cpu license
TF3680	TC3 Filter	cpu license
TF3800	TC3 Machine Learning Inference Engine	cpu license
TF3810	TC3 Neural Network Inference Engine	cpu license
TF3900	TC3 Solar-Position-Algorithm	cpu license
TF4100	TC3 Controller Toolbox	cpu license
TF4110	TC3 Temperature-Controller	cpu license
TF4500	TC3 Speech	cpu license

6. Öffnen Sie die Registerkarte **Order Information (Runtime).**
   ⇨ In der tabellarischen Übersicht der Lizenzen wird die zuvor ausgewählte Lizenz mit dem Status „missing“ angezeigt**.**
7. Klicken Sie auf **7 Days Trial License...**, um die 7-Tage-Testlizenz zu aktivieren.



   ⇨ Es öffnet sich ein Dialog, der Sie auffordert, den im Dialog angezeigten Sicherheitscode einzugeben.



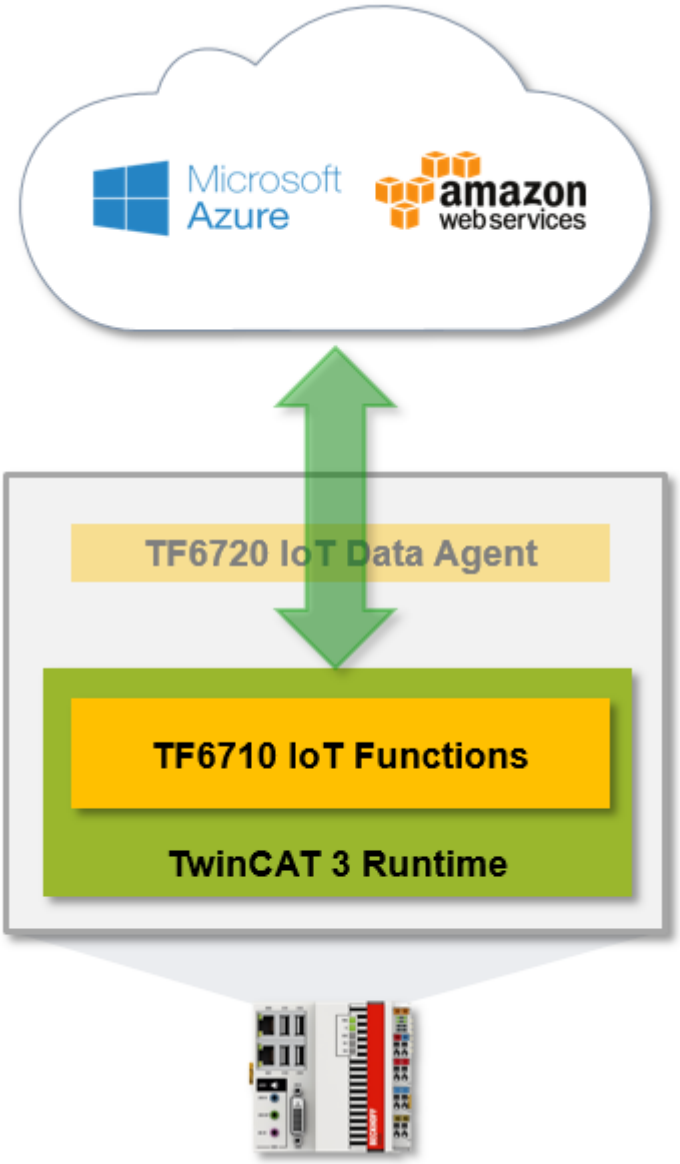
8. Geben Sie den Code genauso ein, wie er angezeigt wird, und bestätigen Sie ihn.
9. Bestätigen Sie den nachfolgenden Dialog, der Sie auf die erfolgreiche Aktivierung hinweist.
   ⇨ In der tabellarischen Übersicht der Lizenzen gibt der Lizenzstatus nun das Ablaufdatum der Lizenz an.
10. Starten Sie das TwinCAT-System neu.

⇨ Die 7-Tage-Testversion ist freigeschaltet.

4 Technische Einführung

TC3 IoT Functions ist ein Produkt für die TwinCAT 3 Runtime, das die bidirektionale Datenkommunikation mit der Cloud ermöglicht. Um einen Konnektivitätskanal mit einem Cloud-Dienst aufzubauen, nutzt das Produkt im Hintergrund technische Funktionalitäten des TC3 IoT Data Agent (TF6720). TC3 IoT Functions kann daher mit jedem Cloud-Dienst verwendet werden, der auch vom TC3 IoT Data Agent unterstützt wird. Beachten Sie, dass der TC3 IoT Data Agent nur für die Bereitstellung der Konnektivitätsschicht zur Cloud verwendet wird und keine TF6720-Lizenz erworben werden muss, um TF6710 zu verwenden.

TC3 IoT Functions und TC3 IoT Data Agent müssen nicht auf demselben System laufen, sondern können auch getrennt voneinander auf verschiedenen Systemen laufen (siehe Setup-Szenarios).



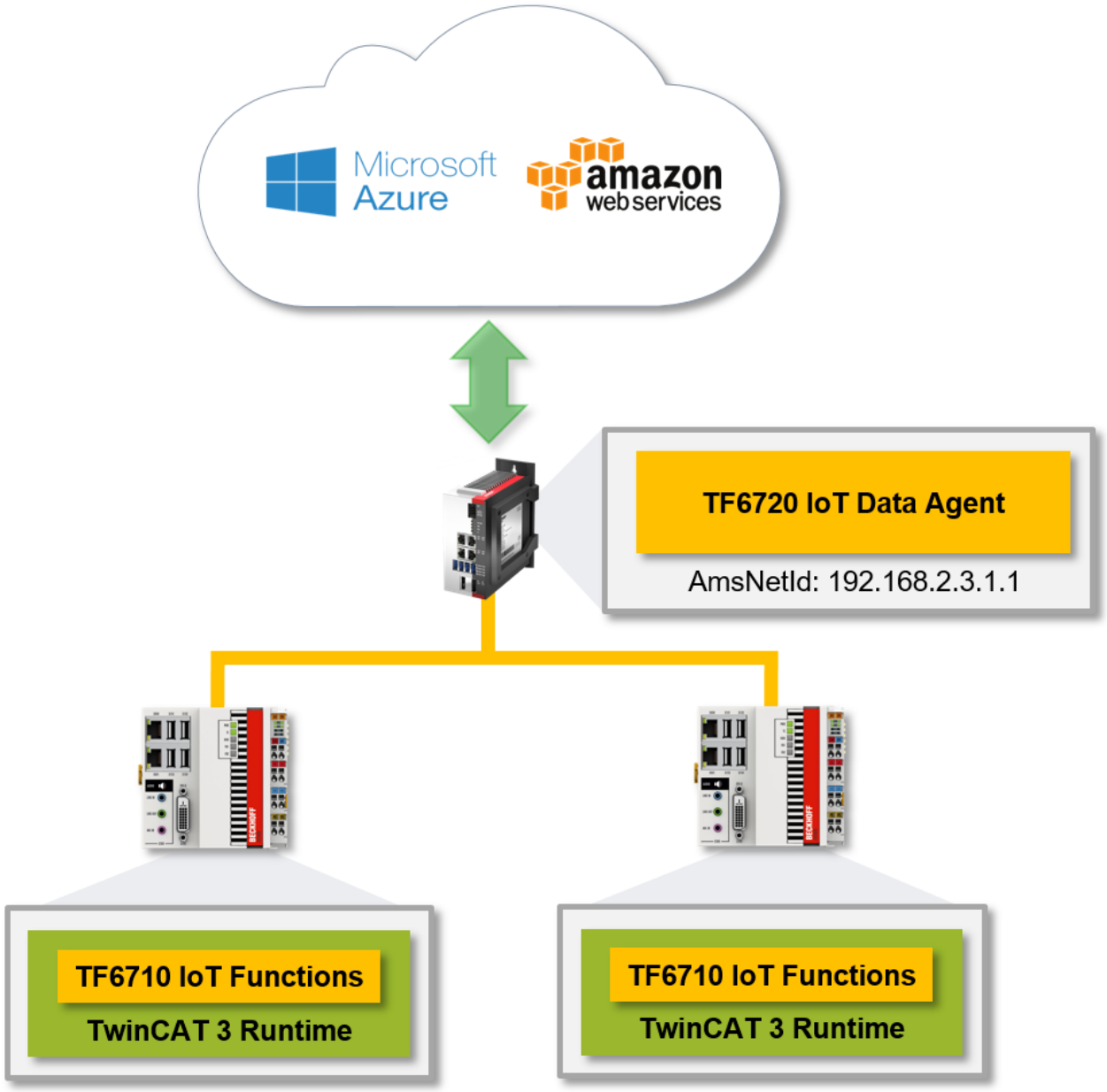
Features

TC3 IoT Functions umfasst die folgenden Features:

- Lese-/Schreibvorgänge für Nachrichten, die an einen Cloud-Dienst gesendet/von diesem empfangen werden sollen
- Bequeme Handhabung von Timeout, Fehlern und Retry

4.1 Reference Data Agent

Werden TC3 IoT Functions und TC3 IoT Data Agent separat installiert und befinden sich die Komponenten auf unterschiedlichen Rechnern, müssen die Funktionsbausteine den Installationsort des TC3 IoT Data Agent angeben. Dies kann über die AMS Net ID des Gerätes erfolgen, das den TC3 IoT Data Agent ausführt. Tragen Sie einfach die AMS Net ID des Gerätes, das den TC3 IoT Data Agent ausführt, in den entsprechenden Eingangsparameter des Funktionsbausteins FB_IotFunctions_Connector [▶ 38] ein. Beachten Sie, dass in diesem Fall eine ADS Route zwischen den Geräten erstellt werden muss.



Wenn TC3 IoT Data Agent und TC3 IoT Functions auf demselben Gerät laufen, wird standardmäßig die lokale AMS NET ID verwendet.

```
fbConnector : FB_IotFunctions_Connector := (sAmsNetId := 192.168.2.3.1.1');
```

4.2 Kommunikationsmuster

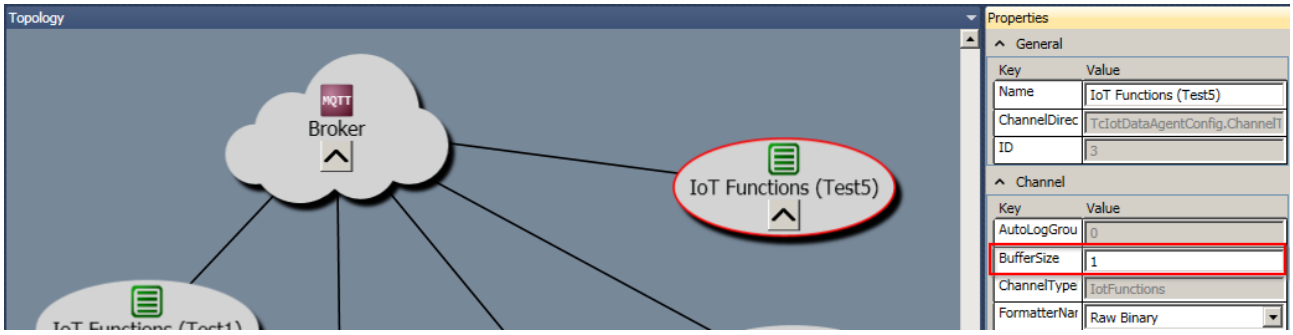
Die obere Verbindungsschicht zu einem Cloud-Dienst basiert in der Regel auf Publisher/Subscriber-Mustern, in einigen Fällen kann sie aber auch auf Polling-Mustern basieren. TC3 IoT Data Agent implementiert die Konnektivität mit dem Cloud-Dienst und stellt diesen Zugang über einen abstrahierten Kommunikationsweg zu TC3 IoT Functions zur Verfügung. Die SPS-Bibliothek von TC3 IoT Functions nutzt diese Schnittstelle dann über ein Polling-Lese-/Schreibmuster.

Beispiel

TC3 IoT Functions sollte verwendet werden, um Nachrichten von Azure IoT Hub zu lesen. Die Konnektivität mit diesem Cloud-Dienst basiert auf dem Publisher/Subscriber-Muster. Das bedeutet, dass der TC3 IoT Data Agent mit Zugangsdaten für den IoT Hub Service konfiguriert ist und somit eine Verbindung zu diesem herstellt. TC3 IoT Functions pollt dann den TC3 IoT Data Agent nach eingehenden Nachrichten ab.

Nachrichtenpuffer

Der TC3 IoT Data Agent enthält einen Nachrichtenpuffer für eingehende Nachrichten, der von TC3 IoT Functions genutzt werden soll. Dieser Nachrichtenpuffer kann direkt auf dem entsprechenden TC3 IoT Functions Kanal eingestellt werden.



4.3 Workflow Programmierung

In diesem Abschnitt wird beschrieben, wie die Funktionsbausteine der TC3 IoT Functions SPS-Bibliothek in bewährter Weise zu verwenden sind. Der Workflow der Programmierung umfasst die folgenden Schritte:

- Aufruf von FB_IotFunctions_Connector.Execute() [▶ 18]
- Überprüfung auf allgemeine Fehler [▶ 18]
- Überprüfung auf Lese-/Schreibfehler [▶ 19]
- Daten lesen [▶ 19]
- Daten schreiben [▶ 19]

Die Code-Ausschnitte in diesem Abschnitt basieren auf den folgenden Deklarationen:

```
PROGRAM MAIN
VAR
    fbConnector : FB_IotFunctions_Connector;
    fbRead      : FB_IotFunctions_Message;
    fbWrite     : FB_IotFunctions_Message;
    nReadError  : UINT;
    nWriteError : UINT;
    nReadData   : UINT;
    nIn         : UINT;
    nOut        : UINT;
    bWrite      : BOOL;
END_VAR
```

Aufruf von FB_IotFunctions_Connector.Execute()

Es wird dringend empfohlen, die Execute-Methode auf der FB_IotFunctions_Connector-Funktionsbausteininstanz als eine der ersten Anweisungen aufzurufen. Diese Methode ist für die Behandlung von Online-Change, die Behandlung von Timeouts und die Kommunikation mit dem TC3 IoT Data Agent zuständig.

```
fbConnector.Execute();
```

Überprüfung auf allgemeine Fehler

Sobald die Execute-Methode ausgelöst wurde, sollte der bError-Ausgang des FB_IotFunctions_Connector-Funktionsbausteins überprüft werden, um Fehler bei der Kommunikation mit dem TC3 IoT Data Agent festzustellen.

```
IF fbConnector.bError THEN
  ...
END_IF
```

Überprüfung auf Lese-/Schreibfehler

Wenn der bError-Ausgang der Funktionsbausteininstanz des Connectors TRUE ist, prüfen Sie die individuelle Fehlermeldung des Request-Funktionsbausteins auf Fehler, um sie richtig zu behandeln. Um einen Fehler zu quittieren und zu verhindern, dass er erneut auftritt, rufen Sie die Reset-Methode des Funktionsbausteins auf.

```
IF fbConnector.bError THEN
  IF fbRead.bError THEN
    nReadError := nReadError + 1;
    fbRead.Reset();
  END_IF
  IF fbWrite.bError THEN
    nWriteError := nWriteError + 1;
    fbWrite.Reset();
  END_IF
END_IF
```

Bevor Sie einen neuen Vorgang wie Lesen oder Schreiben starten, überprüfen Sie alle relevanten Statusabfragen, da diese Vorgänge alle Statusinformationen in den Strukturen ST_IotFunctionsMessage und ST_IotFunctionsRequest zurücksetzen.

Daten lesen

Der Lesevorgang empfängt Daten aus dem Puffer und speichert diese Daten in einem Symbol. Der Wert des angegebenen Symbols wird verwendet, um neue Daten mit früheren Daten zu vergleichen. Wenn neue Daten empfangen wurden, wird bDataAvailable auf TRUE gesetzt. Wenn sich der aktuelle Wert des Symbols vom vorherigen Wert unterscheidet, haben sich die Daten geändert und bDataChanged wird auf TRUE gesetzt.

Das heißt, wenn Sie auf neue Daten reagieren wollen, die sich von den zuvor empfangenen Daten unterscheiden, prüfen Sie den Ausgang bDataChanged. Dieser Ausgang wird nur dann auf TRUE gesetzt, wenn der Empfangspuffer eine Änderung des zuvor empfangenen Pakets anzeigt.

```
IF fbRead.bDataChanged THEN
  ...
END_IF
```

Wenn Sie daran interessiert sind, Daten unabhängig von den Daten zu empfangen, die sich von der zuvor empfangenen Payload unterscheiden, prüfen Sie stattdessen den Ausgang bDataAvailable.

```
IF fbRead.bDataAvailable THEN
  ...
END_IF
```

Daten schreiben

Das folgende Beispiel zeigt, wie man einen bedingten Schreibvorgang einrichtet, der nur ausgeführt wird, wenn das bWrite-Flag auf TRUE gesetzt ist. Stellen Sie vor dem Aufruf der Write-Methode sicher, dass der Funktionsbaustein nicht aktiv ist. Wenn Sie in eine belegte Funktionsbausteininstanz schreiben, kehrt der Aufruf zurück, ohne den Schreibvorgang zu starten.

```
IF bWrite THEN
  IF NOT fbWrite.bBusy THEN
    bWrite := FALSE;
    fbWrite.Write(ADR(nOut), SIZEOF(nOut));
  END_IF
 END_IF
```

4.4 Synchronisierung von Nachrichtenoperationen

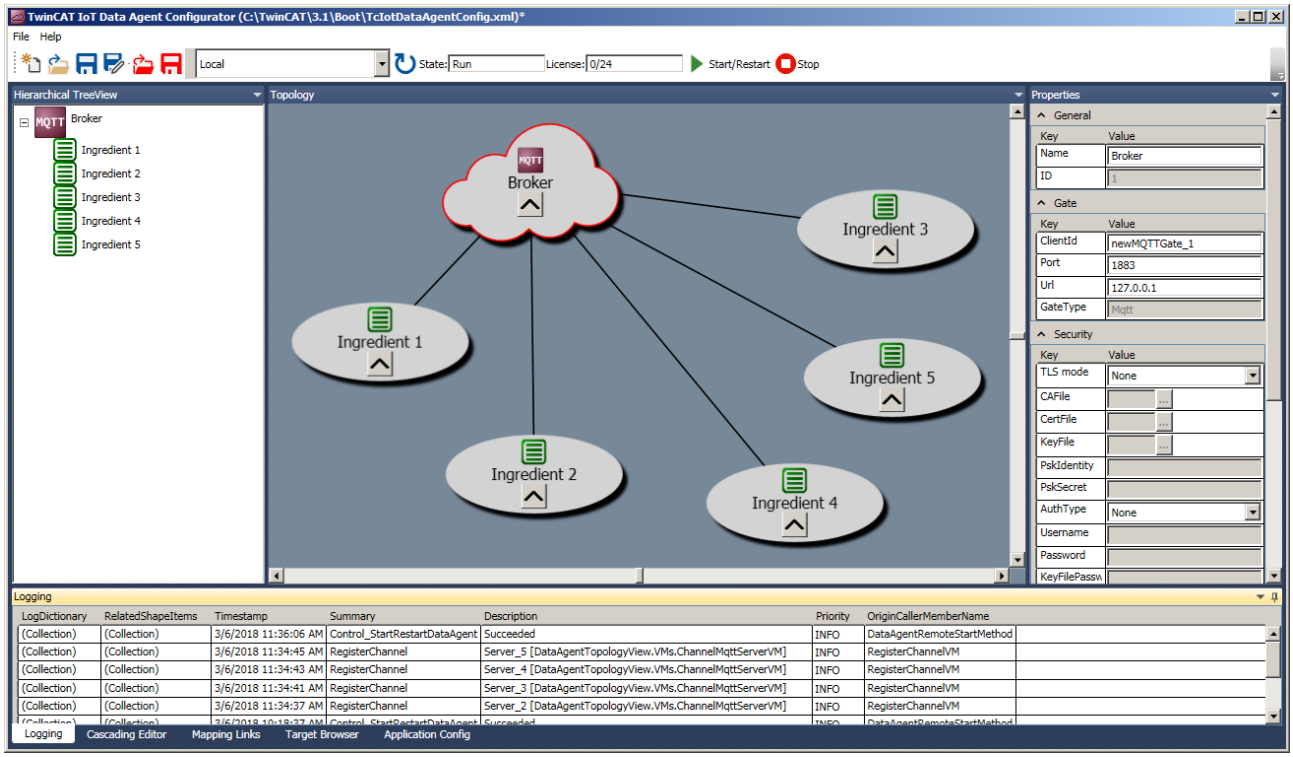
TC3 IoT Functions umfasst Funktionen zur Synchronisierung mehrerer Nachrichtenoperationen. Dies kann in Szenarien, in denen Daten aus verschiedenen Datenquellen/Kanälen eingehen, sehr nützlich sein. Der Funktionsbaustein FB_IotFunctions_Request [▶ 42] bietet verschiedene Mechanismen zur Synchronisierung von Nachrichtenoperationen.

Beispiel-Szenario

Ein Cocktailmixer stellt verschiedene Zutaten zur Verfügung, und jeder Cocktail besteht aus fünf Zutaten. Durch Drücken einer Taste wird dem Rezept eine Zutat hinzugefügt. Der Cocktailmixer beginnt mit dem Mixen des Cocktails, nachdem die fünfte Zutat ausgewählt wurde. Wenn eine Zutat ausgewählt wird, wird eine MQTT-Nachricht an ein (anderes) Topic veröffentlicht. Diese Nachrichten sollten von TC3 IoT Functions empfangen werden.

Grundlegendes Setup

TC3 IoT Data Agent ist mit einem MQTT-Gate und fünf verschiedenen Kanälen für TC3 IoT Functions konfiguriert. Jeder Kanal ist für ein "Zutaten-Topic" konfiguriert.



Um den Lesevorgang für fünf verschiedene Zutaten zu synchronisieren, wird eine Instanz von FB_IotFunctions_Request erstellt. Dann wird für jeden der fünf Kanäle eine Instanz von FB_IotFunctions_Message erstellt.

```
fbRequestIngredients : FB_IotFunctions_Request;

fbReadIngredient : ARRAY[0..4] OF FB_IotFunctions_Message := [(nChannelId := 1),(nChannelId := 2),
(nChannelId := 3),(nChannelId := 4),(nChannelId := 5)];
nIn : ARRAY[0..4] OF STRING;
```

Die synchronisierte Anfrage kann dann wie folgt erstellt werden:

```
IF NOT fbRequestIngredients.bBusy THEN
  IF fbRequestIngredients.bError THEN
    ...
  ELSE
    IF fbRequestIngredients.bTimeoutOccurred THEN
      ...
    ELSE
      // request was successful
      ...
    END_IF
  END_IF

  // Prepare next read operation for ingredients
  fbRequestIngredients.Create();
  fbRequestIngredients.EnqueueRead(ADR(fbReadIngredient [0]),ADR(nIn[0]),SIZEOF(nIn[0]));
  fbRequestIngredients.EnqueueRead(ADR(fbReadIngredient [1]),ADR(nIn[1]),SIZEOF(nIn[1]));
  fbRequestIngredients.EnqueueRead(ADR(fbReadIngredient [2]),ADR(nIn[2]),SIZEOF(nIn[2]));
  fbRequestIngredients.EnqueueRead(ADR(fbReadIngredient [3]),ADR(nIn[3]),SIZEOF(nIn[3]));
```

```
  fbRequestIngredients.EnqueueRead(ADR(fbReadIngredient [4]),ADR(nIn[4]),SIZEOF(nIn[4]));
  fbRequestIngredients.Execute();
END_IF
```

Sample04 zeigt den vollständigen Beispielcode (siehe Beispiele [▶ 48]).

Synchronisationsbedingungen

Die Funktionsbausteininstanz fbRequest enthält verschiedene Synchronisationsbedingungen. Diese können verwendet werden, um festzustellen, ob die Lesevorgänge innerhalb der Anfrage erfolgreich waren, einen Fehler verursachten oder zu einem Timeout führten. Beachten Sie, wie die verschiedenen Timeout- und Retry-Einstellungen zur Unterstützung dieses Anwendungsfalls beitragen können (siehe Timeout Einstellungen [▶ 21]).

Bedingung	Beschreibung
bBusy	TRUE: Die Anfrage ist noch in Bearbeitung und nicht alle Vorgänge wurden erfolgreich abgeschlossen FALSE: Die Anfrage wurde beendet. Um herauszufinden, ob ein Fehler oder eine Zeitüberschreitung aufgetreten ist, müssen weitere Flags ausgewertet werden.
bTimeoutOccurred	TRUE: RequestTimeout wurde für mindestens einen Vorgang ausgelöst. FALSE: Es ist kein Timeout aufgetreten.
bError	TRUE: Bei mindestens einem Vorgang ist ein Fehler aufgetreten. FALSE: Es ist kein Fehler aufgetreten.
Flags der einzelnen Nachrichtenoperationen	Darüber hinaus können die Flags (error, success, bDataAvailable, bDataChanged) jeder Nachrichtenoperation analysiert werden, um herauszufinden, ob eine Anfrageoperation erfolgreich war oder nicht.

4.5 Timeout Einstellungen

Die Funktionsbausteine von TC3 IoT Functions enthalten mehrere Timeout Einstellungen, die dem Benutzer helfen können, Fehler bei der Wiederholung von Operationen zu behandeln. Im folgenden Abschnitt werden die verschiedenen Timeout Einstellungen näher erläutert.

RequestTimeout

Der RequestTimeout kann entweder global auf eine Instanz von FB_IotFunctions_Connector oder individuell auf eine Instanz von FB_IotFunctions_Message gesetzt werden. Individuelle Einstellungen haben immer Vorrang vor globalen Einstellungen. Der RequestTimeout arbeitet eng mit der Einstellung MessageRetryInterval zusammen.

Der RequestTimeout gibt an, wann ein Nachrichtenoperation (Lesen/Schreiben) abläuft. Wenn beispielsweise RequestTimeout auf 10000 eingestellt ist, wird ein Lesevorgang nach 10 Sekunden abgebrochen, wenn keine Daten empfangen wurden. Wenn die Daten innerhalb von 10 Sekunden empfangen werden, wird der Vorgang sofort beendet.

MessageRetryInterval

Das MessageRetryInterval kann entweder global auf eine Instanz von FB_IotFunctions_Connector oder individuell auf eine Instanz von FB_IotFunctions_Message gesetzt werden. Individuelle Einstellungen haben immer Vorrang vor globalen Einstellungen. Das MessageRetryInterval arbeitet eng mit der Einstellung RequestTimeout zusammen.

Das MessageRetryInterval gibt die Zeitspanne in [ms] an, in dem eine Nachrichtenoperation erneut versucht wird. Die Obergrenze des Intervalls ist immer der RequestTimeout. Wenn zum Beispiel RequestTimeout auf 10000 und MessageRetryInterval auf 1000 gesetzt ist, wird ein Lesevorgang zehnmal wiederholt, bevor RequestTimeout ausgelöst wird und der Lesevorgang ein Zeitlimit erreicht.

CumulativeTimeout

Der CumulativeTimeout kann auf Instanzen von FB_IotFunctions_Request gesetzt werden, wenn Nachrichtenoperationen synchronisiert werden (siehe Synchronisierung von Nachrichtenoperationen [▶ 19]).

Der Anwendungsfall sieht folgendermaßen aus (Beispiel):

- Es gibt drei Lesevorgänge, die synchronisiert werden sollten.
- Jeder Lesevorgang kommt von einem anderen Kanal
- RequestTimeout ist global auf 10000 ms eingestellt
- CumulativeTimeout ist global auf 3000 ms eingestellt
- Nach 8000 ms kommt die erste Nachricht über Kanal 1 an.
- Da bis zum Erreichen der RequestTimeout-Zeit nur noch 2000 ms verbleiben, wird die CumulativeTimeout-Zeit zur verbleibenden RequestTimeout-Zeit addiert, um ein Timeout der gesamten Anfrage zu verhindern. Die RequestTimeout-Zeit beträgt dann 5000 ms. Dadurch erhält die Anfrage mehr Zeit, um Daten über die beiden anderen Lesevorgänge zu sammeln. Wenn nach 5000 ms keine Daten für die beiden anderen Operationen vorliegen, wird die gesamte Anfrage abgebrochen.

5 Konfiguration

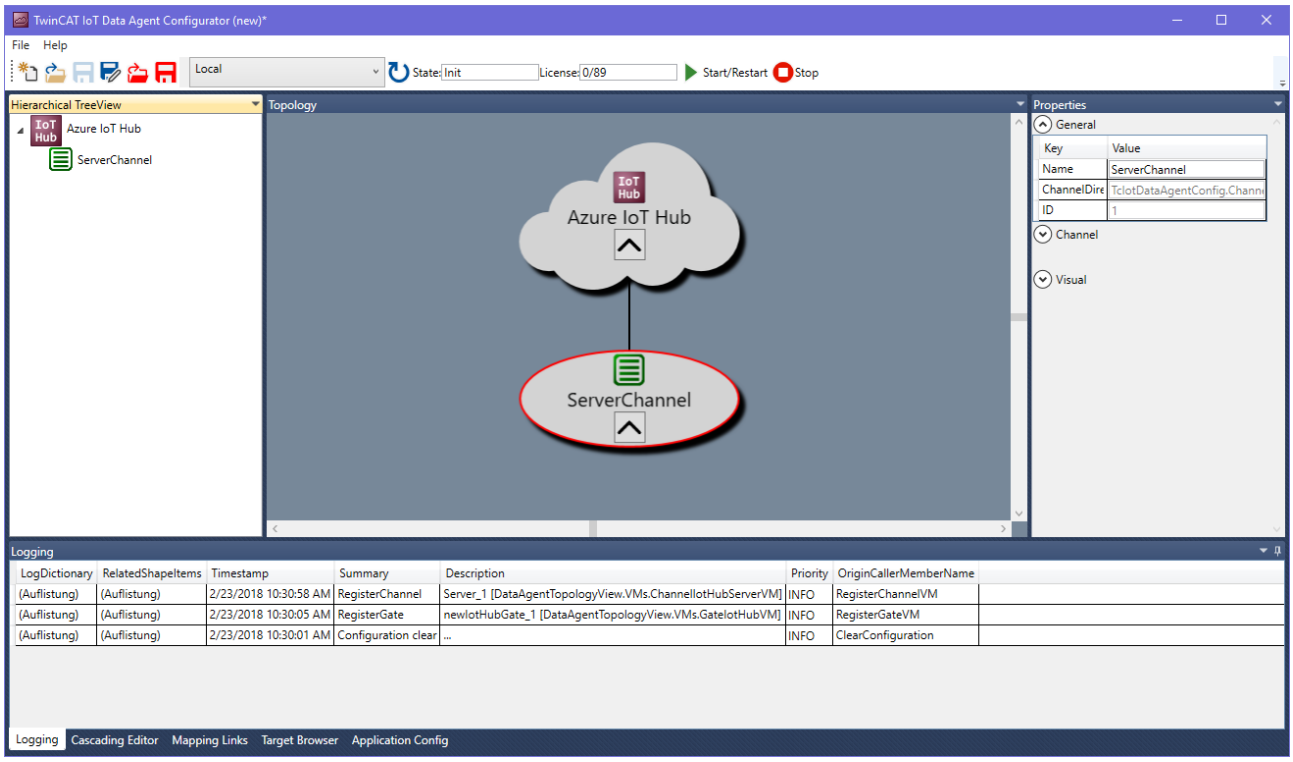
5.1 Übersicht

TC3 IoT Functions nutzt die Funktionalitäten des TC3 IoT Data Agent, um sich mit Cloud-Diensten zu verbinden. Um den Zugang zu einem Cloud-Dienst zu konfigurieren und die Zugangsdaten bereitzustellen, kann der TC3 IoT Data Agent Konfigurator verwendet werden.

Innerhalb des Konfigurators sind die folgenden Konfigurationsschritte erforderlich:

1. Erstellen Sie ein Gate (z. B. Azure IoT Hub) und konfigurieren Sie alle erforderlichen Anmeldeinformationen für die Verbindung.
2. Erstellen Sie einen Server-Kanal am neuen Gate und notieren Sie die Channel-ID.
3. Aktivieren Sie die Konfiguration und starten Sie den TC3 IoT Data Agent.

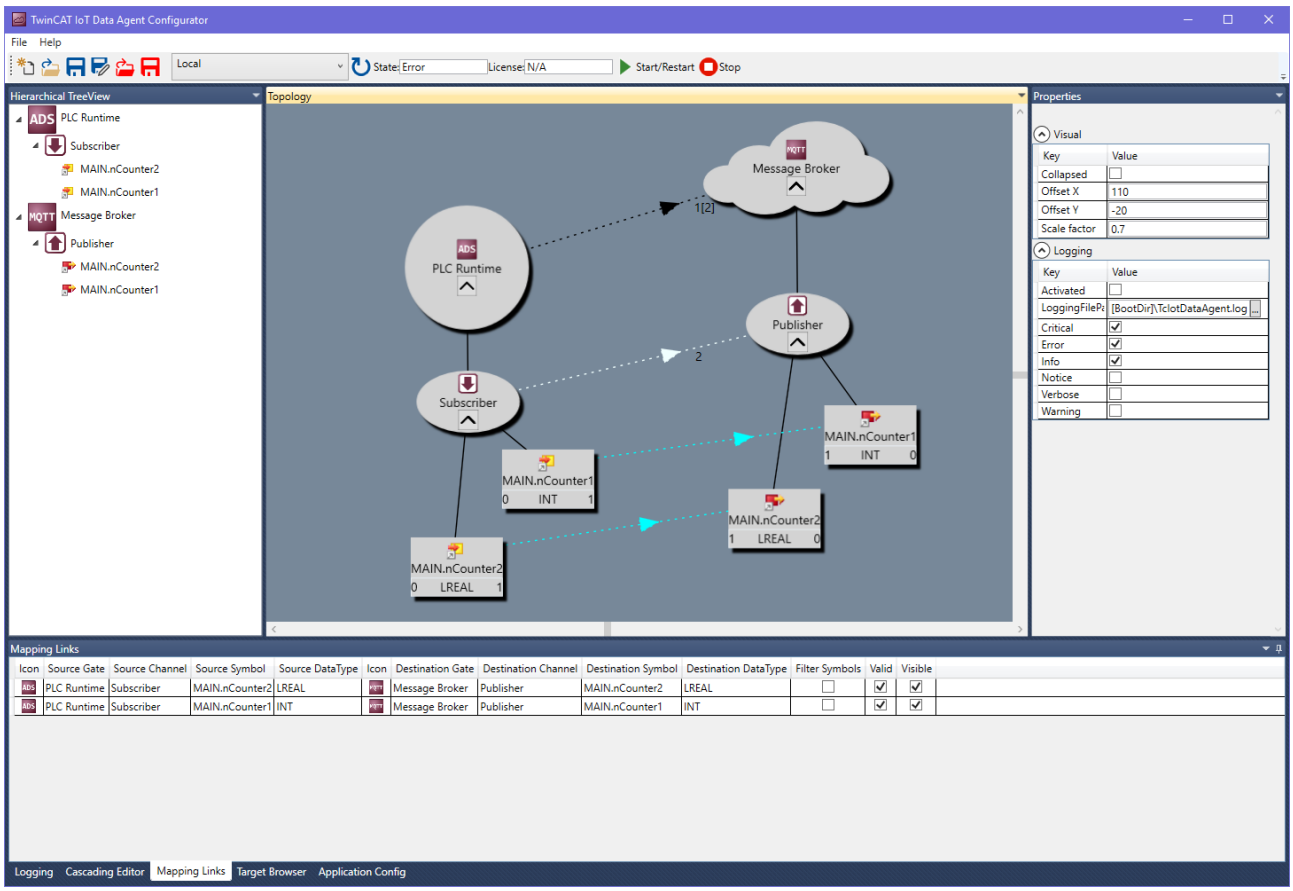
Um diese Konfiguration in TC3 IoT Functions zu verwenden, referenzieren Sie die Kanal-ID im Funktionsbaustein FB_IotFunctions_Message.



5.2 Konfigurator

Der TC3 IoT Data Agent-Konfigurator ist eine benutzerfreundliche grafische Benutzeroberfläche, die die XML-Konfigurationsdatei abstrahiert und eine moderne Schnittstelle bietet, die alle Funktionen zum einfachen Konfigurieren von Symbolen enthält, die an einem Cloud-Dienst gesendet oder von einem solchen Dienst empfangen werden sollen.

Der Konfigurator wird auch für die Konfiguration von TC3 IoT-Funktionen verwendet.



Standardkomponenten

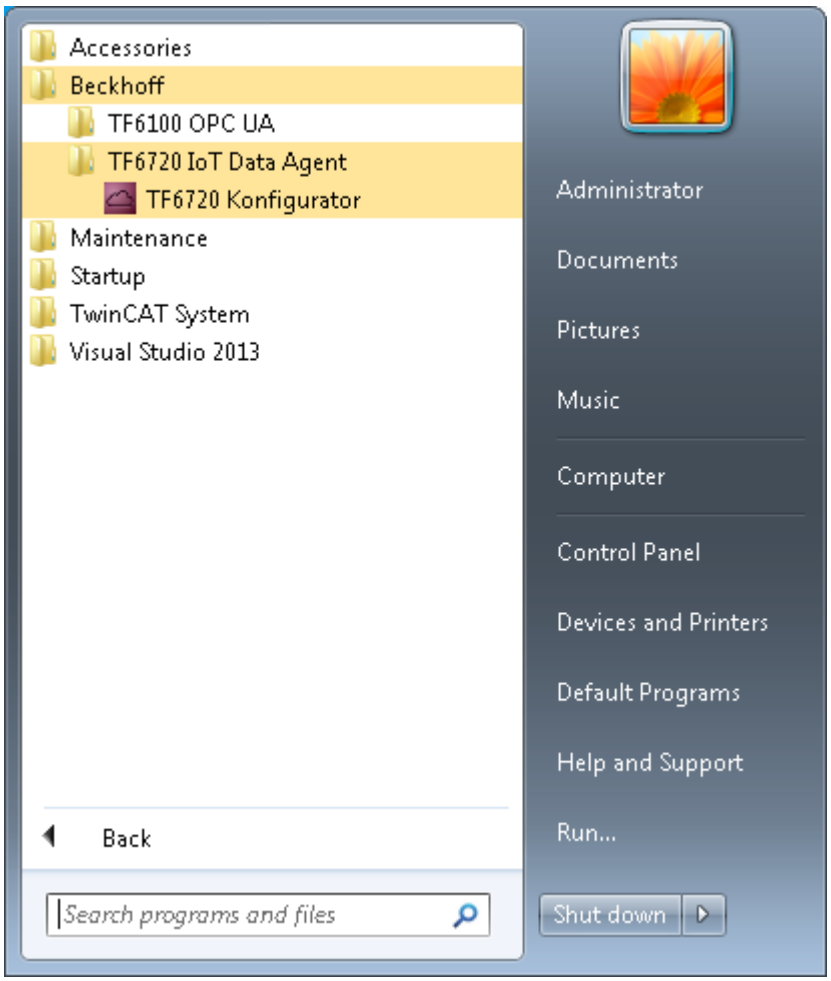
Der TC3 IoT Data Agent-Konfigurator besteht aus den folgenden Bereichen:

Menü und Symbolleiste	Enthält Befehle zum Speichern und Öffnen von Dateien und zum Starten und Stoppen der Anwendung
Hierarchische Baumansicht	Gibt einen hierarchischen Überblick über die Konfiguration, um eine Konfiguration zu erstellen und zu bearbeiten
Topologieansicht	Gibt einen grafischen Überblick über die Konfiguration, um eine Konfiguration zu erstellen und zu bearbeiten
Eigenschaften-Fenster	Zeigt die Eigenschaften einer aktivierten Komponente in der Topologie- oder Baumansicht an.
Logging	Stellt Protokollinformationen vom Konfigurator bereit.
Cascading Editor	Hilft bei großen und komplexen Navigationen, indem Filtermechanismen für Symbole bereitgestellt werden
Mapping Links	Gibt einen Überblick über alle Verbindungen zwischen Symbolen in der Konfiguration
Target Browser	Dient für den symbolischen Zugriff für ADS- und OPC UA-Ziellaufzeiten

Installation

Der Konfigurator wird vom Setup automatisch installiert und ist als Verknüpfung im Windows-Startmenü verfügbar.

Beim ersten Starten des Konfigurators bittet dieser um die Erstellung eines OPC UA-Client-Zertifikats. Dieses Zertifikat wird vom Konfigurator in seinem integrierten OPC UA Target Browser verwendet, um sich mit einem Server zu verbinden und dessen Namensraum zu durchsuchen. Nachdem das Zertifikat erstellt wurde, wird die Konfigurator-Benutzeroberfläche angezeigt.



5.2.1 Topologieansicht

Die Topologieansicht (oder „leere Fläche“) ist der zentrale grafische Konfigurationsbereich des TC3 IoT Data Agent-Konfigurators. Sie zeigt die folgenden Komponenten einer Konfiguration:

- Die konfigurierten Gates, Kanäle und Symbole
- Die Beziehung (Zuordnung bzw. Mapping) zwischen Gates, Kanälen und Symbolen
- Die Kardinalität jeder Beziehung

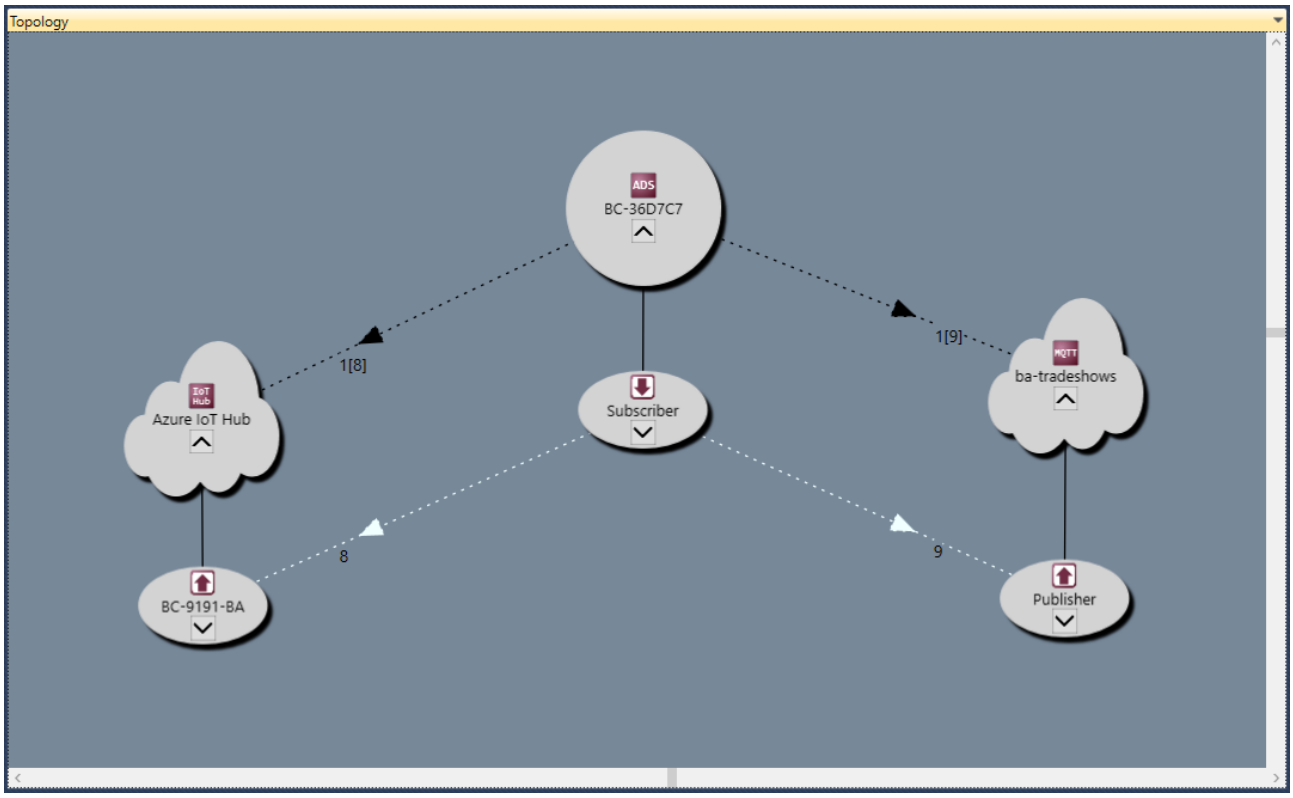
Die Topologieansicht kann zum Erstellen und Bearbeiten einer Konfiguration verwendet werden. Rechtsklicken Sie dazu einfach auf die leere Fläche, um das Kontextmenü zu öffnen, und wählen Sie eine der verschiedenen Konfigurationsoptionen aus, z. B. um ein neues Gate zu erstellen, einem Gate einen neuen Kanal anzufügen oder eine Komponente aus der Konfiguration zu entfernen.

Sie können jedes Objekt in der Topologieansicht verschieben, indem Sie es an eine neue Position ziehen. Jede angefügte Teilkomponente wird zusammen mit ihrer übergeordneten Komponente verschoben. Optional können Sie eine Teilkomponente auch ausblenden, indem Sie auf die Schaltfläche Erweitern ihrer übergeordneten Komponente klicken. Beim Speichern einer Konfiguration wird die Position jedes Objekts in der Konfigurationsdatei gespeichert.

Um die Navigation ein wenig einfacher zu gestalten, unterstützt die Topologieansicht die folgenden Funktionen:

- Bildlaufleisten für vertikale und horizontale Navigation
- Vertikale Navigation über Mausrad
- Horizontale Navigation über Mausrad und SHIFT-Taste (gedrückt halten)
- Heranzoomen/Herauszoomen über Mausrad und STRG-Taste (gedrückt halten)

Anstelle der Topologieansicht kann auch die Baumansicht für die Konfiguration verwendet werden, aber die Topologieansicht bietet einen besseren grafischen Überblick über die aktuelle Konfiguration (siehe Baumansicht [▶ 26])

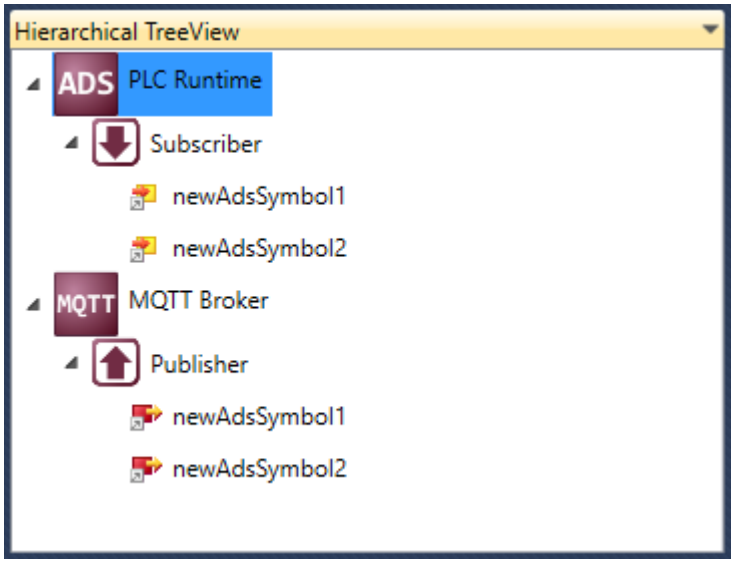


5.2.2 Baumansicht

Die Baumansicht ist eine hierarchische Ansicht der derzeit geöffneten Konfiguration. Sie zeigt die folgenden Komponenten einer Konfiguration:

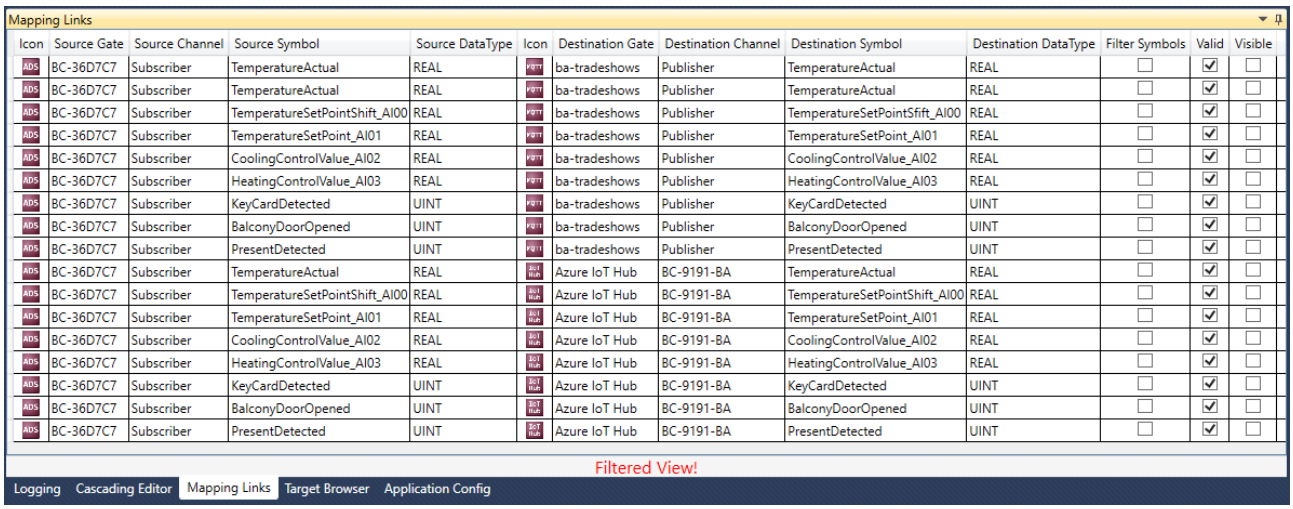
- Die konfigurierten Gates, Kanäle und Symbole
- Das Vorhandensein einer Beziehung (Zuordnung bzw. Mapping) zwischen Symbolen

Die Baumansicht kann auch zur Bearbeitung der Konfiguration verwendet werden, aber es ist womöglich einfacher, stattdessen die Topologieansicht zu verwenden (siehe Topologieansicht [▶ 25]).



5.2.3 Zuordnungen (Mappings)

Das Fenster mit den Zuordnungen gibt einen Überblick über alle Verbindungen zwischen Symbolen in der derzeit geöffneten Konfiguration. Wenn eine Verbindung ausgewählt wird, wird sie in der Topologieansicht automatisch hervorgehoben.



Icon	Source Gate	Source Channel	Source Symbol	Source DataType	Icon	Destination Gate	Destination Channel	Destination Symbol	Destination DataType	Filter Symbols	Valid	Visible
	BC-36D7C7	Subscriber	TemperatureActual	REAL		ba-tradeshows	Publisher	TemperatureActual	REAL	☐	☑	☐
	BC-36D7C7	Subscriber	TemperatureActual	REAL		ba-tradeshows	Publisher	TemperatureActual	REAL	☐	☑	☐
	BC-36D7C7	Subscriber	TemperatureSetPointShift_AI00	REAL		ba-tradeshows	Publisher	TemperatureSetPointSfift_AI00	REAL	☐	☑	☐
	BC-36D7C7	Subscriber	TemperatureSetPoint_AI01	REAL		ba-tradeshows	Publisher	TemperatureSetPoint_AI01	REAL	☐	☑	☐
	BC-36D7C7	Subscriber	CoolingControlValue_AI02	REAL		ba-tradeshows	Publisher	CoolingControlValue_AI02	REAL	☐	☑	☐
	BC-36D7C7	Subscriber	HeatingControlValue_AI03	REAL		ba-tradeshows	Publisher	HeatingControlValue_AI03	REAL	☐	☑	☐
	BC-36D7C7	Subscriber	KeyCardDetected	UINT		ba-tradeshows	Publisher	KeyCardDetected	UINT	☐	☑	☐
	BC-36D7C7	Subscriber	BalconyDoorOpened	UINT		ba-tradeshows	Publisher	BalconyDoorOpened	UINT	☐	☑	☐
	BC-36D7C7	Subscriber	PresentDetected	UINT		ba-tradeshows	Publisher	PresentDetected	UINT	☐	☑	☐
	BC-36D7C7	Subscriber	TemperatureActual	REAL		Azure IoT Hub	BC-9191-BA	TemperatureActual	REAL	☐	☑	☐
	BC-36D7C7	Subscriber	TemperatureSetPointShift_AI00	REAL		Azure IoT Hub	BC-9191-BA	TemperatureSetPointShift_AI00	REAL	☐	☑	☐
	BC-36D7C7	Subscriber	TemperatureSetPoint_AI01	REAL		Azure IoT Hub	BC-9191-BA	TemperatureSetPoint_AI01	REAL	☐	☑	☐
	BC-36D7C7	Subscriber	CoolingControlValue_AI02	REAL		Azure IoT Hub	BC-9191-BA	CoolingControlValue_AI02	REAL	☐	☑	☐
	BC-36D7C7	Subscriber	HeatingControlValue_AI03	REAL		Azure IoT Hub	BC-9191-BA	HeatingControlValue_AI03	REAL	☐	☑	☐
	BC-36D7C7	Subscriber	KeyCardDetected	UINT		Azure IoT Hub	BC-9191-BA	KeyCardDetected	UINT	☐	☑	☐
	BC-36D7C7	Subscriber	BalconyDoorOpened	UINT		Azure IoT Hub	BC-9191-BA	BalconyDoorOpened	UINT	☐	☑	☐
	BC-36D7C7	Subscriber	PresentDetected	UINT		Azure IoT Hub	BC-9191-BA	PresentDetected	UINT	☐	☑	☐

5.2.4 Target Browser

Der Target Browser dient für den symbolischen Zugriff für ADS- und OPC UA-Ziellaufzeiten. Er kann verwendet werden, um Symbole per Drag-and-drop für eine Ziellaufzeit zu konfigurieren.

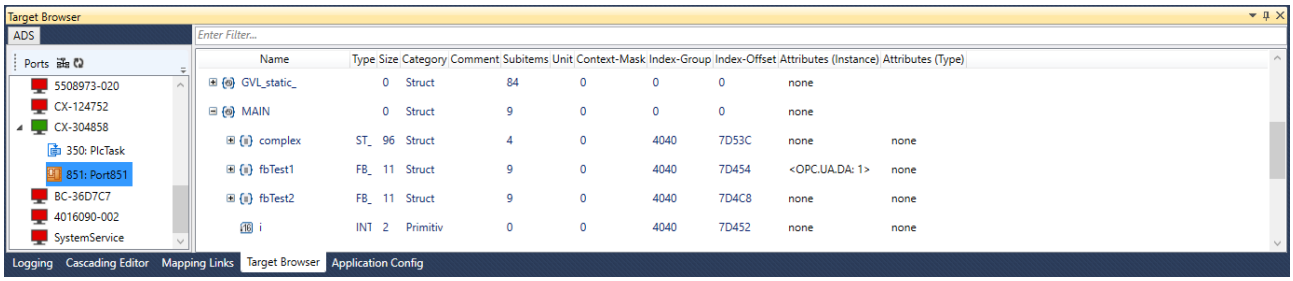
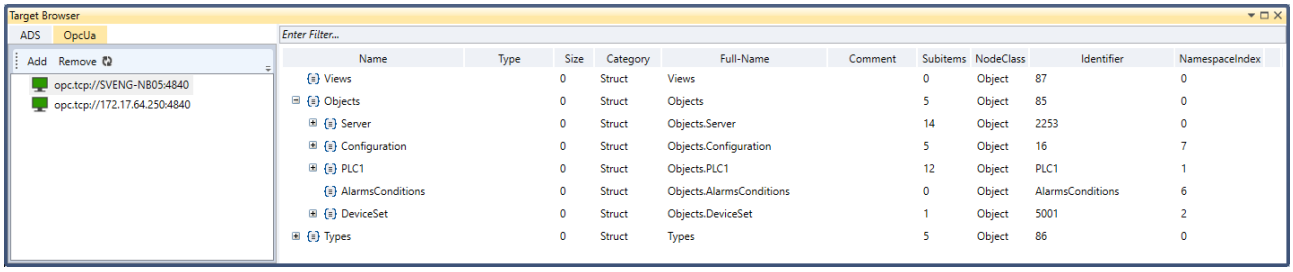
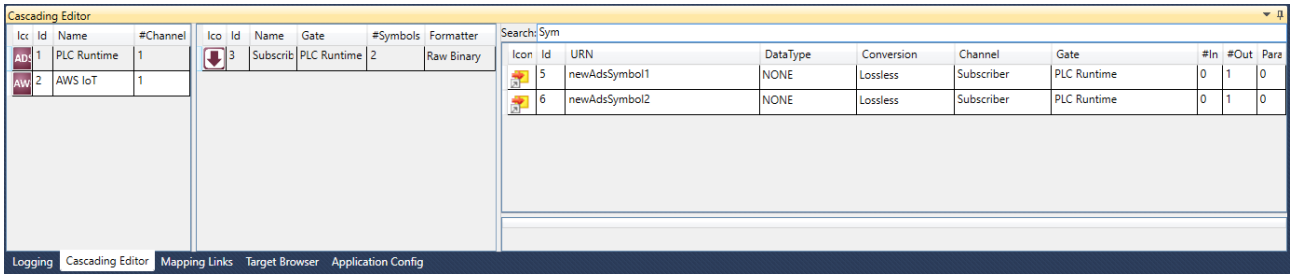


Abb. 1:



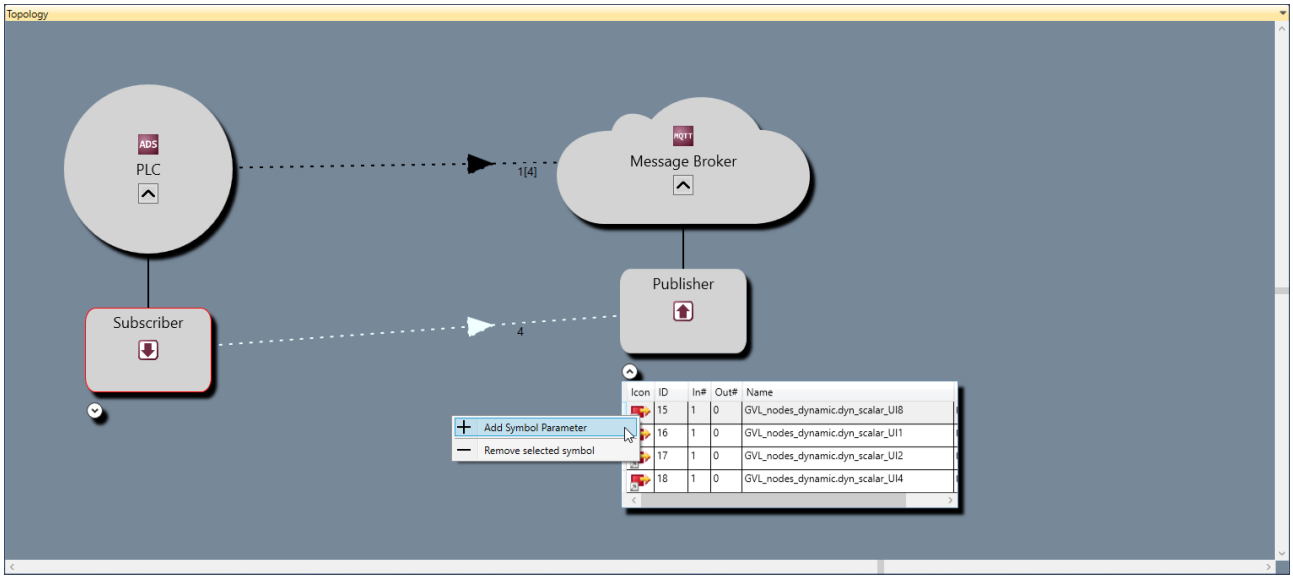
5.2.5 Cascading Editor

Der Cascading Editor hilft beim Navigieren durch große und komplexe Konfigurationen, indem Filtermechanismen für Symbole bereitgestellt werden. Von links nach rechts können Gates und Kanäle ausgewählt werden, um die entsprechenden Symbole anzuzeigen. Außerdem kann mit freiem Text nach Symbolnamen gesucht werden. Wenn eine Komponente ausgewählt wird, wird sie in der Topologieansicht automatisch hervorgehoben.

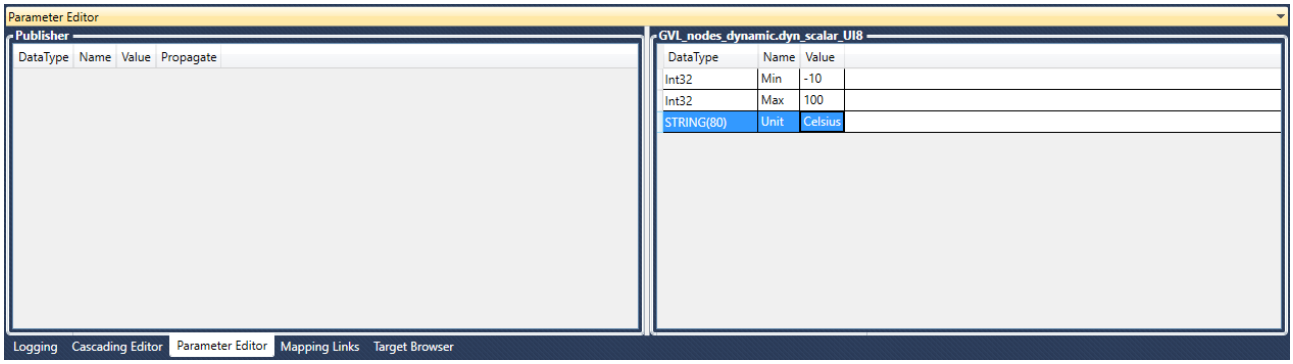


5.2.6 Parameter Editor

Der Parameter Editor ermöglicht die Konfiguration von Symbolmetadaten für Symbole in einem Publisher-Kanal. Als Voraussetzung muss der Publisher-Kanal für die Verwendung des Datenformats „TwinCAT JSON“ konfiguriert werden. Anschließend können durch Rechtsklick auf ein Symbol und Auswahl von **Add Symbol Parameter** neue Symbolparameter hinzugefügt werden.



Dadurch wird dem Parameter Editor ein neuer Symbolparametereintrag hinzugefügt. Danach können der Datentyp, Name und Wert des neuen Parameters festgelegt werden.

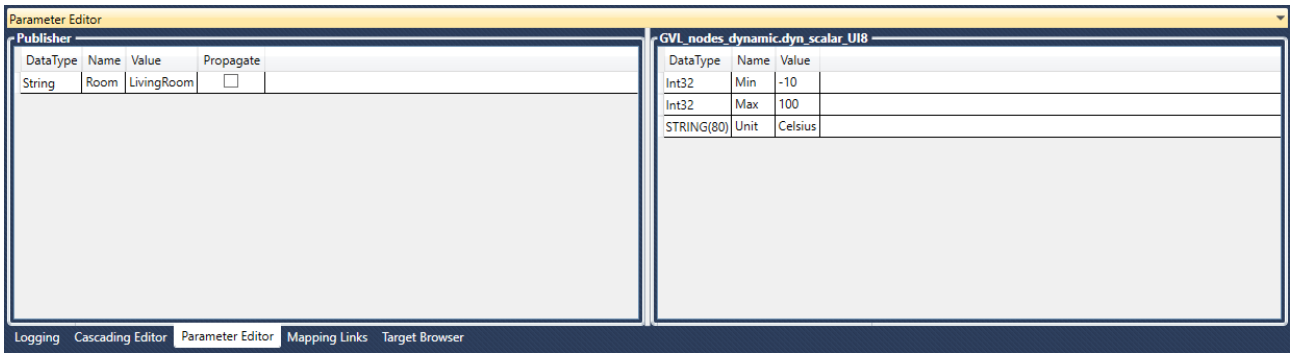


Ein Ergebnis der obigen Konfiguration kann bei Verwendung des Datenformats TwinCAT JSON wie folgt aussehen. In dieser Konfiguration wurde die Variable „GVL_nodes_dynamic.dyn_scalar_UI8“ mit den Symbolparametern Min, Max und Unit konfiguriert. Diese Parameter werden dem Abschnitt „MetaData“ der TwinCAT JSON-formatierten Nachricht hinzugefügt.

```
{
  "Timestamp": "2020-09-28T12:32:49.0370000+02:00",
  "GroupName": "Publisher",
  "Values": {
    "GVL_nodes_dynamic.dyn_scalar_UI8": 259096147,
    "GVL_nodes_dynamic.dyn_scalar_UI1": 83,
    "GVL_nodes_dynamic.dyn_scalar_UI2": 32339,
    "GVL_nodes_dynamic.dyn_scalar_UI4": 259096147
  },
```

```
  "MetaData": {
    "GVL_nodes_dynamic.dyn_scalar_UI8": {
      "Timestamp": "2020-09-28T12:32:49.0350000+02:00",
      "Min": -10,
      "Max": 100,
      "Unit": "Celsius"
    },
    "GVL_nodes_dynamic.dyn_scalar_UI1": {
      "Timestamp": "2020-09-28T12:32:49.0350000+02:00"
    },
    "GVL_nodes_dynamic.dyn_scalar_UI2": {
      "Timestamp": "2020-09-28T12:32:49.0350000+02:00"
    },
    "GVL_nodes_dynamic.dyn_scalar_UI4": {
      "Timestamp": "2020-09-28T12:32:49.0350000+02:00"
    }
  }
}
```


Darüber hinaus können Parameter auch einem Kanal hinzugefügt werden, wodurch die Root-Nachricht um Metadateneigenschaften ergänzt wird. Beispiel:



In der Folge enthält die Root-Nachricht nun eine statische Metadateneigenschaft namens „Room“, die den Wert „LivingRoom“ hat:


```
{
  "Timestamp": "2020-09-30T09:55:58.1160000+02:00",
  "GroupName": "Publisher",
  "Room": "LivingRoom",
  "Values": {
    "GVL_nodes_dynamic.dyn_scalar_UI8": 267266470,
    "GVL_nodes_dynamic.dyn_scalar_UI1": 166,
    "GVL_nodes_dynamic.dyn_scalar_UI2": 10662,
    "GVL_nodes_dynamic.dyn_scalar_UI4": 267266470
    },
  "MetaData": {
    "GVL_nodes_dynamic.dyn_scalar_UI8": {
      "Timestamp": "2020-09-30T09:55:58.1140000+02:00",
      "Min": -10,
      "Max": 100,
      "Unit": "Celsius"
    },
    "GVL_nodes_dynamic.dyn_scalar_UI1": {
      "Timestamp": "2020-09-30T09:55:58.1140000+02:00"
    },
    "GVL_nodes_dynamic.dyn_scalar_UI2": {
      "Timestamp": "2020-09-30T09:55:58.1140000+02:00"
    },
    "GVL_nodes_dynamic.dyn_scalar_UI4": {
      "Timestamp": "2020-09-30T09:55:58.1140000+02:00"
    }
  }
}
```


5.2.7 Settings

Dieser Abschnitt enthält ausführliche Informationen über die verschiedenen Konfigurationsparameter, die für Gates, Kanäle und Symbole eingestellt werden können.

5.2.7.1 Gates

Ein Gate steht für ein Kommunikationsprotokoll oder einen bestimmten Verbindungsdienst, z. B. ADS, OPC UA, MQTT, AWS IoT oder Microsoft Azure IoT Hub. Jedes Gate wird mit Parametern konfiguriert, die für den entsprechenden Gate-Typ spezifisch sind.

ADS

Ein Beckhoff ADS-Gerät ist entweder ein TwinCAT 2/3-Gerät oder ein Beckhoff BC-Gerät. ADS ist das übliche Kommunikationsprotokoll von Beckhoff und kann für den Zugriff auf viele Teile des TwinCAT-Systems verwendet werden. Die gängigsten Anwendungsszenarios für den TC3 IoT Data Agent beinhalten den Zugriff auf eine TwinCAT 2/3 PLC, C++, TcCOM-Module oder das I/O-Prozessabbild, der über ADS erfolgt.

Beim Konfigurieren eines ADS-Gates müssen die folgenden Einstellungen vom TC3 IoT Data Agent vorgenommen werden, um auf das zugrunde liegende ADS-Gerät zuzugreifen:

Einstellung	Beschreibung
AmsNetId	AmsNetId des Zielgeräts, z. B. 127.0.0.1.1.1 für das lokale Gerät.
AdsPort	AdsPort des Zielgeräts, z. B. 801 (TwinCAT 2 PLC) oder 851 (TwinCAT 3 PLC)
IoMode	Legt fest, wie der TC3 IoT Data Agent mit dem ADS-Gerät kommunizieren soll. Folgende Optionen können eingestellt werden: • Direct: greift auf jedes Symbol mit einem separaten ADS-Befehl zu. Vorgeschrieben für BC Controller, erhöht jedoch den ADS-Verkehr • Batched: greift auf in einem ADS-Summenbefehl zusammengefasste Symbole zu, wodurch der ADS-Verkehr optimiert wird, und kann für TwinCAT 2 und 3 PLC oder C++-Laufzeiten verwendet werden

OPC UA

OPC UA ist ein standardisiertes, industrielles Client/Server-Kommunikationsprotokoll, das von vielen Anbietern für verschiedene Anwendungsfälle genutzt wird. Der TC3 IoT Data Agent kann auf OPC UA-Servergeräte zugreifen, um Variablen (sogenannte „Nodes“) auf diesen Geräten mit IoT-Diensten zu verbinden.

Beim Konfigurieren eines OPC UA-Gates müssen die folgenden Einstellungen vom TC3 IoT Data Agent vorgenommen werden, um auf das zugrunde liegende OPC UA-Gerät zuzugreifen:

Einstellung	Beschreibung
Server URL	Die OPC UA-Server-URL, z. B. opc.tcp://localhost:4840
Security policy	Die OPC UA-Sicherheitsrichtlinie, die der TC3 IoT Data Agent während der Verbindungsherstellung mit dem OPC UA-Server verwenden sollte
Security mode	Der OPC UA-Nachrichtensicherheitsmodus, den der TC3 IoT Data Agent während der Verbindungsherstellung mit dem OPC UA-Server verwenden sollte
Authentication mode	Der OPC UA-Authentifizierungsmodus, den der TC3 IoT Data Agent während der Verbindungsherstellung mit dem OPC UA-Server verwenden sollte

MQTT

MQTT kann für die Verbindung mit einem allgemeinen Message-Broker verwendet werden, z. B. Mosquitto, HiveMQ oder ähnliche Broker-Typen.

Einstellung	Beschreibung
Broker address	Die IP-Adresse oder der Hostname des MQTT-Message-Brokers
Port	MQTT spezifiziert Port 1883 für unverschlüsselte Kommunikation und 8883 für verschlüsselte Kommunikation
ClientId	Ein numerischer oder stringbasierter Wert, der den Client identifiziert. Je nach Message-Broker-Typ sollte diese ID eindeutig sein
Authentication mode	Legt fest, ob sich der TC3 IoT Data Agent beim Broker durch eine Benutzername/Passwort-Kombination authentifizieren soll
TLS mode	Legt fest, ob TLS zur Sicherung des Kommunikationskanals zum Message-Broker verwendet werden soll. Zu beachten ist, dass auch der Message-Broker TLS verwenden muss, damit dies funktioniert. Für TLS stehen verschiedene Optionen zur Auswahl: • CA certificate: Verwendet nur das CA-Zertifikat für die Serverauthentifizierung • Client certificate: Verwendet ein Client-Zertifikat für die gegenseitige Client/Server-Authentifizierung • PSK: Verwendet eine gemeinsame PSK-Identität und einen gemeinsamen PSK-Schlüssel, die dem Message-Broker und dem Client bekannt sind

Microsoft Azure IoT Hub

Mit Azure IoT Hub bietet die Cloud-Plattform Microsoft Azure einen Verbindungsdienst in der Cloud an, der bidirektionale Kommunikation, Gerätesicherheit und automatische Skalierbarkeit bietet. Im TC3 IoT Data Agent kann der IoT Hub als spezieller Gate-Typ konfiguriert werden.

Einstellung	Beschreibung
HostName	URL der Azure IoT Hub-Instanz
DeviceId	DeviceId des Geräts, das auf der IoT Hub-Konfigurationswebsite erstellt wurde
SharedAccessKey	Entweder der primäre oder der sekundäre Geräteschlüssel, der zusammen mit dem Gerät auf der IoT Hub-Konfigurationswebsite generiert wurde
CA file	Die CA-Datei, die für die Serverauthentifizierung verwendet wird. Zum Zeitpunkt der Verfassung dieses Artikels spielen bei der Serverauthentifizierung drei Zertifikate eine Rolle, die zur Zertifikatkette gehören und miteinander verbunden sind. • Root CA (Stammzertifikat): Baltimore CyberTrust Root • Intermediate CA (Zwischenzertifikat): Baltimore CyberTrust • Wildcard-Zertifikat Während des anfänglichen TLS-Handshakes werden nur die ersten beiden vom Server an den Client gesendet. Der Client validiert normalerweise nur das Root CA der Kette und bestimmt, ob es vertrauenswürdig ist. Um die CA-Datei für das Root CA zu erwerben, können Sie den Windows-Zertifikatspeicher (certmgr.msc) öffnen, zu den vertrauenswürdigen Stammzertifizierungsstellen browsen und das Zertifikat Baltimore CyberTrust Root exportieren.

AWS IoT

Mit AWS IoT bietet die Cloud-Plattform Amazon Web Services einen Message-Broker-Dienst in der Cloud an, der bidirektionale Kommunikation, Gerätesicherheit und automatische Skalierbarkeit bietet. Im TC3 IoT Data Agent wird AWS IoT als regulärer MQTT-Gate konfiguriert. Für eine erfolgreiche Verbindung mit einer AWS IoT-Instanz müssen jedoch einige spezielle MQTT-Einstellungen konfiguriert werden.

Einstellung	Beschreibung
Broker address	Die URL der AWS IoT-Instanz
Port	Port 8883 für verschlüsselte TLS-Kommunikation ist vorgeschrieben
ClientId	Kann beliebig festgelegt werden, muss jedoch eindeutig sein. In der Regel könnte dies der AWS IoT-Objektname sein.
Authentication mode	Eingestellt auf „No authentication“. Die Authentifizierung bei AWS IoT erfolgt über das TLS-Client-Zertifikat.
TLS mode	Wird vom Konfigurator automatisch auf „Client certificate“ eingestellt. Wählen Sie den Pfad zur CA-Datei, Client-Zertifikatsdatei und Client-Schlüsseldatei aus. Dies sind die Dateien, die auf der Konfigurationswebsite von AWS IoT erzeugt und heruntergeladen werden können.

5.2.7.2 Kanäle

Kanäle werden auf einem Gate zum Senden („Publisher“) oder Empfangen („Subscriber“) von Daten an ein/von einem Gate konfiguriert.

- Source: Dieses Gate ist die Quelle der Daten, d. h. der TC3 IoT Data Agent verbindet sich mit dem Gate und fragt Daten von ihm ab, um diese Daten woandershin (an ein „Destination Gate“) zu senden. Technisch wird dies auch als „Subscriber-Kanal“ bezeichnet.
- Destination: Dieses Gate ist das Ziel der Daten, d. h. der TC3 IoT Data Agent verbindet sich mit dem Gate und sendet ihm Daten, die er von einem anderen Gate (von einem „Source Gate“) erhalten hat. Technisch wird dies auch als „Publisher-Kanal“ bezeichnet.

Jeder Kanal hat andere Einstellungen, die entweder das Datenformat beschreiben, das für diesen Kanal verwendet werden soll, oder die Abtasteinstellungen, die der TC3 IoT Data Agent zum Sammeln der Daten verwenden soll. Diese Einstellungen können auch vom Gate-Typ abhängen, für den der Kanal konfiguriert wurde.

In der folgenden Tabelle sind alle vorhandenen Einstellungen aufgeführt.

Einstellung	Beschreibung	Anwendbar auf Gate-Typ
Direction	Legt fest, ob der Kanal entweder ein Publisher- (Sender-) oder Subscriber- (Empfänger-)Kanal sein soll. Je nach Auswahl und Gate-Typ sind weitere Einstellungen erforderlich oder werden vorausgewählt.	Alle Gates
Einstellung	Beschreibung	Anwendbar auf Gate-Typ
SamplingMode	Wählt aus, ob der Kanal beim Sammeln der Daten von einer Quelle entweder zyklische oder ereignisbasierte Abtastmechanismen verwenden soll. Zu beachten ist, dass je nach Richtung nicht alle Gates beide Arten unterstützen. Ein MQTT-Gate beispielsweise verwendet beim Empfangen von Daten immer den SamplingMode „event“ (wegen des Pub/Sub-Prinzips ist dies immer eventbasiert).	Alle Gates
CycleTime	Nur beim SamplingMode „cyclic“ anwendbar. Legt die Abtastrate in [ms] fest.	Alle Gates
Timeout	Das Timeout für eine Kommunikation mit dem Gate in [ms].	Alle Gates
PartialUpdate	Aktiviert/deaktiviert partielle Updates auf diesem Kanal. Bei Aktivierung (Standard) enthält ein Publish nur das aktualisierte Symbol. Bei Deaktivierung enthält ein Publish alle Symbole eines Kanals mit ihrem zuletzt bekannten Wert. Nur auf Publisher-Kanäle anwendbar.	Alle Gates
BufferSize	Legt die Größe (Anzahl der Nachrichten) des Ringpuffers bei Verbindungsverlust fest.	MQTT, AWS IoT, Azure IoT Hub
Einstellung	Beschreibung	Anwendbar auf Gate-Typ
Formatter	Legt das Datenformat fest, das für diesen Kanal verwendet werden soll, z. B. binär oder JSON. Zu beachten ist, dass einige Gates ihren Kanälen vorschreiben, ein vorgegebenes Datenformat zu verwenden, z. B. ADS- oder OPC UA-Gates, da die Kommunikation mit diesen Geräten ein spezifisches Format erfordert. In diesem Fall ist der Formatter vorgegeben und kann nicht über den Konfigurator geändert werden.	MQTT, AWS IoT, Azure IoT Hub
FormatterType	Legt den Formatter-Typ auf diesem Kanal fest. In den meisten Fällen ist der Formatter-Typ ein InOut-Typ. Für weitere Informationen über diese Einstellung sehen Sie sich unseren Dokumentationsartikel über das Schreiben benutzerdefinierter Plugins über die Formatter-Schnittstelle an.	MQTT, AWS IoT, Azure IoT Hub
Einstellung	Beschreibung	Anwendbar auf Gate-Typ
Topic	Legt das MQTT-Topic fest, das für Publishing oder Subscribing verwendet werden soll.	MQTT
QoS	Legt das QoS (Quality-of-Service)-Niveau fest, das bei Publishing oder Subscribing verwendet werden soll.	MQTT, AWS IoT
Retain	Legt fest, ob eine Nachricht als „Retain“ gesendet werden soll. (Nur für den Publisher-Kanal relevant)	MQTT

Einstellung	Beschreibung	Anwendbar auf Gate-Typ
SendStateInfo	Bei Aktivierung veröffentlicht der TC3 IoT Data Agent seinen „OnlineState“ an das Subtopic /Desc/ und verwendet dieses Subtopic in seinem LastWill. Wenn sich der TC3 IoT Data Agent mit dem Message-Broker verbindet, wird eine JSON-Nachricht an dieses Topic veröffentlicht, die Folgendes enthält { “OnlineState” : true } Wenn der TC3 IoT Data Agent die Verbindung zum Message-Broker ordnungsgemäß trennt, wird die folgende Nachricht an dieses Topic gesendet: { “OnlineState” : false } Wenn der Message-Broker erkennt, dass der TC3 IoT Data Agent die Verbindung verloren hat, wird die folgende Nachricht an dieses Topic gesendet (LastWill): { “OnlineState” : false }	MQTT, AWS IoT

Abtastmodi

Der TC3 IoT Data Agent beinhaltet verschiedene Abtastmodi, die beeinflussen, wie Daten von einer Quelle erfasst oder an ein Ziel geschrieben werden. Der Abtastmodus kann auf einem Kanal eingestellt werden. Gegenwärtig sind die folgenden Abtastmodi vorhanden:

- Cyclic
- OnChange
- TriggerSymbol

Cyclic

Zyklische Abtastung bedeutet, dass der TC3 IoT Data Agent zyklisch das Gate auf Daten abtastet (Subscriber-Kanal) oder zyklisch Daten an ein Gate schreibt (Publisher-Kanal). Auf einem Subscriber-Kanal führt dies zu zyklischen Lesebefehlen, während es auf einem Publisher-Kanal zu zyklischen Schreibbefehlen führt, z. B. auf einem ADS- oder OPC UA-Gate. Auf Gate-Typen, die auf Publisher/Subscriber-Konzepten basieren, z. B. MQTT-, AWS IoT- und Azure IoT Hub-Gates, werden zyklische Anfragen auf einem Subscriber-Kanal automatisch durch Abonnements (Subscriptions) ersetzt, während dies auf einem Publisher-Kanal zu zyklischen Publish-Befehlen führt.

OnChange

OnChange-Abtastung bedeutet, dass der TC3 IoT Data Agent nur Daten mit einem Gate austauscht, wenn sich der Wert einer Variablen geändert hat.

Trigger-Symbole

Trigger-Symbole ermöglichen eine Abtastung „auf Anforderung“, z. B. wenn eine bestimmte Bedingung für ein bestimmtes Symbol (das sogenannte „Trigger-Symbol“) erfüllt ist. Es können verschiedene Arten von Bedingungen festgelegt werden. Sie werden als Teil eines Kanals konfiguriert und ermöglichen die Festlegung der folgenden Bedingungsarten.

Bedingungsart	Beschreibung
EQ	Wert des Trigger-Symbols ist gleich einem bestimmten Wert
NE	Wert des Trigger-Symbols weicht von einem bestimmten Wert ab
LE	Wert des Trigger-Symbols ist kleiner gleich einem bestimmten Wert
GE	Wert des Trigger-Symbols ist größer gleich einem bestimmten Wert
LT	Wert des Trigger-Symbols ist kleiner als ein bestimmter Wert
GT	Wert des Trigger-Symbols ist größer als ein bestimmter Wert

Wenn die Bedingung erfüllt ist, werden alle Symbole in diesem Kanal an das entsprechende Gate veröffentlicht. Außerdem kann festgelegt werden, wie oft die Symbolwerte gesendet werden sollen.

Sendeverhalten	Beschreibung
risingEdge	Die Symbole werden nur einmal gesendet, wenn die Bedingung erfüllt ist
continuous	Die Symbole werden gesendet, solange die Bedingung erfüllt ist

Verwendung von Trigger-Symbolen

Trigger-Symbole können nur für ADS- und OPC UA-Subscriber-Symbole konfiguriert werden.

- Fügen Sie das Symbol, das als Trigger-Symbol fungieren soll, dem ADS- oder OPC UA-Subscriber-Kanal hinzu.
- Konfigurieren Sie den verbundenen Publisher-Kanal mit dem SamplingMode „OnTrigger“, um das zuvor hinzugefügte Symbol als Trigger-Symbol festzulegen.

5.2.7.3 Symbole

Symbole stehen für Variablen von einem Gate, z. B. eine TwinCAT SPS-Variable. Die Symbolkonfiguration enthält die Adressinformationen, die der Data Agent benötigt, um den Wert eines Symbols zu lesen oder zu schreiben. Diese Adressinformationen hängen daher vom Gate-Typ ab.

Auf Gates, die einen Target Browser [▶ 27] unterstützen, erkennt der Browser automatisch die richtigen Adressinformationen für ein Symbol. Bei allen anderen Gates müssen diese Informationen manuell eingegeben werden.

Gate-Typ	Einstellung	Beschreibung
ADS	URN	Symbolname-Adressinformation der ADS-Variable. Funktioniert nicht bei allen ADS-Geräten, z. B. unterstützen BC-Geräte keine Symbolnamen.
ADS	IndexGroup IndexOffset	Die IndexGroup/IndexOffset-Kombination kann verwendet werden, um auf Daten eines ADS-Geräts zuzugreifen, das Symbol-Adressinformationen nicht unterstützt. Im Fall von TwinCAT PLC steht die IndexGroup/IndexOffset-Kombination direkt für eine Speicheradresse, z. B. von einer SPS-Variablen, die sich nach einer Neukompilierung des TwinCAT-Projekts ändern kann. Daher ist es gängige Praxis, stattdessen den TwinCAT PLC-Symbolserver zu verwenden, der Symbolinformationen für seine SPS-Variablen bereitstellt, was bedeutet, dass auf eine Variable über ihren Symbolnamen zugegriffen werden kann, der auch nach einer Neukompilierung oder Online-Änderung gültig bleibt (falls das Symbol noch vorhanden ist). Einige ADS-Dienste umfassen jedoch keinen solchen Symbolserver, z. B. kleine Beckhoff BC-Geräte. In diesen Fällen muss die IndexGroup/IndexOffset-Kombination verwendet werden.
ADS	DataType	Datentyp des Symbols
ADS	Conversion	Legt den Konvertierungsmodus für dieses Symbol fest.
OPC UA	Name	Beschreibender Name des OPC UA-Nodes. Wird nur im Konfigurator verwendet, stellt keine Online-Adressinformation dar.
OPC UA	Identifier	Identifizierung des OPC UA-Symbols auf dem Server, z. B.: • s = MAIN.nCounter (wenn der IdentifierType „String“ ist) • n = 42 (wenn der IdentifierType „Numeric“ ist)
OPC UA	NsName	Name des Namensraums, in dem sich das Symbol befindet. Dieser entspricht dem Namensraum-Index, der Teil einer OPC UA NodeId ist. Die Übersetzung kann über das NamespaceArray erfolgen.
OPC UA	AttributeId	Die AttributeId definiert das OPC UA-Attribut, das vom Data Agent beim Lesen eines Symbols verwendet werden soll. In den meisten Szenarios ist dies der „Wert“ eines Symbols.
OPC UA	Conversion	Legt den Konvertierungsmodus für dieses Symbol fest.
MQTT	URN	Name des MQTT-Symbols. Dieser entspricht dem Namen, der im JSON-Format als Schlüssel verwendet wird, auch beim Empfang von Daten vom MQTT-Gate.
MQTT	DataType	Datentyp des Symbols
MQTT	Conversion	Legt den Konvertierungsmodus für dieses Symbol fest.
IoT Hub	URN	Name des IoT Hub-Symbols. Dieser entspricht dem Namen, der im JSON-Format als Schlüssel verwendet wird, auch beim Empfang von Daten vom IoT Hub.
IoT Hub	DataType	Datentyp des Symbols
IoT Hub	Conversion	Legt den Konvertierungsmodus für dieses Symbol fest.

Beim Konfigurieren eines Kanals können Symbole über einen Target Browser oder manuell hinzugefügt werden, indem die richtigen Adressinformationen angegeben werden. Zu beachten ist, dass nicht alle Gate-Typen Target Browser-Funktionen umfassen. In diesem Fall müssen Symbole manuell konfiguriert werden.

Manuelle Symbolkonfiguration

Wenn das Zielgerät nicht online ist oder keine Symbol-Adressinformationen bereitstellt (z. B. das BC9191), können Symbole durch Eingabe der Symboladresse auch manuell hinzugefügt werden. Die Tabellen am Anfang dieses Dokuments zeigen, welche Informationen in diesem Fall benötigt werden.

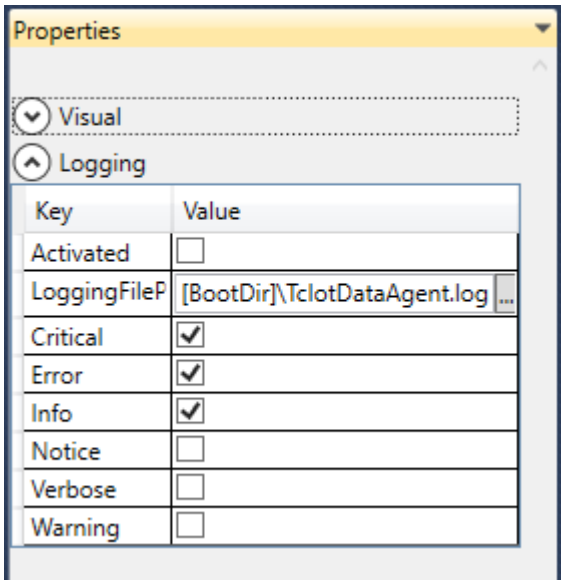
Typkonvertierung

Der TC3 IoT Data Agent unterstützt die Konvertierung des Datentyps, bevor die Daten an ein Gate veröffentlicht werden. Die Typkonvertierung erfolgt auf Symbolebene, was bedeutet, dass verschiedene Symbole verschiedene Konvertierungsmodi verwenden können. Die folgende Tabelle zeigt die verschiedenen vorhandenen Konvertierungsmodi.

Konvertierungsmodus	Beschreibung
Losless	Standardeinstellung. „Kleinere“ Typen können in „größere“ Typen konvertiert werden. Beispielsweise kann ein Subscriber-Symbol vom Datentyp INT als ein Symbol vom Datentyp Int32 veröffentlicht werden (2 Byte in 4 Byte).
Lossy	Falls erforderlich, können auch „größere“ Symbole in „kleinere“ Symbole konvertiert werden. Beispielsweise kann ein Subscriber-Symbol vom Datentyp DINT als ein Symbol vom Datentyp Int16 veröffentlicht werden (4 Byte in 2 Byte). Je nach Wert des Subscriber-Symbols kann dies natürlich zu abgeschnittenen Werten führen.
Strict	Falls erforderlich, können Symbole auch so konfiguriert werden, dass sie den „strikten Modus“ verwenden. In diesem Konvertierungsmodus muss die Datentypgröße eines Subscriber-Symbols exakt mit dem Datentyp des zugeordneten Publisher-Symbols übereinstimmen. Beispielsweise kann ein Subscriber-Symbol vom Datentyp INT nur als ein Symbol vom Datentyp Int16 veröffentlicht werden (2 Byte in 2 Byte).

5.2.8 Fehlerprotokollierung

Zur Fehlersuche kann der TC3 IoT Data Agent eine Protokolldatei erzeugen, die auf der Grundlage verschiedener Protokollierungsebenen gefüllt werden kann. Alle erforderlichen Einstellungen können über das Eigenschaften-Fenster des Konfigurators konfiguriert werden. Klicken Sie dazu einfach auf eine leere Stelle in der Topologieansicht und konfigurieren Sie die Protokollierungseinstellungen im Eigenschaften-Fenster.



Zu beachten ist, dass alle Einstellungen mit der derzeit geöffneten Konfiguration verbunden sind und für jede Konfiguration individuell festgelegt werden müssen. Das Standardverzeichnis, in dem die Protokolldatei erstellt wird, ist das TwinCAT-Bootverzeichnis. Der Platzhalter [BootDir] wählt das TwinCAT-Bootverzeichnis automatisch aus.

Logging-Fenster

Das Logging-Fenster im Konfigurator zeigt nur protokollierte Ereignisse an, die mit dem Konfigurator in Zusammenhang stehen. Um ein Protokoll für den TC3 IoT Data Agent-Hintergrunddienst zu erstellen, sind die vorstehenden Einstellungen erforderlich.

6 SPS API

6.1 Funktionsbausteine

6.1.1 FB_IotFunctions_Connector

Der Funktionsbaustein ermöglicht die Kommunikation mit einer local/remote TC3 IoT Data Agent Installation. Die Execute-Methode des Funktionsbausteins muss zyklisch aufgerufen werden, um die Hintergrundkommunikation mit dem TC3 IoT Data Agent sicherzustellen und den Empfang von Nachrichten zu ermöglichen. Alle Verbindungsparameter sowie die globalen Einstellungen sind als Eingabeparameter vorhanden.

Syntax

```
FUNCTION_BLOCK FB_IotFunctions_Connector
VAR_INPUT
    sAmsNetId                       : STRING;
    nDefaultRequestTimeout          : UDINT;
    nDefaultMessageRetryInterval    : UDINT;
    nDefaultMessageCumulativeTimeout : UDINT;
END_VAR
VAR_OUTPUT
    hrInitializationErrorCode       : HRESULT;
    hrLastMessageErrorCode          : HRESULT;
    driverOTCID                     : OTCID;
    pStIotFunctionsRequests         : POINTER TO ST_IotFunctionsRequestContainer;
END_VAR
```

Eingänge

Bezeichnung	Typ	Beschreibung
sAmsNetId	STRING	Target Data Agent Instance AmsNetId [127.0.0.1.1.1]
nDefaultRequestTimeout	UDINT	Standard-Timeout-Wert in Millisekunden [10000]
nDefaultMessageRetryInterval	UDINT	Standardintervall für die Wiederholung von Nachrichten in Millisekunden (0 deaktiviert Wiederholungen) [0]
nDefaultMessageCumulativeTimeout	UDINT	Standardmäßiges kumulatives Timeout in Millisekunden (bei erfolgreichem Empfang von Nachrichtendaten wird das Timeout der betreffenden Anfrage um diesen Wert verlängert) [0]

Ausgänge

Bezeichnung	Typ	Beschreibung
hrInitializationErrorCode	HRESULT	HRESULT der Treiberinstanziierung und -konfiguration
hrLastMessageErrorCode	HRESULT	HRESULT des letzten gemeldeten Fehlers, der in einer beliebigen Nachricht während des Execute-Aufrufs aufgetreten ist.
driverOTCID	OTCID	OTCID des instanziierten Treibers (kann in Umgebungen mit mehreren Datenagenten an FB-Eingangsvariablen für Anfragen und Nachrichten weitergegeben werden)
pStIotFunctionsRequests	POINTER TO ST_IotFunctionsRequestContainer [▶ 47]	Beschreibung des letzten Fehlercodes.

Methoden

Bezeichnung	Beschreibung
Execute [▶ 39]	Ermöglicht die Hintergrundkommunikation mit dem TC3 IoT Data Agent. Die Methode muss zyklisch aufgerufen werden.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.1.4022.14	IPC oder CX (c86, x64, ARM)	Tc3_IotFunctions

6.1.1.1 Execute

Diese Methode ermöglicht die Hintergrundkommunikation mit dem TC3 IoT Data Agent. Sie muss zyklisch aufgerufen werden.

Syntax

```
METHOD Execute : HRESULT
```

Rückgabewert

Name	Typ	Beschreibung
Execute	HRESULT	Gibt an, ob der Aufruf erfolgreich war.

6.1.2 FB_IotFunctions_Message

Der Funktionsbaustein bietet Lese- und Schreiboperationen für Nachrichten. Alle Nachrichtenparameter sowie die globalen Einstellungen sind als Eingabeparameter vorhanden.

Syntax

```
FUNCTION_BLOCK FB_IotFunctions_Message
VAR_INPUT
  nChannelId                : UINT;
  nRequestTimeout           : UDINT;
  nMessageRetryInterval     : UDINT;
  nMessageCumulativeTimeout : UDINT;
  iotFunctionsDriverOTCID   : OTCID;
END_VAR
VAR_OUTPUT
  hResult                   : HRESULT;
  pStMessageDetails         : POINTER TO ST_IotFunctionsMessage;
  pStRequestDetails         : POINTER TO ST_IotFunctionsRequest;
  bInitialized              : BOOL;
  fbTcResultEvent           : FB_TcIotFunctionsResultEvent;
END_VAR
```

Eingänge

Bezeichnung	Typ	Beschreibung
nChannelId	UINT	Kanal-ID des entsprechenden IotFunctions-Kanals in der Konfiguration der Zieldatenagenteninstanz.
nRequestTimeout	UDINT	Timeout-Wert in Millisekunden.
nMessageRetryInterval	UDINT	Meldungsintervall in Millisekunden (0 schaltet Wiederholungen aus).
nMessageCumulativeTimeout	UDINT	Kumulatives Timeout (der erfolgreiche Empfang von Nachrichtendaten verlängert den Timeout der betreffenden Anfrage um diesen Wert).
iotFunctionsDriverOTCID	OTCID	OTCID der Ziel-IotFunctions-Treiberinstanz (siehe driverOTCID von FB_IotFunctions_Connector)

Ausgänge

Bezeichnung	Typ	Beschreibung
hResult	HRESULT	Enthält das letzte hResultat (wird aktualisiert, wenn auf bBusy oder bError zugegriffen wird oder Read/Write aufgerufen wird)
pStMessageDetails	POINTER TO ST_IotFunctionsMessage [▶ 45]	Zeiger auf Struktur mit detaillierten Informationen über die zugrunde liegende Nachricht
pStRequestDetails	POINTER TO ST_IotFunctionsRequest [▶ 46]	Zeiger auf Struktur mit detaillierten Informationen über die zugrunde liegende Anfrage, in der diese Nachricht ausgeführt wird
bInitialized	BOOL	Zeigt an, ob die Schnittstellensuche und -initialisierung erfolgreich war
fbTcResultEvent	FB_TcIotFunctionsResultEvent	Instanz eines Helferbausteins, welcher Detailinformationen zum Sende- oder Empfangsvorgang enthält.

Methoden

Bezeichnung	Beschreibung
Read [▶ 40]	Liest eine Nachricht aus dem Kanal
Acknowledge [▶ 41]	Quittiert den Fehler-/Erfolgsstatus und gibt die zugehörigen Nachrichtenobjekte frei. Rufen Sie diese Methode nach der Statusquittierung auf, wenn in diesem Zyklus kein neuer Lese-/Schreibvorgang gestartet wird, um eine mehrfache Auswertung des letzten Status zu verhindern.
Write [▶ 41]	Schreibt eine Nachricht in den Kanal

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.1.4022.14	IPC oder CX (c86, x64, ARM)	Tc3_IotFunctions

6.1.2.1 Read

Die Methode liest eine Nachricht aus dem Kanal.

Syntax

```
METHOD Read : HRESULT
VAR_INPUT
    pPayload : PVOID;
    nPayload : UDINT;
END_VAR
```

Rückgabewert

Name	Typ	Beschreibung
Read	HRESULT	Gibt an, ob der Aufruf erfolgreich war.

Eingänge

Name	Typ	Beschreibung
pPayload	PVOID	Pointer auf den Zielspeicher.
nPayload	UDINT	Groesse des Zielspeichers.

6.1.2.2 Acknowledge

Die Methode quittiert den Fehler-/Erfolgsstatus und gibt die zugehörigen Nachrichtenobjekte frei. Rufen Sie diese Methode nach der Statusquittierung auf, wenn in diesem Zyklus kein neuer Lese-/Schreibvorgang gestartet wird, um eine mehrfache Auswertung des letzten Status zu verhindern.

Syntax

```
METHOD Acknowledge : BOOL
```

Rückgabewert

Name	Typ	Beschreibung
Acknowledge	BOOL	Gibt an, ob der Aufruf erfolgreich war.

6.1.2.3 Write

Die Methode schreibt eine Nachricht in den Kanal.

Syntax

```
METHOD Write : BOOL
VAR_INPUT
    pPayload     :     PVOID;
    nPayloadSize : UDINT;
END_VAR
```

Rückgabewert

Name	Typ	Beschreibung
Write	BOOL	Gibt an, ob der Aufruf erfolgreich war.

Eingänge

Name	Typ	Beschreibung
pPayload	PVOID	Adresse des Zielspeichers.
nPayloadSize	UDINT	Groesse des Zielspeichers.

6.1.3 FB_IotFunctions_Request

Der Funktionsbaustein ermöglicht die Synchronisierung mehrerer Nachrichten (siehe Synchronisierung von Nachrichtenoperationen [▶ 19]).

Syntax

```
FUNCTION_BLOCK FB_IotFunctions_Request
VAR_INPUT
    iotFunctionsDriverOTCID : OTCID;
    nRequestTimeout         : UDINT;
END_VAR
VAR_OUTPUT
    hResult             : HRESULT;
    pStRequestDetails   : POINTER TO ST_IotFunctionsRequest;
    bInitialized        : BOOL;
    fbTcResultEvent     : FB_TcIotFunctionsResultEvent;
END_VAR
```

Eingänge

Bezeichnung	Typ	Beschreibung
iotFunctionsDriverOTCID	OTCID	OTCID der Ziel-IotFunctions-Treiberinstanz [Standardwert ist das zuerst erstellte Objekt des FB_IotFunctions_Connector]
nRequestTimeout	UDINT	Timeout-Wert in Millisekunden

Ausgänge

Bezeichnung	Typ	Beschreibung
hResult	HRESULT	Enthält das hresult der letzten Nachricht (wird aktualisiert, wenn auf bBusy oder bError zugegriffen wird oder Create/ Execute/EnqueueRead/EnqueueWrite aufgerufen wird)
pStRequestDetails	POINTER TO ST_IotFunctionsRequest [▶ 46]	Zeiger auf Struktur, die Detailinformationen über die zugrunde liegende Anfrage enthält
bInitialized	BOOL	Zeigt an, ob die Schnittstellensuche und -initialisierung erfolgreich war
fbTcResultEvent	FB_TcIotFunctionsResultEvent	Instanz eines Helferbausteins, welcher Detailinformationen zum Sende- oder Empfangsvorgang enthält.

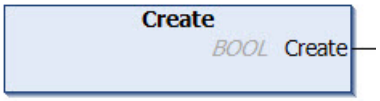
Methoden

Bezeichnung	Beschreibung
Create [▶ 43]	Erstellt eine neue Anfrage
EnqueueRead [▶ 43]	Fügt eine Instanz von FB_IotFunctions_Message [▶ 39] für Leseoperationen hinzu
EnqueueWrite [▶ 44]	Fügt eine Instanz von FB_IotFunctions_Message [▶ 39] für Schreiboperationen hinzu
Execute [▶ 44]	Führt die Anfrage aus
Acknowledge [▶ 41]	Quittiert den Fehler-/Erfolgsstatus und gibt die zugehörigen Nachrichtenobjekte frei. Rufen Sie diese Methode nach der Statusquittierung auf, wenn in diesem Zyklus kein neuer Lese-/Schreibvorgang gestartet wird, um eine mehrfache Auswertung des letzten Status zu verhindern.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.1.4022.14	IPC oder CX (c86, x64, ARM)	Tc3_IotFunctions

6.1.3.1 Create



Diese Methode erstellt eine neue Anfrage.

Syntax

```
METHOD Create : BOOL
```

Rückgabewert

Name	Typ	Beschreibung
Create	BOOL	Gibt an, ob der Aufruf erfolgreich war.

6.1.3.2 EnqueueRead

Die Methode fügt eine Instanz von FB_IotFunctions_Message [▶ 39] für Leseoperationen hinzu.

Syntax

```
METHOD EnqueueRead : BOOL
VAR_INPUT
    pMessageFB  :  POINTER TO FB_IotFunctions_Message;
    pPayload    : POINTER TO PVOID;
    nPayloadSize: UDINT;
END_VAR
```

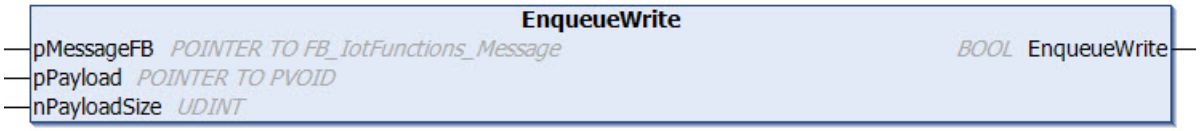
Rückgabewert

Name	Typ	Beschreibung
EnqueueRead	BOOL	Gibt an, ob der Aufruf erfolgreich war.

Eingänge

Name	Typ	Beschreibung
pMessageFB	POINTER TO FB_IotFunctions_Message	Pointer auf ein Objekt vom Typ FB_IotFunctions_Message.
pPayload	POINTER TO PVOID	Adresse des Inhalts.
nPayloadSize	UDINT	Groesse des Inhalts.

6.1.3.3 EnqueueWrite



Fügt eine Instanz von FB_IotFunctions_Message [▶ 39] für Schreiboperationen hinzu.

Syntax

```
METHOD EnqueueWrite : BOOL
VAR_INPUT
    pMessageFB  :  POINTER TO FB_IotFunctions_Message;
    pPayload    : POINTER TO PVOID;
    nPayloadSize: UDINT;
END_VAR
```

Rückgabewert

Name	Typ	Beschreibung
EnqueueWrite	BOOL	Gibt an, ob der Aufruf erfolgreich war.

Eingänge

Name	Typ	Beschreibung
pMessageFB	POINTER TO FB_IotFunctions_Message	Pointer auf ein Objekt vom Typ FB_IotFunctions_Message.
pPayload	POINTER TO PVOID	Adresse des Inhalts.
nPayloadSize	UDINT	Groesse des Inhalts.

6.1.3.4 Execute

Syntax

```
METHOD EnqueueWrite : BOOL
```

Rückgabewert

Name	Typ	Beschreibung
Execute	BOOL	Gibt an, ob der Aufruf erfolgreich war.

6.2 Datentypen

6.2.1 ST_IotFunctionsEvent

Der Datentyp enthält detaillierte Informationen über die Ursache des letzten Fehlercodeergebnisses.

Syntax

```
(* guid of underlying event class *)
uuidEventClass: GUID;

(* event id of underlying event *)
nEventId      : UDINT;

(* severity level of underlying event *)
nSeverity     : UDINT;

(* verbose message of underlying event *)
sEventMsg     : STRING(255);

(* verbose event class name of underlying event *)
sEventClass   : STRING(255);

(* verbose origin of underlying event *)
sSourcePath   : STRING(255);

(* cycle in which underlying event occurred *)
nCycle        : ULINT;
```

Parameter

Name	Typ	Beschreibung
uuidEventClass	GUID	GUID der Event-Klasse.
nEventId	UDINT	ID des Events.
nSeverity	UDINT	Schweregrad des Events.
sEventMsg	STRING	Ausführliche Nachricht des Events.
sEventClass	STRING	Ausführlicher Klassenname des Events.
sSourcePath	STRING	Ausführlicher Herkunftspfad des Events.
nCycle	ULINT	Zyklus, in welchem das Event aufgetreten ist.

6.2.2 ST_IotFunctionsMessage

Enthält detaillierte Informationen über das zugrunde liegende Nachrichtenobjekt.

Syntax

```
(* request Id of governing request containing this message *)
nRequestId            : ULINT;

(* cycle in which this message was created *)
nCycleCreated         : ULINT;

(* time passed since creation (in milliseconds) *)
nAge                  : ULINT;

(* corresponding IoTDataAgent channel *)
nChannelId            : UDINT;

(* hrResult of latest action *)
hrResultCode          : HRESULT;
```

```
(* message Id *)
nMessageId            : UINT;

(* amount of initiated Ads requests during message lifetime *)
nAdsRequestCount      : UINT;

(* amount of received Ads confirmations during message lifetime *)
nAdsConfirmationCount : UINT;

(* current (internal) state of message object. *)
eMessageState         : EIotFunctionsMessageState;

(* message direction [read/write] *)
eMessageDirection     : EIotFunctionsMessageDirection;

(* indicates if the accepted data differes from the previously contained data (relevant for reading
data) *)
bBufferChanged        : BOOL;

(* name of the corresponding symbol *)
sSymbolName           : STRING(255);
```

Parameter

Name	Typ	Beschreibung
nRequestId	ULINT	ID der Anfrage, welche diese Nachricht enthält.
nCycleCreated	ULINT	Zyklus, in welchem diese Nachricht erzeugt wurde.
nAge	ULINT	Verstrichene Zeit seit Erzeugung dieser Nachricht in Millisekunden.
nChannelId	UDINT	ID des zu verwendenden Kanals im Data Agent.
hrResultCode	HRESULT	HRESULT der letzten durchgeführten Operation.
nMessageId	UINT	ID dieser Nachricht.
nAdsRequestCount	UINT	Anzahl an ADS Operationen während der Lebensdauer dieser Nachricht (ADS Requests).
nAdsConfirmationCount	UINT	Anzahl an ADS Operationen während der Lebensdauer dieser Nachricht (ADS Confirmations).
eMessageState	EIotFunctionsMessageState	Aktueller interner Zustand dieses Nachrichtenobjekts.
eMessageDirection	EIotFunctionsMessageDirection	Richtung dieser Nachricht (Lesen oder Schreiben).
bBufferChanged	BOOL	Zeigt an, ob sich der Inhalt des Nachrichtenpuffers geändert hat (Relevant für Lesevorgänge).
sSymbolName	STRING	Name des verwendeten Symbols.

6.2.3 ST_IotFunctionsRequest

Der Datentyp enthält detaillierte Informationen über das zugrunde liegende Anfrageobjekt.

Syntax

```
(* request Id *)
nRequestId    : ULINT;

(* cycle in which this request was created *)
nCycleCreated : ULINT;

(* time passed since creation (in milliseconds) *)
```

```
nAge          : ULINT;

(* time until request expires (timeout) (in milliseconds) *)
nTimeToLive   : ULINT;

(* count of currently pending messages in this request *)
nPendingCount : UDINT;

(* count of currently contained messages in this request *)
nTotalCount   : UINT;

(* indicates if the corresponding internal object has been removed *)
bIsRemoved    : BOOL;

(* indicates if any contained message has timed out *)
bIsTimedOut   : BOOL;

(* indicates if all contained messages have been processed (regardless of success, error or timeout
states) *)
bIsCompleted  : BOOL;
```

Parameter

Name	Typ	Beschreibung
nRequestId	ULINT	ID dieser Anfrage.
nCycleCreated	ULINT	Zyklus, in welchem diese Anfrage erzeugt wurde.
nAge	ULINT	Verstrichene Zeit seit Erzeugung dieser Anfrage in Millisekunden.
nTimeToLive	ULINT	Verbleibende Zeit bis zum Ablaufen dieser Anfrage in Millisekunden.
nPendingCount	UDINT	Anzahl der momentan ausstehenden Nachrichten dieser Anfrage.
nTotalCount	UINT	Anzahl aller enthaltenen Nachrichten dieser Anfrage.
bIsRemoved	BOOL	Zeigt an, ob das interne Objekt dieser Anfrage bereits entfernt wurde.
bIsTimedOut	BOOL	Zeigt an, ob irgendeine enthaltene Nachricht in einen Timeout gelaufen ist.
bIsCompleted	BOOL	Zeigt an, ob alle enthaltenen Nachrichten abgearbeitet worden sind (Unabhängig vom Bearbeitungsendstatus).

6.2.4 ST_IotFunctionsRequestContainer

Der Datentyp enthält ein Array von ST_IotFunctionsRequests, das die von der enthaltenen FB_IotFunctions_Connector-Instanz bearbeiteten Anfragen darstellt.

Syntax

```
(* array of ST_IotFunctionsRequests *)
apIotRequests : ARRAY [0..99] OF POINTER TO ST_IotFunctionsRequest;
```

Parameter

Name	Typ	Beschreibung
apIotRequests	ARRAY	Array von ST_IotFunctionsRequests, welches die von der enthaltenen FB_IotFunctions_Connector-Instanz bearbeiteten Anfragen darstellt.

7 Beispiele

Beispiele für TC3 IoT Functions können als Einzelcontainerlösung heruntergeladen werden: https://infosys.beckhoff.com/content/1031/TF6710_TC3_IoT_Functions/Resources/5247017867.zip

8 Anhang

8.1 Support und Service

Beckhoff und seine weltweiten Partnerfirmen bieten einen umfassenden Support und Service, der eine schnelle und kompetente Unterstützung bei allen Fragen zu Beckhoff Produkten und Systemlösungen zur Verfügung stellt.

Downloadfinder

Unser Downloadfinder beinhaltet alle Dateien, die wir Ihnen zum Herunterladen anbieten. Sie finden dort Applikationsberichte, technische Dokumentationen, technische Zeichnungen, Konfigurationsdateien und vieles mehr.

Die Downloads sind in verschiedenen Formaten erhältlich.

Beckhoff Niederlassungen und Vertretungen

Wenden Sie sich bitte an Ihre Beckhoff Niederlassung oder Ihre Vertretung für den lokalen Support und Service zu Beckhoff Produkten!

Die Adressen der weltweiten Beckhoff Niederlassungen und Vertretungen entnehmen Sie bitte unserer Internetseite: www.beckhoff.com

Dort finden Sie auch weitere Dokumentationen zu Beckhoff Komponenten.

Beckhoff Support

Der Support bietet Ihnen einen umfangreichen technischen Support, der Sie nicht nur bei dem Einsatz einzelner Beckhoff Produkte, sondern auch bei weiteren umfassenden Dienstleistungen unterstützt:

- Support
- Planung, Programmierung und Inbetriebnahme komplexer Automatisierungssysteme
- umfangreiches Schulungsprogramm für Beckhoff Systemkomponenten

Hotline:	+49 5246 963-157
E-Mail:	support@beckhoff.com

Beckhoff Service

Das Beckhoff Service-Center unterstützt Sie rund um den After-Sales-Service:

- Vor-Ort-Service
- Reparaturservice
- Ersatzteilservice
- Hotline-Service

Hotline:	+49 5246 963-460
E-Mail:	service@beckhoff.com

Beckhoff Unternehmenszentrale

Beckhoff Automation GmbH & Co. KG

Hülshorstweg 20
33415 Verl
Deutschland

Telefon: +49 5246 963-0
E-Mail: info@beckhoff.com
Internet: www.beckhoff.com

Mehr Informationen:
www.beckhoff.com/tf6710.html

Beckhoff Automation GmbH & Co. KG
Hülshorstweg 20
33415 Verl
Deutschland
Telefon: +49 5246 9630
info@beckhoff.com
www.beckhoff.com