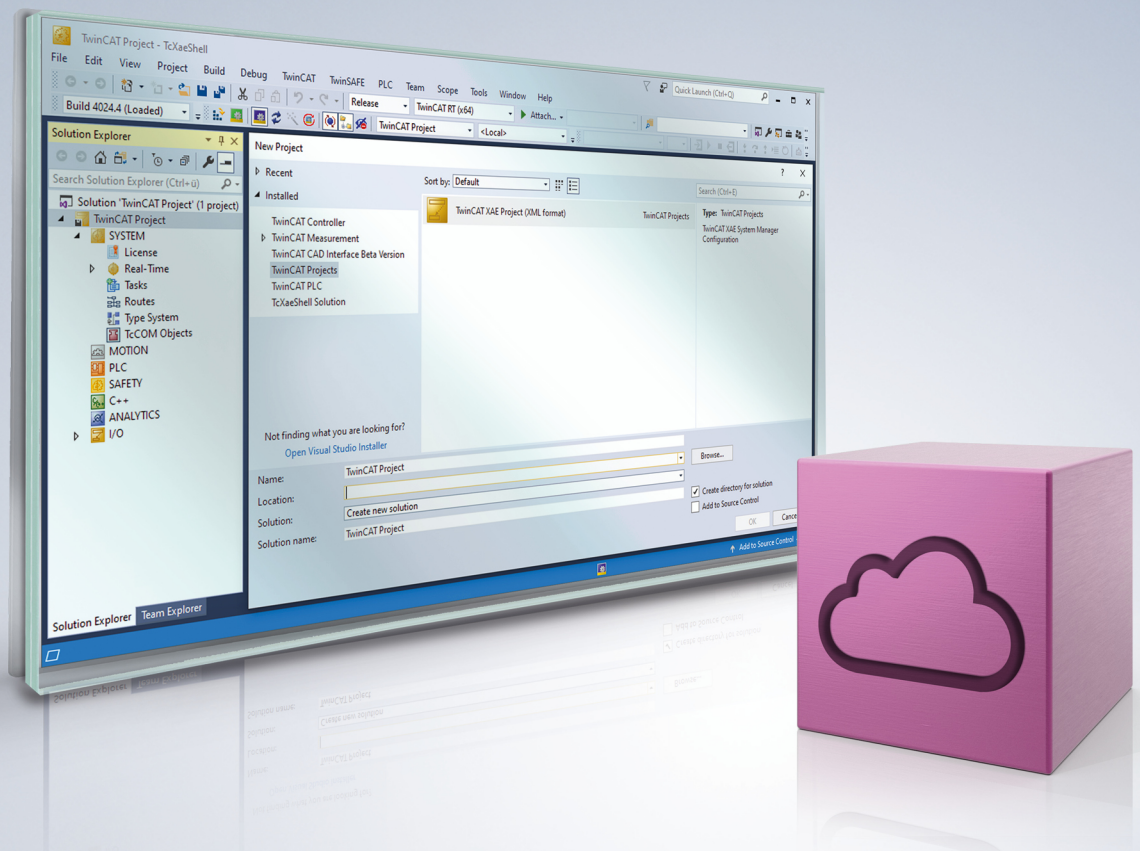
BECKHOFF New Automation Technology

Handbuch | DE

TF6770

TwinCAT 3 | IoT WebSockets



15.04.2024 | Version: 1.0.0

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.

2 Übersicht

Die Funktionsbausteine der SPS-Bibliothek Tc3_IotBase können verwendet werden, um als Client eine WebSocket-Verbindung mit einem Server herzustellen und Daten auszutauschen.

Produktkomponenten

Die Funktion TF6770 IoT WebSockets besteht aus den folgenden Komponenten, die ab TwinCAT-Version 3.1.4026.x verwendet werden können:

- **Treiber:** TcIotDrivers.sys (in der Installation von TwinCAT.Standard.XAR im Paket TwinCAT.XAR.DriversBase enthalten)
- **SPS-Bibliothek:** Tc3_IotBase (in der Installation von TF6770.IotWebSockets.XAE enthalten)

3 Installation

3.1 Systemanforderungen

Technische Daten	Beschreibung
Betriebssystem	Windows 7/10, Windows Embedded Standard 7, TwinCAT/BSD
Zielplattform	PC-Architektur (x86, x64 oder ARM)
TwinCAT-Version	TwinCAT 3.1 Build 4026.3 oder höher
Erforderliches TwinCAT-Setup-Level	TwinCAT 3 XAE, XAR
Erforderliche TwinCAT-Lizenz	TF6770 TC3 IoT WebSockets

3.2 Installation

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 das im Workload enthaltene 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 TF6770 Iot WebSockets-Function installieren.

```
tcpkg install TF6770.IotWebSockets.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.

3.3 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

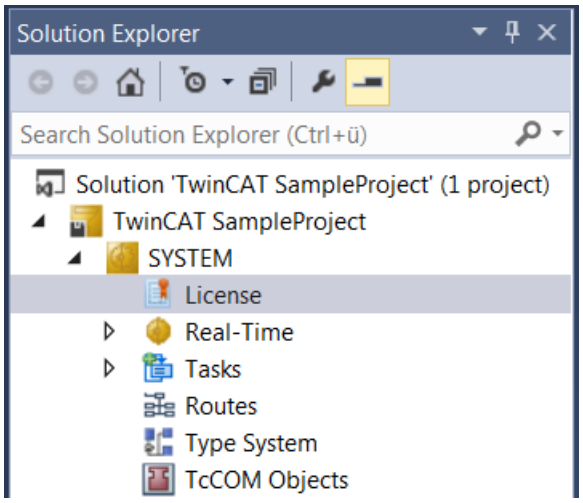
Eine 7-Tage-Testversion kann nicht für einen TwinCAT-3-Lizenz-Dongle freigeschaltet werden.

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.

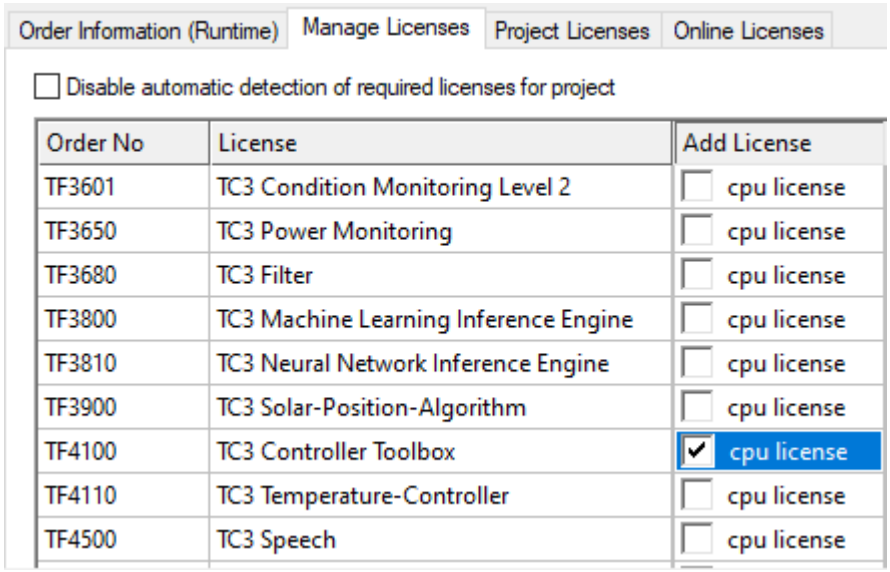
   ⇨ Die Lizenzierungseinstellungen beziehen sich immer auf das eingestellte Zielsystem. Mit der Aktivierung des Projekts auf dem Zielsystem werden automatisch auch die zugehörigen TwinCAT-3-Lizenzen auf dieses System kopiert.

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“).

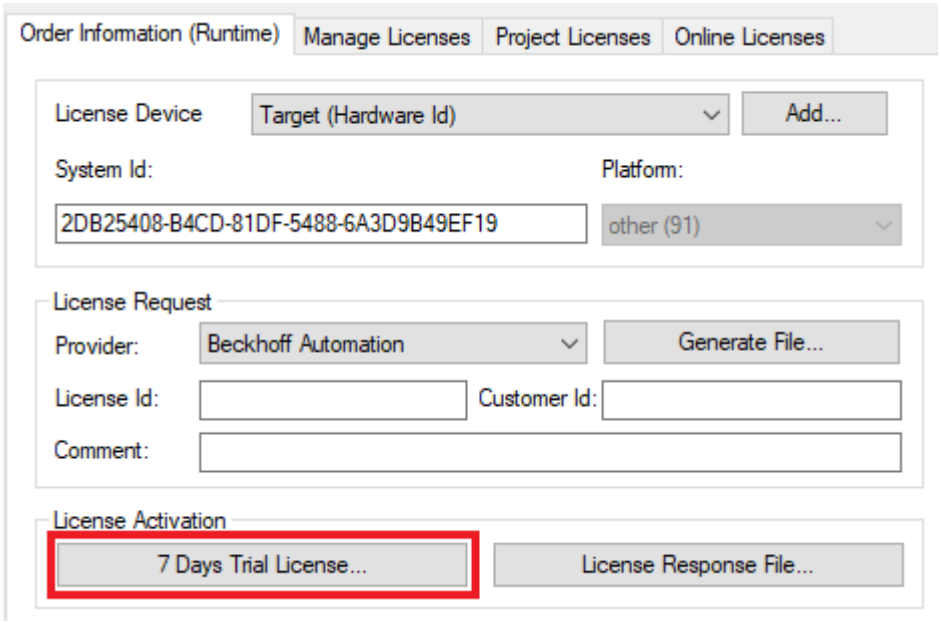


Order No	License	Add License
TF3601	TC3 Condition Monitoring Level 2	cpu license
TF3650	TC3 Power Monitoring	cpu license
TF3680	TC3 Filter	cpu license
TF3800	TC3 Machine Learning Inference Engine	cpu license
TF3810	TC3 Neural Network Inference Engine	cpu license
TF3900	TC3 Solar-Position-Algorithm	cpu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

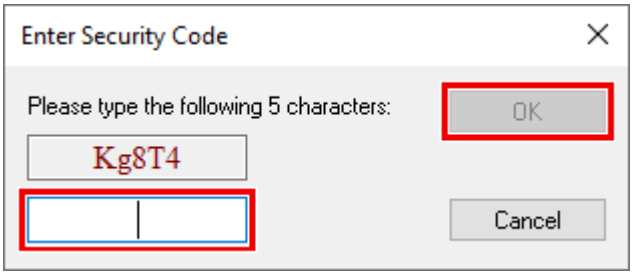
6. Öffnen Sie die Registerkarte **Order Information (Runtime).**

   ⇨ 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

4.1 WebSocket

WebSocket ist ein auf TCP basierendes Netzwerkprotokoll. Im Gegensatz zum zustandslosen HTTP-Protokoll ermöglicht WebSocket eine dauerhafte bidirektionale Verbindung zwischen Server und Client.

Eine WebSocket-Verbindung entsteht aus einem WebSocket-Handshake. In diesem WebSocket-Handshake wird zunächst ein HTTP GET-Request an den Server gesendet, der Informationen über ein Update dieser Verbindung enthält. Wenn der Server WebSocket-Verbindungen unterstützt und die Anfrage akzeptiert, wird eine HTTP-Response mit dem Statuscode 101 Switching Protocols zurück an den Client geschickt. Nach Abschluss des Handshakes wird von HTTP zu WebSocket gewechselt.

Bei hergestellter Verbindung kann im Gegensatz zum HTTP-Protokoll sowohl der Client als auch der Server ohne vorherige Anfrage Daten an den jeweils anderen schicken. Beide Kommunikationsteilnehmer können die Verbindung auch wieder beenden.

Anwendung findet das WebSocket-Protokoll beispielsweise in Chat-Anwendungen, Online-Spielen oder Sport-Livetickern. Im Beispiel des Sport-Livetickers kann der Server dem verbundenen Client Updates mitteilen, ohne dass der Client dafür wie bei HTTP immer eine Anfrage schicken muss.

4.2 Kompression

Im Allgemeinen bezeichnet das Wort „Datenkompression“ die Fähigkeit, die Anzahl an Bits zu verringern, die zur Darstellung von Daten notwendig sind. Dazu werden beispielsweise wiederkehrende Zeichenfolgen durch einen Kompressionsalgorithmus mit einer Referenz auf die erste dieser Zeichenfolgen versehen. Eine geeignete Kompression muss ohne Informationsverlust passieren.

Die RFC 7692-Spezifikation definiert die “permessage-deflate”-Extension für die Komprimierung von WebSocket-Nachrichten. Die Komprimierungsmöglichkeit kann im FB_IotWebSocketClient [▶ 22] durch die Variable bPerMessageDeflate aktiviert werden. Pro Nachricht kann die Kompression dann noch in der SendMessage [▶ 27]()-Methode aktiviert oder deaktivert werden.

4.3 Sicherheit

Bei der Betrachtung der Absicherung einer Datenkommunikation lassen sich zwei Ebenen voneinander unterscheiden. Zum einen die Absicherung des Transportkanals [▶ 12] und zum anderen die Absicherung auf Applikationsebene.

4.3.1 Transportebene

Für die sichere Übertragung von Daten wird im TwinCAT IoT-Treiber der weltweit gängige Standard Transport Layer Security (TLS) verwendet. Das folgende Kapitel beschreibt den TLS-Kommunikationsablauf exemplarisch am Beispiel von TLS-Version 1.2.

TLS ist ein Standard, welcher eine Kombination aus symmetrischer und asymmetrischer Kryptographie darstellt und versendete Daten vor unbefugtem Zugriff und Manipulation durch Dritte schützt. Außerdem wird die Authentifizierung der Kommunikationsteilnehmer zur gegenseitigen Identitätsprüfung von TLS unterstützt.

Inhalt dieses Kapitels

Die folgenden Informationen in diesem Kapitel beziehen sich in allgemeiner Weise auf den TLS-Kommunikationsablauf und haben keinen Bezug zu der Implementierung in TwinCAT. Sie sollen lediglich für ein grundlegendes Verständnis sorgen, um den in den folgenden Unterkapiteln erläuterten Bezug zu der TwinCAT-Implementierung besser nachvollziehen zu können.

Unterstützte Funktionen

Der TwinCAT IoT -Treiber ermöglicht die Verwendung der folgenden TLS-Funktionen.

Funktion	Beschreibung
Selbst signierte Clientzertifikate	Verwendung eines selbst-signierten Clientzertifikats zur Authentifizierung am Message Broker.
CA-signierte Clientzertifikate	Verwendung eines CA-signierten Clientzertifikats zur Authentifizierung am Message Broker. Das CA-Zertifikat kann zum Herstellen einer Vertrauensstellung ebenfalls angegeben werden.
Zertifikatssperrlisten	Verwendung von Zertifikatssperrlisten (englisch: Certificate Revocation Lists (CRL)).
Pre-Shared Key (PSK)	Verwendung eines Pre-Shared Key (PSK) zur Authentifizierung am Message Broker.

Definition Cipher-Suite

Eine Cipher Suite ist per Definition eine Zusammensetzung von Algorithmen (Schlüsselaustausch, Authentifizierung, Verschlüsselung, MAC) zur Verschlüsselung. Auf diese einigen sich Client und Server während des TLS-Verbindungsaufbaus. Weitere Informationen zu Cipher Suites entnehmen Sie bitte der Fachliteratur.

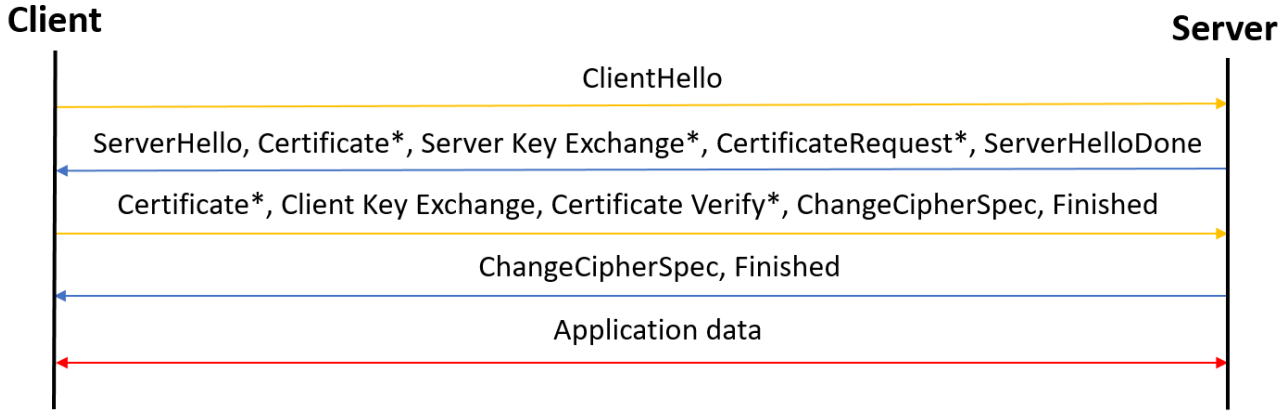
TLS-Kommunikationsablauf

Am Anfang einer Kommunikation mit TLS-Verschlüsselung steht der sogenannte TLS-Handshake zwischen Server und Client. Während des Handshakes wird asymmetrische Kryptographie verwendet, nach erfolgreichem Abschluss des Handshakes kommunizieren Server und Client mit symmetrischer Kryptographie, da diese um ein Vielfaches schneller ist als asymmetrische Kryptographie.

Es gibt drei verschiedene Arten der Authentisierung für das TLS-Protokoll:

- Der Server weist sich per Zertifikat aus (siehe Server-Zertifikat [▶ 16])
- Client und Server weisen sich per Zertifikat aus (siehe Client/Server-Zertifikat [▶ 17])
- Pre-Shared-Keys (siehe Pre-Shared-Keys [▶ 18])

Über Vor- und Nachteile der verschiedenen Authentisierungsarten informieren Sie sich bitte in der Fachliteratur.



Beispielhafte Erläuterung anhand von RSA

Alle mit * gekennzeichneten Nachrichten sind optional und werden nicht zwingend benötigt. Die nachfolgenden Schritte beziehen sich auf das RSA-Verfahren und haben keine allgemeine Gültigkeit für andere Verfahren.

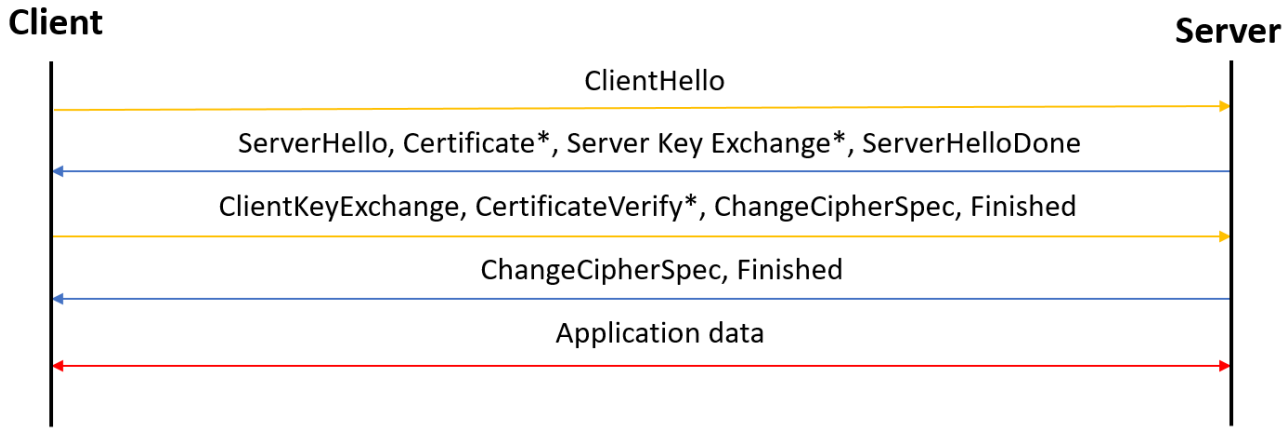
Die folgende Tabelle erläutert die einzelnen Schritte aus dem oben dargestellten Kommunikationsablauf.

Schritt	Beschreibung
ClientHello	Der Client initiiert einen Verbindungsaufbau zum Server. Dabei werden unter anderem die verwendete TLS-Version, eine Zufallsfolge von Bytes (Client Random) und die vom Client unterstützten Cipher Suites übertragen.
ServerHello	Der Server sucht sich aus den vom Client offerierten Cipher Suites eine aus und legt diese für die Kommunikation fest. Besteht keine Schnittmenge zwischen den vom Client und Server unterstützten Cipher Suites, wird der TLS-Verbindungsaufbau abgebrochen. Zusätzlich kommuniziert auch der Server eine Zufallsfolge von Bytes (Server Random).
Certificate	Der Server präsentiert dem Client sein Zertifikat, damit der Client verifizieren kann, dass der Server auch der erwartete Server ist. Vertraut der Client dem Server-Zertifikat nicht, wird der TLS-Verbindungsaufbau abgebrochen. Das Server-Zertifikat enthält zusätzlich den öffentlichen Schlüssel des Servers.
ServerKeyExchange	Bei bestimmten Schlüsselaustauschalgorithmen reichen die Informationen aus dem Zertifikat nicht aus, damit der Client das sogenannte Pre-Master Secret erzeugen kann. In diesem Fall werden die fehlenden Informationen mittels Server Key Exchange übertragen.
CertificateRequest	Der Server fordert vom Client ein Zertifikat an, um die Identität des Clients verifizieren zu können.
ServerHelloDone	Der Server teilt dem Client mit, dass sein Versenden der initialen Informationen beendet ist.
Certificate	Der Client kommuniziert sein Zertifikat inklusive des öffentlichen Schlüssels an den Server. Der Ablauf ist der gleiche wie in entgegengesetzter Richtung: Vertraut der Server dem vom Client versendeten Zertifikat nicht, wird der Verbindungsaufbau abgebrochen.
ClientKeyExchange	Der Client verwendet den öffentlichen Schlüssel des Servers, um durch asymmetrische Verschlüsselung ein von ihm generiertes Pre-Master Secret verschlüsselt an den Server zu schicken. Aus diesem Pre-Master Secret, dem Server Random und dem Client Random wird dann der symmetrische Schlüssel berechnet, der nach dem Verbindungsaufbau für die Kommunikation verwendet wird.
CertificateVerify	Der Client signiert die bisherigen Nachrichten des Handshakes mit seinem privaten Schlüssel. Da der Server den öffentlichen Schlüssel des Clients durch das Versenden des Zertifikats erhalten hat, kann er verifizieren, dass das präsentierte Zertifikat auch wirklich dem Client „gehört“.
ChangeCipherSpec	Der Client teilt dem Server mit, dass er auf symmetrische Kryptographie umsteigt. Jede Nachricht vom Client an den Server ab hier ist signiert und verschlüsselt.
Finished	Der Client teilt dem Server verschlüsselt mit, dass der TLS-Verbindungsaufbau auf seiner Seite beendet ist. Die Nachricht enthält einen Hash und einen MAC über die bisherigen Nachrichten des Handshakes.

Schritt	Beschreibung
ChangeCipherSpec	Der Server entschlüsselt das Pre-Master Secret, das der Client mit seinem öffentlichen Schlüssel verschlüsselt hat. Da der Server der Einzige ist, der seinen privaten Schlüssel besitzt, kann nur der Server dieses Pre-Master Secret entschlüsseln. So wird sichergestellt, dass der symmetrische Schlüssel nur Client und Server bekannt ist. Anschließend berechnet auch der Server den symmetrischen Schlüssel aus Pre-Master Secret und den beiden Zufallsfolgen und teilt dem Client mit, dass auch er jetzt mit symmetrischer Kryptographie kommuniziert. Jede Nachricht vom Server an den Client ab hier ist signiert und verschlüsselt. Durch die Generierung des symmetrischen Schlüssels kann der Server die Finished-Nachricht des Clients entschlüsseln und sowohl Hash als auch MAC verifizieren. Sollte diese Verifizierung fehlschlagen, wird die Verbindung abgebrochen.
Finished	Der Server teilt dem Client mit, dass der TLS-Verbindungsaufbau auf seiner Seite ebenfalls beendet ist. Die Nachricht enthält so wie beim Client einen Hash und einen MAC über die bisherigen Nachrichten des Handshakes. Auf der Seite des Clients wird dann dieselbe Verifikation vollzogen wie beim Server, auch hier gilt: Wenn Hash und MAC nicht erfolgreich entschlüsselt werden, wird die Verbindung abgebrochen.
ApplicationData	Client und Server kommunizieren nach Abschluss des TLS-Verbindungsaufbaus mit symmetrischer Kryptographie.

4.3.1.1 Server-Zertifikat

Dieser Abschnitt behandelt den Fall, dass nur der Client das Server-Zertifikat verifizieren will, der Server aber das Client-Zertifikat nicht. Der Kommunikationsablauf aus dem Kapitel Transportebene [▶ 12] verkürzt sich zu folgendem Ablauf.



Verifikation des Server-Zertifikats

Das Server-Zertifikat wird auf Client-Seite verifiziert. Dabei wird überprüft, ob es von einer bestimmten Certificate Authority signiert ist. Ist das nicht der Fall, wird die Verbindung vom Client abgebrochen, da er dem Server dann nicht vertraut.

Verwendung in TwinCAT

In TwinCAT wird in dem Fall der Dateipfad zum CA-Zertifikat angegeben (.PEM oder .DER-Datei) oder der Inhalt der .PEM-Datei als String. Im IoT-Treiber wird dann das vom Server präsentierte Zertifikat überprüft. Wenn die Zertifikatskette nicht von der angegebenen CA signiert ist, wird die Verbindung zum Server abgebrochen. Der nachfolgende Code stellt nur exemplarisch die beschriebenen Verbindungsparameter dar. Der Beispielcode bezieht sich auf den HTTP-Client, den MQTT-Client und den WebSocket-Client, exemplarisch wird der HTTP-Client verwendet.

```
PROGRAM MAIN
VAR
    fbClient : FB_IotHttpClient;
END_VAR
```

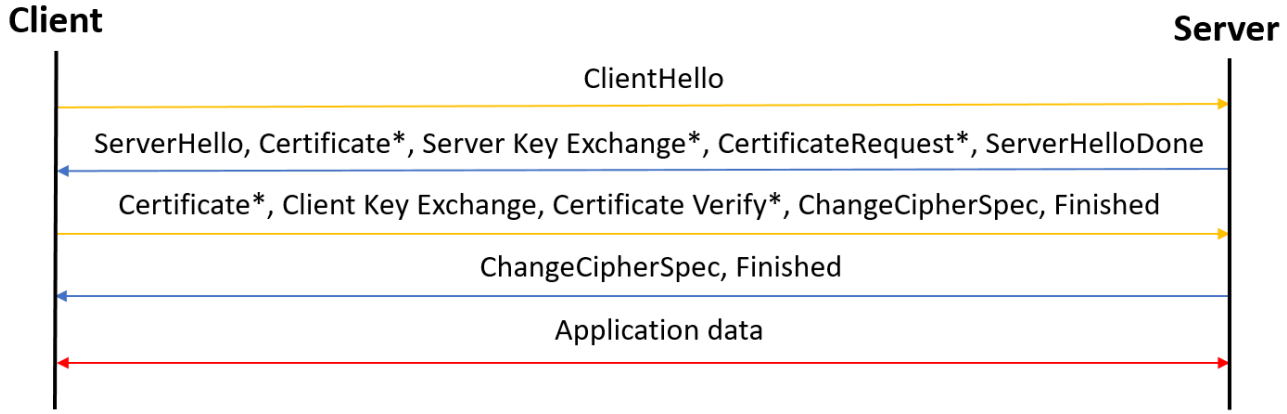
```
fbClient.stTLS.sCA:= 'C:\TwinCAT\3.1\Config\Certificates\someCA.pem';
```

Hat der Benutzer das CA-Zertifikat nicht zur Verfügung, kann trotzdem eine Verbindung hergestellt werden. Dazu gibt es eine boolsche Variable, mit der TwinCAT die Verifikation des Server-Zertifikats verboten wird. Die Verbindung wird zwar mit dem öffentlichen Schlüssel des unverifizierten Server-Zertifikats verschlüsselt hergestellt, ist aber anfälliger gegen Man-in-the-middle-Attacken.

```
fbClient.stTLS.sCA.bNoServerCertCheck:= TRUE;
```

4.3.1.2 Client/Server-Zertifikat

In diesem Abschnitt wird der Fall betrachtet, dass sowohl Client- als auch Server-Zertifikat verifiziert werden. Der im Vergleich zum Server-Zertifikat [▶ 16]-Kapitel leicht abgeänderte Kommunikationsablauf wird in der folgenden Grafik visualisiert. Die einzelnen Schritte des TLS-Verbindungsaufbaus sind im Kapitel Transportebene [▶ 12] beschrieben.



Verwendung in TwinCAT

In TwinCAT wird im Falle der Verwendung eines Client-Zertifikats ebenso wie beim CA-Zertifikat der Dateipfad (.PEM- oder .DER-Datei) oder der Inhalt der .PEM-Datei als String übergeben. Dieses Zertifikat präsentiert TwinCAT als Client dann dem Server. Für das Certificate Verify muss zusätzlich der private Schlüssel des Clients referenziert werden. Optional kann im Falle eines Passwortschutzes für den privaten Schlüssel auch dieses Passwort übergeben werden. Der Beispielcode bezieht sich auf den HTTP-Client, den MQTT-Client und den WebSocket-Client, exemplarisch wird der HTTP-Client verwendet.

```
PROGRAM MAIN
VAR
    fbClient : FB_IotHttpClient;
END_VAR
```

```
fbClient.stTLS.sCA:= 'C:\TwinCAT\3.1\Config\Certificates\someCA.pem';
fbClient.stTLS.sCert:= 'C:\TwinCAT\3.1\Config\Certificates\someCRT.pem';
fbClient.stTLS.sKeyFile:= 'C:\TwinCAT\3.1\Config\Certificates\someprivatekey.pem.key';
fbClient.stTLS.sKeyPwd:= 'yourkeyfilepasswordhere';
```

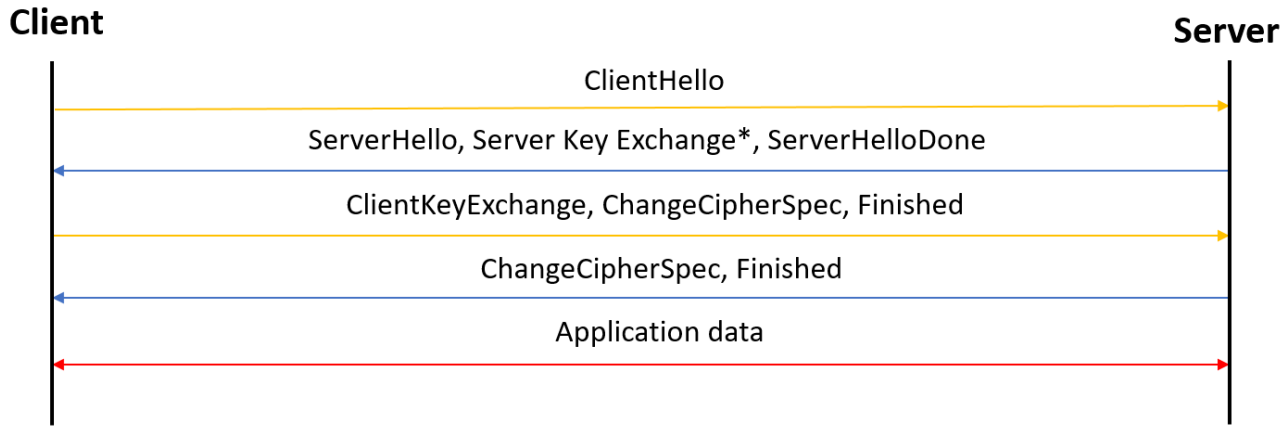
In dem Fall, dass ein Client-Zertifikat gesetzt ist, muss auch ein CA-Zertifikat zur Validierung des Server-Zertifikats gesetzt werden. Das ist durch das Verhalten des im IoT-Treiber verwendeten Security-Frameworks bedingt.

Wenn in diesem Fall die Validierung des Server-Zertifikats abgeschaltet werden soll, kann zusätzlich ein Flag zum Überspringen der Validierung gesetzt werden. Es ist aber nicht möglich, das CA-Zertifikat wegzulassen.

4.3.1.3 Pre-Shared-Keys

Standardmäßig werden beim TLS-Verbindungsaufbau asymmetrische Schlüsselpaare verwendet. Asymmetrische Kryptographie verbraucht höhere Rechenleistung, daher kann es in Umgebungen mit wenig CPU-Leistung eine Möglichkeit sein, Pre-Shared Keys (PSK) zu verwenden. Pre-Shared-Keys sind vorher geteilte, symmetrische Schlüssel.

Verglichen mit dem Kommunikationsablauf bei asymmetrischer Verschlüsselung fällt bei Verwendung von PSK das Zertifikat weg. Client und Server müssen sich über die sogenannte Identität auf einen PSK einigen. Der PSK ist per Definition vorher beiden Parteien bekannt.



Server Key Exchange: In dieser optionalen Nachricht kann der Server dem Client einen Hinweis auf die Identität des verwendeten PSK geben.

Client Key Exchange: Der Client gibt die Identity des PSK an, mit dem die Verschlüsselung durchgeführt werden soll.

Verwendung in TwinCAT

In TwinCAT wird die Identität des PSK als String angegeben, der PSK selbst wird als ByteArray in der Steuerung abgespeichert. Zusätzlich wird noch die Länge des PSK angegeben. Der Beispielcode bezieht sich auf den HTTP-Client, den MQTT-Client und den WebSocket-Client, exemplarisch wird der HTTP-Client verwendet.

```
PROGRAM MAIN
VAR
    fbClient : FB_IotHttpClient;
    cMyPskKey : ARRAY[1..64] OF BYTE := [16#1B, 16#D0, 16#6F, 16#D2, 16#56, 16#16, 16#7D, 16#C1, 16#
E8, 16#C7, 16#48, 16#2A, 16#8E, 16#F5, 16#FF];
END_VAR
```

```
fbClient.stTLS.sPskIdentity:= identityofPSK';
fbClient.stTLS.aPskKey:= cMyPskKey;
fbClient.stTLS.nPskKeyLen:= 15;
```

4.3.1.4 Unterstützte Cipher-Suites

Der TwinCAT IoT-Treiber unterstützt eine sichere Datenübertragung unter Verwendung des TLS-Standards. Nachfolgend finden Sie eine Übersicht zu allen vom IoT-Treiber unterstützten Cipher-Suites, abhängig von der TwinCAT Version.

TwinCAT 3.1 Build 4024.x

Cipher-Suite
AES128-GCM-SHA256
AES128-SHA
AES128-SHA256
AES256-SHA
AES256-SHA256
DES-CBC3-SHA
DHE-RSA-AES128-GCM-SHA256
DHE-RSA-AES128-SHA
DHE-RSA-AES128-SHA256
DHE-RSA-AES256-SHA
DHE-RSA-AES256-SHA256
ECDHE-ECDSA-AES128-GCM-SHA256
ECDHE-ECDSA-AES128-SHA
ECDHE-ECDSA-AES128-SHA256
ECDHE-ECDSA-AES256-SHA
ECDHE-ECDSA-DES-CBC3-SHA
ECDHE-RSA-AES128-GCM-SHA256
ECDHE-RSA-AES128-SHA
ECDHE-RSA-AES128-SHA256
ECDHE-RSA-AES256-SHA
ECDHE-RSA-DES-CBC3-SHA
EDH-RSA-DES-CBC3-SHA
PSK-3DES-EDE-CBC-SHA
PSK-AES128-CBC-SHA
PSK-AES128-CBC-SHA256
PSK-AES128-GCM-SHA256
PSK-AES256-CBC-SHA

TwinCAT 3.1 Build 4026.x

Cipher-Suite
AES128-GCM-SHA256
AES128-SHA
AES128-SHA256
AES256-GCM-SHA384
AES256-SHA
AES256-SHA256
DHE-RSA-AES128-GCM-SHA256
DHE-RSA-AES128-SHA
DHE-RSA-AES128-SHA256
DHE-RSA-AES256-GCM-SHA384
DHE-RSA-AES256-SHA
DHE-RSA-AES256-SHA256
ECDHE-ECDSA-AES128-GCM-SHA256
ECDHE-ECDSA-AES128-SHA
ECDHE-ECDSA-AES128-SHA256
ECDHE-ECDSA-AES256-GCM-SHA384
ECDHE-ECDSA-AES256-SHA
ECDHE-ECDSA-AES256-SHA384
ECDHE-RSA-AES128-GCM-SHA256
ECDHE-RSA-AES128-SHA
ECDHE-RSA-AES128-SHA256
ECDHE-RSA-AES256-GCM-SHA384
ECDHE-RSA-AES256-SHA
ECDHE-RSA-AES256-SHA384
PSK-AES128-CBC-SHA
PSK-AES128-CBC-SHA256
PSK-AES128-GCM-SHA256
PSK-AES256-CBC-SHA
PSK-AES256-CBC-SHA384
PSK-AES256-GCM-SHA384

4.3.2 Applikationsebene

Auch auf der Applikationsebene bieten sich verschiedene Sicherheits-Mechanismen an. Im Folgenden werden diese Sicherheitsmechanismen beschrieben.

4.3.2.1 JSON Web Token (JWT)

JSON Web Token (JWT) sind ein offener Standard (nach RFC 7519), welche ein kompaktes und sich selbst beschreibendes Format definieren, um Informationen sicher zwischen Kommunikationsteilnehmern in Form eines JSON Objekts zu übertragen. Die Authentizität der übertragenen Information kann hierbei verifiziert und sichergestellt werden, da ein JWT mit einer digitalen Signatur versehen wird. Die Signatur kann hierbei über ein Shared Secret (via HMAC Algorithmus) oder einen Public/Private Key (via RSA) erfolgen.

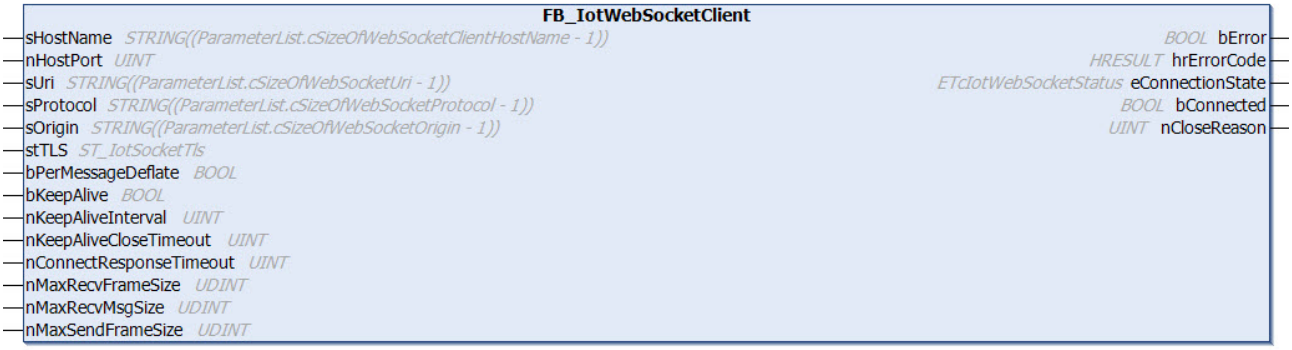
Das am weitesten verbreitete Anwendungsbeispiel für JWT ist die Autorisierung eines Geräts oder Benutzers an einem Service. Sobald sich ein Benutzer an dem Service angemeldet hat, beinhalten alle weiteren Anfragen an den Service das JWT. Anhand des JWT kann der Service dann entscheiden, auf welche weiteren Dienste oder Ressourcen der Benutzer zugreifen darf. Hierdurch können zum Beispiel Single Sign On Lösungen in Cloud-Diensten realisiert werden.

Die SPS Bibliothek Tc3_JsonXml stellt über die Methode FB_JwtEncode die Möglichkeit bereit ein JWT zu erzeugen und signieren.

5 SPS API

5.1 Funktionsbausteine

5.1.1 FB_IotWebSocketClient



Dieser Funktionsblock ermöglicht die Kommunikation mit einem WebSocket-Server. Ein Client kann eine Verbindung zu genau einem WebSocket-Server verwalten. Die Execute [▶ 25]()-Methode muss als Hintergrundkommunikation zum Server zyklisch aufgerufen werden.

Alle Eingabeparameter werden nur dann verarbeitet, wenn eine Verbindung hergestellt wird. Alle Parameter vom Typ STRING erwarten das UTF-8-Format. Dieses entspricht dem STRING-Format für die 7-Bit-ASCII-Zeichen.

Syntax

```
FUNCTION_BLOCK FB_IotWebSocketClient
VAR_INPUT
    sHostName              : STRING;
    nHostPort              : UINT;
    sUri                   : STRING;
    sProtocol              : STRING;
    sOrigin                : STRING;
    stTLS                  : ST_IotSocketTls;
    bPerMessageDeflate     : BOOL;
    bKeepAlive             : BOOL;
    nKeepAliveInterval     : UINT;
    nKeepAliveCloseTimeout : UINT;
    nConnectResponseTimeout: UINT;
    nMaxRecvFrameSize      : UDINT;
    nMaxRecvMsgSize        : UDINT;
    nMaxSendFrameSize      : UDINT;
END_VAR
VAR_OUTPUT
    bError                 : BOOL;
    hrErrorCode            : HRESULT;
    eConnectionState       : ETcIotWebSocketStatus;
    bConnected             : BOOL;
    nCloseReason           : UINT;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
sHostName	STRING	sHostName kann als Name oder als IP-Adresse angegeben werden. Bei fehlender Angabe wird Local Host verwendet.
nHostPort	UINT	Hier wird der Host Port angegeben. Dieser ist standardmäßig 80.
sUri	STRING	Optionaler Parameter, um beim WebSocket-Opening-Handshake eine URI anzugeben.
sProtocol	STRING	Optionaler Parameter, um beim WebSocket-Opening-Handshake ein Unterprotokoll anzugeben.
sOrigin	STRING	Optionaler Parameter, um beim WebSocket-Opening-Handshake eine Origin anzugeben.
stTLS	ST_IotSocketTls	Wenn der Server eine TLS-gesicherte Verbindung anbietet, kann hier die nötige Konfiguration vorgenommen werden.
bPerMessageDeflate	BOOL	Aktiviert die Permessage-Deflate-Komprimierungserweiterung.
bKeepAlive	BOOL	Aktiviert vom Client gesendete Keep-Alive-Ping-Nachrichten. Standardeinstellung ist TRUE.
nKeepAliveInterval	UINT	Legt fest, wie viele Sekunden, nachdem keine Nachrichten vom Server empfangen wurden, der Client eine Ping-Nachricht senden soll.
nKeepAliveCloseTimeout	UINT	Legt fest, wie viele Sekunden der Client auf die Ping-Antwort warten soll.
nConnectResponseTimeout	UINT	Legt fest, wie viele Sekunden der Client auf die Handshake-Antwort des Servers warten soll.
nMaxRecvFrameSize	UDINT	Maximal empfangbare Frame-Größe. Standardgröße ist 16#10000.
nMaxRecvMsgSize	UDINT	Maximal empfangbare Message-Größe. Standardgröße ist 16#100000.
nMaxSendFrameSize	UDINT	Maximal sendbare Frame-Größe. Standardgröße ist 16#4000.

Ausgänge

Name	Typ	Beschreibung
bError	BOOL	Wird TRUE, sobald eine Fehlersituation auftritt.
hrErrorCode	HRESULT	Gibt einen ADS Return Code zurück. Eine Erläuterung zu den möglichen ADS Return Codes befindet sich im Anhang.
eConnectionState	ETcIotWebSocketStatus	Gibt den Zustand der Verbindung vom Client zum Server als Enumeration ETcIotWebSocketState an.
bConnected	BOOL	TRUE, wenn die Verbindung zum Host hergestellt ist und der WebSocket-Handshake erfolgreich war.
nCloseReason	UINT	WebSocket-Statuscode gemäß der Definition in RFC 6455.

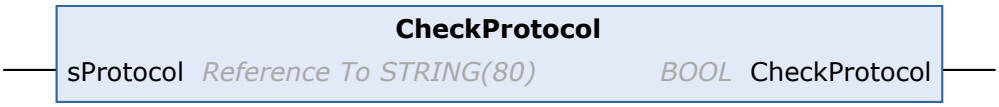
Methoden

Name	Beschreibung
CheckProtocol [▶ 24]	Diese Methode kann optional überschrieben werden, um das vom Server zurückgegebene Protokoll bei der Herstellung der Verbindung zu überprüfen.
Connect [▶ 24]	Methode zum Herstellen der Verbindung zum WebSocket-Server.
Disconnect [▶ 25]	Methode zum Trennen der Verbindung zum WebSocket-Server.
Execute [▶ 25]	Muss zyklisch als Hintergrundkommunikation zum WebSocket-Server aufgerufen werden. Wenn Nachrichten empfangen wurden, wird die Callback-Methode für jede Nachricht einmal ausgelöst.
OnWebSocketClose [▶ 25]	Callback-Methode, die beim Schließen der WebSocket-Verbindung ausgelöst wird.
OnWebSocketMessage [▶ 26]	Callback-Methode, die beim Erhalt einer Nachricht ausgelöst wird.
SendMessage [▶ 27]	Methode zum Senden einer Nachricht.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.1.4026.x	IPC oder CX (x86, x64, ARM)	Tc3_IotBase (3.5.1 oder höher)

5.1.1.1 CheckProtocol



Die Methode muss überschrieben werden, wenn das vom WebSocket-Server zurückgegebene Protokoll überprüft werden soll. Im Falle einer fehlgeschlagenen Überprüfung muss der Wert FALSE zurückgegeben werden.

Syntax

```
METHOD CheckProtocol: BOOL
VAR_IN_OUT CONSTANT
    sProtocol       : STRING;
END_VAR
```

Rückgabewert

Name	Typ	Beschreibung
CheckProtocol	BOOL	TRUE, wenn die Überprüfung des vom WebSocket-Server zurückgegebenen Protokolls erfolgreich war. FALSE, wenn die Überprüfung nicht erfolgreich war.

/ Ein-/Ausgänge

Name	Typ	Beschreibung
sProtocol	STRING	Das zu überprüfende Protokoll.

5.1.1.2 Connect

Diese Methode wird aufgerufen, wenn eine Verbindung vom Client zum Server aufgebaut werden soll.

Syntax

```
METHOD Connect: BOOL
VAR_IN
    bReconfig : BOOL;
END_VAR
```

Rückgabewert

Name	Typ	Beschreibung
Connect	BOOL	Bei erfolgreichem Verbindungsaufbau gibt die Methode TRUE zurück.

Eingänge

Name	Typ	Beschreibung
bReconfig	BOOL	Dieser Parameter muss auf TRUE gesetzt werden, wenn die Verbindungsparameter nach dem initialen Verbindungsaufbau verändert wurden und beim nächsten Verbindungsaufbau verwendet werden sollen.

5.1.1.3 Disconnect

Diese Methode wird aufgerufen, wenn eine Verbindung vom Client zum Server beendet werden soll.

Syntax

```
METHOD Disconnect: BOOL
```

Rückgabewert

Name	Typ	Beschreibung
Disconnect	BOOL	Bei erfolgreicher Trennung der Verbindung gibt die Methode TRUE zurück.

5.1.1.4 Execute

Diese Methode muss zyklisch als Hintergrundkommunikation zum WebSocket-Server aufgerufen werden. Wenn Nachrichten empfangen wurden, wird die Callback-Methode OnWebSocketMessage [▶ 26]() für jede Nachricht einmal ausgelöst.

5.1.1.5 OnWebSocketClose

Diese Methode darf nicht vom Anwender aufgerufen werden. Stattdessen kann vom Funktionsbaustein FB_IotWebSocketClient abgeleitet und diese Methode überschrieben werden. Während dem Aufruf der Methode Execute [▶ 25]()hat der zuständige TwinCAT-Treiber die Möglichkeit, im Fall eines eingehenden Close-Frames, die Methode OnWebSocketClose() aufzurufen.

Syntax

```
METHOD OnWebSocketClose: HRESULT
VAR_IN
    statusCode     : UINT;
    content        : PVOID;
    contentLength  : UDINT;
END_VAR
```

Rückgabewert

Name	Typ	Beschreibung
OnWebSocketClose	HRESULT	Über diesen Rückgabewert kann frei entschieden werden.

Eingänge

Name	Typ	Beschreibung
statusCode	UINT	WebSocket-Statuscode gemäß der Definition in RFC 6455.
content	PVOID	Zeiger auf den Inhalt.
contentLength	UDINT	Größe des Inhalts in Bytes.

5.1.1.6 OnWebSocketMessage

Diese Methode darf nicht vom Anwender aufgerufen werden. Stattdessen kann vom Funktionsbaustein FB_IotWebSocketClient abgeleitet und diese Methode überschrieben werden. Während dem Aufruf der Methode Execute()hat der zuständige TwinCAT-Treiber die Möglichkeit, im Fall von neuen eingehenden Nachrichten, die Methode OnWebSocketMessage() aufzurufen. Bei mehreren eingehenden Nachrichten wird die Callback-Methode mehrfach, je Nachricht einmal, aufgerufen. Die Implementierung der Methode muss dies berücksichtigen.

```
METHOD OnWebSocketMessage: HRESULT
VAR_IN
    content              : PVOID;
    contentLength        : UDINT;
    contentType          : ETcIotWebSocketContentType;
END_VAR
```

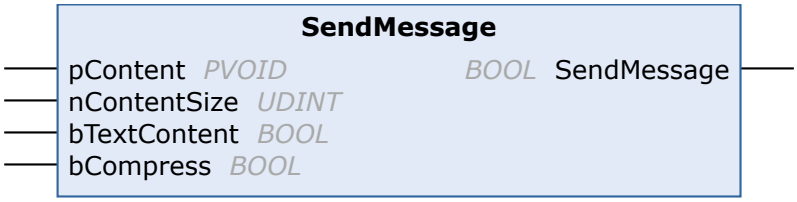
Rückgabewert

Name	Typ	Beschreibung
OnWebSocketMessage	HRESULT	Der Rückgabewert der Methode ist mit S_OK zu belegen, sofern die Nachricht angenommen wurde. Soll die Nachricht im Kontext des nächsten Execute()-Aufrufs erneut ausgegeben werden, kann der Rückgabewert mit S_FALSE belegt werden.

Eingänge

Name	Typ	Beschreibung
content	PVOID	Zeiger auf den Inhalt.
contentLength	UDINT	Größe des Inhalts in Bytes.
contentType	ETcIotWebSocketContentType	Legt fest, ob der Inhalt binär oder als Text vorliegt.

5.1.1.7 SendMessage



Diese Methode wird aufgerufen, wenn eine Nachricht an den WebSocket-Server gesendet werden soll.

Syntax

```
METHOD SendMessage: BOOL
VAR_IN
    pContent      : PVOID;
    nContentSize  : UDINT;
    bTextContent  : BOOL;
    bCompress     : BOOL;
END_VAR
```

Rückgabewert

Name	Typ	Beschreibung
SendMessage	BOOL	Bei erfolgreichem Senden der Nachricht gibt die Methode den Wert TRUE zurück.

Eingänge

Name	Typ	Beschreibung
pContent	PVOID	Zeiger auf den Inhalt.
nContentSize	UDINT	Größe des Inhalts in Bytes.
bTextContent	BOOL	TRUE: Inhalt ist Text, FALSE: Inhalt ist binär.
bCompress	BOOL	Wenn auf TRUE gesetzt, wird Komprimierung verwendet.

5.1.1.8 ST_IotSocketTls

Der folgende Typ enthält die TLS-Sicherheitseinstellungen für den HTTP-Client und den WebSocket-Client. Entweder CA (Zertifizierungsstelle) oder PSK (PreSharedKey) können verwendet werden.

Syntax

Definition:

```
TYPE ST_IotSocketTls :
STRUCT
    sCA                : STRING(255*);
    sCert              : STRING(255*);
    sKeyFile           : STRING(255*);
    sKeyPwd            : STRING(255*);
    sCrl               : STRING(255*);
    sCiphers           : STRING(255*);
    sVersion           : STRING(80) := 'tlsv1.2';
    bNoServerCertCheck : BOOL := FALSE;

    sPskIdentity       : STRING(255*);
    aPskKey            : ARRAY[1..64*] OF BYTE;
    nPskKeyLen         : USINT;
END_STRUCT
END_TYPE
```

Parameter

Name	Typ	Beschreibung
sCA	STRING(255)	Zertifikat der Certificate Authority (CA)
sCert	STRING(255)	Client-Zertifikat für die Authentifizierung beim Server
sKeyFile	STRING(255)	Private Key des Clients
sKeyPwd	STRING(255)	Passwort des Private Keys, falls anwendbar
sCrl	STRING(255)	Pfad zur Certificate Revocation List, welche im Format PEM oder DER vorliegen kann
sCiphers	STRING(255)	Zu verwendende Ciper Suites, angegeben im OpenSSL-String-Format
sVersion	STRING(80)	Zu verwendende TLS-Version
bNoServerCertCheck	BOOL	Deaktiviert die Überprüfung des Server-Zertifikats auf Gültigkeit. Wenn ohne TLS-Verschlüsselung (HTTP/WebSockets) kommuniziert werden soll, muss dieser Wert auf FALSE bleiben.
sPskIdentity	STRING(255)	PreSharedKey-Identität für TLS-PSK Verbindung
aPskKey	ARRAY[1..64] OF BYTE	PreSharedKey für TLS-PSK-Verbindung
nPskKeyLen	USINT	Länge des PreSharedKey in Bytes

Alle Strings und Arrays, die mit einem * gekennzeichnet worden sind, werden mit dem Wert in Klammern initialisiert. Über die Parameterliste kann auf diese Werte zugegriffen werden und können diese geändert werden. Dies ist nicht während der Laufzeit, sondern nur vor der Kompilierung des Codes möglich.

5.2 Datentypen

5.2.1 ETclotWebSocketStatus

```
TYPE ETcIotWebSocketStatus :
(
    WS_STATUS_BUSY:=-1,
    WS_STATUS_SUCCESS:=0,
    WS_STATUS_SOCK_NOMEM:=1,
    WS_STATUS_SOCK_ERR_CREATE_PROTOCOL:=2,
    WS_STATUS_SOCK_CONN_INVAL:=3,
    WS_STATUS_SOCK_NO_CONN:=4,
    WS_STATUS_SOCK_CONN_REFUSED:=5,
    WS_STATUS_SOCK_NOT_FOUND:=6,
    WS_STATUS_SOCK_CONN_LOST:=7,
    WS_STATUS_SOCK_ERR_TLS:=8,
    WS_STATUS_SOCK_NOT_SUPPORTED:=10,
    WS_STATUS_SOCK_ERR_AUTH:=11,
    WS_STATUS_SOCK_ERRACL_DENIED:=12,
    WS_STATUS_SOCK_ERR_UNKNOWN:=13,
    WS_STATUS_SOCK_ERRNO:=14,
    WS_STATUS_SOCK_ERR_EAI:=15,
    WS_STATUS_SOCK_ERR_PROXY:=16,
    WS_STATUS_TLS_CA_NOTFOUND:=17,
    WS_STATUS_TLS_CERT_NOTFOUND:=18,
    WS_STATUS_TLS_KEY_NOTFOUND:=19,
    WS_STATUS_TLS_CA_INVALID:=20,
    WS_STATUS_TLS_CERT_INVALID:=21,
    WS_STATUS_TLS_KEY_INVALID:=22,
    WS_STATUS_TLS_VERIFY_FAIL:=23,
    WS_STATUS_TLS_SETUP:=24,
    WS_STATUS_TLS_HANDSHAKE_FAIL:=25,
    WS_STATUS_TLS_CIPHER_INVALID:=26,
    WS_STATUS_TLS_VERSION_INVALID:=27,
    WS_STATUS_TLS_PSK_INVALID:=28,
    WS_STATUS_TLS_CRL_NOTFOUND:=29,
    WS_STATUS_TLS_CRL_INVALID:=30,
    WS_STATUS_FINALIZE_DISCONNECT:=31,
    WS_STATUS_SOCK_ERR_BIND:=32,
```

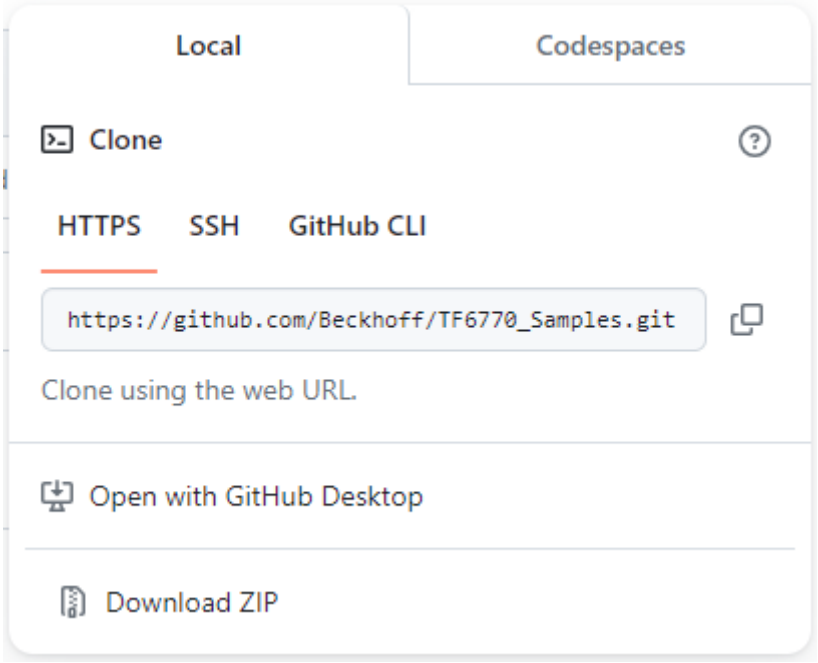
```
    WS_STATUS_SOCK_BIND_ADDR_INUSE:=33,
    WS_STATUS_SOCK_BIND_ADDR_INVAL:=34,
    WS_STATUS_SOCK_ERR_CREATE:=35,
    WS_STATUS_SOCK_ERR_CREATE_TYPE:=36,
    WS_STATUS_SOCK_CONN_FAILED:=37,
    WS_STATUS_SOCK_CONN_TIMEDOUT:=38,
    WS_STATUS_SOCK_CONN_HOSTUNREACH:=39,
    WS_STATUS_TLS_CERT_EXPIRED:=40,
    WS_STATUS_TLS_CN_MISMATCH:=41,
    WS_STATUS_INTERNAL_ERROR:=1000,
    WS_STATUS_CONNECT_REQ_INTERNAL_ERROR:=1001,
    WS_STATUS_CONNECT_REQ_SEND_ERROR:=1002,
    WS_STATUS_CONNECT_RES_PARSE_ERROR:=1003,
    WS_STATUS_CONNECT_RES_INVALID:=1004,
    WS_STATUS_CONNECT_RES_INVALID_STATUS:=1005,
    WS_STATUS_CONNECT_RES_ACCEPT_INVALID:=1006,
    WS_STATUS_CONNECT_RES_ACCEPT_INVALID_HASH:=1007,
    WS_STATUS_CONNECT_RES_INVALID_EXTENSION:=1008,
    WS_STATUS_CONNECT_RES_REJECT:=1009,
    WS_STATUS_CONNECT_RES_TIMEDOUT:=1010,
    WS_STATUS_KEEP_ALIVE_TIMEDOUT:=1011,
    WS_STATUS_NOMEMORY:=1012,
    WS_STATUS_INVALID_MSG_SIZE:=1013,
    WS_STATUS_PROTOCOL_ERROR:=1014,
    WS_STATUS_RCV_QUEUE_FULL:=1015,
    WS_STATUS_DECOMPRESS_ERROR:=1016
) DINT;
END_TYPE
```

5.2.2 ETcIotWebSocketContentType

```
TYPE ETcIotWebSocketContentType :
(
    WS_CONTENT_CONTINUATION:=0,
    WS_CONTENT_TEXT:=1,
    WS_CONTENT_BINARY:=2
) DINT;
END_TYPE
```

6 Beispiele

Beispielcode und -konfigurationen für dieses Produkt können über das entsprechende Repository auf GitHub bezogen werden: https://github.com/Beckhoff/TF6770_Samples. Sie haben dort die Möglichkeit das Repository zu clonen oder ein ZIP File mit dem Sample herunterzuladen.



7 Anhang

7.1 ADS Return Codes

Gruppierung der Fehlercodes:
Globale Fehlercodes: 0x0000 [▶ 31]... (0x9811_0000 ...)
Router Fehlercodes: 0x0500 [▶ 31]... (0x9811_0500 ...)
Allgemeine ADS Fehler: 0x0700 [▶ 32]... (0x9811_0700 ...)
RTime Fehlercodes: 0x1000 [▶ 34]... (0x9811_1000 ...)

Globale Fehlercodes

Hex	Dec	HRESULT	Name	Beschreibung
0x0	0	0x98110000	ERR_NOERROR	Kein Fehler.
0x1	1	0x98110001	ERR_INTERNAL	Interner Fehler.
0x2	2	0x98110002	ERR_NORTIME	Keine Echtzeit.
0x3	3	0x98110003	ERR_ALLOCLOCKEDMEM	Zuweisung gesperrt - Speicherfehler.
0x4	4	0x98110004	ERR_INSERTMAILBOX	Postfach voll – Es konnte die ADS Nachricht nicht versendet werden. Reduzieren der Anzahl der ADS Nachrichten pro Zyklus bringt Abhilfe.
0x5	5	0x98110005	ERR_WRONGRECEIVEHMSG	Falsches HMSG.
0x6	6	0x98110006	ERR_TARGETPORTNOTFOUND	Ziel-Port nicht gefunden – ADS Server ist nicht gestartet oder erreichbar.
0x7	7	0x98110007	ERR_TARGETMACHINENOTFOUND	Zielrechner nicht gefunden – AMS Route wurde nicht gefunden.
0x8	8	0x98110008	ERR_UNKNOWNCMDID	Unbekannte Befehl-ID.
0x9	9	0x98110009	ERR_BADTASKID	Ungültige Task-ID.
0xA	10	0x9811000A	ERR_NOIO	Kein IO.
0xB	11	0x9811000B	ERR_UNKNOWNAMSCMD	Unbekannter AMS-Befehl.
0xC	12	0x9811000C	ERR_WIN32ERROR	Win32 Fehler.
0xD	13	0x9811000D	ERR_PORTNOTCONNECTED	Port nicht verbunden.
0xE	14	0x9811000E	ERR_INVALIDAMSLENGTH	Ungültige AMS-Länge.
0xF	15	0x9811000F	ERR_INVALIDAMSNETID	Ungültige AMS Net ID.
0x10	16	0x98110010	ERR_LOWINSTLEVEL	Installations-Level ist zu niedrig –TwinCAT 2 Lizenzfehler.
0x11	17	0x98110011	ERR_NODEBUGINTAVAILABLE	Kein Debugging verfügbar.
0x12	18	0x98110012	ERR_PORTDISABLED	Port deaktiviert – TwinCAT System Service nicht gestartet.
0x13	19	0x98110013	ERR_PORTALREADYCONNECTED	Port bereits verbunden.
0x14	20	0x98110014	ERR_AMSSYNC_W32ERROR	AMS Sync Win32 Fehler.
0x15	21	0x98110015	ERR_AMSSYNC_TIMEOUT	AMS Sync Timeout.
0x16	22	0x98110016	ERR_AMSSYNC_AMSERROR	AMS Sync Fehler.
0x17	23	0x98110017	ERR_AMSSYNC_NOINDEXINMAP	Keine Index-Map für AMS Sync vorhanden.
0x18	24	0x98110018	ERR_INVALIDAMSPORT	Ungültiger AMS-Port.
0x19	25	0x98110019	ERR_NOMEMORY	Kein Speicher.
0x1A	26	0x9811001A	ERR_TCPSEND	TCP Sendefehler.
0x1B	27	0x9811001B	ERR_HOSTUNREACHABLE	Host nicht erreichbar.
0x1C	28	0x9811001C	ERR_INVALIDAMSFRAGMENT	Ungültiges AMS Fragment.
0x1D	29	0x9811001D	ERR_TLSSEND	TLS Sendefehler – Secure ADS Verbindung fehlgeschlagen.
0x1E	30	0x9811001E	ERR_ACCESSDENIED	Zugriff Verweigert – Secure ADS Zugriff verweigert.

Router Fehlercodes

Hex	Dec	HRESULT	Name	Beschreibung
0x500	1280	0x98110500	ROUTERERR_NOLOCKEDMEMORY	Lockierter Speicher kann nicht zugewiesen werden.
0x501	1281	0x98110501	ROUTERERR_RESIZEMEMORY	Die Größe des Routerspeichers konnte nicht geändert werden.
0x502	1282	0x98110502	ROUTERERR_MAILBOXFULL	Das Postfach hat die maximale Anzahl der möglichen Meldungen erreicht.
0x503	1283	0x98110503	ROUTERERR_DEBUGBOXFULL	Das Debug Postfach hat die maximale Anzahl der möglichen Meldungen erreicht.
0x504	1284	0x98110504	ROUTERERR_UNKNOWNPORTTYPE	Der Porttyp ist unbekannt.
0x505	1285	0x98110505	ROUTERERR_NOTINITIALIZED	Router ist nicht initialisiert.
0x506	1286	0x98110506	ROUTERERR_PORTALREADYINUSE	Die Portnummer ist bereits vergeben.
0x507	1287	0x98110507	ROUTERERR_NOTREGISTERED	Der Port ist nicht registriert.
0x508	1288	0x98110508	ROUTERERR_NOMOREQUEUES	Die maximale Portanzahl ist erreicht.
0x509	1289	0x98110509	ROUTERERR_INVALIDPORT	Der Port ist ungültig.
0x50A	1290	0x9811050A	ROUTERERR_NOTACTIVATED	Der Router ist nicht aktiv.
0x50B	1291	0x9811050B	ROUTERERR_FRAGMENTBOXFULL	Das Postfach hat die maximale Anzahl für fragmentierte Nachrichten erreicht.
0x50C	1292	0x9811050C	ROUTERERR_FRAGMENTTIMEOUT	Fragment Timeout aufgetreten.
0x50D	1293	0x9811050D	ROUTERERR_TOBEREMOVED	Port wird entfernt.

Allgemeine ADS Fehlercodes

Hex	Dec	HRESULT	Name	Beschreibung
0x700	1792	0x98110700	ADSERR_DEVICE_ERROR	Allgemeiner Gerätefehler.
0x701	1793	0x98110701	ADSERR_DEVICE_SRVNOTSUPP	Service wird vom Server nicht unterstützt.
0x702	1794	0x98110702	ADSERR_DEVICE_INVALIDGRP	Ungültige Index-Gruppe.
0x703	1795	0x98110703	ADSERR_DEVICE_INVALIDOFFSET	Ungültiger Index-Offset.
0x704	1796	0x98110704	ADSERR_DEVICE_INVALIDACCESS	Lesen oder Schreiben nicht gestattet.
0x705	1797	0x98110705	ADSERR_DEVICE_INVALIDSIZE	Parametergröße nicht korrekt.
0x706	1798	0x98110706	ADSERR_DEVICE_INVALIDDATA	Ungültige Daten-Werte.
0x707	1799	0x98110707	ADSERR_DEVICE_NOTREADY	Gerät nicht betriebsbereit.
0x708	1800	0x98110708	ADSERR_DEVICE_BUSY	Gerät beschäftigt.
0x709	1801	0x98110709	ADSERR_DEVICE_INVALIDCONTEXT	Ungültiger Kontext vom Betriebssystem - Kann durch Verwendung von ADS Bausteinen in unterschiedlichen Tasks auftreten. Abhilfe kann die Multitasking-Syncronisation in der SPS geben.
0x70A	1802	0x9811070A	ADSERR_DEVICE_NOMEMORY	Nicht genügend Speicher.
0x70B	1803	0x9811070B	ADSERR_DEVICE_INVALIDPARM	Ungültige Parameter-Werte.
0x70C	1804	0x9811070C	ADSERR_DEVICE_NOTFOUND	Nicht gefunden (Dateien,...).
0x70D	1805	0x9811070D	ADSERR_DEVICE_SYNTAX	Syntax-Fehler in Datei oder Befehl.
0x70E	1806	0x9811070E	ADSERR_DEVICE_INCOMPATIBLE	Objekte stimmen nicht überein.
0x70F	1807	0x9811070F	ADSERR_DEVICE_EXISTS	Objekt ist bereits vorhanden.
0x710	1808	0x98110710	ADSERR_DEVICE_SYMBOLNOTFOUND	Symbol nicht gefunden.
0x711	1809	0x98110711	ADSERR_DEVICE_SYMBOLVERSIONINVALID	Symbol-Version ungültig – Kann durch einen Online-Change auftreten. Erzeuge einen neuen Handle.
0x712	1810	0x98110712	ADSERR_DEVICE_INVALIDSTATE	Gerät (Server) ist im ungültigen Zustand.
0x713	1811	0x98110713	ADSERR_DEVICE_TRANSMODENOTSUPP	AdsTransMode nicht unterstützt.
0x714	1812	0x98110714	ADSERR_DEVICE_NOTIFYHNDINVALID	Notification Handle ist ungültig.
0x715	1813	0x98110715	ADSERR_DEVICE_CLIENTUNKNOWN	Notification-Client nicht registriert.
0x716	1814	0x98110716	ADSERR_DEVICE_NOMOREHDLS	Keine weiteren Handles verfügbar.
0x717	1815	0x98110717	ADSERR_DEVICE_INVALIDWATCHSIZE	Größe der Notification zu groß.
0x718	1816	0x98110718	ADSERR_DEVICE_NOTINIT	Gerät nicht initialisiert.
0x719	1817	0x98110719	ADSERR_DEVICE_TIMEOUT	Gerät hat einen Timeout.
0x71A	1818	0x9811071A	ADSERR_DEVICE_NOINTERFACE	Interface Abfrage fehlgeschlagen.
0x71B	1819	0x9811071B	ADSERR_DEVICE_INVALIDINTERFACE	Falsches Interface angefordert.
0x71C	1820	0x9811071C	ADSERR_DEVICE_INVALIDCLSID	Class-ID ist ungültig.
0x71D	1821	0x9811071D	ADSERR_DEVICE_INVALIDOBJID	Object-ID ist ungültig.
0x71E	1822	0x9811071E	ADSERR_DEVICE_PENDING	Anforderung steht aus.
0x71F	1823	0x9811071F	ADSERR_DEVICE_ABORTED	Anforderung wird abgebrochen.
0x720	1824	0x98110720	ADSERR_DEVICE_WARNING	Signal-Warnung.
0x721	1825	0x98110721	ADSERR_DEVICE_INVALIDARRAYIDX	Ungültiger Array-Index.
0x722	1826	0x98110722	ADSERR_DEVICE_SYMBOLNOTACTIVE	Symbol nicht aktiv.
0x723	1827	0x98110723	ADSERR_DEVICE_ACCESSDENIED	Zugriff verweigert.
0x724	1828	0x98110724	ADSERR_DEVICE_LICENSENOTFOUND	Fehlende Lizenz.
0x725	1829	0x98110725	ADSERR_DEVICE_LICENSEEXPIRED	Lizenz abgelaufen.
0x726	1830	0x98110726	ADSERR_DEVICE_LICENSEEXCEEDED	Lizenz überschritten.
0x727	1831	0x98110727	ADSERR_DEVICE_LICENSEINVALID	Lizenz ungültig.
0x728	1832	0x98110728	ADSERR_DEVICE_LICENSESYSTEMID	Lizenzproblem: System-ID ist ungültig.
0x729	1833	0x98110729	ADSERR_DEVICE_LICENSENOTIMELIMIT	Lizenz nicht zeitlich begrenzt.
0x72A	1834	0x9811072A	ADSERR_DEVICE_LICENSEFUTUREISSUE	Lizenzproblem: Zeitpunkt in der Zukunft.
0x72B	1835	0x9811072B	ADSERR_DEVICE_LICENSETIMETOLONG	Lizenz-Zeitraum zu lang.
0x72C	1836	0x9811072C	ADSERR_DEVICE_EXCEPTION	Exception beim Systemstart.
0x72D	1837	0x9811072D	ADSERR_DEVICE_LICENSEDUPLICATED	Lizenz-Datei zweimal gelesen.
0x72E	1838	0x9811072E	ADSERR_DEVICE_SIGNATUREINVALID	Ungültige Signatur.
0x72F	1839	0x9811072F	ADSERR_DEVICE_CERTIFICATEINVALID	Zertifikat ungültig.
0x730	1840	0x98110730	ADSERR_DEVICE_LICENSEOEMNOTFOUND	Public Key vom OEM nicht bekannt.
0x731	1841	0x98110731	ADSERR_DEVICE_LICENSERESTRICTED	Lizenz nicht gültig für diese System.ID.
0x732	1842	0x98110732	ADSERR_DEVICE_LICENSEDEMODENIED	Demo-Lizenz untersagt.
0x733	1843	0x98110733	ADSERR_DEVICE_INVALIDFNCID	Funktions-ID ungültig.
0x734	1844	0x98110734	ADSERR_DEVICE_OUTOFRANGE	Außerhalb des gültigen Bereiches.
0x735	1845	0x98110735	ADSERR_DEVICE_INVALIDALIGNMENT	Ungültiges Alignment.

Hex	Dec	HRESULT	Name	Beschreibung
0x736	1846	0x98110736	ADSERR_DEVICE_LICENSEPLATFORM	Ungültiger Plattform Level.
0x737	1847	0x98110737	ADSERR_DEVICE_FORWARD_PL	Kontext – Weiterleitung zum Passiv-Level.
0x738	1848	0x98110738	ADSERR_DEVICE_FORWARD_DL	Kontext – Weiterleitung zum Dispatch-Level.
0x739	1849	0x98110739	ADSERR_DEVICE_FORWARD_RT	Kontext – Weiterleitung zur Echtzeit.
0x740	1856	0x98110740	ADSERR_CLIENT_ERROR	Clientfehler.
0x741	1857	0x98110741	ADSERR_CLIENT_INVALIDPARM	Dienst enthält einen ungültigen Parameter.
0x742	1858	0x98110742	ADSERR_CLIENT_LISTEMPTY	Polling-Liste ist leer.
0x743	1859	0x98110743	ADSERR_CLIENT_VARUSED	Var-Verbindung bereits im Einsatz.
0x744	1860	0x98110744	ADSERR_CLIENT_DUPLINVOKEID	Die aufgerufene ID ist bereits in Benutzung.
0x745	1861	0x98110745	ADSERR_CLIENT_SYNCTIMEOUT	Timeout ist aufgetreten – Die Gegenstelle antwortet nicht im vorgegebenen ADS Timeout. Die Routeneinstellung der Gegenstelle kann falsch konfiguriert sein.
0x746	1862	0x98110746	ADSERR_CLIENT_W32ERROR	Fehler im Win32 Subsystem.
0x747	1863	0x98110747	ADSERR_CLIENT_TIMEOUTINVALID	Ungültiger Client Timeout-Wert.
0x748	1864	0x98110748	ADSERR_CLIENT_PORTNOTOPEN	Port nicht geöffnet.
0x749	1865	0x98110749	ADSERR_CLIENT_NOAMSADDR	Keine AMS Adresse.
0x750	1872	0x98110750	ADSERR_CLIENT_SYNCINTERNAL	Interner Fehler in Ads-Sync.
0x751	1873	0x98110751	ADSERR_CLIENT_ADDHASH	Überlauf der Hash-Tabelle.
0x752	1874	0x98110752	ADSERR_CLIENT_REMOVEHASH	Schlüssel in der Tabelle nicht gefunden.
0x753	1875	0x98110753	ADSERR_CLIENT_NOMORESYM	Keine Symbole im Cache.
0x754	1876	0x98110754	ADSERR_CLIENT_SYNCRESINVALID	Ungültige Antwort erhalten.
0x755	1877	0x98110755	ADSERR_CLIENT_SYNCPORTLOCKED	Sync Port ist verriegelt.
0x756	1878	0x98110756	ADSERR_CLIENT_REQUESTCANCELLED	Die Anfrage wurde abgebrochen.

RTime Fehlercodes

Hex	Dec	HRESULT	Name	Beschreibung
0x1000	4096	0x98111000	RTERR_INTERNAL	Interner Fehler im Echtzeit-System.
0x1001	4097	0x98111001	RTERR_BADTIMERPERIODS	Timer-Wert nicht gültig.
0x1002	4098	0x98111002	RTERR_INVALIDTASKPTR	Task-Pointer hat den ungültigen Wert 0 (null).
0x1003	4099	0x98111003	RTERR_INVALIDSTACKPTR	Stack-Pointer hat den ungültigen Wert 0 (null).
0x1004	4100	0x98111004	RTERR_PRIOEXISTS	Die Request Task Priority ist bereits vergeben.
0x1005	4101	0x98111005	RTERR_NOMORETCB	Kein freier TCB (Task Control Block) verfügbar. Maximale Anzahl von TCBs beträgt 64.
0x1006	4102	0x98111006	RTERR_NOMORESEMAS	Keine freien Semaphoren zur Verfügung. Maximale Anzahl der Semaphoren beträgt 64.
0x1007	4103	0x98111007	RTERR_NOMOREQUEUES	Kein freier Platz in der Warteschlange zur Verfügung. Maximale Anzahl der Plätze in der Warteschlange beträgt 64.
0x100D	4109	0x9811100D	RTERR_EXTIRQALREADYDEF	Ein externer Synchronisations-Interrupt wird bereits angewandt.
0x100E	4110	0x9811100E	RTERR_EXTIRQNOTDEF	Kein externer Sync-Interrupt angewandt.
0x100F	4111	0x9811100F	RTERR_EXTIRQINSTALLFAILED	Anwendung des externen Synchronisierungs-Interrupts ist fehlgeschlagen.
0x1010	4112	0x98111010	RTERR_IRQLNOTLESSOREQUAL	Aufruf einer Service-Funktion im falschen Kontext
0x1017	4119	0x98111017	RTERR_VMXNOTSUPPORTED	Intel VT-x Erweiterung wird nicht unterstützt.
0x1018	4120	0x98111018	RTERR_VMXDISABLED	Intel VT-x Erweiterung ist nicht aktiviert im BIOS.
0x1019	4121	0x98111019	RTERR_VMXCONTROLSMISSING	Fehlende Funktion in Intel VT-x Erweiterung.
0x101A	4122	0x9811101A	RTERR_VMXENABLEFAILS	Aktivieren von Intel VT-x schlägt fehl.

Spezifische positive HRESULT Return Codes:

HRESULT	Name	Beschreibung
0x0000_0000	S_OK	Kein Fehler.
0x0000_0001	S_FALSE	Kein Fehler. Bsp.: erfolgreiche Abarbeitung, bei der jedoch ein negatives oder unvollständiges Ergebnis erzielt wurde.
0x0000_0203	S_PENDING	Kein Fehler. Bsp.: erfolgreiche Abarbeitung, bei der jedoch noch kein Ergebnis vorliegt.
0x0000_0256	S_WATCHDOG_TIMEOUT	Kein Fehler. Bsp.: erfolgreiche Abarbeitung, bei der jedoch eine Zeitüberschreitung eintrat.

TCP Winsock-Fehlercodes

Hex	Dec	Name	Beschreibung
0x274C	10060	WSAETIMEDOUT	Verbindungs Timeout aufgetreten - Fehler beim Herstellen der Verbindung, da die Gegenstelle nach einer bestimmten Zeitspanne nicht ordnungsgemäß reagiert hat, oder die hergestellte Verbindung konnte nicht aufrecht erhalten werden, da der verbundene Host nicht reagiert hat.
0x274D	10061	WSAECONNREFUSED	Verbindung abgelehnt - Es konnte keine Verbindung hergestellt werden, da der Zielcomputer dies explizit abgelehnt hat. Dieser Fehler resultiert normalerweise aus dem Versuch, eine Verbindung mit einem Dienst herzustellen, der auf dem fremden Host inaktiv ist—das heißt, einem Dienst, für den keine Serveranwendung ausgeführt wird.
0x2751	10065	WSAEHOSTUNREACH	Keine Route zum Host - Ein Socketvorgang bezog sich auf einen nicht verfügbaren Host.
Weitere Winsock-Fehlercodes: Win32-Fehlercodes			

Mehr Informationen:
www.beckhoff.com/tf6770/

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