BECKHOFF New Automation Technology

Handbuch | DE

TS6100

TwinCAT 2 | OPC UA Server

Supplement | Communication

25.04.2025 | Version: 1.3

Inhaltsverzeichnis

1 Vorwort ... 7
1.1 Hinweise zur Dokumentation ... 7
1.2 Zu Ihrer Sicherheit ... 8
1.3 Hinweise zur Informationssicherheit ... 9
2 Übersicht ... 10
3 Installation ... 12
3.1 Systemvoraussetzungen ... 12
3.2 Installation ... 13
3.3 Updateinstallation ... 16
3.4 Installationsvarianten ... 16
3.5 Lizenzierung ... 17
4 Technische Einführung ... 21
4.1 Quick Start ... 21
4.2 Quick Start (TwinCAT 2) ... 24
4.3 Initialisierung ... 28
4.4 Empfohlene Schritte ... 34
4.5 Unterstützte Features ... 37
4.6 Softwarearchitektur ... 39
4.7 Konfigurator ... 40
4.8 Optimierungen ... 42
4.9 Applikationsverzeichnisse ... 47
4.10 Data Access ... 49
4.10.1 Überblick ... 49
4.10.2 Verbindung mit der Laufzeit ... 50
4.10.3 Freigabe von Symbolen ... 53
4.10.4 Nodesets ... 70
4.10.5 Datentypen ... 71
4.10.6 Arrays ... 72
4.10.7 Enums ... 74
4.10.8 Strukturen ... 75
4.10.9 Properties ... 78
4.10.10 StatusCode ... 79
4.10.11 AnalogItemType ... 82
4.10.12 Description ... 83
4.10.13 ReadOnly ... 84
4.10.14 Alias ... 85
4.10.15 Pointer und Referenzen ... 87
4.10.16 Typsystem ... 87
4.10.17 DI Components ... 90
4.10.18 DeviceState ... 91
4.10.19 ServerState ... 92
4.11 Historical Access ... 93
4.11.1 Überblick ... 93

4.11.2 Unterstützte Funktionen ... 93
4.11.3 Konfiguration ... 95
4.11.4 HistoryUpdate ... 96
4.11.5 TwinCAT Analytics ... 97
4.11.6 Zugriff auf historische Daten ... 97
4.12 Alarms and Conditions ... 100
4.12.1 Überblick ... 100
4.12.2 Unterstützte Funktionen ... 101
4.12.3 Konfiguration ... 101
4.12.4 Zusätzliche Applikationsdaten ... 104
4.12.5 Zugriff auf Alarme und Events ... 106
4.13 Methodenaufrufe ... 107
4.13.1 Überblick ... 107
4.13.2 Job-Methoden ... 107
4.13.3 RPC-Methoden ... 110
4.14 TwinCAT Eventlogger ... 114
4.14.1 Überblick ... 114
4.14.2 Konfiguration ... 114
4.14.3 Zugriff auf Alarme und Events ... 115
4.15 Global Discovery Server ... 119
4.15.1 Überblick ... 119
4.15.2 Push ... 119
4.15.3 Pull ... 120
4.16 Security ... 124
4.16.1 Übersicht ... 124
4.16.2 Endpunkte ... 125
4.16.3 Zertifikatsaustausch ... 126
4.16.4 Authentifizierung ... 128
4.16.5 Zugriffsrechte ... 131
4.17 File Transfer ... 133
4.17.1 Überblick ... 133
4.17.2 Konfiguration ... 134
4.18 Reverse Connect ... 135
4.19 Redundanz ... 138
4.20 Logging ... 142
4.21 System Tray ... 143

5 SPS API ... 144
5.1 Tc2_OpcUa ... 144
5.1.1 Datentypen ... 144
5.1.2 Funktionsbausteine ... 145

6 Beispiele ... 148

7 Tutorials ... 149

8 Anhang ... 150
8.1 Attribute und Kommentare ... 150
8.2 32-bit und 64-bit Prozess ... 151

8.3 Fehlerdiagnose 153
8.4 ADS Return Codes 157
8.5 Support und Service 161

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.

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

Copyright

© Beckhoff Automation GmbH & Co. KG, Deutschland.
Weitergabe sowie Vervielfältigung dieses Dokuments, Verwertung und Mitteilung seines Inhalts sind verboten, soweit nicht ausdrücklich gestattet.
Zuwiderhandlungen verpflichten zu Schadenersatz. Alle Rechte für den Fall der Patent-, Gebrauchsmuster- oder Geschmacksmustereintragung vorbehalten.

Fremdmarken

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

1.2 Zu Ihrer Sicherheit

Sicherheitsbestimmungen

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

Haftungsausschluss

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

Qualifikation des Personals

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

Signalwörter

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

Warnungen vor Personenschäden

⚠ GEFAHR

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

⚠ WARNUNG

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

⚠ VORSICHT

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

Warnung vor Umwelt- oder Sachschäden

HINWEIS

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

Information zum Umgang mit dem Produkt

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

1.3 Hinweise zur Informationssicherheit

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

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

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

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

2 Übersicht

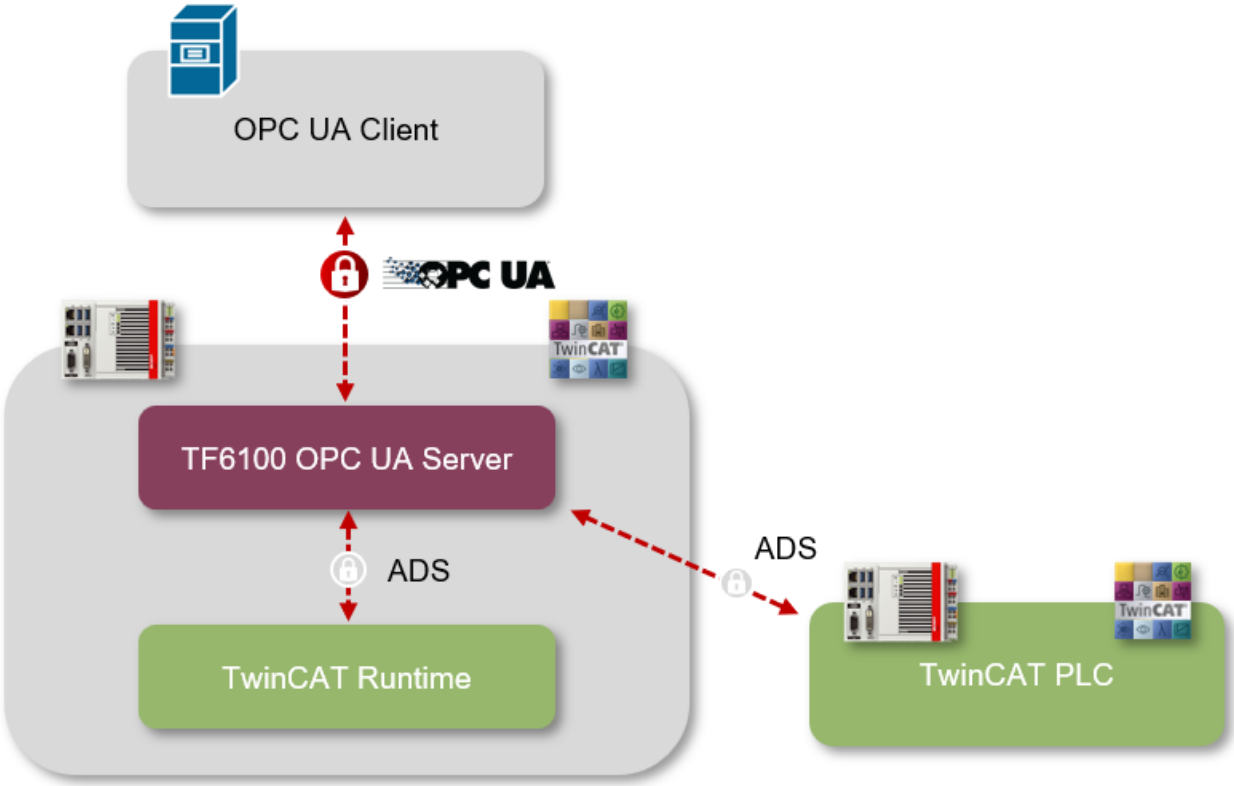
OPC Unified **A**rchitecture (OPC UA) ist die nächste Generation des klassischen OPC-Standards. Es handelt sich hierbei um ein weltweit standardisiertes Kommunikationsprotokoll, über das Maschinendaten hersteller- und plattformunabhängig ausgetauscht werden können. OPC UA integriert gängige Sicherheitsstandards bereits direkt im Protokoll. Ein weiterer großer Vorteil von OPC UA gegenüber dem klassischen OPC-Standard ist die Unabhängigkeit vom COM/DCOM-System.

Detaillierte Informationen zu OPC UA finden Sie auf der Webseite der OPC Foundation.

Die TwinCAT 3 Function TF6100 OPC UA besteht aus mehreren Softwarekomponenten, welche einen Datenaustausch mit TwinCAT, basierend auf OPC UA, ermöglichen.
Die folgende Tabelle gibt einen Überblick über die einzelnen Produktbestandteile.

Software-Komponente	Beschreibung
TwinCAT OPC UA Server	Stellt eine OPC-UA-Server-Schnittstelle zur Verfügung, damit UA-Clients auf die TwinCAT-Laufzeit zugreifen können.
TwinCAT OPC UA Client	Stellt eine OPC-UA-Client-Funktionalität zur Verfügung, damit die Kommunikation mit anderen OPC UA Servern auf der Grundlage von PLCopen-genormten Funktionsbausteinen sowie einem einfach zu konfigurierenden I/O-Gerät möglich ist.
TwinCAT OPC UA Configurator	Grafische Benutzerschnittstelle für die Konfiguration des TwinCAT OPC UA Servers.
TwinCAT OPC UA Sample Client	Grafische Beispielimplementierung eines OPC UA Clients um einen ersten Verbindungstest mit dem TwinCAT OPC UA Server durchführen zu können.
TwinCAT OPC UA Gateway	Wrapper-Technologie, die sowohl eine OPC-COM-DA-Server-Schnittstelle als auch OPC-UA-Server-Aggregationsfähigkeiten zur Verfügung stellt.

Diese Dokumentation beschreibt den TwinCAT OPC UA Server, bei welchem es sich um eine OPC-UA-Serverapplikation handelt, die den Zugriff auf Symbole aus der TwinCAT-Echtzeitumgebung ermöglicht. Der Server kann hierbei entweder direkt auf der Steuerung oder auf einem Gateway-PC betrieben werden. Beide Zusammenhänge sind in folgendem Schaubild vereinfacht dargestellt und werden im Kapitel Installationsvarianten [▶ 16] näher erläutert.


OPC UA Client
OPC UA
TwinCAT
TF6100 OPC UA Server
ADS
TwinCAT Runtime
ADS
TwinCAT
TwinCAT PLC

3 Installation

3.1 Systemvoraussetzungen

Für die Installation und den Betrieb dieses Produkts gelten die folgenden Systemvoraussetzungen.

Server

Technische Daten	Beschreibung
Betriebssystem	Windows 10 Windows CE 6/7 (nur bis inklusive Setup-Version 5.1.x) Windows Server 2022 TwinCAT/BSD TwinCAT/Linux®
Zielplattformen	PC-Architektur (x86, x64, Arm®)
.NET Framework	---
TwinCAT-Version	TwinCAT 2, TwinCAT 3
TwinCAT-Installationslevel	TwinCAT 2 CP, PLC, NC-PTP TwinCAT 3 XAE, XAR, ADS
Benötigte TwinCAT-Lizenz	TS6100 TwinCAT OPC UA (für TwinCAT 2) TF6100 TC3 OPC UA (für TwinCAT 3)

Sample Client

Technische Daten	Beschreibung
Betriebssystem	Windows 10 Windows Server 2022
Zielplattformen	PC-Architektur (x86, x64, Arm®)
.NET Framework	4.7.2
TwinCAT-Version	TwinCAT 2, TwinCAT 3
TwinCAT-Installationslevel	TwinCAT 2 CP, PLC, NC-PTP TwinCAT 3 XAE, XAR, ADS
Benötigte TwinCAT-Lizenz	---

Kompatibilität TwinCAT 2/TwinCAT 3

Diese Dokumentation hat sowohl für das TwinCAT 2 Supplement (TS6100) als auch für die TwinCAT 3 Function (TF6100) Gültigkeit.
Falls in einzelnen Teilfunktionen Unterschiede bestehen, weist eine entsprechende Informationsbox an direkter Stelle darauf hin und enthält gegebenenfalls zusätzliche Erläuterungen.

Firewall-Port

Um eine Kommunikation mit dem TwinCAT OPC UA Server zu ermöglichen, muss der folgende Netzwerkport in der Firewall des Geräts geöffnet werden:

```
4840/tcp (incoming)
```

Wenn der TwinCAT OPC UA Server zum Beispiel auf einem Beckhoff Industrie-PC installiert wird, muss in der Firewall des Betriebssystems dieser Port als eingehende Kommunikation geöffnet werden.

Hardwareanforderungen

Die Anforderungen des TwinCAT OPC UA Servers an die unterlagerte Industrie-PC-Hardware hängen stark vom jeweiligen Einsatzszenario ab. Generell kann man zwei Metriken identifizieren, welche für den Einsatz des Servers entscheidend sind:
Arbeitsspeicher- und CPU-Auslastung.

Die Arbeitsspeicherauslastung des Servers variiert zum einen je nach Anzahl an Symbolen, die über die Data Access [▶ 49] Geräte in den Adressraum des Servers importiert werden, und zum anderen je nach Anzahl an gleichzeitigen Verbindungen, Subscriptions, Historical Access Aufrufen, ..., die ein OPC UA Client anlegt. Die Grundlast des Servers orientiert sich jedoch an der Symbolanzahl und kann mit circa 1024 Bytes pro Symbol festgelegt werden. Bei einer Symbolanzahl von 1.000.000 Symbolen bedeutet dies zum Beispiel eine Arbeitsspeicherauslastung von circa

```
1024 Bytes * 1.000.000 Symbole = 1.024 Mbyte
```

Arbeitsspeicherauslastung des Industrie-PC beachten

Bitte stellen Sie sicher, dass der entsprechend genutzte Industrie-PC einen ausreichend großen Arbeitsspeicher aufweist.

Für einen ersten Test können Sie Ihr Projekt auf Ihrem Engineering-PC in Betrieb nehmen und dort die Arbeitsspeicherauslastung des TwinCAT OPC UA Servers im Windows-Task-Manager auslesen. Beachten Sie hierbei etwaige Grenzen des Betriebssystems, zum Beispiel die Arbeitsspeichergrenze von 2 GB für 32-bit Prozesse. Hiervon ist zum Beispiel die 32-bit Variante des TwinCAT OPC UA Servers betroffen. Benötigen Sie mehr als 2 GB Arbeitsspeicher für den Prozess, so müssen Sie auf die 64-bit Variante des Servers zurückgreifen.

Die CPU-Auslastung des Servers hängt hingegen ausschließlich von den zur Laufzeit herrschenden Bedingungen ab, insbesondere der Anzahl an Subscriptions und MonitoredItems sowie deren Konfigurationsparameter, die ein OPC UA Client beim Server anfordert. Das Kapitel Optimierungen [▶ 42] gibt Ihnen hierzu weitere Informationen.

3.2 Installation

Die Installation dieser TwinCAT 3 Function kann, abhängig von der verwendeten TwinCAT-Version und dem Betriebssystem, auf unterschiedliche Arten erfolgen, welche im Folgenden näher beschrieben werden sollen.

HINWEIS

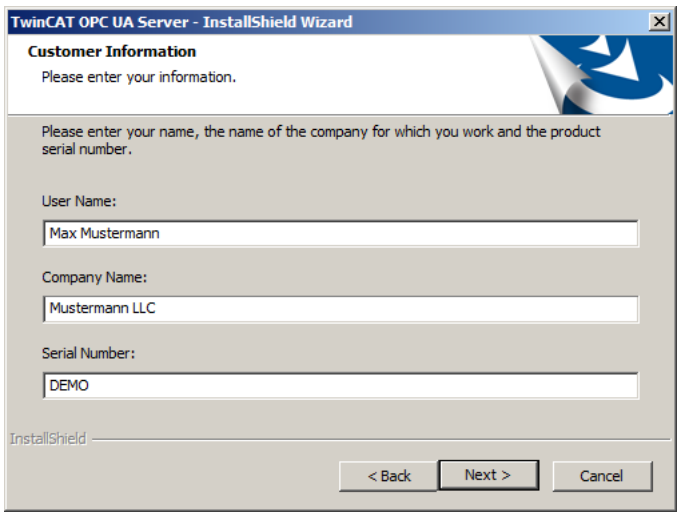
Updateinstallation

Bei einer Updateinstallation wird immer die vorherige Installation deinstalliert. Bitte stellen Sie sicher, dass Sie vorher ein Backup Ihrer Konfigurationsdateien erstellt haben.

TwinCAT 2 Setup

Wenn Sie TwinCAT 2 verwenden, so können Sie dieses Supplement über ein Setup-Paket installieren, welches Sie auf der Beckhoff Webseite unter https://www.beckhoff.com/download herunterladen können.

Die Installation kann hierbei auf Engineering- als auch Runtime-Seite erfolgen, je nachdem, auf welchem System Sie das Supplement benötigen. Bitte beachten Sie, dass unter TwinCAT 2 die Lizensierung des Supplements direkt während der Installation erfolgt. Ein separater Dialog fordert Sie entsprechend zur Eingabe des Lizenzschlüssels auf. Wenn Sie eine 30-Tage Testversion des Supplements installieren möchten, so geben Sie als Lizenzschlüssel bitte DEMO ein.



Für TwinCAT 2 auf Windows CE gibt es einen separaten Setup-Download mit dem Namen TS6100-0030 OPC UA. Dieses Setup installiert die CAB-Files, welche Sie dann auf ein Windows CE Gerät übertragen und dort installieren können.

TwinCAT 3 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.

HINWEIS

Unvorbereiteter TwinCAT-Neustart kann Datenverlust erzeugen

Die Installation dieser Function hat unter Umständen einen TwinCAT-Neustart zur Folge.
Stellen Sie sicher, dass keine kritischen TwinCAT-Applikationen auf dem System laufen oder fahren Sie diese zunächst geordnet herunter.

Kommandozeilenprogramm TcPkg

Über das TcPkg **C**ommand **L**ine **I**nterface (CLI) können Sie sich die verfügbaren Workloads auf dem System anzeigen lassen, sowie eine bestimmte Workload installieren.

```
tcpkg list -t workload
tcpkg install TF6100.OpcUaServer.XAE
tcpkg install TF6100.OpcUaServer.XAR
```

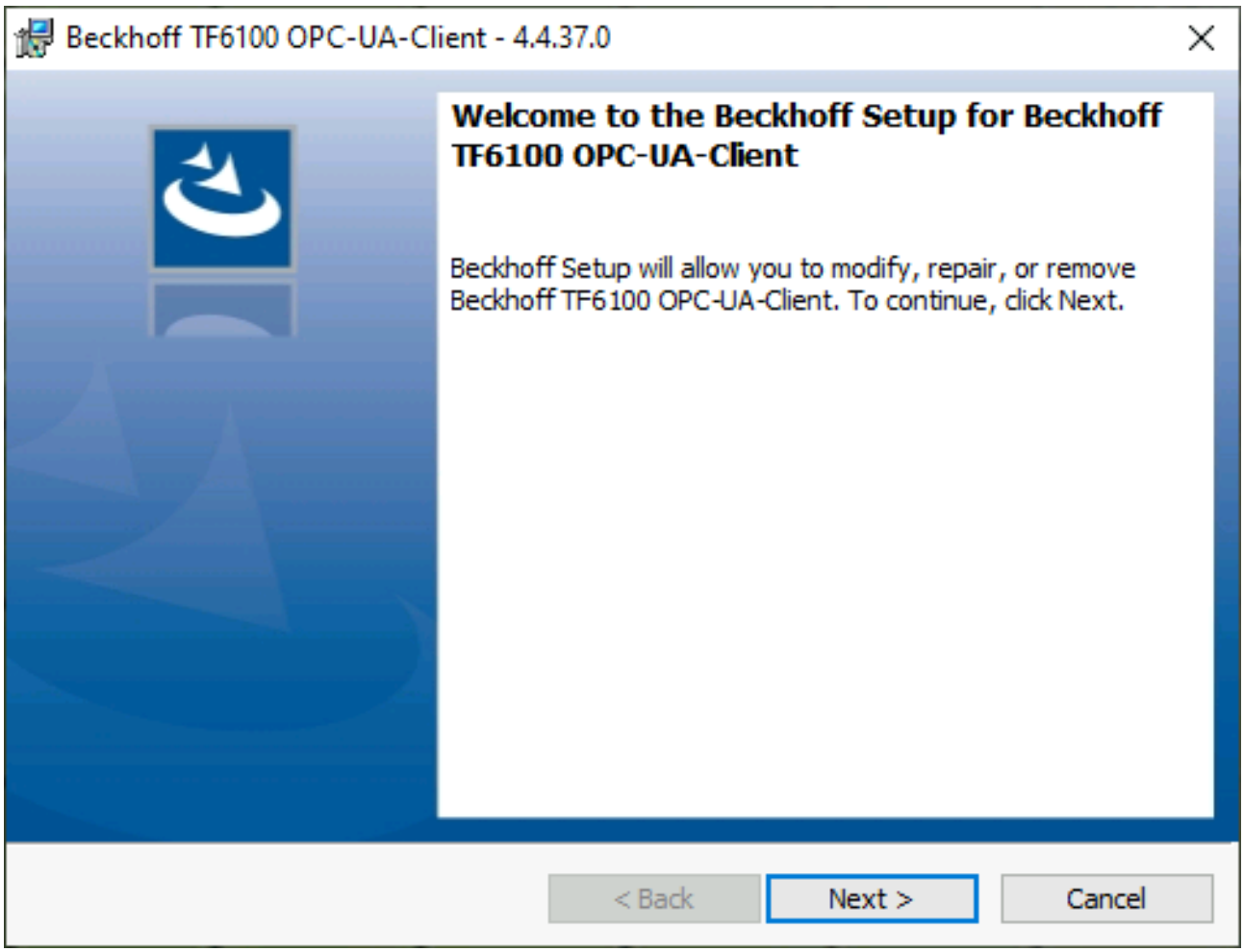
TwinCAT Package Manager UI

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.

TwinCAT 3 Setup

Wenn Sie TwinCAT 3.1 Build 4024 auf dem Betriebssystem Microsoft Windows verwenden, können Sie diese Function über ein Setup-Paket installieren, welches Sie auf der Beckhoff Webseite unter https://www.beckhoff.com/download herunterladen können.

Die Installation kann hierbei sowohl auf Engineering- als auch Runtime-Seite erfolgen, je nachdem, auf welchem System Sie die Function benötigen. Der folgende Screenshot zeigt exemplarisch die Setup-Oberfläche am Beispiel des TF6100 TwinCAT OPC UA Client-Setups.



Zur Durchführung des Installationsvorgangs, folgen Sie den entsprechenden Anweisungen im Setup-Dialog.

HINWEIS

Unvorbereiteter TwinCAT-Neustart kann Datenverlust erzeugen

Die Installation dieser Function hat unter Umständen einen TwinCAT-Neustart zur Folge.
Stellen Sie sicher, dass keine kritischen TwinCAT-Applikationen auf dem System laufen oder fahren Sie diese zunächst geordnet herunter.

TwinCAT/BSD

Wenn Sie als Betriebssystem TwinCAT/BSD verwenden, so können Sie diese Function über den TwinCAT/BSD Package Server installieren. Weitere Informationen zum TwinCAT/BSD Package Server und dessen Konfiguration entnehmen Sie bitte der TwinCAT/BSD Dokumentation.

Zum Suchen eines Packages können Sie in der TwinCAT/BSD Konsole den folgenden Aufruf verwenden:

```
pkg search TF6100
```

Zur Installation einer Function können Sie in der TwinCAT/BSD Konsole den folgenden Aufruf verwenden:

```
doas pkg install TF6100-OPC-UA-Server-<version>
```

Windows CE

Wenn Sie als Betriebssystem Microsoft Windows CE verwenden, können Sie diese Function über die jeweiligen CAB-Dateien installieren, welche sie als separate Dateien auf der Beckhoff Webseite unter https://www.beckhoff.com/download herunterladen können.

Nach dem Download können die CAB-Dateien per Dateitransfer auf das Windows CE-Gerät übertragen und dort ausgeführt werden. Die CAB-Dateien installieren und registrieren dann die Function auf dem jeweiligen System.

Verwenden Sie jeweils die CAB-Datei passend zu Ihrem System. Konkret bedeutet dies:

- CE-ARMV4I: Arm®-basierte Geräte, z. B. CX8190, CX9020
- CE-X86: x86-basierte Geräte, z. B. CX51xx, CX52xx, CX20xx

Der Dateitransfer der CAB-Datei auf das Gerät kann entweder über die CF/SD-Karte oder den im Windows CE integrierten FTP-Server erfolgen.

Geräteneustart

Nach der Installation dieser Function ist ein Geräteneustart erforderlich, damit die Function verwendet werden kann.

3.3 Updateinstallation

Bei einer Updateinstallation werden alte Applikationsdateien durch neue ersetzt. Hierbei werden von existierenden Konfigurationsdateien Backups erstellt und in der Form *.xml.bak im Applikationsverzeichnis [▶ 47] abgespeichert. Dies gilt ebenfalls für die Zertifikatsverzeichnisse.

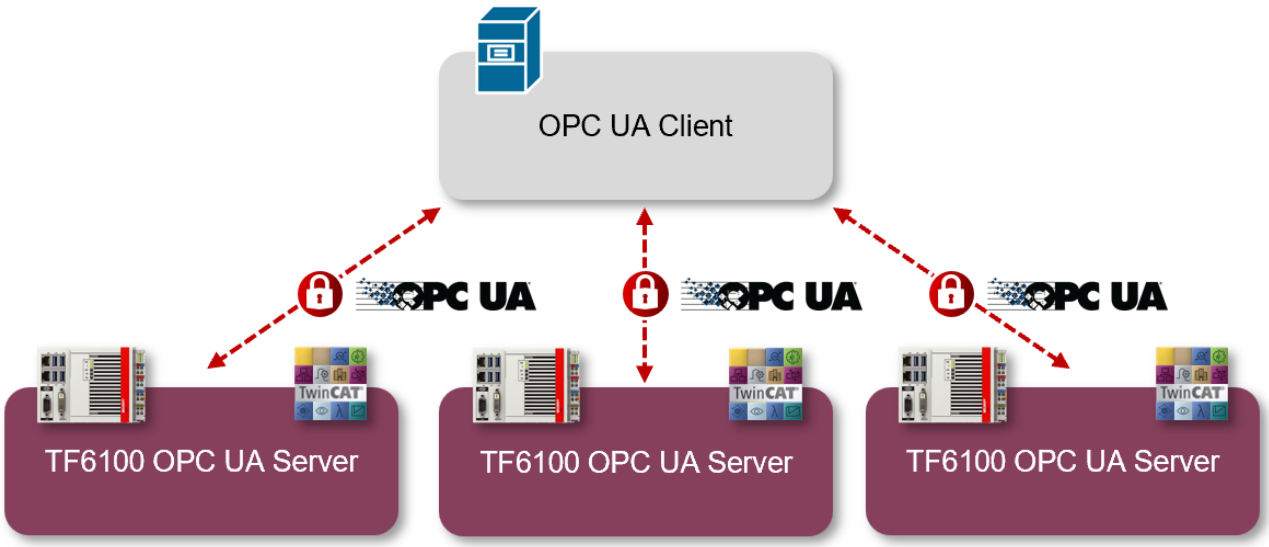
3.4 Installationsvarianten

Im Folgenden werden zwei Installationsvarianten erläutert, gemäß derer der TwinCAT OPC UA Server je nach Anwendungsfall und Infrastruktur installiert werden kann.

1. Server direkt in die Steuerung integriert
2. Betrieb des Servers auf einem Gateway-PC

Server direkt in die Steuerung integriert

Dieses Szenario beschreibt, wie der TwinCAT OPC UA Server üblicherweise betrieben werden sollte. Einer der größten Vorteile dieser Software besteht darin, dass sie selbst in die kleinste Embedded-Plattform, z. B. die CX8000-Baureihe, integriert werden kann. Dank dieser Integration ist das allgemeine Handling sehr einfach und komfortabel. OPC UA Clients, z. B. HMI-oder MES/ERP-Systeme, können eine Verbindung mit den entsprechenden TwinCAT OPC UA Servern im Netzwerk herstellen und aus der TwinCAT-Laufzeit stammende Symbolinformationen lesen und schreiben.



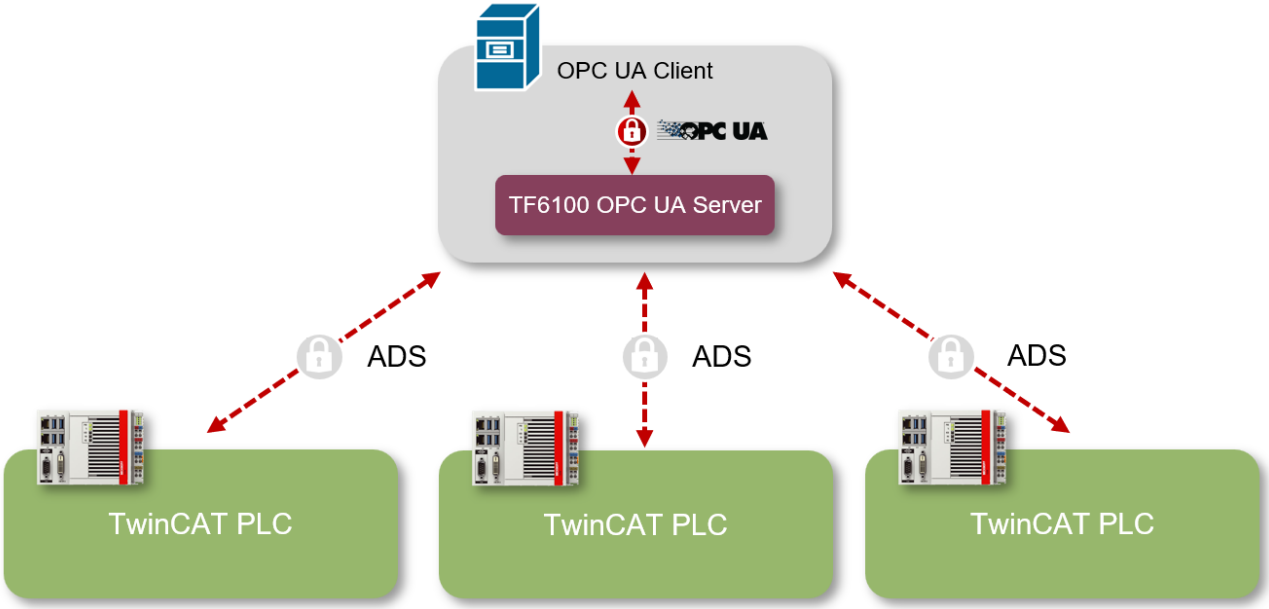
Dieses Szenario hat folgende Vorteile:

- Eine optimierte Netzwerkauslastung, da auf OPC UA Mechanismen, wie Subscriptions („OnDataChange Kommunikation“), aufgebaut werden kann.
- Eine dezentralisierte Speichernutzung. Jedes TwinCAT OPC UA Server Gerät ist ausschließlich für den eigenen Speicherbedarf zuständig, da nur die „eigene“ SPS-Symbolik im Adressraum des Servers bereitgestellt werden muss.

- Eine sichere Kommunikation bis in die Steuerung. OPC UA verfügt über direkt in das Protokoll integrierte Sicherheitsmechanismen. In der Firewall der Steuerung wird nur der OPC UA Server Port freigegeben, worüber dann die sichere Kommunikation erfolgt. Im Fall der Reverse Connect [▶ 135]-Funktionalität entfällt das Öffnen des (eingehenden) Firewall Ports sogar.

Betrieb des Servers auf einem Gateway-PC

Dieses Szenario beschreibt die Verwendung des TwinCAT OPC UA Servers auf einem Gateway-PC. Dieser Anwendungsfall kommt häufig in Brownfield-Szenarien zum Einsatz, bei denen existierende Steuerungssysteme eine OPC UA Schnittstelle erhalten sollen. Der TwinCAT OPC UA Server wird in diesem Fall auf einem Gateway-PC (oftmals auch als „Edge-PC“ bezeichnet) installiert und bindet eine oder mehrere unterlagerte TwinCAT-SPS-Steuerungen an. Die Kommunikation zwischen Server und SPS erfolgt dann über TwinCAT-ADS.



Aus monetärer Sicht ist dieses Szenario sehr attraktiv, denn es muss nur eine TwinCAT OPC UA Server Softwarelizenz erworben werden. Dieses Szenario weist jedoch im Vergleich zu einer Integration des Servers in die Steuerung auch einige technische Nachteile auf:

- Die Netzwerknutzung kann abhängig von der Anzahl vorhandener Geräte und Symbole sehr hoch sein. Der TwinCAT OPC UA Server verwendet ein zyklisches ADS-Sampling, um die Symbolwerte schnell und effizient aus der TwinCAT-Laufzeit abzufragen und muss dabei auch gleichzeitig in der Lage sein, tausende von Symbolen gleichzeitig bedienen zu können. Je mehr Symbole (und je mehr angebundene SPS-Steuerungen) vorhanden sind, desto mehr zyklische Kommunikation ist im Netzwerk unterwegs.
- Der Speicherbedarf auf dem zentralen Server ist sehr hoch, weil der TwinCAT OPC UA Server die Symbolinformationen von jeder TwinCAT-SPS importieren und in seinem Adressraum vorhalten muss.
- Die Kommunikation zwischen Server und SPS erfolgt auf Basis von TwinCAT-ADS, welches nur in neueren TwinCAT-Versionen eine integrierte Verschlüsselung des Transportkanals ermöglicht. Für ältere Systeme im Rahmen einer Brownfield-Anwendung steht diese womöglich noch nicht zur Verfügung.
- Bei jeder SPS-Programmänderung müssen die Symboldateien zwischen der TwinCAT-SPS und dem zentralen Server neu ausgetauscht werden. Dieser Schritt entfällt, wenn der Server direkt in der SPS-Steuerung betrieben wird.

3.5 Lizenzierung

Dieses TwinCAT Produkt benötigt für den vollumfänglichen Betrieb eine Lizenz. Zusätzlich wird die Freischaltung einer Testlizenz ermöglicht, um das Produkt evaluieren zu können.

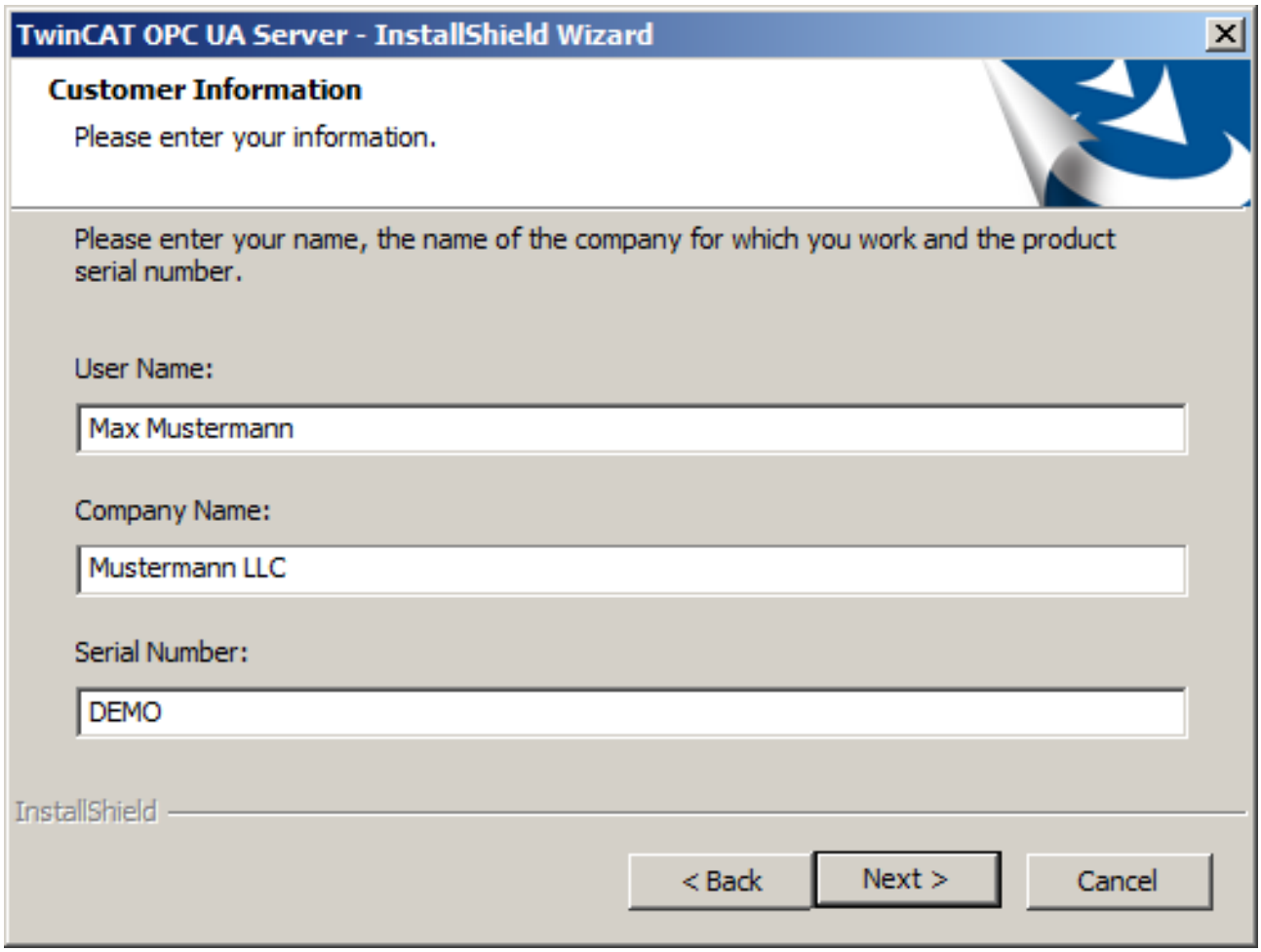
In den nachfolgenden Abschnitten finden Sie weitere Informationen zur Lizenzierung des Produkts:

- Lizenzierung unter TwinCAT 2
- Lizenzierung einer 7-Tage Testversion unter TwinCAT 3
- Lizenzierung einer Vollversion unter TwinCAT 3

Weitere Informationen zur TwinCAT-3-Lizenzierung finden Sie im Beckhoff Information System in der Dokumentation „Lizenzierung“ (TwinCAT 3 > Lizenzierung).

Lizenzierung unter TwinCAT 2

Die Lizenzierung des Produkts erfolgt unter TwinCAT 2 während der Installation, ein entsprechendes Textfeld ermöglicht die Eingabe eines Lizenzschlüssels. Um dieses Produkt zu evaluieren, können Sie es für einen Zeitraum von 30-Tagen freischalten. Hierfür geben Sie als Lizenzschlüssel „Demo“ ein.

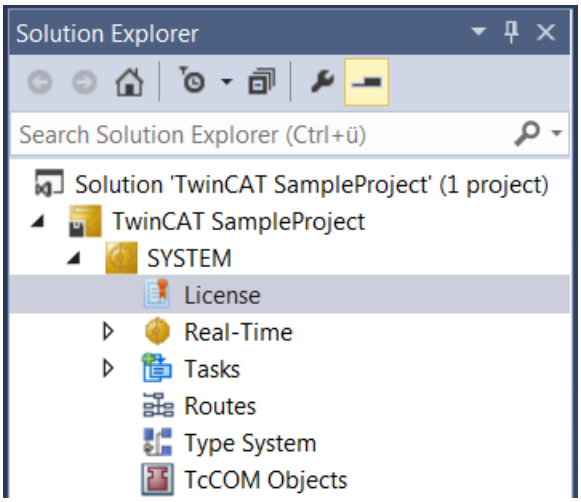


Lizenzierung der 7-Tage-Testversion einer TwinCAT 3 Function

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

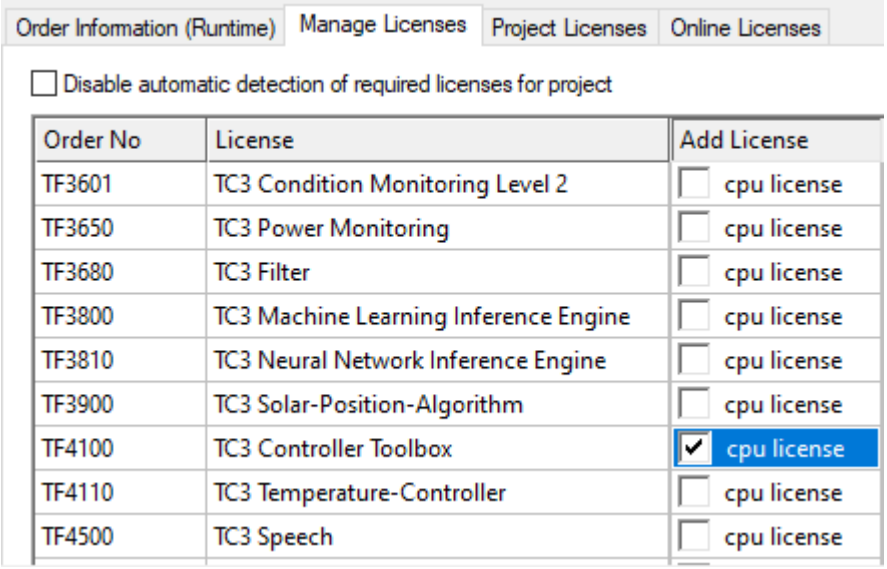
1. Starten Sie die TwinCAT-3-Entwicklungsumgebung (XAE).
2. Öffnen Sie ein bestehendes TwinCAT-3-Projekt oder legen Sie ein neues Projekt an.
3. Wenn Sie die Lizenz für ein Remote-Gerät aktivieren wollen, stellen Sie das gewünschte Zielsystem ein. Wählen Sie dazu in der Symbolleiste in der Drop-down-Liste **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“).

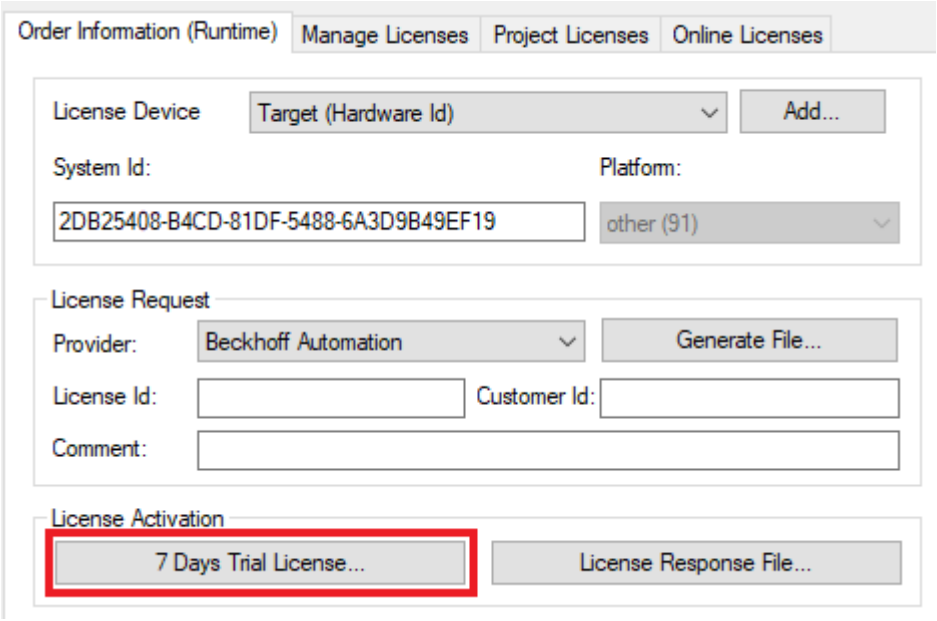


Order No	License	Add License
TF3601	TC3 Condition Monitoring Level 2	cpu license
TF3650	TC3 Power Monitoring	cpu license
TF3680	TC3 Filter	cpu license
TF3800	TC3 Machine Learning Inference Engine	cpu license
TF3810	TC3 Neural Network Inference Engine	cpu license
TF3900	TC3 Solar-Position-Algorithm	cpu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

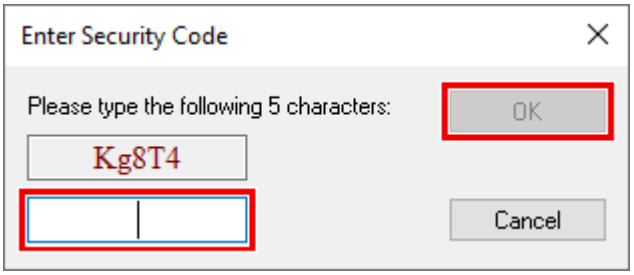
6. Öffnen Sie die Registerkarte **Order Information (Runtime).**

⇨ 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.

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“.

4 Technische Einführung

4.1 Quick Start

Das folgende Kapitel ermöglicht einen Schnelleinstieg in den TwinCAT OPC UA Server. In dieser Anleitung initialisieren Sie den TwinCAT OPC UA Server im Auslieferungszustand, erstellen anschließend ein TwinCAT-SPS-Projekt und geben dann eine SPS-Variable durch Setzen eines Pragmas über OPC UA frei. Die Variable ist anschließend im Adressraum des Servers verfügbar.
Im Folgenden werden die Handlungsschritte ihrer Reihenfolge nach genauer beschrieben:

- Initialisieren des Servers
- Erstellen eines TwinCAT-SPS-Projekts
- Aktivieren des Symboldateien-Downloads
- Hinzufügen einer Produktlizenz
- Freigabe einer Variablen
- Verbinden eines OPC UA Clients

Initialisieren des Servers

Nach der Installation ist eine einmalige Initialisierung [▶ 28] des Servers nach dem sogenannten TOFU (**T**rust-**O**n-**F**irst-**U**se)-Prinzip notwendig. Hierbei konfigurieren Sie ein Benutzerkonto, welches anschließend für den Verbindungsaufbau mit dem Server benötigt wird.

Da es sich hierbei um ein zentrales- und Security-relevantes Thema handelt, wird die Initialisierung [▶ 28] in einem separaten Dokumentationskapitel im Detail erläutert. Die weiteren Schritte in diesem Abschnitt gehen davon aus, dass dieser Vorgang einmal durchgeführt wurde und Sie den Server mit einem Benutzerkonto initialisiert haben.

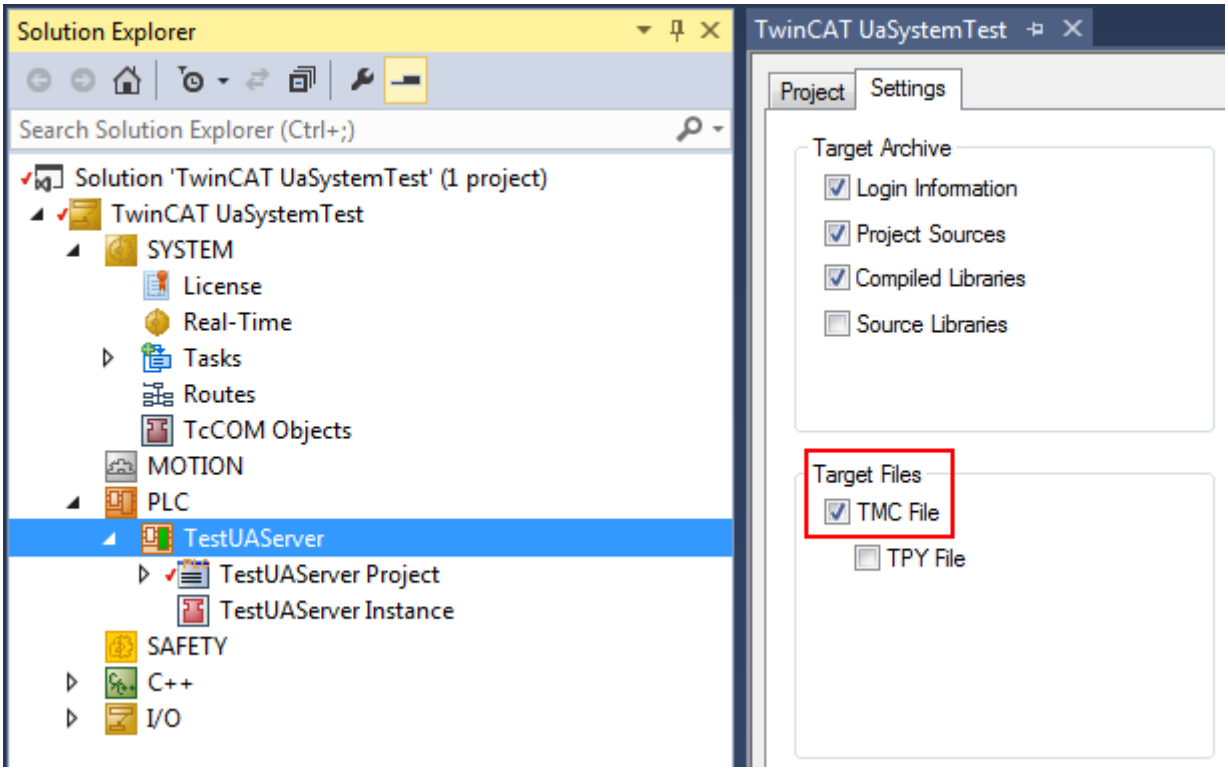
Tools für die Initialisierung des Servers

Für die Initialisierung des Servers können Sie den TwinCAT OPC UA Configurator verwenden. Hierfür ist eventuell die Installation eines weiteren Softwarepakets notwendig.

Erstellen eines TwinCAT-SPS-Projekts (Symboldateien-Download)

1. Öffnen Sie die TwinCAT XAE Shell (oder das Visual Studio).
2. Wählen Sie im Menü **Datei** den Befehl **Neu > Projekt**.
3. Fügen Sie dem Projekt ein leeres SPS-Projekt hinzu.
4. Aktivieren Sie in den Eigenschaften des SPS-Projekts den automatischen Download der TMC-Datei, wie im folgenden Screenshot zu sehen ist.

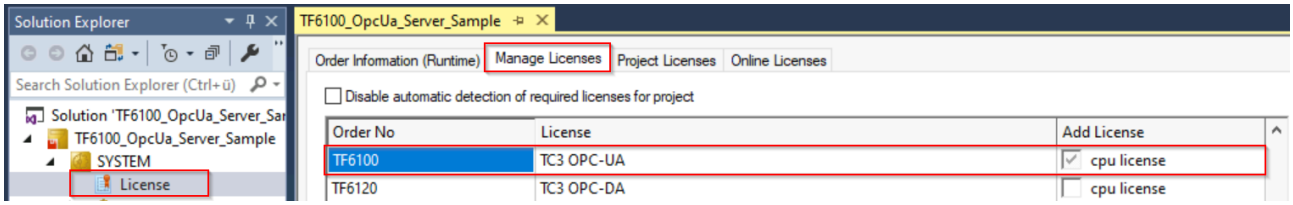
⇨ Ein neues TwinCAT-Projekt wurde erstellt.



Hinzufügen einer Produktlizenz

✓ Prüfen Sie, ob im TwinCAT-XAE-Lizenzdialog eine TF6100 Lizenz vorhanden ist.

1. Falls nicht, können Sie im Rahmen dieses Quick Start Tutorials eine 7-Tage-Testlizenz verwenden.



Freigabe einer Variablen

2. Fügen Sie zum MAIN Programm eine neue Variable vom Datentyp INT hinzu.
3. Setzen Sie an dieser Variable das Pragma zur Freigabe der Variablen über OPC UA.

```
{attribute 'OPC.UA.DA' := '1'}
nMyCounter : INT;
```

4. Inkrementieren Sie diese Variable im Implementierungsteil des MAIN Programms jeden Zyklus um 1.

```
nMyCounter := nMyCounter + 1;
```

5. Aktivieren Sie das TwinCAT-Projekt auf Ihrem System.

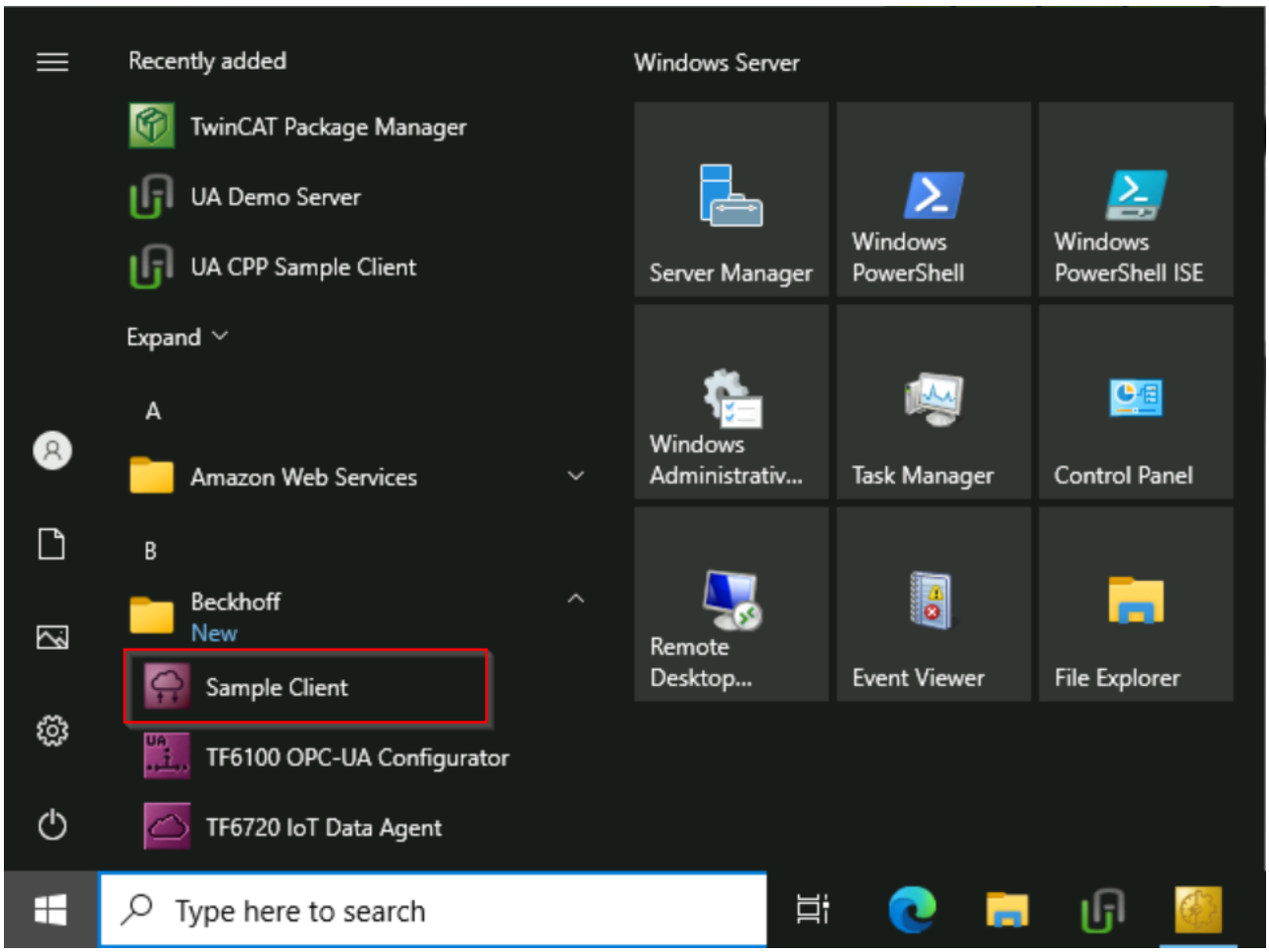
Verbinden eines OPC UA Clients

Ein OPC UA Client verwendet die sogenannte ServerURL, um sich mit einem Server zu verbinden. Sie ServerURL beinhaltet hierbei die IP-Adresse oder den Hostnamen des Geräts auf dem der Server installiert wurde. In diesem Tutorial gehen wir davon aus, dass Client und Server auf demselben System laufen. Somit verbindet sich der Client mit folgender ServerURL:

```
opc.tcp://localhost:4840
```

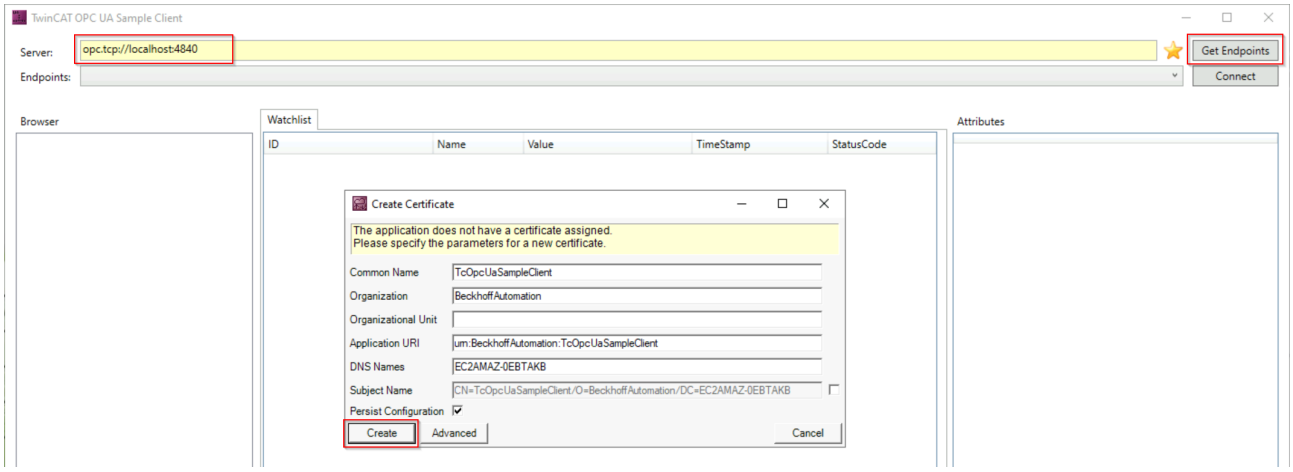
Wir verwenden als Client den TwinCAT OPC UA Sample Client, welcher Bestandteil des TF6100 Produktpakets ist.

Nach der Installation ist dieser über das Windows-Startmenü aufrufbar.

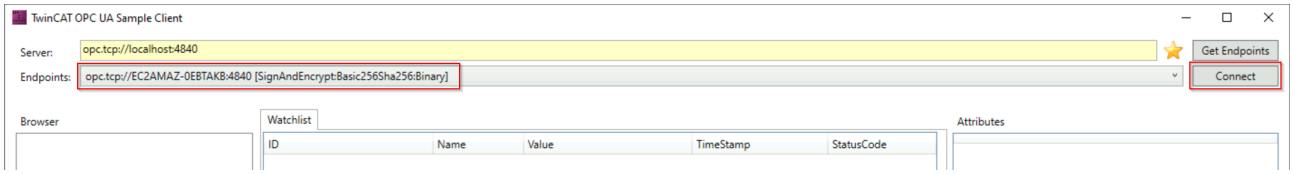


Im TwinCAT OPC UA Sample Client ist die ServerURL zum Localhost bereits standardmäßig eingetragen.

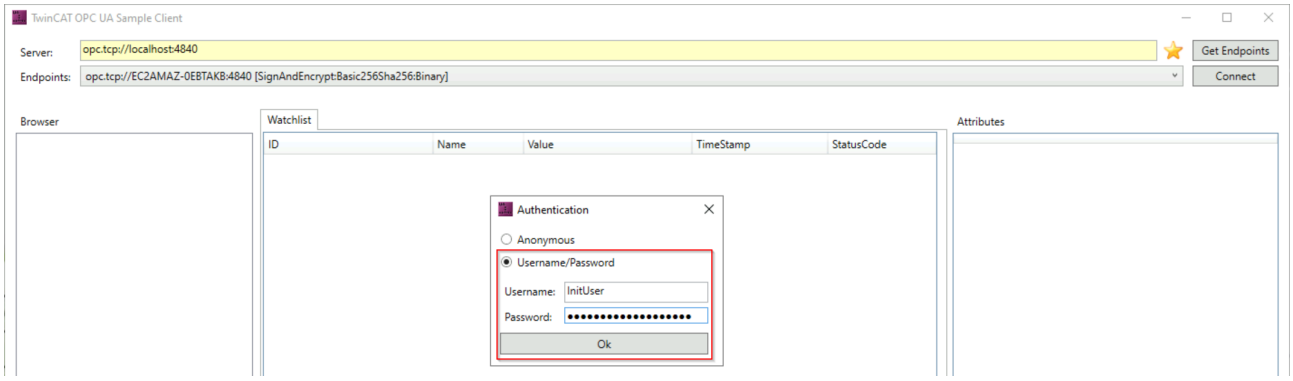
1. Klicken Sie somit auf den Button **Get Endpoints**.
   ⇨ Falls Sie den Sample Client zum ersten Mal verwenden, wird eine Dialogbox angezeigt, welche Einstellungen für die Generierung der Applikationszertifikate entgegen nimmt.



2. Bestätigen Sie die Dialogbox mit dem Button **Create**.
   ⇨ Es werden nun alle Verbindungsendpunkte vom Server ausgelesen und angezeigt.
3. Wählen Sie den Endpunkt „SignAndEncrypt:Basic256Sha256:Binary“ aus und klicken Sie auf den Button **Connect**.



4. Geben Sie die Daten des Benutzerkontos ein, welches Sie im ersten Schritt dieses Dokumentationsartikels für die Initialisierung des Servers konfiguriert haben.

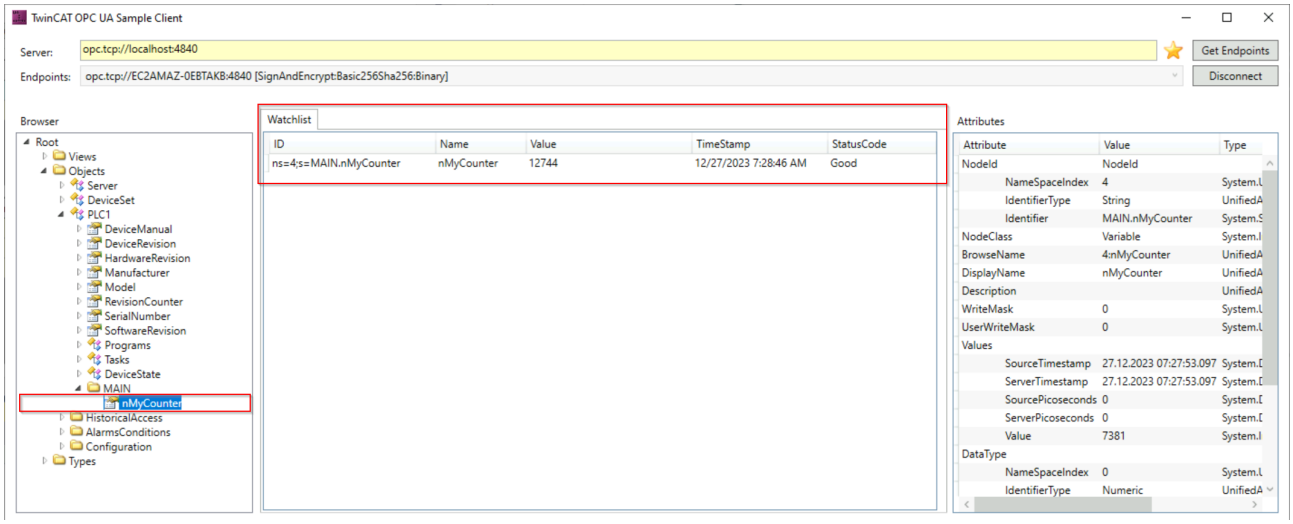


5. Klicken Sie auf **Ok**.

⇨ Sie sind nun mit dem Server verbunden.
Der Adressraum des Servers wird Ihnen in der linken Hälfte der Applikation in einer Baumstruktur angezeigt und Sie können durch die Symbolik des SPS-Programms navigieren. In diesem Beispiel haben wir eine SPS Variable für OPC UA freigegeben. Diese finden Sie unterhalb des folgenden Pfads:

```
Root \ Objects \ PLC1 \ MAIN \ nMyCounter
```

Durch einen Doppelklick auf die Variable können Sie diese zur „Watchlist“ hinzufügen. Das bedeutet, dass eine Subscription auf die Variable angelegt wird und im Fall einer Werteänderung der Variablenwert vom Server an den Client übermittelt wird.



4.2 Quick Start (TwinCAT 2)

Das folgende Kapitel ermöglicht einen Schnelleinstieg in den TwinCAT OPC UA Server, wenn dieser unter TwinCAT 2 betrieben wird. In dieser Anleitung initialisieren Sie den TwinCAT OPC UA Server im Auslieferungszustand, erstellen anschließend ein TwinCAT-SPS-Projekt und geben dann eine SPS-Variable durch Setzen eines Kommentars über OPC UA frei. Die Variable ist anschließend im Adressraum des Servers verfügbar.

Im Folgenden werden die Handlungsschritte ihrer Reihenfolge nach genauer beschrieben:

- Initialisieren des Servers
- Erstellen eines TwinCAT-SPS-Projekts
- Aktivieren des Symboldateien-Downloads
- Freigabe einer Variablen
- Verbinden eines OPC UA Clients

Initialisieren des Servers

Nach der Installation ist eine einmalige Initialisierung [▶ 28] des Servers nach dem sogenannten TOFU (**T**rust-**O**n-**F**irst-**U**se)-Prinzip notwendig. Hierbei konfigurieren Sie ein Benutzerkonto, welches anschließend für den Verbindungsaufbau mit dem Server benötigt wird.

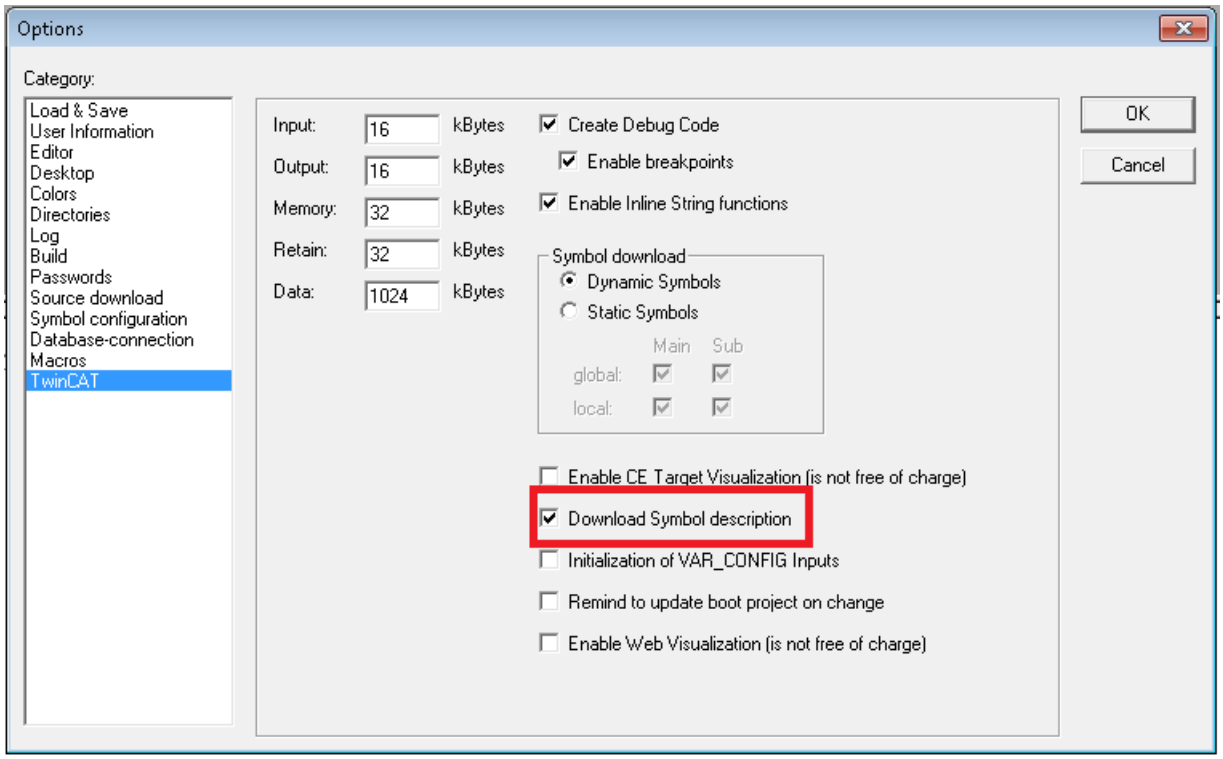
Da es sich hierbei um ein zentrales- und Security-relevantes Thema handelt, wird die Initialisierung [▶ 28] in einem separaten Dokumentationskapitel im Detail erläutert. Die weiteren Schritte in diesem Abschnitt gehen davon aus, dass dieser Vorgang einmal durchgeführt wurde und Sie den Server mit einem Benutzerkonto initialisiert haben.

Tools für die Initialisierung des Servers

Für die Initialisierung des Servers können Sie den TwinCAT OPC UA Configurator verwenden. Hierfür ist eventuell die Installation eines weiteren Softwarepakets notwendig.

Erstellen eines TwinCAT-SPS-Projekts und Aktivieren des Symboldateien Downloads

Legen Sie im TwinCAT PLC Control ein neues Projekt an, oder öffnen Sie ihr existierendes SPS-Projekt. Öffnen Sie die Projekteigenschaften und aktivieren Sie die Option „Download Symbol description“, damit die Symboldatei (*.tpy) des SPS-Programms beim Aktivieren in das Bootverzeichnis des Zielgeräts übertragen wird.



Freigabe einer Variablen

1. Fügen Sie zum MAIN Programm eine neue Variable vom Datentyp INT hinzu.
2. Setzen Sie an dieser Variable den Kommentar zur Freigabe der Variablen über OPC UA. Über die „Description“ lässt sich ein Text festlegen, welcher an der jeweiligen Node im OPC UA Adressraum ebenfalls als Description Attribute angezeigt wird.

```
nMyCounter : INT; (*~ (OPC:1:some description) *)
```

3. Durch einen zusätzlichen Kommentar können Sie die Variable als read-only markieren. Schreibvorgänge über OPC UA werden dann vom Server mit dem Statuscode BadNotWriteable abgelehnt.

```
nMyCounter : INT; (*~ (OPC:1:some description)
                      (OPC_PROP[0005]:1:read-only flag) *)
```

4. Über den folgenden Kommentar können Sie einen Alias für die Variable definieren, d.h. einen anderen Namen für die Variable im OPC UA Namensraum festlegen. Der Wert, den Sie hierbei für x festlegen, entspricht dann dem neuen Variablennamen.

```
nMyCounter : INT; (*~ (OPC:1:some description)
                      (OPC_PROP[0005]:1:read-only flag)
                      (OPC_UA_PROP[5100]:x:alias name) *)
```

5. Inkrementieren Sie diese Variable im Implementierungsteil des MAIN Programms jeden Zyklus um 1.

```
nMyCounter := nMyCounter + 1;
```

6. Aktivieren Sie das TwinCAT-Projekt auf Ihrem System.

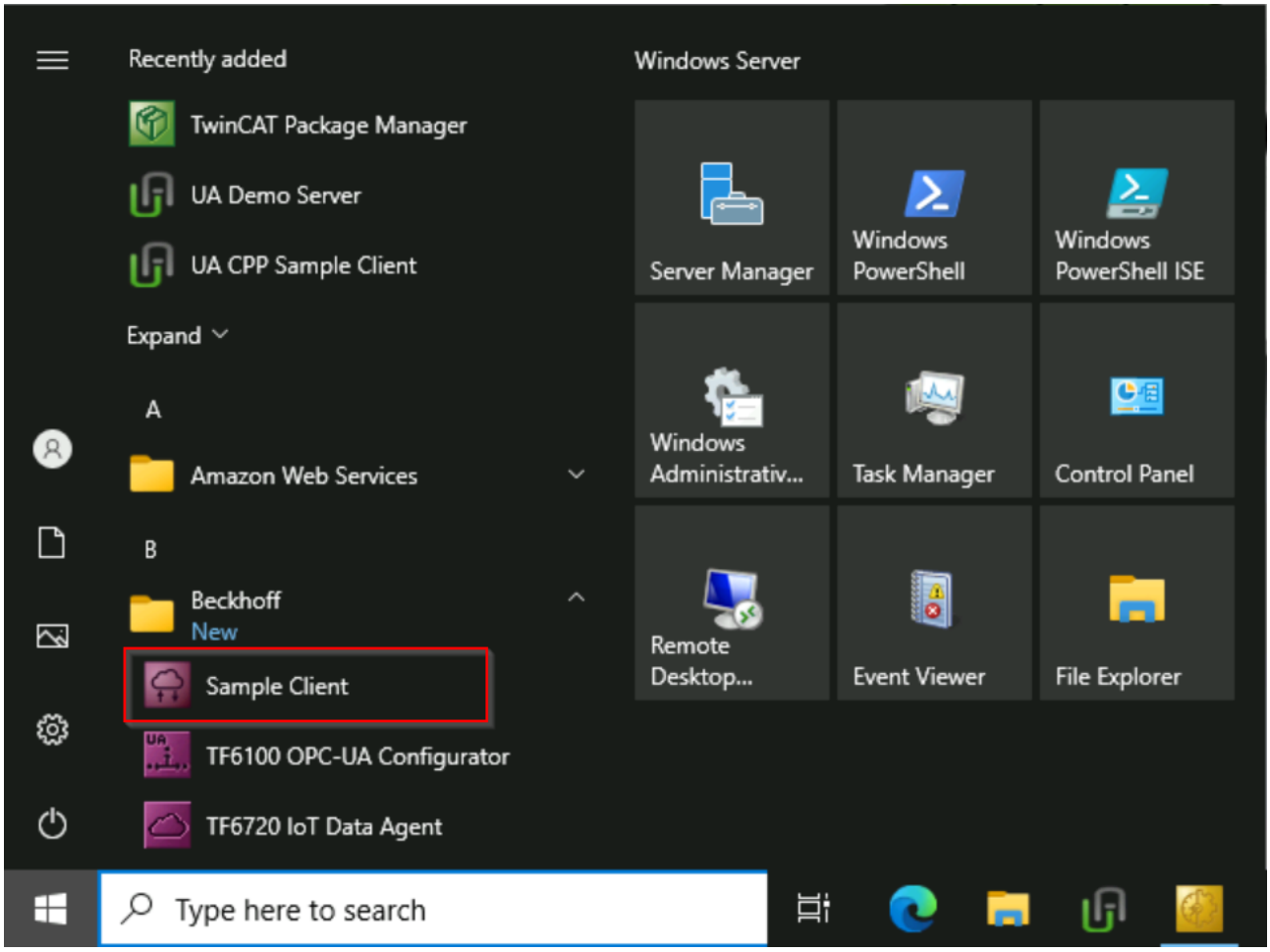
Verbinden eines OPC UA Clients

Ein OPC UA Client verwendet die sogenannte ServerURL, um sich mit einem Server zu verbinden. Sie ServerURL beinhaltet hierbei die IP-Adresse oder den Hostnamen des Geräts auf dem der Server installiert wurde. In diesem Tutorial gehen wir davon aus, dass Client und Server auf demselben System laufen. Somit verbindet sich der Client mit folgender ServerURL:

```
opc.tcp://localhost:4840
```

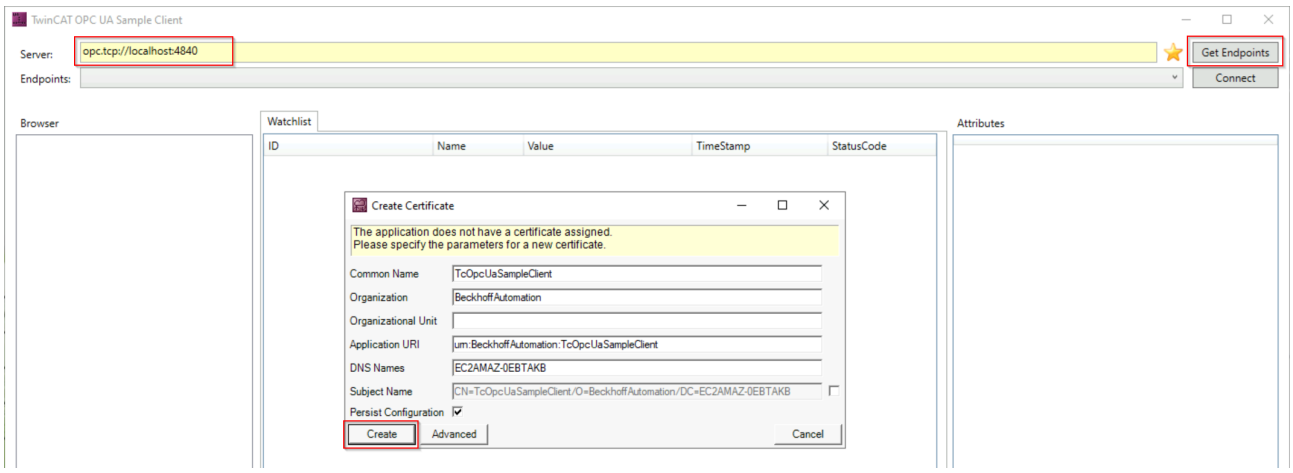
Wir verwenden als Client den TwinCAT OPC UA Sample Client, welcher Bestandteil des TF6100 Produktpakets ist.

Nach der Installation ist dieser über das Windows-Startmenü aufrufbar.

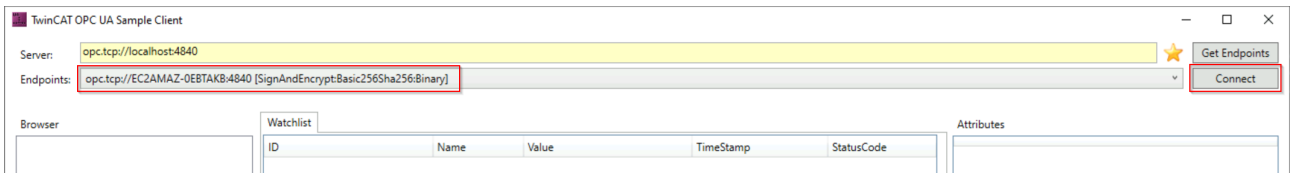


Im TwinCAT OPC UA Sample Client ist die ServerURL zum Localhost bereits standardmäßig eingetragen.

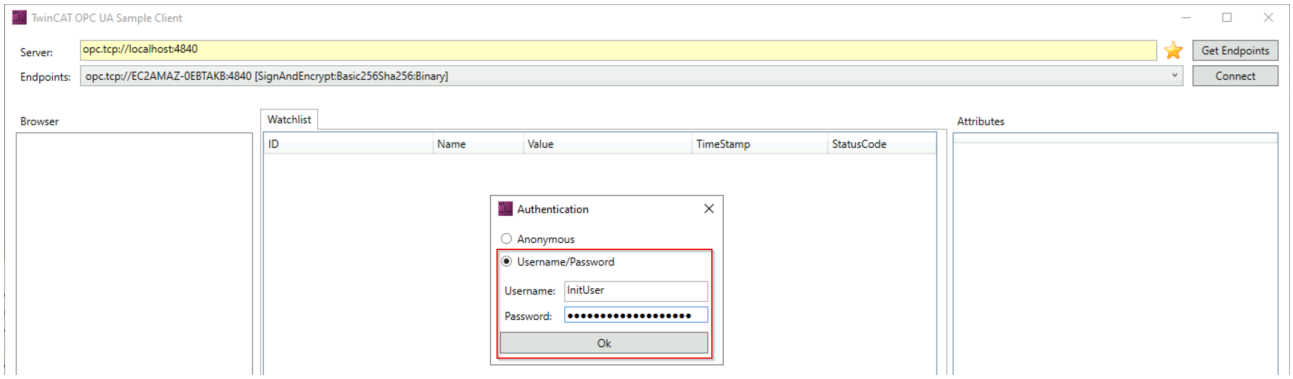
1. Klicken Sie somit auf den Button **Get Endpoints**.
   ⇨ Falls Sie den Sample Client zum ersten Mal verwenden, wird eine Dialogbox angezeigt, welche Einstellungen für die Generierung der Applikationszertifikate entgegen nimmt.



2. Bestätigen Sie die Dialogbox mit dem Button **Create**.
   ⇨ Es werden nun alle Verbindungsendpunkte vom Server ausgelesen und angezeigt.
3. Wählen Sie den Endpunkt „SignAndEncrypt:Basic256Sha256:Binary“ aus und klicken Sie auf den Button **Connect**.



4. Geben Sie die Daten des Benutzerkontos ein, welches Sie im ersten Schritt dieses Dokumentationsartikels für die Initialisierung des Servers konfiguriert haben.

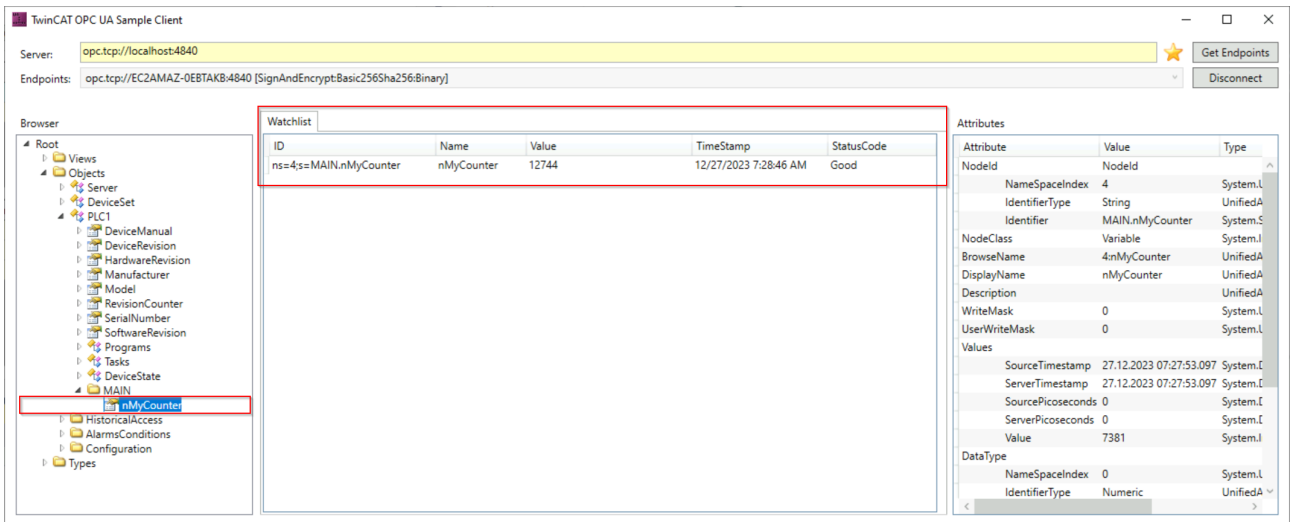


5. Klicken Sie auf **Ok**.

⇨ Sie sind nun mit dem Server verbunden.
Der Adressraum des Servers wird Ihnen in der linken Hälfte der Applikation in einer Baumstruktur angezeigt und Sie können durch die Symbolik des SPS-Programms navigieren. In diesem Beispiel haben wir eine SPS Variable für OPC UA freigegeben. Diese finden Sie unterhalb des folgenden Pfads:

```
Root \ Objects \ PLC1 \ MAIN \ nMyCounter
```

Durch einen Doppelklick auf die Variable können Sie diese zur „Watchlist“ hinzufügen. Das bedeutet, dass eine Subscription auf die Variable angelegt wird und im Fall einer Werteänderung der Variablenwert vom Server an den Client übermittelt wird.

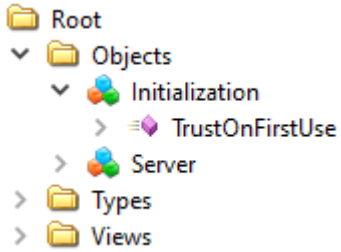


4.3 Initialisierung

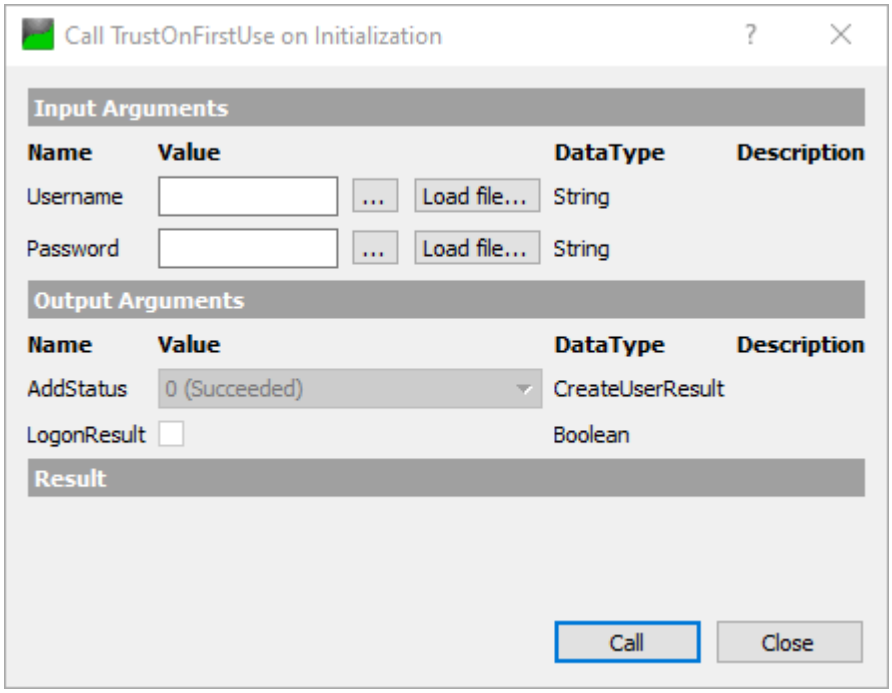
Beginnend mit Setupversion 4.4.0 benötigt der TwinCAT OPC UA Server die Durchführung einer Initialisierungsphase, welche sich an dem TOFU-Prinzip (**T**rust **O**n **F**irst **U**se) orientiert. Das bedeutet, dass der Server aktiv durch den Anwender initialisiert werden muss, damit er für seine verschiedenen Funktionen (Data Access, Historical Access, usw.) verwendet werden kann.

Im Auslieferungszustand ermöglicht es der Server den Clients, eine unauthentifizierte Verbindung („Anonymous“) aufzubauen. Die einmalig durchzuführende TOFU-Initialisierung erfordert nun die Konfiguration eines Betriebssystem-Benutzers, den ein OPC UA Client anschließend verwenden muss, um sich erfolgreich am Server anzumelden.

Hierfür stellt der Server im uninitialisierten Zustand ausschließlich einen speziellen Initialisierungs-Namensraum zur Verfügung. Dieser Namensraum beinhaltet ein Objekt „Initialization“ mit einer Methode „TrustOnFirstUse“.



Die Methode definiert die folgenden Ein-/Ausgabeparameter:



Parameter	Description
[in] Username	Benutzername für den anzulegenden Betriebssystem-Benutzer. Existiert der Benutzer bereits, so versucht der Server einen Test-Login mit dem angegebenen Passwort durchzuführen und übernimmt, falls erfolgreich, den bereits vorhandenen Benutzer in seine Security-Konfiguration.
[in] Password	Passwort für den Betriebssystem-Benutzer. **Das Passwort wird nicht in der Serverkonfiguration gespeichert, sondern ist ausschließlich in der Benutzerdatenbank des Betriebssystems vorhanden. Beachten Sie, dass die Art des Passworts abhängig von etwaigen Sicherheitseinstellungen des Betriebssystems sein kann (Stichwort „komplexe Passwörter“).**
[out] AddStatus	Gibt an, ob das Anlegen des Betriebssystem-Benutzers erfolgreich war oder ob der Benutzer ggf. schon existiert.
[out] LogonResult	Gibt an, ob sich der Server mit der angegebenen Benutzername/ Passwort-Kombination am Betriebssystem anmelden konnte. Hiermit lässt sich gut überprüfen, ob man eventuell das falsche Passwort angegeben hat, falls der Benutzer schon existiert.
[out] OPC UA Statuscode	Der reguläre OPC UA Statuscode beim Methodenaufruf. Ist die Methode auf OPC UA Ebene erfolgreich aufgerufen worden, gibt dieser Statuscode GOOD zurück, anderenfalls BAD.

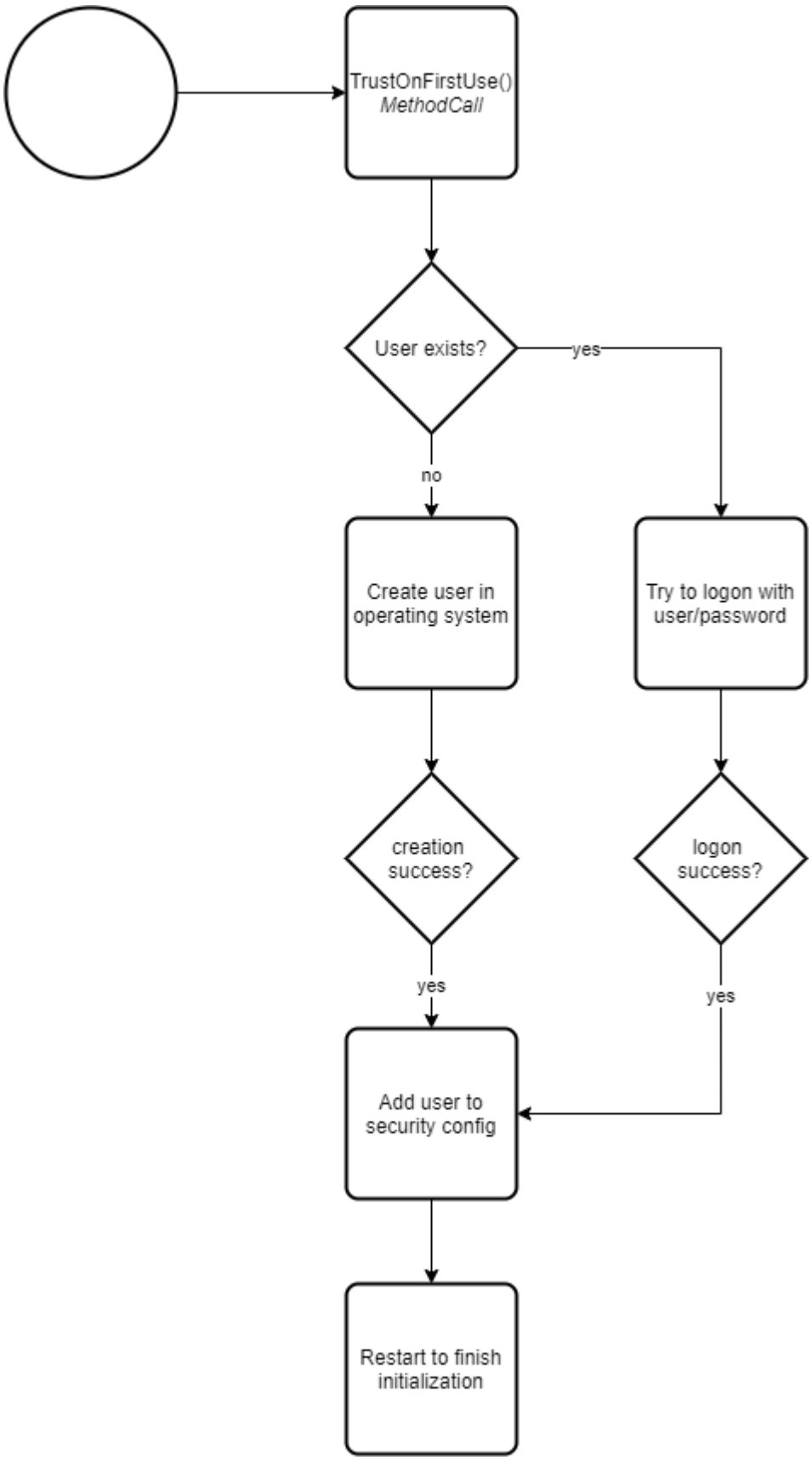
Durch den Aufruf dieser Methode wird der Server initialisiert. Die Methode versucht hierbei einen vom Anwender angegebenen Benutzer im unterlagerten Betriebssystem des Servers anzulegen. Ist dies erfolgreich, so wird der Benutzer automatisch der Security-Konfiguration (TcUaSecurityConfig.xml) des Servers hinzugefügt und als Serveradministrator definiert. Nach einem automatischen Neustart des Servers am Ende des Methodenaufrufs kann sich ein OPC UA Client dann mit diesem Benutzer am Server anmelden.

Existiert ein angegebener Benutzer schon im Betriebssystem, so wird dies über einen Ausgabeparameter (AddStatus) angezeigt. Der Server versucht in diesem Fall, sich mit dem angegebenen Passwort am Betriebssystem anzumelden. Ist dieser Anmeldevorgang erfolgreich, wird der Benutzer in der Security-Konfiguration des Servers eingetragen und die Initialisierung durch einen automatischen Neustart des Servers erfolgreich beendet. Schlägt die Anmeldung am Betriebssystem fehl (z. B., weil das falsche Passwort angegeben wurde), wird dies über einen Ausgabeparameter (LogonResult) angezeigt und die Initialisierung wird nicht fortgesetzt. So wird verhindert, dass man versehentlich versucht, den Server mit einer falschen Benutzername/Passwort-Kombination zu initialisieren und sich dadurch „auszusperren“.

Ablaufen eines Benutzerpasswords

Wenn der OPC UA Server einen Betriebssystembenutzer anlegt, wird für diesen Benutzer **nicht** explizit aktiviert, dass das Passwort nicht abläuft. Hier werden die Einstellungen des Betriebssystems übernommen, wo das maximale Kennwortalter in den Kennwortrichtlinien festgelegt ist. Wenn das maximale Kennwortalter auf 0 steht, laufen Passwörter nicht ab; ansonsten tun sie dies nach der im Betriebssystem angegebenen Anzahl von Tagen.

Das folgende Diagramm veranschaulicht diesen Prozess noch einmal in stark vereinfachter Form:

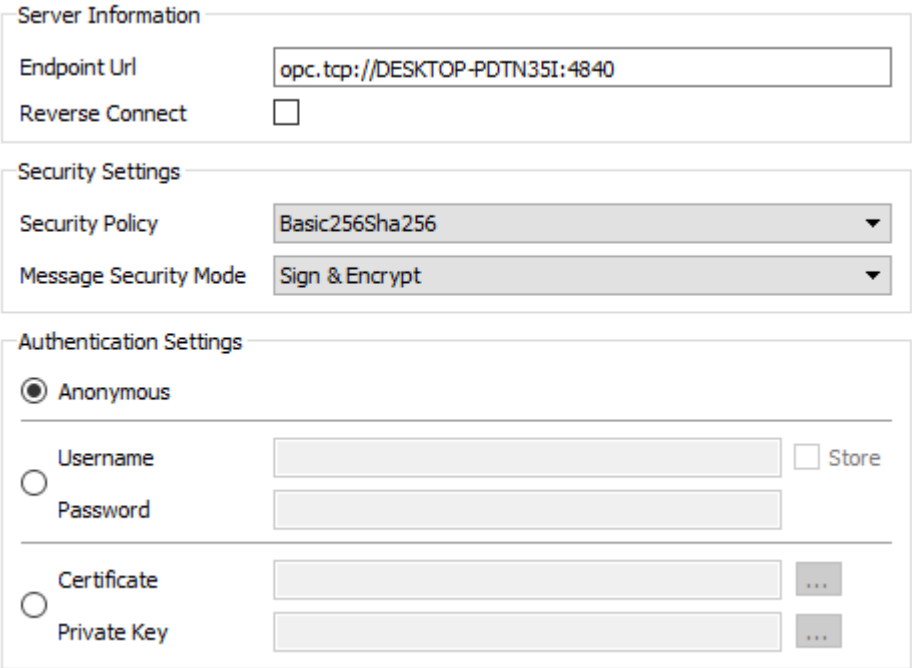


Nach dem Neustart des Servers muss ein OPC UA Client beim Verbindungsaufbau den zur Initialisierung verwendeten Betriebssystembenutzer zur Authentifizierung verwenden.

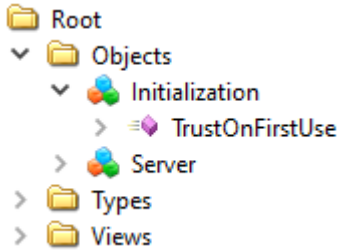
Die folgenden Screenshots zeigen den gesamten Vorgang exemplarisch am Beispiel des OPC UA Clients „UA Expert“. In diesem Beispiel gehen wir davon aus, dass der Benutzer noch nicht im Betriebssystem existiert und somit durch den Server angelegt wird.

Schritt 1: OPC UA Client verbindet sich erstmalig mit dem Server

Der Server wurde installiert und der UA Expert baut zum ersten Mal eine Verbindung mit dem Server auf. Für diese Verbindung kann noch der Anonymous-Zugriff verwendet werden.

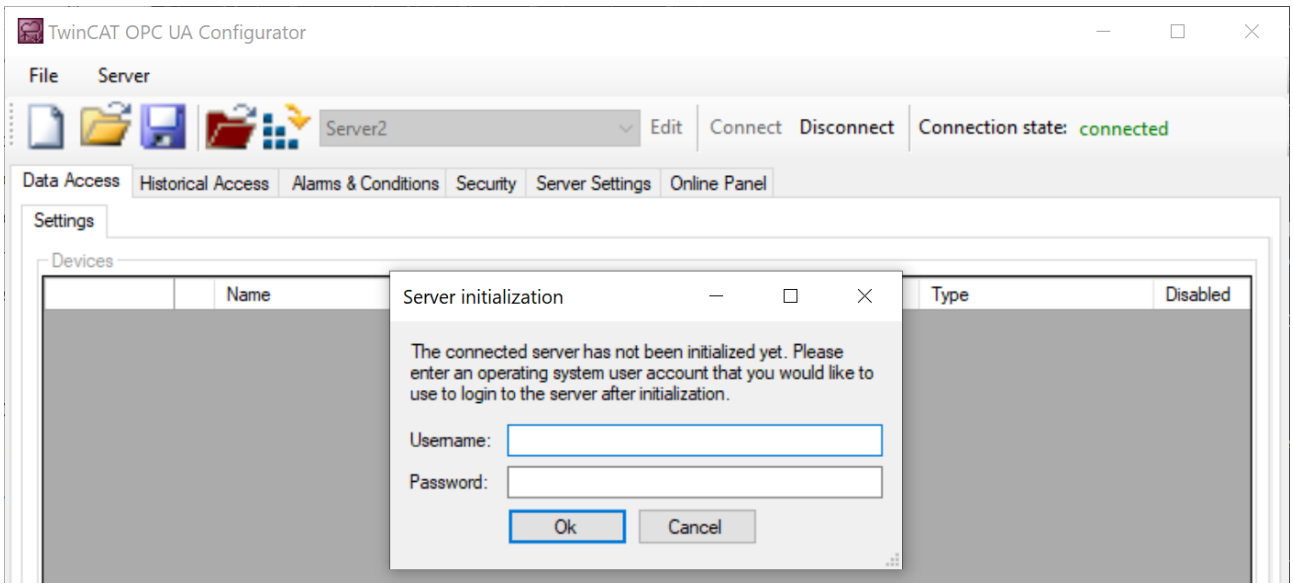


Nach dem Herstellen der Verbindung findet man im Adressraum des Servers das Initialization-Objekt mitsamt der TrustOnFirstUse Methode.



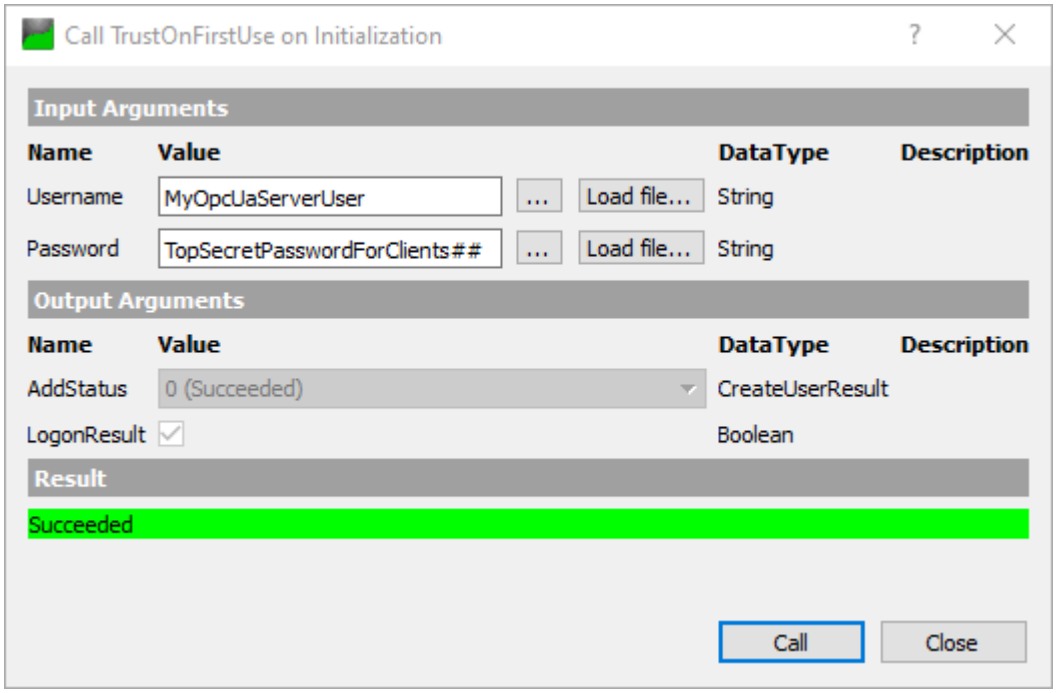
Schritt 2: OPC UA Client startet TrustOnFirstUse

Der Aufruf der TrustOnFirstUse Methode kann über einen beliebigen OPC UA Client erfolgen, z. B. den UA Expert. Aber auch die Beckhoff-eigenen Konfigurationstools erlauben die Verwendung dieser Initialisierungsschnittstelle. Der TwinCAT OPC UA Configurator (Standalone oder Visual Studio integriert) erkennt beim Herstellen einer Verbindung automatisch einen uninitialisierten Server und ermöglicht die Initialisierung über eine entsprechende Konfigurationsoberfläche:



Die folgenden Schritte zeigen denselben Prozess, wie er z. B. in der Software UA Expert manuell durchgeführt werden kann:

Im UA Expert wird die TrustOnFirstUse Methode aufgerufen, um einen Benutzer zu erzeugen und den Server für diesen Benutzer zu konfigurieren. Als Benutzername wurde in diesem Beispiel „MyOpcUaServerUser“ verwendet. Das Passwort muss den Komplexitätsanforderungen des Betriebssystems entsprechen, anderenfalls schlägt die Initialisierung fehl. Der folgende Screenshot zeigt den erfolgreichen Aufruf der Methode.



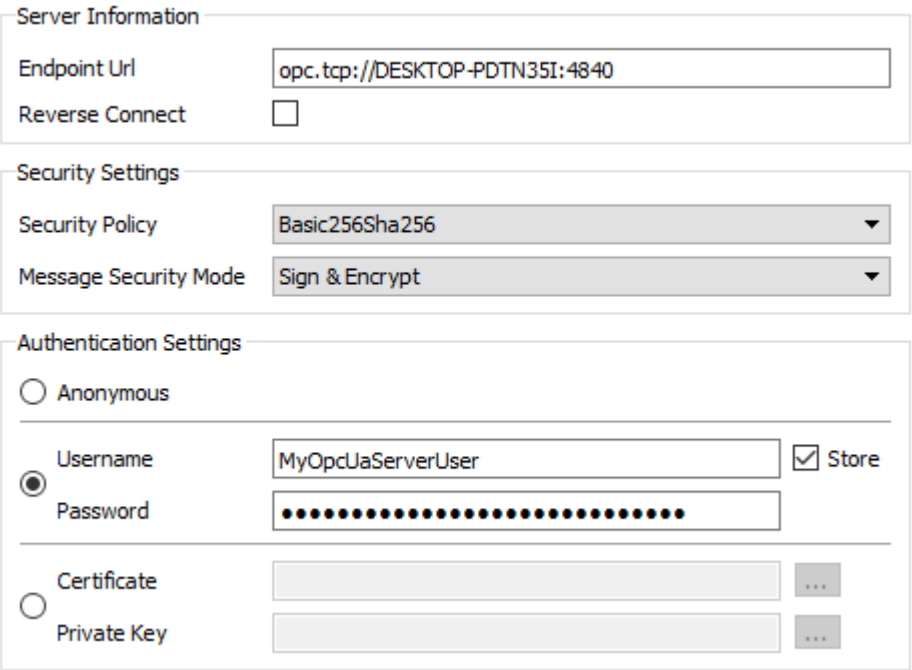
Der Parameter AddStatus zeigt, dass das Anlegen des Benutzers in der Benutzerdatenbank des Betriebssystems erfolgreich war. Der Parameter LogonResult zeigt, dass eine initiale Test-Authentifizierung des Servers mit den angegebenen Benutzerinformationen erfolgreich war.

Der Server startet nach diesem erfolgreichen Methodenaufruf automatisch neu.

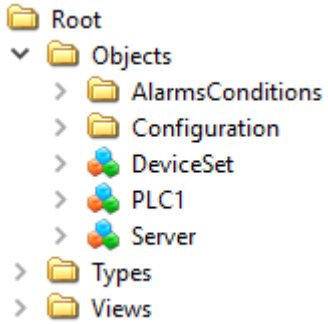
Schritt 3: OPC UA Client meldet sich am initialisierten Server an

Benutzername/Passwort deaktiviert Anonymous-Zugriff

Bitte beachten Sie, dass der UA Expert sich nach dem Methodenaufruf nicht automatisch neu mit dem Server verbinden kann, da der Anonymous-Zugriff deaktiviert wurde und von nun an die Anmeldung über den angegebenen Benutzernamen erfolgen muss.



Nach dem Verbindungsaufbau sind im Adressraum des Servers die regulären Namensräume und Objekte wieder zu finden und die Projektierung der Applikation kann beginnen.



Berechtigungen des TOFU-Benutzers

Der durch den TOFU-Mechanismus konfigurierte Benutzer hat Vollzugriff auf den Server, was unter Umständen nicht gewünscht ist. Beckhoff empfiehlt daher in einem nächsten Schritt die Erstellung eines expliziten Benutzers für den reinen Datenzugriff, siehe Empfohlene Schritte.

4.4 Empfohlene Schritte

Nach der Erstinbetriebnahme empfiehlt Beckhoff die Beachtung der folgenden Punkte, um den Server weiter zu konfigurieren und eine stabile und sichere Betriebsumgebung zu gewährleisten.

Data Access

Data Access beschreibt eine Funktion von OPC UA zur Darstellung von Symbolen und den entsprechenden Zugriff darauf im Adressraum des Servers. Im TwinCAT OPC UA Server ist die Konfiguration von Data-Access-Geräten ein elementarer Bestandteil und Grundlage für weitere Funktionalitäten. Wir empfehlen Ihnen daher im nächsten Schritt unser Kapitel zum Themengebiet Data Access [▶ 49], in welchem auch beschrieben wird, wie Sie eine Verbindung mit der Laufzeit [▶ 50] herstellen können.

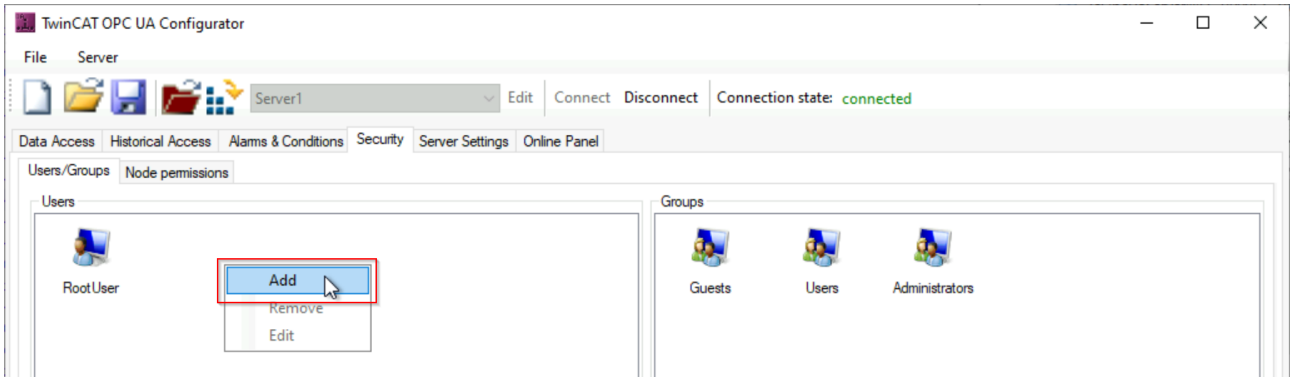
Nur sichere IdentityToken verwenden

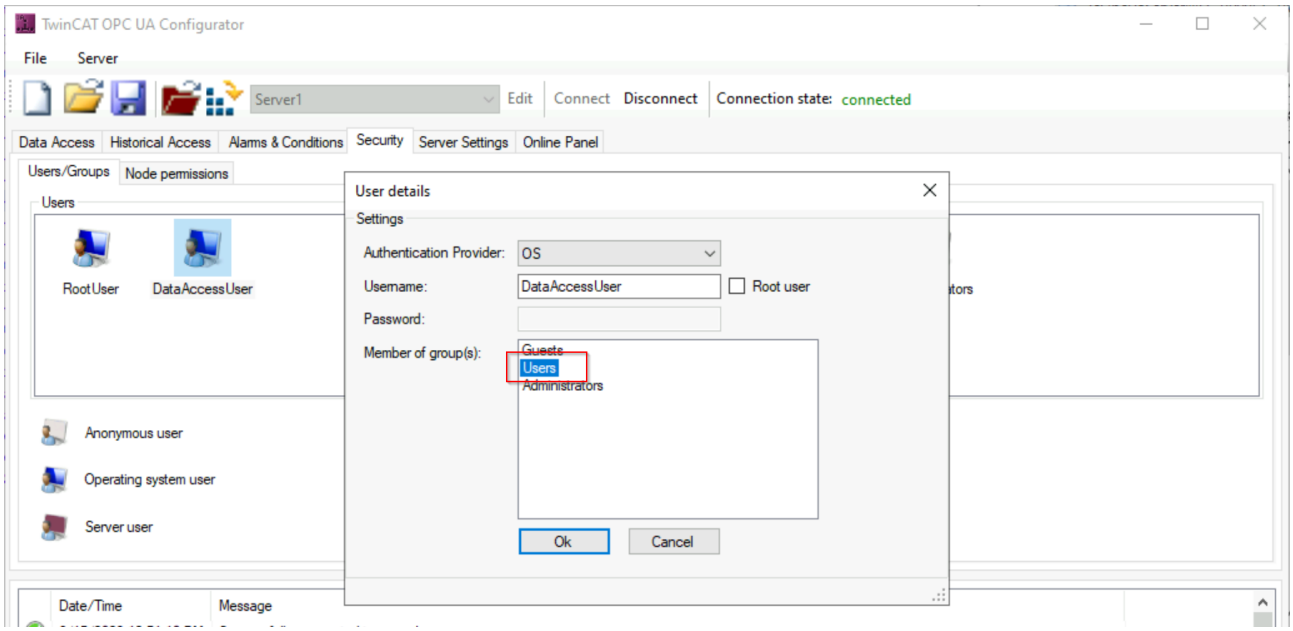
Durch die einmalige Initialisierung des Servers wird das IdentityToken „Anonymous“ deaktiviert. **Aus Sicherheitsgründen sollten Sie dieses deaktiviert lassen.** Der Zugriff auf den Server sollte ausschließlich durch authentifizierte Client-Applikationen erfolgen, wie z. B. der standardmäßig durch die Initialisierung konfigurierten Benutzername-/Passwort-Authentifizierung.

Erstellung eines Benutzers für den reinen Datenzugriff

Durch die bereits genannte Initialisierung des Servers wird ein Benutzer für den Zugriff auf den Server konfiguriert und anschließend der Anonymous-Zugriff auf den Server deaktiviert. Der konfigurierte Benutzer hat hierbei Vollzugriff auf alle Objekte im Namensraum des Servers. In den meisten Anwendungsszenarien ist dies nicht gewünscht und der Administrator-Benutzer soll vom Anwendungsbenutzer separiert werden.

Beckhoff empfiehlt daher die Konfiguration eines zusätzlichen, dedizierten Benutzers, welcher die notwendigen Berechtigungen für den Zugriff auf Variablen eines Data-Access-Geräts bekommt, dabei jedoch nicht auf den Konfigurations-Namensraum zugreifen darf. Diese Einstellung kann über den Konfigurator durchgeführt werden, indem Sie einen neuen Benutzer hinzufügen, welcher der Benutzergruppe „Users“ zugewiesen wird.





Der neu konfigurierte Benutzer hat dann alle notwendigen Berechtigungen auf TwinCAT-Variablen zuzugreifen und das Typsystem auszulesen, jedoch nicht die Konfiguration des Servers zu beeinflussen. Bitte beachten Sie, dass Sie bei Verwendung des Authentifzierungsproviders „OS“ den Benutzer ebenfalls im Betriebssystem anlegen müssen, d. h., dass er dort vorhanden sein muss.

HINWEIS

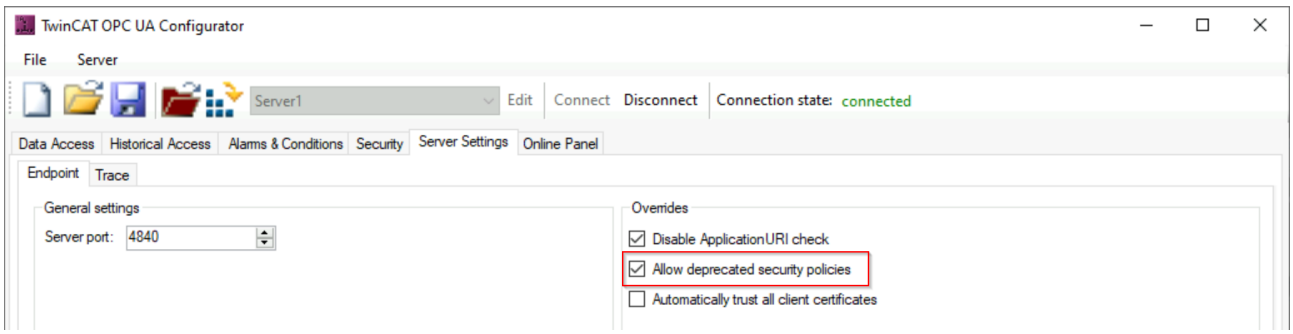
Unsichere Endpunkte deaktiviert lassen

✓ Als unsicher eingestufte Endpunkte werden standardmäßig nicht vom TwinCAT OPC UA Server angeboten. Über einen Konfigurationsschalter lassen sich diese im Server verfügbar machen – Beckhoff rät ausdrücklich davon ab!

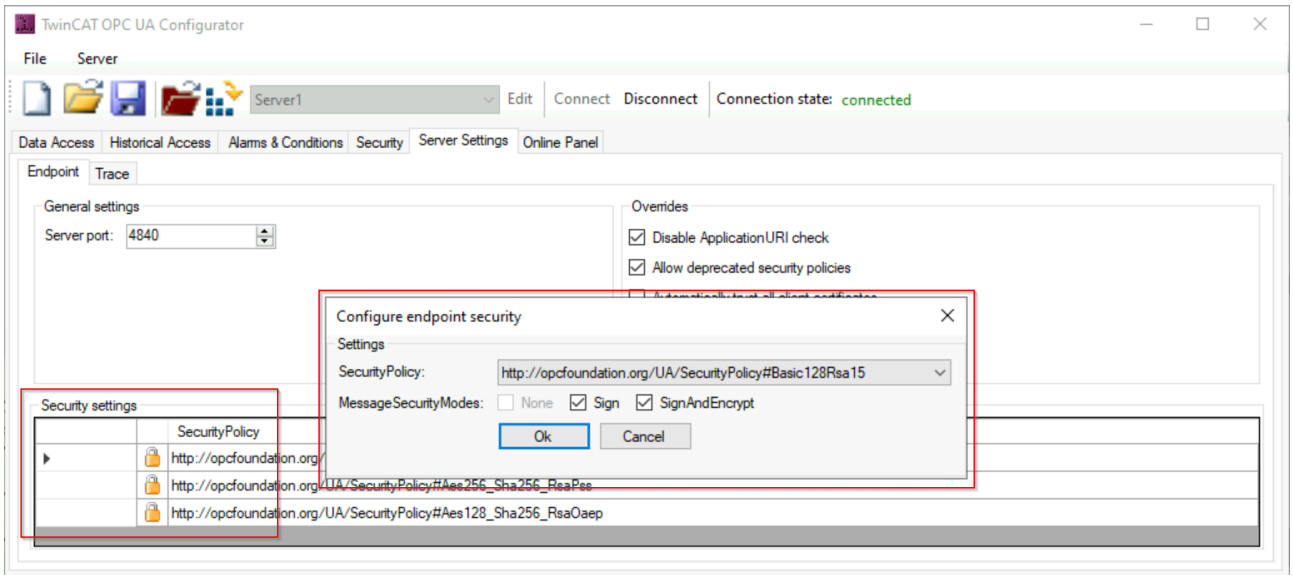
a) Verwenden Sie nur die aktuell als sicher geltenden Endpunkte.

b) Beachten und befolgen Sie die weiteren sicherheitsrelevanten Empfehlungen im folgenden Abschnitt.

Der folgende Screenshot zeigt Ihnen, wie Sie ältere Server Endpunkte im TwinCAT OPC UA Configurator aktivieren können.



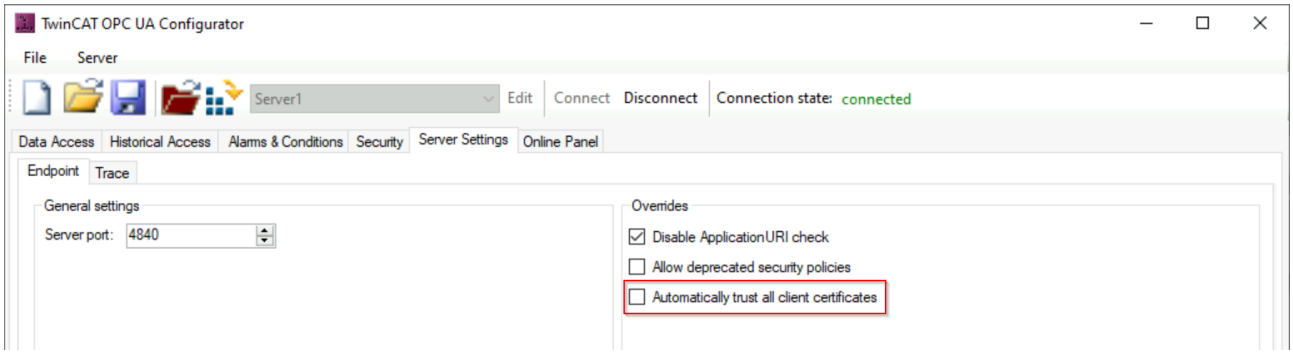
Anschließend können Sie die unsicheren Endpunkte wieder zur Konfiguration des Servers hinzufügen, zum Beispiel über das Kontextmenü im Konfigurator im Bereich „Security Settings“:



Desweiteren ist auch der None-/None-Endpunkt im Auslieferungszustand des Servers deaktiviert. Aus Sicherheitsgründen empfiehlt Beckhoff, diesen Endpunkt ebenfalls deaktiviert zu lassen und den Zugriff auf den Server nur über einen sicheren Endpunkt zu erlauben. Bei Bedarf kann der None-/None-Endpunkt über den oben genannten Weg wieder zur Konfiguration des Servers hinzugefügt werden.

'AutomaticallyTrustAllClientCertificates' deaktivieren

Im Auslieferungszustand wird der Server für die einfache Inbetriebnahme so konfiguriert, dass dieser allen Client-Zertifikaten automatisch vertraut, ohne auf Serverseite einen manuellen Zertifikatsaustausch durchführen zu müssen. Aus Sicherheitsgründen empfiehlt Beckhoff die Deaktivierung dieser Einstellung. Diese Einstellung kann über den TwinCAT OPC UA Configurator vorgenommen werden, wie der folgende Screenshot zeigt:



Nach der Deaktivierung dieser Einstellung muss eine Vertrauensstellung zwischen Client und Server hergestellt werden, indem beide Applikationen gegenseitig ihren Zertifikaten vertrauen.

4.5 Unterstützte Features

OPC UA verfügt über eine große Anzahl von Features. Es kann sein, dass dieses Produkt nicht alle Features von Anfang an unterstützt. Weitere Features werden im Laufe der Zeit durch Produktaktualisierungen hinzugefügt. Die folgende Tabelle zeigt eine Liste aller derzeit unterstützten Funktionalitäten der aktuellen Produktversion.

Kompatibilität zur OPC UA Pub/Sub Spezifikation

Die aktuelle Version dieses Produkts ist kompatibel zur OPC UA Pub/Sub Spezifikation Version 1.05.03.

Zertifizierungsprofil

Die aktuelle Version dieses Produkts ist zertifiziert für das Profil „Standard UA Server“. Die in dieser Zertifizierung überprüften Conformance Units sind seitens der OPC Foundation dokumentiert.

Spezifikation

Die OPC UA Spezifikation ist in verschiedene Teile („Parts“) unterteilt. Die Parts 1 bis 5 definieren hierbei die Kernfunktionalitäten von OPC UA. Die folgende Tabelle gibt einen Überblick über die verschiedenen Parts der Spezifikation und markiert, ob dieses Produkt eine entsprechende Unterstützung anbietet. Weiter unten finden Sie dann, falls anwendbar, zusätzliche Detailinformationen zu den unterstützten Funktionen aus dem jeweiligen Part. Bitte beachten Sie, dass naturgemäß nicht alle Parts unterstützt werden können, da manche Teile der Spezifikation sich außerhalb der Produktdefinition bewegen.

Part	Unterstützt
OPC10000-1 Overview and Concepts	Ja
OPC10000-2 Security Model	Ja
OPC10000-3 Address Space Model	Ja
OPC10000-4 Services	Ja
OPC10000-5 Information Model	Ja
OPC10000-6 Mappings	Ja
OPC10000-7 Profiles	Ja
OPC10000-8 Data Access	Ja
OPC10000-9 Alarms & Conditions	Ja
OPC10000-10 Programs	Nein
OPC10000-11 Historical Access	Ja
OPC10000-12 Discovery and Global Services	Ja
OPC10000-13 Aggregates	Nein
OPC10000-14 Pub/Sub	Nein *1
OPC10000-15 Safety	Nein
OPC10000-16 State Machines	Nein
OPC10000-17 Alias Names	Nein
OPC10000-18 Role-Based Security	Nein *2
OPC10000-19 Dictionary References	Nein
OPC10000-20 File Transfer	Ja
OPC10000-21 Device Onboarding	Nein
OPC10000-22 Base Network Model	---
OPC10000-23 Common ReferenceTypes	---
OPC10000-24 Scheduler	Nein

*1: OPC UA Pub/Sub wird durch das Produkt TF6105 TC3 OPC UA Pub/Sub abgebildet

*2: Dieses Produkt beinhaltet eine eigene Benutzerrechte-Implementierung, welche schon vor der Veröffentlichung Part 18 existierte und ein Benutzer-/Gruppen-basiertes Berechtigungssystem [▶ 131] implementiert.

OPC10000-4 Services

Die folgende Tabelle gibt einen Überblick über die von diesem Produkt unterstützten Services (Service Sets, Service Behaviors, etc.) wie in Part 4 der OPC UA Spezifikation beschrieben.

Feature	Unterstützt
Attributes Read/Write	Ja
Attributes HistoryRead	Ja (Value Attribut)
Attributes HistoryUpdate	Ja (Value Attribut)
Subscriptions	Ja
Durable Subscriptions	Nein
Monitored Items	Ja
Method Calls	Ja
Auditing	Nein
Redundancy	Nein

OPC10000-8 Data Access

Die folgende Tabelle gibt einen Überblick über die von diesem Produkt unterstützten Data Access Funktionen wie in Part 8 der OPC UA Spezifikation beschrieben.

Feature	Unterstützt
DataItemType	Ja
AnalogItemType	Ja
DiscreteItemType	Ja *1
ArrayItemType	Ja
EUInformation	Ja *1
ComplexNumberType	Ja *1
DoubleComplexNumberType	Ja *1
AxisInformation	Ja *1
AxisScaleEnumeration	Ja *1
XVType	Ja *1
ObjectTypes	Ja (Function Blocks)
StructuredTypes	Ja
ServerTimestamp	Ja
SourceTimestamp	Ja
Quality	Ja

*1: Wird unterstützt über den Nodeset-Import und TE6100 OPC UA Nodeset Editor.

OPC10000-9 Alarms & Conditions

Die folgende Tabelle gibt einen Überblick über die von diesem Produkt unterstützten Alarms & Conditions Funktionen wie in Part 9 der OPC UA Spezifikation beschrieben.

Feature	Unterstützt
LimitAlarmType	Ja
OffNormalAlarmType	Ja
Multi-Language Alarm-/Eventtexts	Ja
TwinCAT 3 Eventlogger	Ja *1
Custom EventTypes	Ja *2
Custom AlarmTypes	Ja *2

*1: Unterstützung des TwinCAT 3 Eventlogger über proprietäre Alarm-/EventTypes

*2: Wird unterstützt über den Nodeset-Import und TE6100 OPC UA Nodeset Editor.

OPC10000-11 Historical Access

Die folgende Tabelle gibt einen Überblick über die von diesem Produkt unterstützten Historical Access Funktionen wie in Part 11 der OPC UA Spezifikation beschrieben.

Feature	Unterstützt
HistoricalDataNodes [▶ 93]	Ja
HistoricalEventNodes	Nein
Historical Audit Events	Nein
HistoryUpdate [▶ 96]	Ja
External History Sources	Nein

OPC10000-20 File Transfer

Dieses Produkt unterstützt den OPC UA seitigen Austausch von Konfigurationsdateien, welche zur Inbetriebnahme des Produkts verwendet werden. Eine generische File Transfer [▶ 133] Funktionalität ist im OPC UA Server der Software Beckhoff Device Manager implementiert.

Security

Die folgende Tabelle gibt einen Überblick über die von diesem Produkt unterstützten Security Funktionen. Die hier dargestellten Features sind teils auf mehrere Parts der Spezifikation aufgeteilt.

Feature	Unterstützt
SecureChannel	Ja
SecureEndpoints [▶ 125]	Ja
Application Instance Certificate	Ja
Certificate Trust Relationship	Ja
IdentityToken Anonymous	Ja
IdentityToken Username/Password	Ja
IdentityToken Benutzerzertifikate	Ja
Benutzerrechte auf Namespace-Level	Ja
Benutzerrechte auf Node-Level	Ja

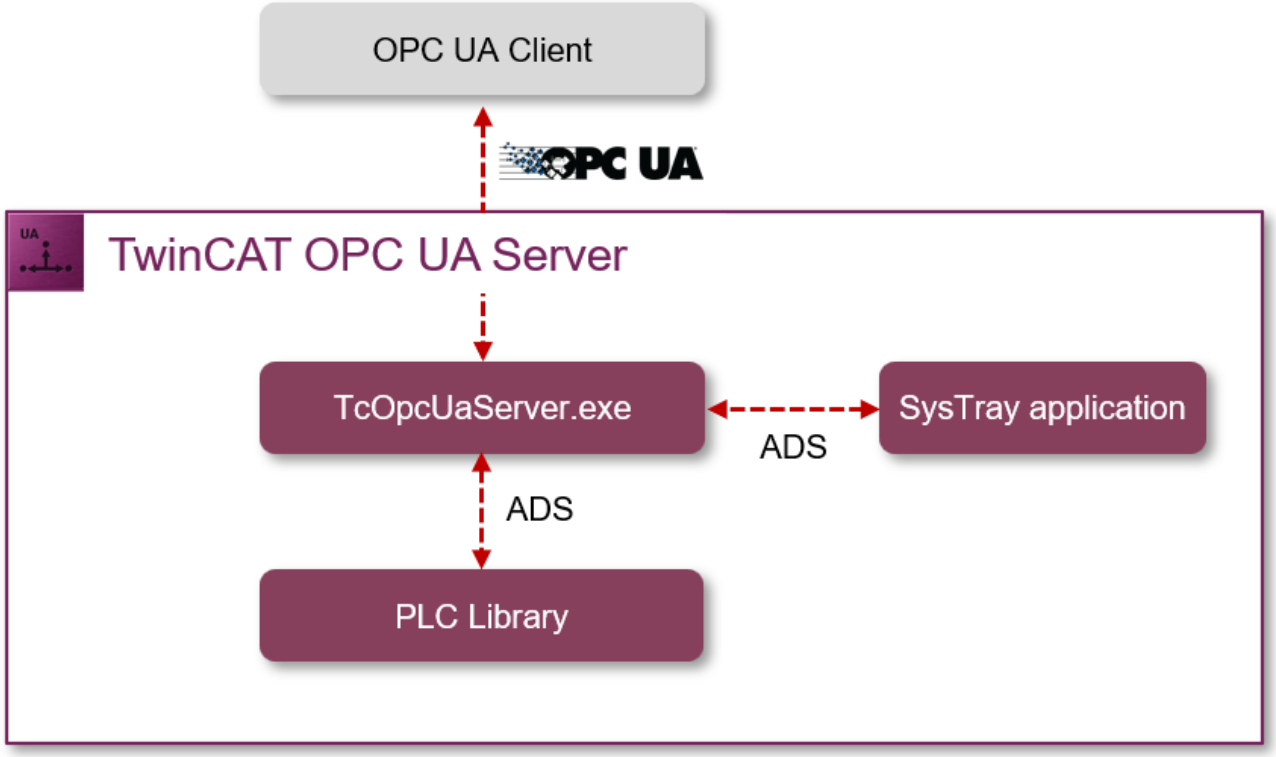
4.6 Softwarearchitektur

Die interne Softwarearchitektur dieses Produkts müssen Sie für den Betrieb der Software nicht kennen – sie kann jedoch im Einzelfall von Interesse sein. Deshalb stellen wir Sie Ihnen im Folgenden kurz vor.

Der TwinCAT OPC UA Server besteht im Wesentlichen aus folgenden Komponenten::

- Dem Prozess im Betriebssystem
- Der Applikation im Windows System Tray [▶ 143]
- Der SPS-Bibliothek Tc2_OpcUa [▶ 144]

Das Zusammenspiel der einzelnen Komponenten wird durch das folgende Schaubild näher beschrieben:



Prozess im Betriebssystem

Der Prozess im Betriebssystem (TcOpcUaServer.exe) kümmert sich um die OPC UA Protokollfunktionen (also die Kommunikation mit den OPC UA Clients), das Bereitstellen des OPC UA Adressraums und um die Kommunikation mit den unterlagerten Geräten [▶ 49]. Desweiteren stellt der Prozess eine ADS-Server-Schnittstelle zur Verfügung, sodass die SPS-Bibliothek Tc2_OpcUa sowie die Windows-System-Tray-Applikation mit der Serverapplikation interagieren können.

Windows System Tray Applikation

Diese Applikation ermöglicht das Antriggern eines Neustarts des TwinCAT OPC UA Servers. Die entsprechende Funktion ist über das Icon im Windows-System-Tray aufrufbar.

SPS-Bibliothek

Die SPS-Bibliothek Tc2_OpcUa ermöglicht eine Interaktion mit dem TwinCAT OPC UA Server sowohl zum Abrufen von Statusinformationen als auch zum Antriggern eines Neustarts.

4.7 Konfigurator

Für die einfache Konfiguration des TwinCAT OPC UA Servers stehen zwei grafische Benutzeroberflächen im Rahmen des TwinCAT OPC UA Configurators zur Verfügung: eine in das Visual Studio (bzw. die XAE Shell) integrierte Oberfläche sowie ein Standalone Tool.

Dokumentation zum TwinCAT OPC UA Configurator

Obwohl in verschiedenen Teilbereichen dieser Dokumentation auf den TwinCAT OPC UA Configurator eingegangen wird, besitzt dieser auch eine eigene Produktdokumentation, welche Sie im Beckhoff Informationssystem finden.

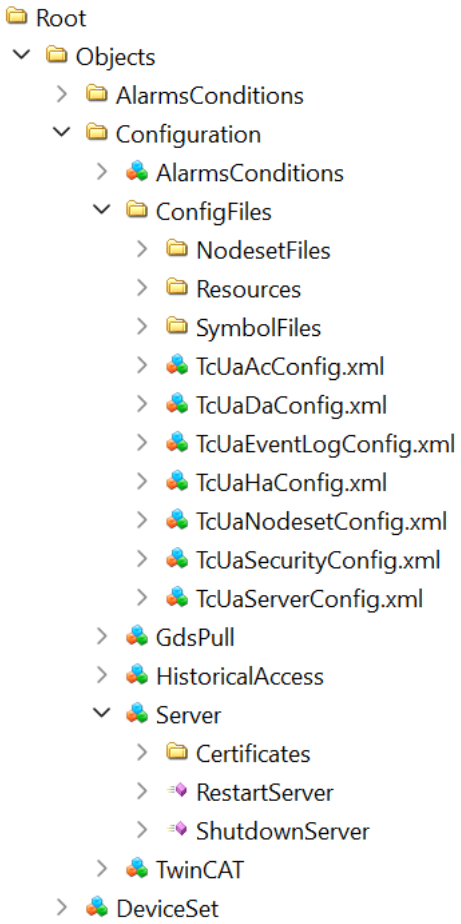
Konfiguration über OPC UA

Der TwinCAT OPC UA Server beinhaltet einen sogenannten Konfigurationsnamensraum, welcher eine lokale und remote Konfiguration des Servers über OPC UA ermöglicht. Der TwinCAT OPC UA Configurator macht sich diese Schnittstelle zunutze, um auf die einzelnen Konfigurationsdateien des Servers zugreifen zu können.

Die folgenden Funktionalitäten sind im Konfigurationsnamensraum abgebildet:

- Verwaltung der Serverkonfigurationsdateien
- Verwaltung der Zertifikate
- Neustart des Servers

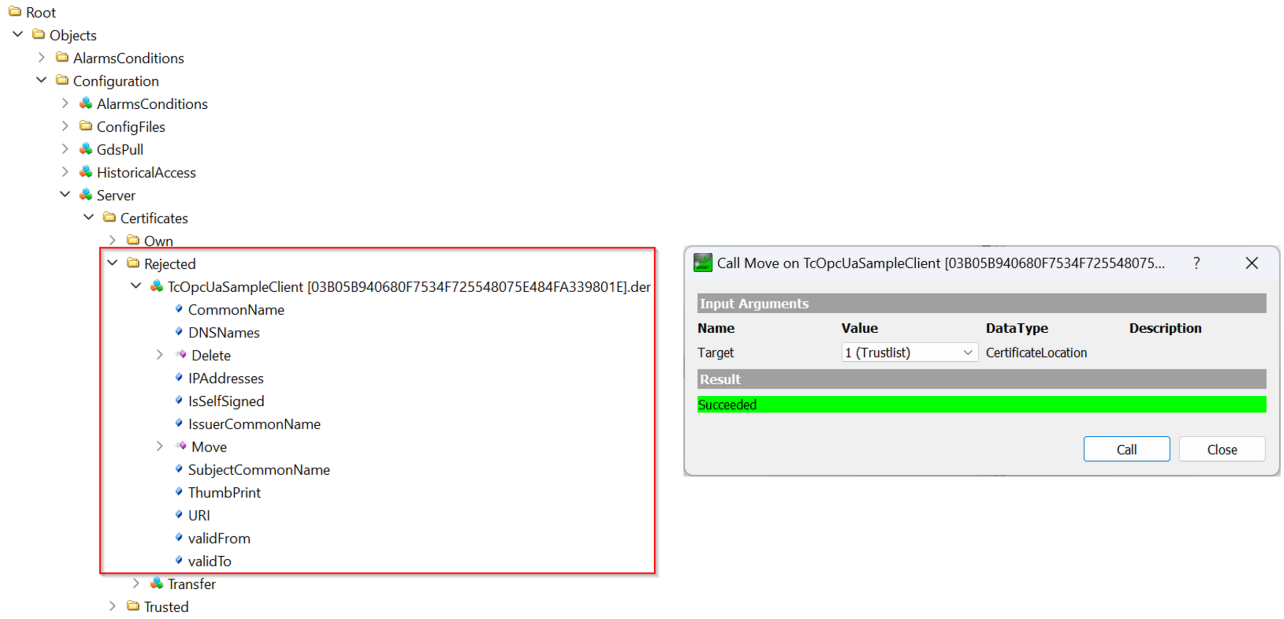
Nach der Initialisierung [▶ 28] des Servers hat der dort konfigurierte Benutzer Zugriff auf den Konfigurationsnamensraum und kann verwendet werden, um den Server weiter zu konfigurieren. Der folgende Screenshot zeigt den Konfigurationsnamensraum aus Sicht eines OPC UA Clients (in diesem Beispiel der UA Expert von Unified Automation):



Alle Konfigurationsdateien des Servers sind als Objekte vom Type FileType verfügbar. Dieser Objekttyp und der Zugriff darauf ist durch OPC UA standardisiert. Zertifikatsdateien werden im Konfigurationsnamensraum als CertificateType angelegt.

Verwaltung von Zertifikaten

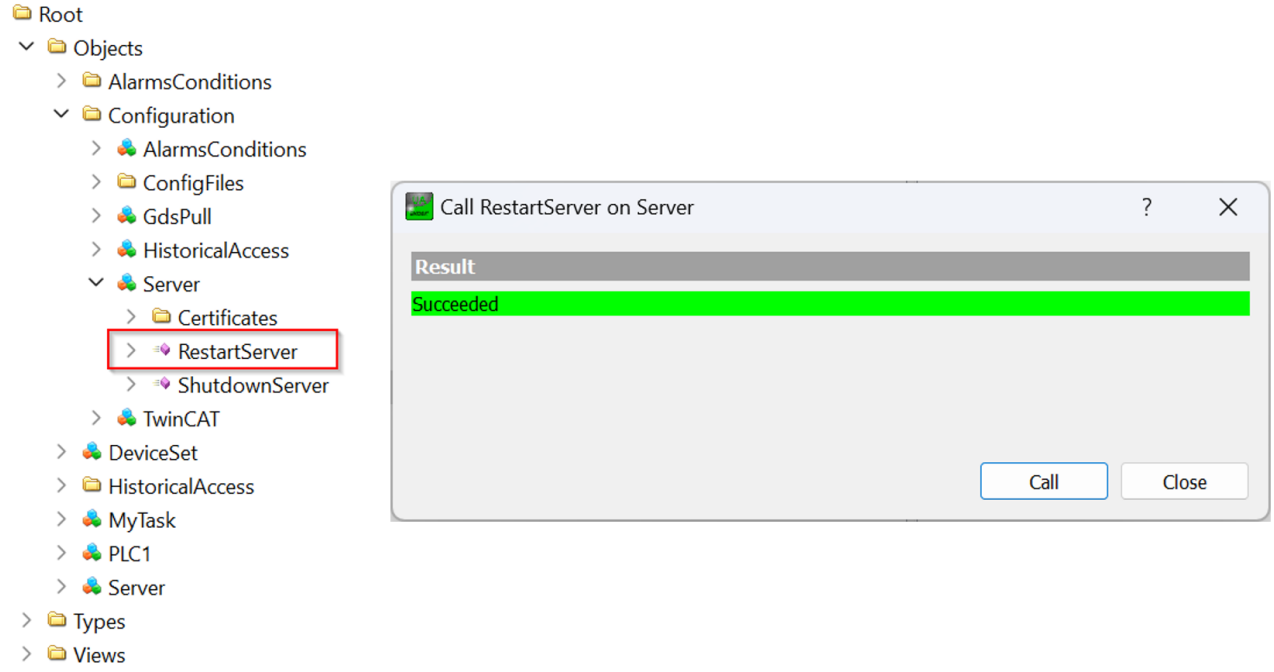
Clientzertifikate werden in „Rejected" und „Trusted" Zertifikate unterteilt, was durch einen separaten Ordner im Namensraum repräsentiert wird. Durch Aufruf der Methode `Move()` an einem Zertifikatsobjekt kann ein Zertifikat zwischen den Vertrauenslisten verschoben werden. Zudem bieten verschiedene Eigenschaften zur einfacheren Identifikation weitere Informationen über die Zertifikate selbst.



Im TwinCAT OPC UA Configurator steht Ihnen eine Benutzeroberfläche zur Verfügung, um das Trusten/ Rejecten von Zertifikaten einfacher zu gestalten. Weitere Informationen hierzu finden Sie im Kapitel Zertifikatsaustausch [▶ 126].

Neustart des Servers

Der Konfigurationsnamensraum beinhaltet eine Methode, mit der sich der TwinCAT OPC UA Server neu starten lässt, ohne dass ein TwinCAT-Neustart erforderlich ist.



4.8 Optimierungen

Es gibt verschiedene Möglichkeiten, die Kommunikationsverbindung zwischen OPC UA Client- und -Server bzw. SPS zu optimieren. Auch das Laufzeitverhalten, insbesondere die CPU- und Arbeitsspeicherauslastung vom TwinCAT OPC UA Server, kann durch verschiedene Parameter optimiert werden. Dieses Kapitel stellt verschiedene Optimierungsmöglichkeiten vor, insbesondere zu den folgenden Themen:

- SamplingInterval vs. PublishingInterval
- StructuredTypes

- StructuredTypes und deren Membervariablen

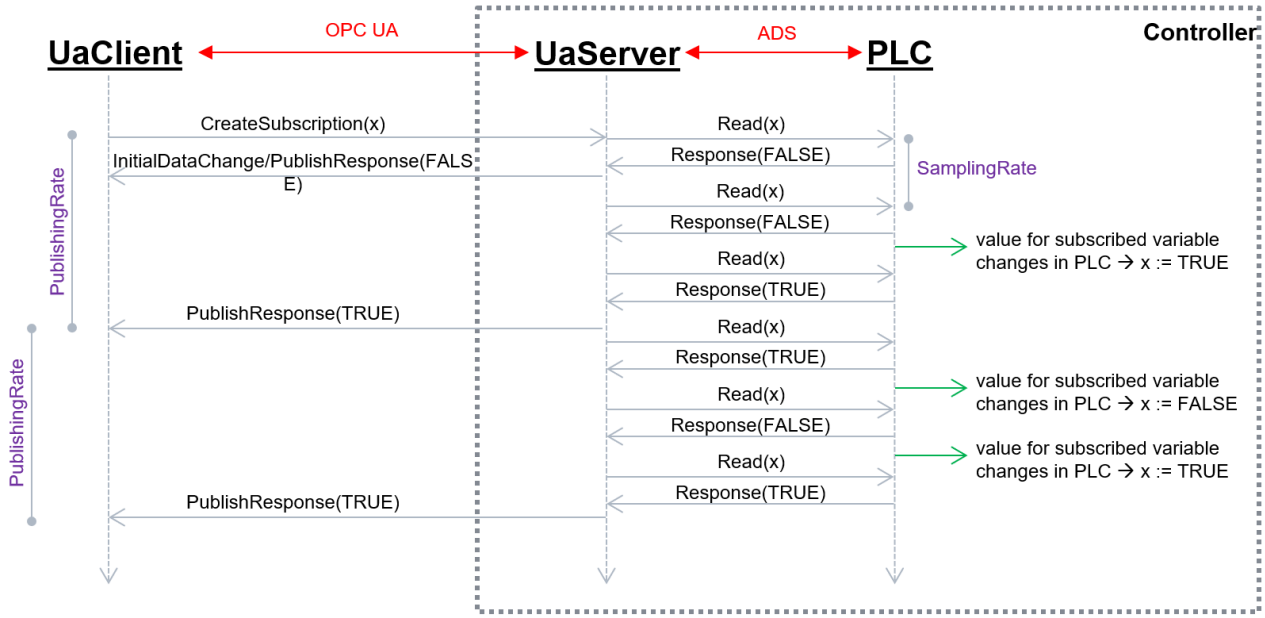
> Die hier dargestellten Screenshots und Performancewerte stellen Beispiele unter Laborbedingungen dar, welche auf unterschiedlichen Hardwaregeräten ausgeführt wurden. Sie lassen sich daher nicht 1:1 auf Kundenprojekte übertragen und dienen nur zur Veranschaulichung bestimmter Sachverhalte.

SamplingInterval vs. PublishingInterval

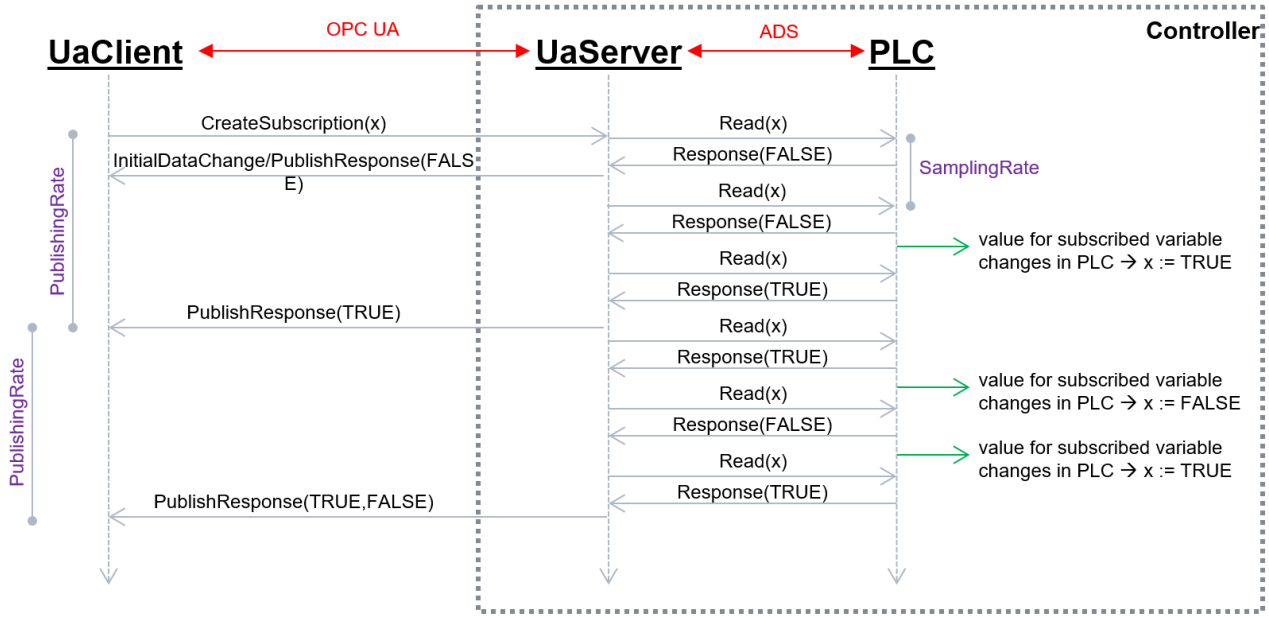
Beim Anlegen einer Subscription verwendet ein OPC UA Client verschiedene Parameter für die Subscription und die darin enthaltenen, sogenannten MonitoredItems, um Benachrichtigungen über Variablenänderungen zu erhalten. Die folgende Tabelle erklärt zwei dieser Parameter, welche anschließend näher beschrieben werden.

Parameter	Beschreibung
PublishingInterval	Das PublishingInterval gibt die Rate an, mit der ein OPC UA Client vom Server über Wertänderungen informiert wird. Das PublishingInterval wird in Part 4 der OPC UA Spezifikation detailliert beschrieben.
SamplingInterval	Das SamplingInterval gibt die Rate an, mit der der OPC UA Server seine unterliegende Datenquelle nach Wertänderungen abtasten soll, im Falle des TwinCAT OPC UA Servers über die ADS-Verbindung. Das SamplingInterval wird in Part 4 der OPC UA Spezifikation detailliert beschrieben.

Die folgende Grafik stellt den Zusammenhang zwischen diesen beiden Parametern noch einmal anschaulich dar. Es wird hierbei davon ausgegangen, dass der TwinCAT OPC UA Server auf der SPS-Steuerung installiert wurde und der OPC UA Client von einem externen System aus auf den Server zugreift.



Wie in dem Schaubild erkennbar, kann hierbei durchaus die Situation entstehen, dass der OPC UA Client bestimmte Wertänderungen in der SPS nicht mitbekommt, z. B. wenn diese entweder „zu schnell“ in der SPS passieren bzw. die SamplingRate nicht hoch genug ist bzw. ein Variablenwert wieder auf den ursprünglichen Wert zurück geht (wie oben erkennbar). Über einen weiteren Parameter, die sogenannte QueueSize, lassen sich mehrere Wertänderungen zwischen den PublishingIntervallen erfassen und an den OPC UA Client übertragen. In obigem Beispiel wurde eine QueueSize von 1 gewählt, d. h. es wird immer der „Last Known Value“ an den Client übertragen. In dem folgenden Schaubild wurde hingegen eine QueueSize von 2 gewählt, d. h. es werden die letzten zwei bekannten Wertänderungen an den Client übertragen.

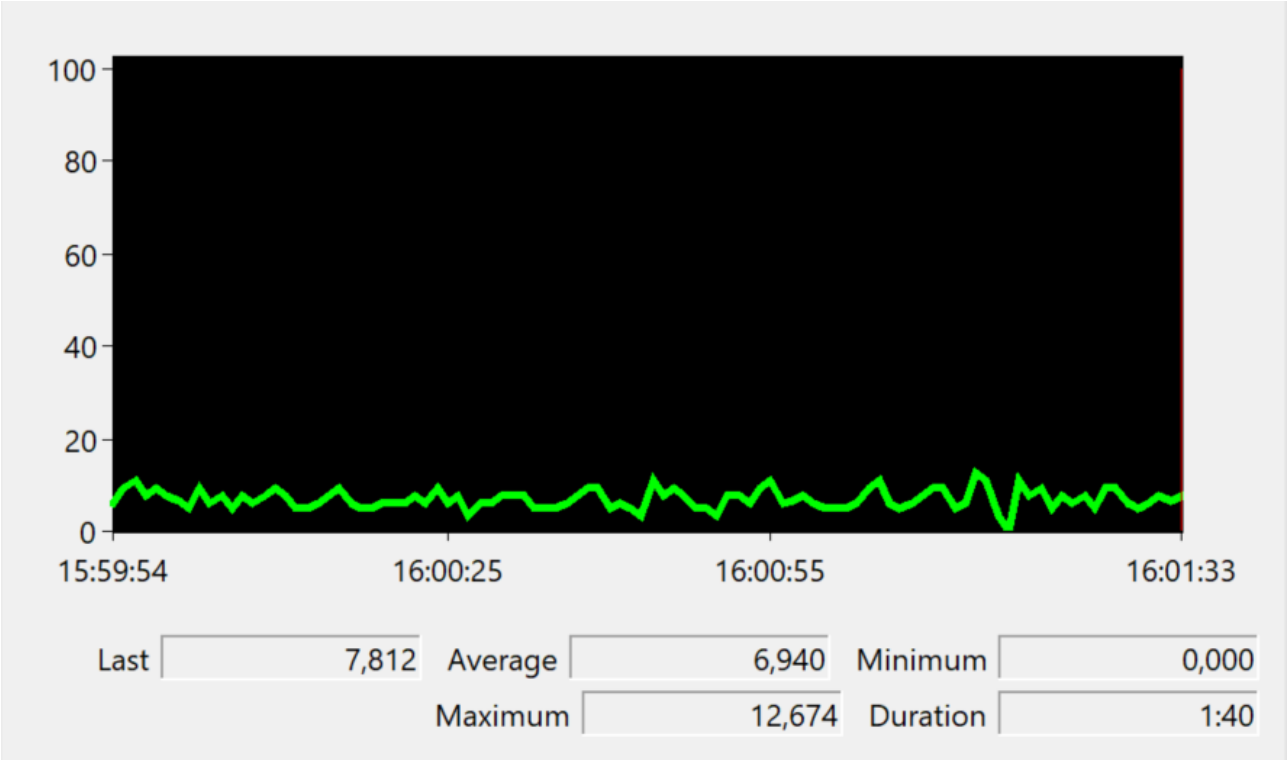


Bei dem ersten PublishResponse lässt sich erkennen, dass nur eine Wertänderung an den Client übermittelt wird, da auch nur eine Wertänderung in der SPS stattgefunden hat. Beim zweiten PublishResponse lässt sich erkennen, dass zwei Wertänderungen in der SPS stattgefunden haben und beide auch an den Client übertragen werden.

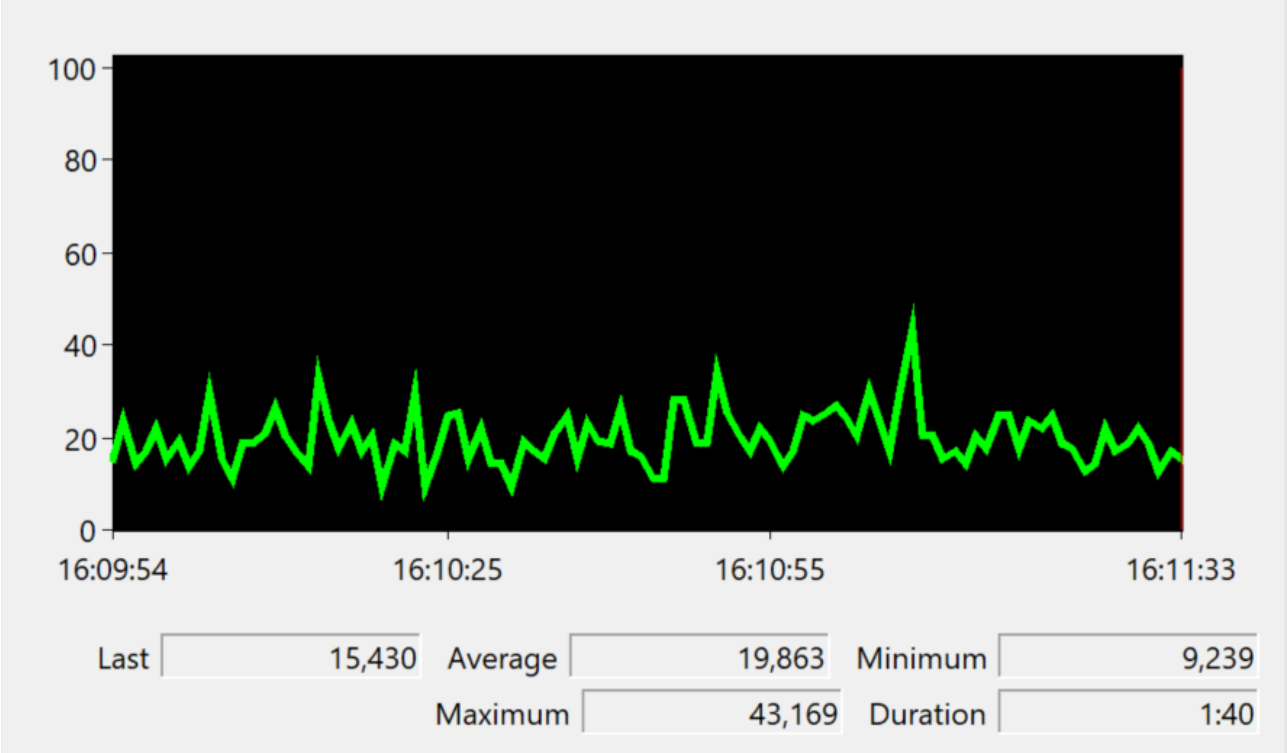
Bei den oben beschriebenen Parametern handelt es sich um Einstellungen, die der Client üblicherweise selbst in der Hand hat und beim Server anfragt. Und genau hier lassen sich viele Optimierungen vornehmen, denn beide Parameter haben einen starken Einfluss darauf, wie viel CPU-Zeit der OPC UA Client- und -Server benötigen, da entsprechend viele Informationen verarbeitet bzw. angefragt werden müssen.

Abhängig vom verwendeten Einsatzszenario sollten die beiden Parameter entsprechend sinnvoll eingestellt werden. Wenn es sich bei dem OPC UA Client zum Beispiel um eine Visualisierung handelt, dann machen schnelle PublishingIntervalle und SamplingRaten nur bedingt Sinn, da das menschliche Auge ohnehin keine Informationen schneller als ~200 ms verarbeiten kann. Auch die Verwendung der QueueSize sollte in Abhängigkeit zum Einsatzszenario sinnvoll gewählt werden. Wenn der OPC UA Client ohnehin keine Werte aus der Queue verarbeitet, ist eine QueueSize von 1 ausreichend und sinnvoll, da entsprechend weniger Informationen übertragen werden müssen und dies das System weiter optimiert.

In dem folgenden Beispiel hat ein OPC UA Client eine Subscription mit 10.000 Variablen auf dem TwinCAT OPC UA Server angelegt. Als PublishingInterval wurden hierbei 500 ms und als SamplingInterval 250 ms gewählt. Die CPU-Auslastung des TwinCAT OPC UA Servers lag bei einem Durchschnittswert von ca. 6,9 %, siehe folgenden Screenshot vom Windows Performance Monitor.



Anschließend wurde das PublishingInterval auf 200 ms und das SamplingInterval auf 100 ms eingestellt. Die CPU-Auslastung des TwinCAT OPC UA Servers erhöhte sich hierdurch auf einen Durchschnittswert von ca. 19 %.



StructuredTypes

Standardmäßig stellt der TwinCAT OPC UA Server Strukturen [▶ 75] als FolderType in seinem Adressraum bereit. Die Membervariablen werden dann als separate Nodes unterhalb des Folders dargestellt und auf sie kann zugegriffen werden. Der Server ermöglicht auch die Verwendung von StructuredTypes für die Struktur, wodurch die Typinformationen der Struktur vom Client verarbeitet werden und die Struktur datenkonsistent gelesen/geschrieben werden kann. Bei der Kommunikation mit der SPS (über das TwinCAT-ADS-Protokoll) verhalten sich beide Varianten grundlegend unterschiedlich.

Bei dem Zugriff eines OPC UA Clients auf einzelne Membervariablen kann es, je nach Anzahl der Variablen, durchaus passieren, dass die ADS-Kommunikation auf mehrere ADS Read/Write Kommandos aufgeteilt wird. Dies kann wiederum zur Folge haben, dass die ADS-Kommandos in unterschiedlichen SPS-Zyklen von der SPS abgearbeitet werden. Eine Datenkonsistenz kann hierbei also nicht garantiert werden.

Bei dem Zugriff eines OPC UA Clients auf einen StructuredType wird hingegen sichergestellt, dass der StructuredType in einem einzigen ADS Read/Write Kommando von der SPS abgearbeitet wird, wodurch eine Datenkonsistenz sichergestellt wird.

Bei der Verwendung von StructuredTypes für große SPS-Datenstrukturen sollten die folgenden Dinge beachtet werden:

- Das ADS Read/Write Kommando bzw. die entsprechende Antwort kann je nach Strukturgröße sehr groß werden und entsprechend viel Speicher im TwinCAT-ADS-Router benötigen. **Bitte achten Sie daher auf den Routerspeicher.**
- StructuredTypes müssen bei der Kommunikation mit der TwinCAT-Echtzeitumgebung vom TwinCAT OPC UA Server kodiert bzw. dekodiert werden, was zusätzliche CPU-Zeit benötigt.
- Je nach Größe der Struktur können Lese-/Schreibkommandos lange dauern, was auch stark abhängig von der Zeit ist die der Server zum Codieren/Decodieren benötigt.

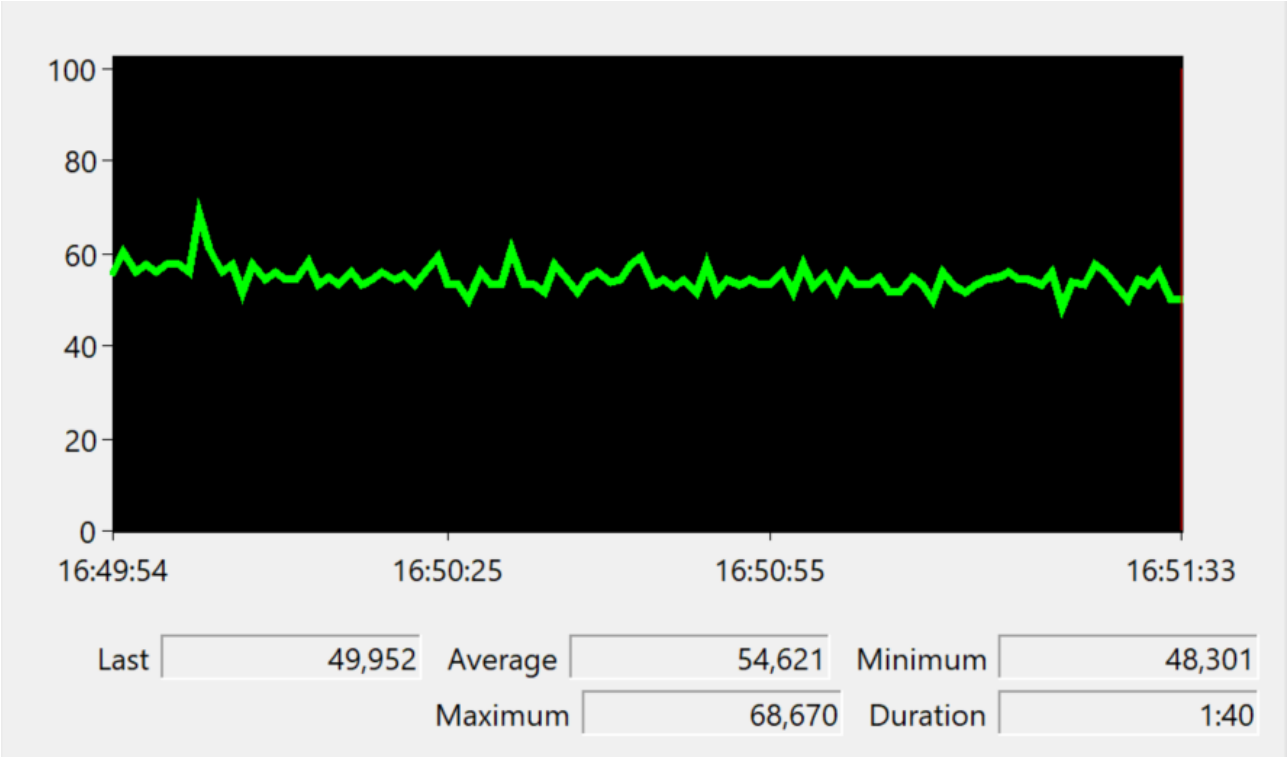
Aus diesen Gründen ist die Maximalgröße eines StructuredTypes im Auslieferungszustand des Servers limitiert, kann jedoch bei Bedarf über den Parameter <MaxStructureSize> in der Konfigurationsdatei TcUaDaConfig.xml angepasst werden. Dieser Parameter gibt die maximale Größe einer Struktur in Bytes an. Wenn eine Struktur die Größe <MaxStructureSize> überschreitet, wird sie als FolderType importiert, wo jedes Strukturelement als einzelner Knoten verfügbar ist. Für weitere Informationen siehe Kapitel Strukturen [▶ 75].

Gerade im Zusammenhang mit den bei einer Subscription einstellbaren Parametern zum SamplingInterval und PublishingInterval lassen sich bei StructuredTypes einige Optimierungen vornehmen, welche einen großen Einfluss auf das Laufzeitverhalten des Servers haben können.

In dem folgenden Beispiel hat ein OPC UA Client eine Subscription auf einen StructuredType angelegt. Die unterlagerte SPS-Datenstruktur ist hierbei wie folgt aufgebaut:

```
TYPE ST_TEST :
STRUCT
  stComplex : ST_Complex_1;
  strString5 : STRING[5];
  eEnum : E_Enum_1;
  strString3 : STRING[3];
  arrComplex : ARRAY[0..9999] OF ST_Complex_1;
  arrDint : ARRAY[0..9999] OF DINT;
END_STRUCT
END_TYPE
```

Die als Membervariable verwendete Struktur ST_Complex_1 ist ca. 91 Byte groß. Insgesamt handelt es sich hierbei also um eine Datenstruktur mit einer Gesamtgröße von ca. 1 Mbyte. Als PublishingInterval wurden 500 ms und als SamplingInterval 250 ms gewählt. Die CPU-Auslastung des TwinCAT OPC UA Servers lag nach dem Anlegen der Subscription bei einem Durchschnittswert von ca. 54,6 %, siehe folgenden Screenshot vom Windows Performance Monitor.



Entsprechend des gewählten SamplingIntervals wird bei der unterlagerten ADS-Kommunikation ca. alle 250 ms ein ADS Read abgesetzt. Bei dem zugehörigen Response erkennt man deutlich die Größe der Datenstruktur von ca. 1 Mbyte, d. h. dass alle 250 ms ein 1 Mbyte großes Datenpaket durch den TwinCAT-ADS-Router transportiert wird (und vom Server entsprechend verarbeitet werden muss). Der Codierungs/Decodierungs-Vorgang vom Server benötigt entsprechend viel CPU-Leistung, was die hohe CPU-Auslastung erklärt.

15/06/2022 16:55:17 326 ms	R	Req	10.0.2.15.1.1 (33391)	10.0.2.15.1.1 (851)	0	0x755	12	IG 0xf005 IO 0x1a000210 Len 1000108
15/06/2022 16:55:17 327 ms	R	Res	10.0.2.15.1.1 (851)	10.0.2.15.1.1 (33391)	0	0x755	1000116	Res 0x0, Len 1000108 + 03 00 00 00 2a 00 00 00 48
15/06/2022 16:55:17 574 ms	R	Req	10.0.2.15.1.1 (33359)	10.0.2.15.1.1 (851)	0	0x852	12	IG 0xf005 IO 0x1a000210 Len 1000108
15/06/2022 16:55:17 575 ms	R	Res	10.0.2.15.1.1 (851)	10.0.2.15.1.1 (33359)	0	0x852	1000116	Res 0x0, Len 1000108 + 03 00 00 00 2a 00 00 00 48
15/06/2022 16:55:17 822 ms	R	Req	10.0.2.15.1.1 (33391)	10.0.2.15.1.1 (851)	0	0x756	12	IG 0xf005 IO 0x1a000210 Len 1000108
15/06/2022 16:55:17 823 ms	R	Res	10.0.2.15.1.1 (851)	10.0.2.15.1.1 (33391)	0	0x756	1000116	Res 0x0, Len 1000108 + 03 00 00 00 2a 00 00 00 48
15/06/2022 16:55:18 70 ms	R	Req	10.0.2.15.1.1 (33359)	10.0.2.15.1.1 (851)	0	0x853	12	IG 0xf005 IO 0x1a000210 Len 1000108
15/06/2022 16:55:18 71 ms	R	Res	10.0.2.15.1.1 (851)	10.0.2.15.1.1 (33359)	0	0x853	1000116	Res 0x0, Len 1000108 + 03 00 00 00 2a 00 00 00 48
15/06/2022 16:55:18 318 ms	R	Req	10.0.2.15.1.1 (33391)	10.0.2.15.1.1 (851)	0	0x757	12	IG 0xf005 IO 0x1a000210 Len 1000108
15/06/2022 16:55:18 319 ms	R	Res	10.0.2.15.1.1 (851)	10.0.2.15.1.1 (33391)	0	0x757	1000116	Res 0x0, Len 1000108 + 03 00 00 00 2a 00 00 00 48

StructuredTypes und deren Membervariablen

Standardmäßig werden die Membervariablen eines StructuredType im Adressraum des Servers als eigene Nodes dargestellt und verfügbar gemacht. Dies benötigt zusätzlichen Arbeitsspeicher, da der TwinCAT OPC UA Server für jede Node Arbeitsspeicher allokiert. Ein OPC UA Client, der ausschließlich mit dem StructuredType, also dem „Rootelement“ einer Struktur arbeitet, benötigt diese zusätzlichen Nodes nicht. Diese lassen sich über ein spezielles Pragma explizit ausblenden, was die Arbeitsspeicherauslastung des Servers verringert. Das Pragma ist im Kapitel Strukturen [▶ 75] näher beschrieben.

4.9 Applikationsverzeichnisse

Diese Applikation verwendet verschiedene Verzeichnisse, um relevante Informationen abzuspeichern, z. B. Konfigurations- oder Zertifikatsdateien.

Installationsverzeichnis

Das Basis-Installationsverzeichnis der Applikation ist auf allen Betriebssystemen immer relativ zum TwinCAT-Installationsverzeichnis.

```
%TcInstallDir%\Functions\TF6100-OPC-UA
```

Unterhalb dieses Verzeichnisses wird die Applikation dann in folgendes Verzeichnis installiert. Hierbei wird eine Unterscheidung nach Plattform (x86/x64) gemacht.

```
%TcInstallDir%\Functions\TF6100-OPC-UA\Win32\Server
%TcInstallDir%\Functions\TF6100-OPC-UA\Win64\Server
```

Basisverzeichnis für PKI Infrastruktur

Zertifikatsdateien, welche zum Aufbau einer gesicherten Kommunikationsverbindung verwendet werden, werden auf allen Betriebssystemen in folgendem Verzeichnis abgelegt:

```
%TcInstallDir%\Functions\TF6100-OPC-UA\Win32\Server\PKI
%TcInstallDir%\Functions\TF6100-OPC-UA\Win64\Server\PKI
```

Verzeichnis für Zertifikats-Vertrauensstellung (trusted)

Clientzertifikate in diesem Verzeichnis werden als „vertrauenswürdig“ deklariert. Dieser Pfad ist auf allen Betriebssystemen identisch.

```
%TcInstallDir%\Functions\TF6100-OPC-UA\Win32\Server\PKI\CA\trusted
%TcInstallDir%\Functions\TF6100-OPC-UA\Win64\Server\PKI\CA\trusted
```

Verzeichnis für Zertifikats-Vertrauensstellung (rejected)

Clientzertifikate in diesem Verzeichnis werden als „nicht vertrauenswürdig“ deklariert. Dieser Pfad ist auf allen Betriebssystemen identisch.

```
%TcInstallDir%\Functions\TF6100-OPC-UA\Win32\Server\PKI\CA\rejected
%TcInstallDir%\Functions\TF6100-OPC-UA\Win64\Server\PKI\CA\rejected
```

Verzeichnis für das Serverzertifikat

Die Verzeichnisse für das Serverzertifikat sind wie folgt festgelegt, wobei zwischen dem Verzeichnis für den Public-Key („certs“) und Private-Key („private“) unterschieden wird.

```
%TcInstallDir%\Functions\TF6100-OPC-UA\Win32\Server\PKI\CA\own\certs
%TcInstallDir%\Functions\TF6100-OPC-UA\Win64\Server\PKI\CA\own\private
```

Logdateien

Logdateien werden in dem folgenden Verzeichnis abgelegt. Hierbei gibt es eine Unterscheidung nach Betriebssystem.

```
%TcInstallDir%\Functions\TF6100-OPC-UA\Win32\Server (Windows)
%TcInstallDir%\Functions\TF6100-OPC-UA\Win64\Server (Windows)
/var/log/TF6100-OPC-UA-Server (TwinCAT/BSD)
```

Konfigurationsdateien

Der TwinCAT OPC UA Server verwendet zur Konfiguration der einzelnen Funktionalitäten verschiedene Konfigurationsdateien, welche, wie in unten stehender Tabelle aufgelistet, definiert sind. Alle Konfigurationsdateien liegen hierbei im Installationsverzeichnis des Servers (siehe oben) und werden üblicherweise über den TwinCAT OPC UA Configurator konfiguriert.

Datei	Beschreibung
TcUaAcConfig.xml	Konfigurationsdatei für die Alarms & Conditions [▶ 100] (A&C) Funktionalität des Servers. In dieser Datei werden die für A&C konfigurierten Nodes gespeichert.
TcUaDaConfig.xml	Konfigurationsdatei für die Data Access [▶ 49] (DA) Funktionalität des Servers. In dieser Datei werden die Kommunikationsverbindungen mit den unterlagerten ADS-Geräten konfiguriert.
TcUaGdsClientConfig.xml	Konfigurationsdatei für die Global Discovery Server [▶ 119] Konfiguration. In dieser Datei wird die Verbindung zu einem GDS konfiguriert.
TcUaHaConfig.xml	Konfigurationsdatei für die Historical Access [▶ 93] (HA) Konfiguration des Servers. In dieser Datei werden sowohl die für HA verwendeten Datenspeicher als auch die hierfür konfigurierten Nodes gespeichert.
TcUaNodesetConfig.xml	Konfigurationsdatei für den Nodeset [▶ 70]-Import des Servers. In dieser Datei werden die zu importierenden Nodesets und deren Verknüpfung mit einem ADS-Gerät definiert.
TcUaSecurityConfig.xml	Konfigurationsdatei für Benutzer, Gruppen und Access Control Lists (ACL) im Rahmen der Definition von Zugriffsrechten [▶ 131] und Authentifizierungs [▶ 128]-Mechanismen.
TcUaServerConfig.xml	Konfigurationsdatei für verschiedene Parameter des Servers, z. B. die Konfiguration der Endpunkte [▶ 125], Logging [▶ 142], Reverse Connect [▶ 135].

4.10 Data Access

4.10.1 Überblick

In diesem Kapitel werden die notwendigen Schritte zur Konfiguration der Variablen im Namensraum des TwinCAT OPC UA Servers für **D**ata **A**ccess (DA) beschrieben.

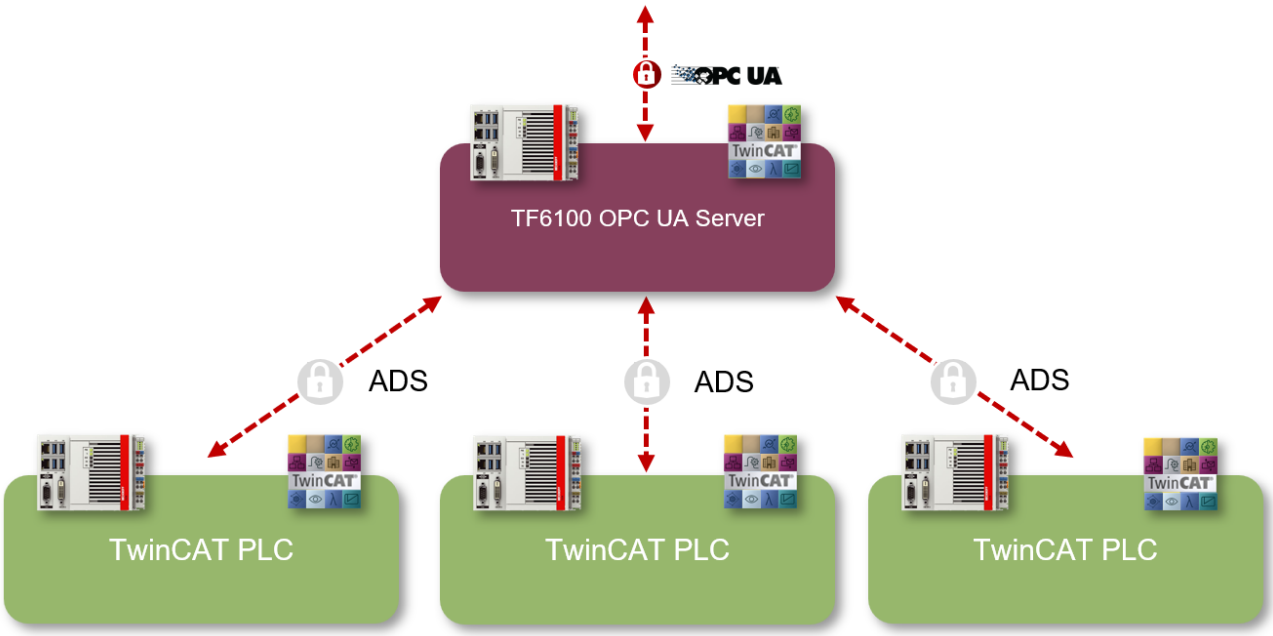
Data Access ist eine Funktion von OPC UA, welche die Darstellung und Verwendung von Variablenwerten beschreibt. Hier geht es zum Beispiel um die Dienstfunktionalitäten, wie ein OPC UA Client auf Variablenwerte zugreifen kann.

Der TwinCAT OPC UA Server kann Variablen aus allen TwinCAT Echtzeitumgebungen bereitstellen. Dazu gehören zum Beispiel die TwinCAT SPS, TwinCAT 3 C++, TwinCAT 3 Matlab®/Simulink® oder auch eine TwinCAT I/O Task. Hierbei kann der Server auf mehrere Echtzeitumgebungen zugreifen und deren Symbolik in seinem Adressraum bereitstellen.

Grundlage für weitere Funktionen

Die Verbindung mit der Laufzeit [▶ 50] und Freigabe von Symbolen [▶ 53] für Data Access ist Grundlage für weitere Funktionen, wie zum Beispiel Historical Access und Alarms & Conditions. Bitte stellen Sie sicher, dass Sie sich mit den hier notwendigen Einstellungen vertraut machen.

Die verschiedenen Echtzeitumgebungen müssen sich nicht zwangsläufig auf demselben System befinden, sondern können auf verschiedene Steuerungen verteilt sein. In diesem Fall muss zu jedem System eine ADS-Route aufgebaut werden. Das folgende Schaubild verdeutlicht diesen Zusammenhang.



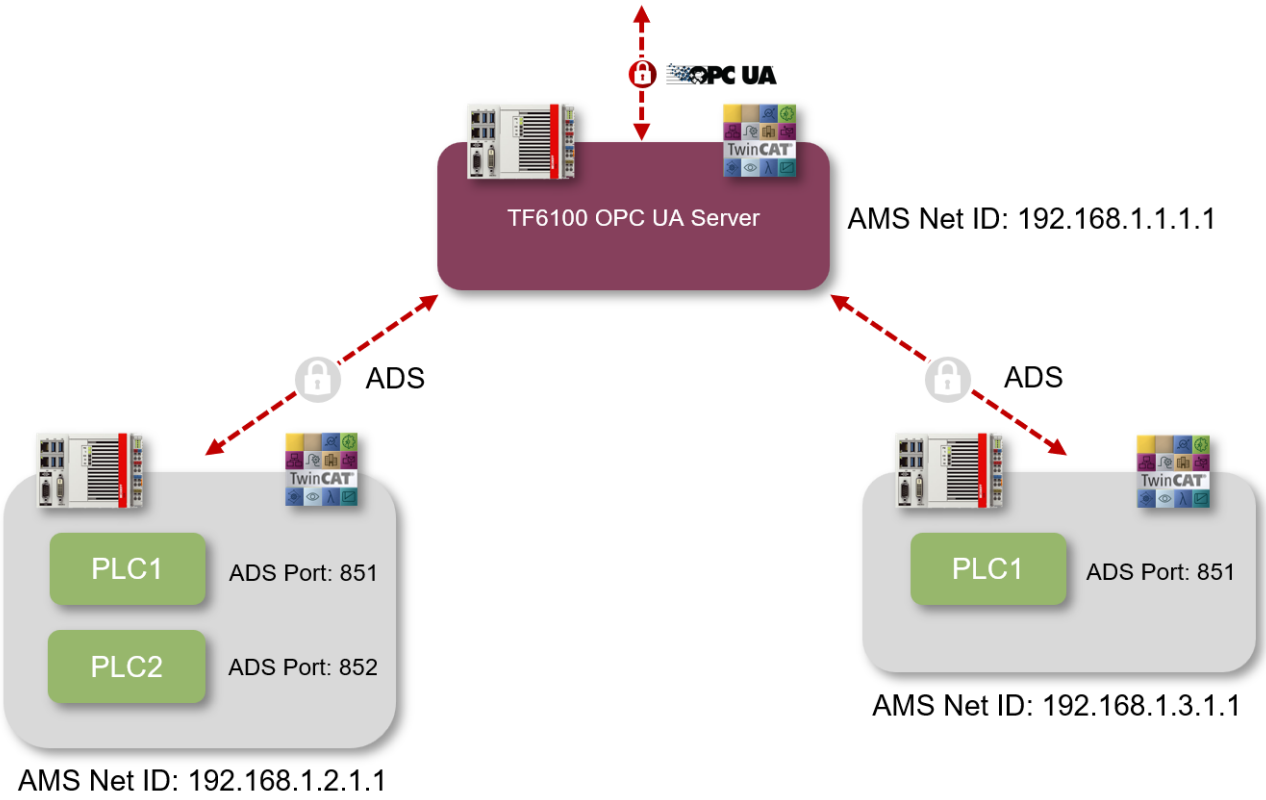
Im TwinCAT OPC UA Configurator kann die Data-Access-Konfiguration in der Registerkarte **Data Access** vorgenommen werden. Alle angeschlossenen Echtzeitumgebungen werden dort als eigenständiges „Gerät“ angezeigt. Im Kapitel Verbindung mit der Laufzeit [▶ 50] erfahren Sie, wie Sie Data-Access-Geräte hinzufügen können und welche Parameter dafür notwendig sind. Anschließend können Sie mit der Freigabe von Symbolen [▶ 53] beginnen.

Sehen Sie dazu auch

- Überblick [▶ 93]

4.10.2 Verbindung mit der Laufzeit

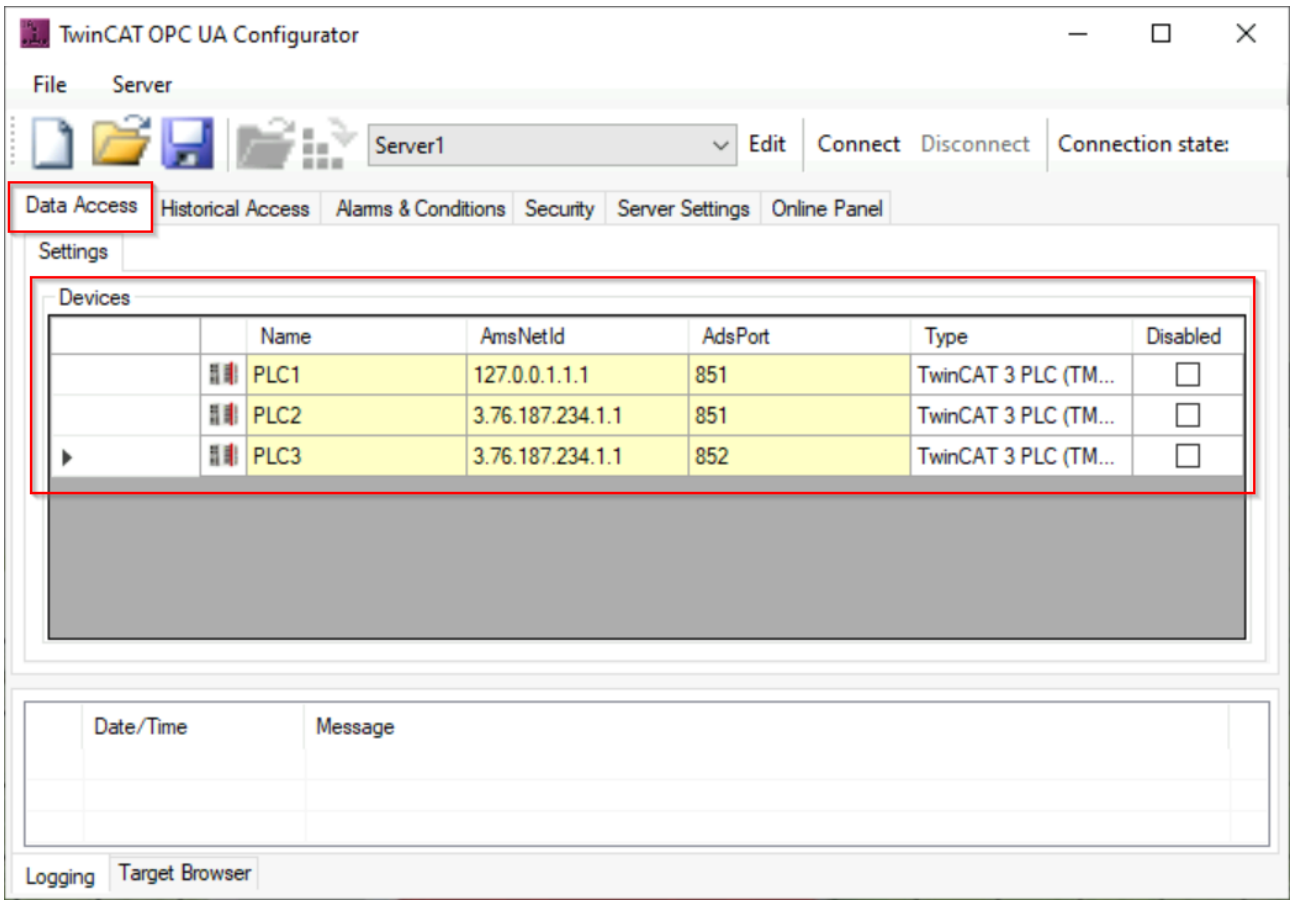
Der TwinCAT OPC UA Server kann Symbole aus einer oder mehreren Echtzeitumgebungen bereitstellen. Diese können sich auch auf unterschiedlichen physikalischen Steuerungssystemen befinden. In diesem Fall muss eine ADS-Route zu dem jeweiligen Zielsystem hergestellt werden. Die sogenannte „AMS Net ID“ identifiziert hierbei ein Steuerungssystem eindeutig im Netzwerk und wird benötigt, um eine ADS-Route zu dem System aufzubauen. Der ADS Port hingegen identifiziert dann eine bestimmte Applikation auf diesem System, z. B. die TwinCAT-SPS.
Das folgende Schaubild verdeutlicht diesen Zusammenhang.



In diesem Beispiel gibt es drei Steuerungsgeräte. Auf dem ersten Gerät ist der TwinCAT OPC UA Server installiert und dieses Gerät wird durch die AMS Net ID 192.168.1.1.1.1 identifiziert. Auf dem zweiten Gerät, welches durch die AMS Net ID 192.168.1.2.1.1 identifiziert wird, sind zwei SPS-Laufzeiten gestartet und werden durch jeweils einen eigenen ADS Port angesprochen. Auf dem dritten Gerät (192.168.1.3.1.1) läuft nur eine SPS-Laufzeit.

Konfiguration

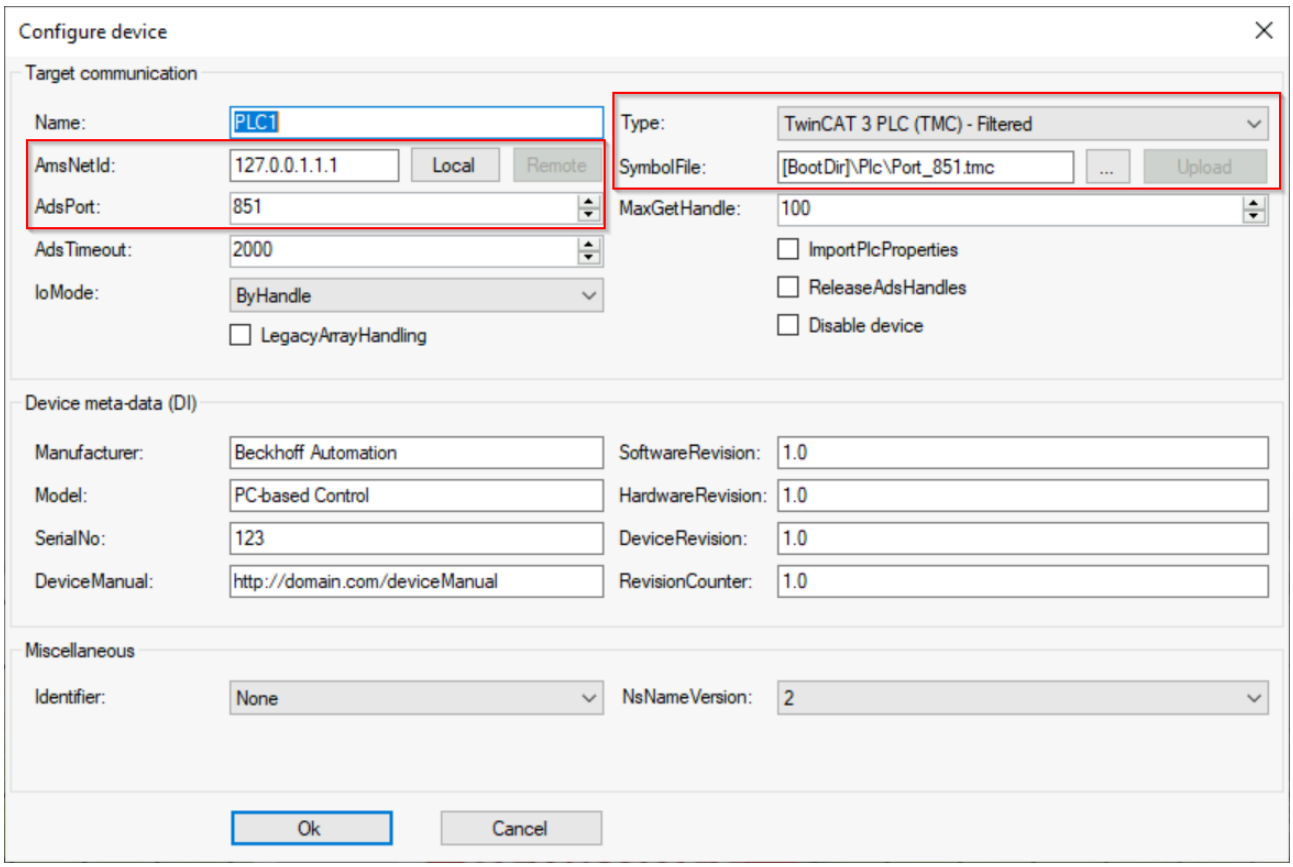
Die Konfiguration für ein Data-Access-Gerät kann mit Hilfe des TwinCAT OPC UA Configurators erfolgen. In der Registerkarte **Data Access** erhalten Sie einen Überblick über alle konfigurierten Geräte und können Geräte hinzufügen oder entfernen.



In diesem Screenshot sehen Sie zum Beispiel eine Konfiguration mit drei Data-Access-Geräten. Diese sind wie folgt konfiguriert worden:

- PLC1: Lokale SPS-Laufzeit (127.0.0.1.1.1) auf Port 851
- PLC2: Remote SPS-Laufzeit (3.76.187.234.1.1) auf Port 851
- PLC3: Remote SPS-Laufzeit (3.76.187.234.1.1) auf Port 852

Alle für die Verbindung mit einem Gerät und dessen Symbolik benötigten Parameter können nun in den Eigenschaften des Geräts konfiguriert werden.



Abhängig von der verwendeten Echtzeitumgebung können Sie hier zum Beispiel eine Symboldatei auswählen, die zu verwendende ADS-Route selektieren und den ADS Port der jeweiligen Applikation eintragen. Mehr Informationen zu den verschiedenen Symboldateien erhalten Sie im Kapitel Freigabe von Symbolen [▶ 53].

4.10.3 Freigabe von Symbolen

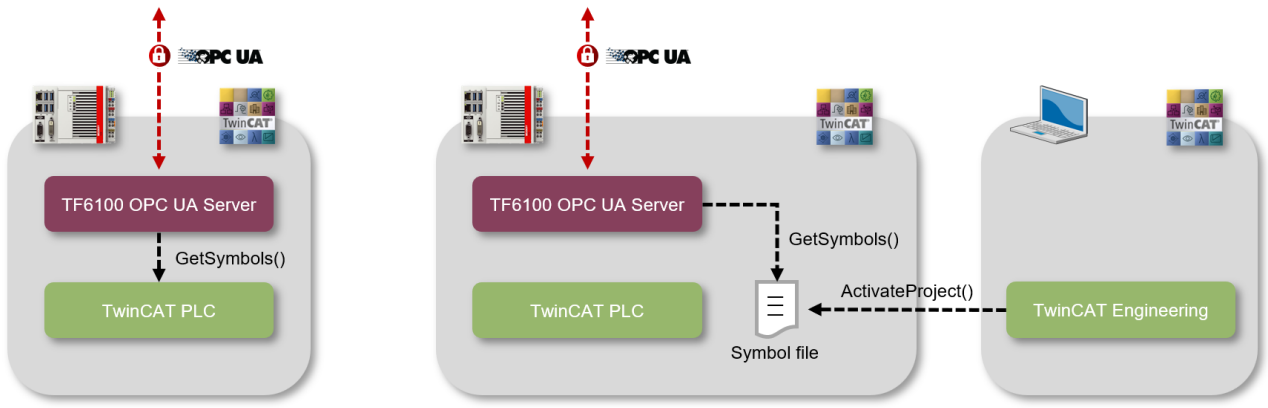
Wie Sie bereits im Quick Start [▶ 21] Tutorial erfahren haben, erfolgt die Freigabe von Symbolen in der TwinCAT-3-SPS über ein sogenanntes Pragma (auch „Attribut“ genannt). In anderen Echtzeitumgebungen kann der Freigabemechanismus variieren. Der folgende Kapitel gibt Ihnen hierzu einen Überblick.

Der Begriff „Symbol“

Der Begriff „Symbol“ wird in dieser Dokumentation stellvertretend für Variable, Struktur, Funktionsbausteininstanz, Methode, usw., verwendet.

Symbolik

Der TwinCAT OPC UA Server erhält seine Informationen über Echtzeitvariablen (z. B. Adressen und Datentypen), die sogenannte „Symbolik“. Abhängig von der verwendeten Echtzeitumgebung (z. B. TwinCAT-SPS oder TwinCAT 3 C++) stehen verschiedene sogenannte „Symboldateien“ für das Auslesen der Symbolik bereit (auch als „Offline-Symbolik“ bezeichnet). Zusätzlich kann die Symbolik über TwinCAT-ADS ausgelesen werden, wenn die Laufzeitumgebung gestartet ist (auch als „Online-Symbolik“ bezeichnet).



Online Symbolic Offline Symbolic

Der Vorteil bei der Verwendung der Online-Symbolik ist, dass keinerlei Symboldateien ausgetauscht werden müssen. Gerade in der Installationsvariante [▶ 16], bei der der TwinCAT OPC UA Server auf einem Gateway-PC betrieben wird, entfällt somit der manuelle Kopiervorgang der Symboldateien aller angeschlossenen Steuerungen. Der Nachteil von Online-Symbolik ist, dass die Symbole erst bei laufender SPS-Umgebung zur Verfügung stehen.

Die Symboldatei muss vom Server importiert werden, damit dieser auf die Adressinformationen von Symbolen zugreifen kann. Alternativ kann der Server auch direkt mit der Online-Symbolik arbeiten. Die Symbole stehen dann erst im Adressraum des Servers zur Verfügung, wenn die Echtzeitumgebung verfügbar ist.

Für die Konfiguration des Servers und welche Symboldatei dieser einlesen soll, kann der TwinCAT OPC UA Configurator verwendet werden. Der entsprechende Pfad zur Symboldatei ist dort als Teil des Data-Access-Geräts konfigurierbar.

Abhängig von der verwendeten Echtzeitumgebung variiert somit die Symboldatei, welche vom Server importiert werden muss, um Adressinformationen zu den vorhandenen Symbolen zu erhalten und auch der Freigabemechanismus von Symbolen. Während man bei der TwinCAT-3-SPS mit sogenannten Pragmas arbeitet, um ein Symbol für OPC UA freizugeben, so werden hierfür unter TwinCAT 2 noch speziell formatierte Kommentare an dem Symbol verwendet. Unter TwinCAT 3 C++ hingegen verwendet man den sogenannten TMC-Code-Generator, um Symbole für OPC UA freizugeben.

Symboldateien können in der Engineeringoberfläche von TwinCAT so konfiguriert werden, dass sie beim Aktivieren des Projekts automatisch in das Bootverzeichnis vom Zielgerät kopiert werden. Der TwinCAT OPC UA Server wird üblicherweise so konfiguriert, dass er die Symboldatei aus dem Bootverzeichnis ausliest.

Die folgende Tabelle gibt einen Überblick über die verschiedenen Symboldateien in Bezug auf ihre Echtzeitumgebung und den Freigabemechanismus.

Echtzeitumgebung	Symboldatei	Freigabe über…	Pfad im Bootverzeichnis
TwinCAT 2 SPS	TPY	Kommentar	CurrentPlc_1.tpy
TwinCAT 3 SPS [▶ 55]	TMC	Pragma	Plc\Port_%AdsPort%.tmc
TwinCAT 3 C++ [▶ 57]	TMI	TMC-Code-Generator	Tmi\%ObjectId%.tmi
TwinCAT 3 Matlab®/ Simulink [▶ 61]®	TMI	TMC-Code-Generator	Tmi\%ObjectId%.tmi
TwinCAT 3 I/O Task [▶ 67]	XML	Kommentar	CurrentConfig.xml
Online-Symbolik [▶ 69]	---	Pragma TMC-Code-Generator	---

Weitere Informationen finden Sie in den Unterkapiteln zu der jeweiligen Echtzeitumgebung.

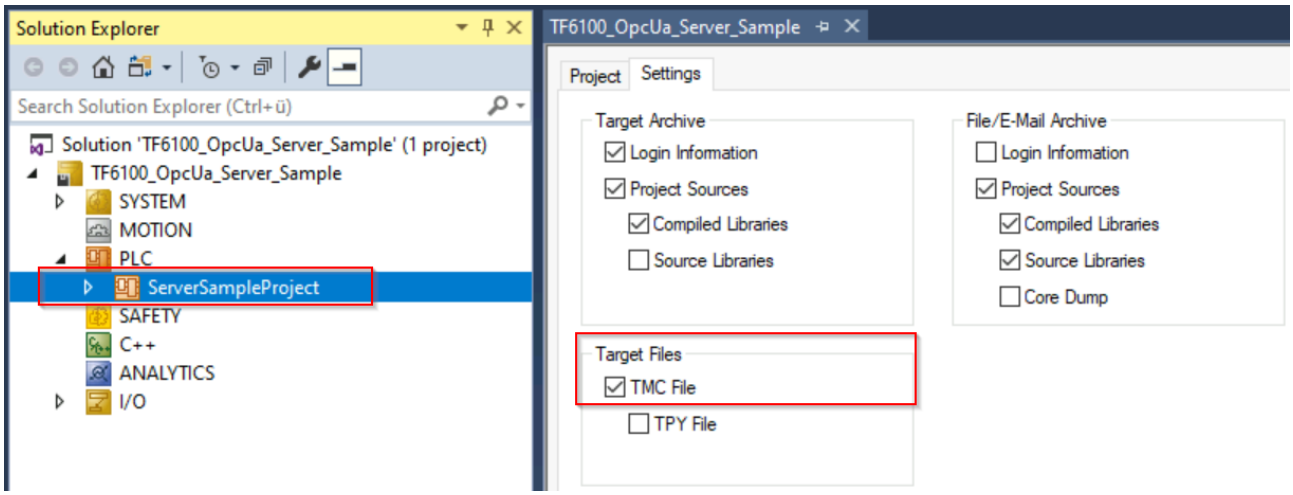
4.10.3.1 SPS

In diesem Kapitel wird beschrieben, wie Sie den Namensraum des TwinCAT OPC UA Servers konfigurieren, um Zugriff auf die Symbole eines SPS Projekts zu erhalten. Dies erfordert die Durchführung der folgenden Schritte:

- Aktivieren der Symboldatei
- Freigabe von Symbolen
- Konfiguration des Servers

Aktivieren der Symboldatei

Um die Symbole eines SPS Projekts über OPC UA verfügbar zu machen, ist es zunächst einmal erforderlich, daß Sie den Download der Symboldatei (TMC) in den Eigenschaften des SPS Projekts aktivieren.



Hierdurch wird beim Aktivieren des Projekts die Symboldatei automatisch auf das entsprechende Zielgerät übertragen und liegt dort dann im TwinCAT Bootverzeichnis vor. Der Name der Symboldatei orientiert sich hierbei am ADS Port (siehe unten) der SPS Laufzeit, zum Beispiel:

```
%TwinCATInstallDir%\3.1\Boot\Plc\Port_851.tmc
```

Quick Start

Ist der TwinCAT OPC UA Server ebenfalls auf demselben Gerät installiert, so liest er im Auslieferungszustand automatisch immer die Symboldatei der ersten SPS Laufzeit ein, wodurch keine weiteren Einstellungen am Server notwendig sind. Durch das Aktivieren des Projekts wird somit die Symboldatei auf das Zielgerät übertragen und durch einen anschließenden TwinCAT Neustart auch der TwinCAT OPC UA Server neu gestartet. Dieser findet nun beim Startvorgang die Symboldatei im Bootverzeichnis und liest diese ein. Durch die gesetzten Pragmas erkennt der Server welche Symbole er in seinem Adressraum bereitstellen soll.

Freigabe von Symbolen

In der TwinCAT 3 SPS werden Symbole über sogenannte Pragmas für OPC UA freigegeben. Ein solches Pragma wurde zum Beispiel auch im Quick Start [▶ 21] Tutorial verwendet, um eine Variable für OPC UA freizugeben. Das Pragma wird vor dem Symbol eingefügt für das es gelten soll. Das folgende Pragma kann verwendet werden, um eine Variable, Array, Struktur für OPC UA freizugeben. Der TwinCAT OPC UA Server erkennt beim Auslesen der Symbolik dieses Pragma und importiert das zugehörige Symbol in seinen Namensraum. Die entsprechende Typinformation [▶ 71] wird hierbei übernommen und auf OPC UA abgebildet.

```
{attribute 'OPC.UA.DA' := '1'}
nMyCounter : INT;
```

Für einzelne Teilfunktionen des Data Access können zusätzliche Pragmas erforderlich sein, zum Beispiel Strukturen [▶ 75], Properties [▶ 78], AnalogItemType [▶ 82], StatusCode [▶ 79], das Setzen einer Description [▶ 83], Alias [▶ 85], oder auch des Read-Only [▶ 84] Flags einer Node. Die zusätzlichen Pragmas werden dann in einer separaten Zeile eingefügt, zum Beispiel:

```
{attribute 'OPC.UA.DA' := '1'}
{attribute 'OPC.UA.DA.Access' := '1'}
nMyCounter : INT;
```

(Setzt das Read-Only Flag für diese Variable)

Das Pragma zur Freigabe eines Symbols wird auf alle Kind-Symbole automatisch weitervererbt. Möchte man die Vererbung ab einer bestimmten Stelle blockieren, zum Beispiel ab einem bestimmten Member einer Struktur, so kann mit dem folgenden Pragma gearbeitet werden:

```
{attribute 'OPC.UA.DA' := '0'}
stChild : ST_ChildStruct;
```

Im Falle von Funktionsbausteinen und Strukturen ist der Definitionsort des Pragmas entscheidend. Definiert man das Pragma an einer Instanz, so wird nur diese Instanz (und alle Kindelemente) für OPC UA freigegeben. Definiert man das Pragma hingegen an der Funktionsbaustein- oder Strukturdefinition, so werden alle Instanzen des Funktionsbausteins oder der Struktur freigegeben. In dem folgenden Beispiel sollen die beiden Instanzen fbTest1 und fbTest2 des Funktionsbausteins FB_BLOCK1 über OPC UA verfügbar gemacht werden. Wenn eine ganze Instanz freigegeben wird, sind über OPC UA auch alle ihre Symbole verfügbar. Das SPS-Programm sieht wie folgt aus:

```
PROGRAM MAIN
VAR
  {attribute 'OPC.UA.DA' := '1'}
  fbTest1 : FB_BLOCK1;
  fbTest2 : FB_BLOCK1;
END_VAR
```

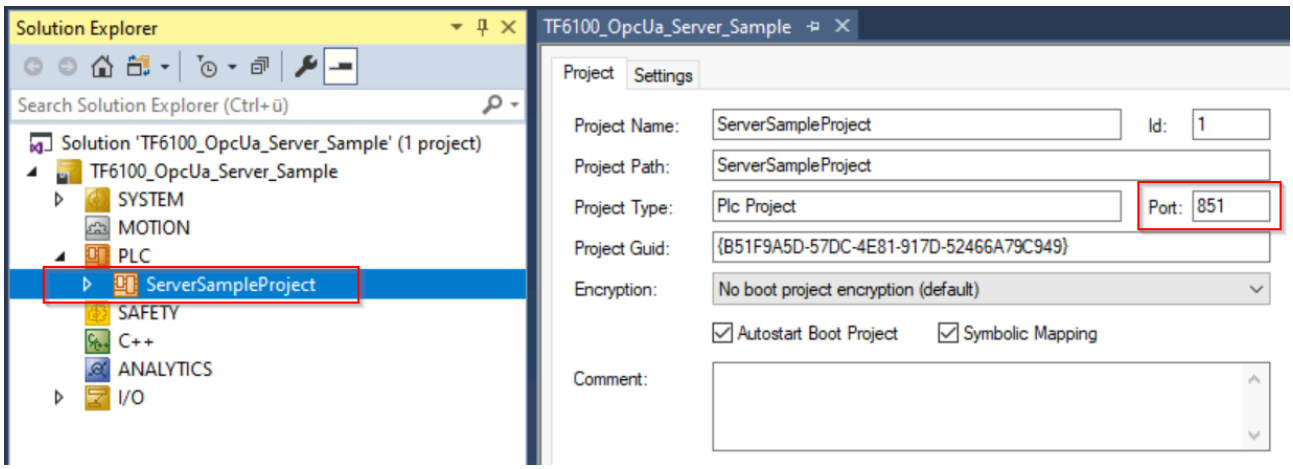
```
FUNCTION_BLOCK FB_BLOCK1
VAR_INPUT
  {attribute 'OPC.UA.DA' := '1'}
  ni1 : INT;
  ni2 : INT;
END_VAR
VAR_OUTPUT
  {attribute 'OPC.UA.DA' := '1'}
  no1 : INT;
  no2 : INT;
END_VAR
VAR
  {attribute 'OPC.UA.DA' := '1'}
  nx1 : INT;
  nx2 : INT;
END_VAR
```

Die Instanz fbTest1 erhält nun das Pragma, wodurch alle enthaltenen Symbole automatisch ebenfalls für OPC UA freigegeben werden, sprich fbTest.ni1, fbTest.ni2, usw. Die Instanz fbTest2 erhält hingegen nicht das Pragma, allerdings wurden die drei darin enthaltenen Variablen ni1, no1 und nx1 mit dem Pragma gekennzeichnet. Diese sind demzufolge in allen Instanzen über OPC UA verfügbar.

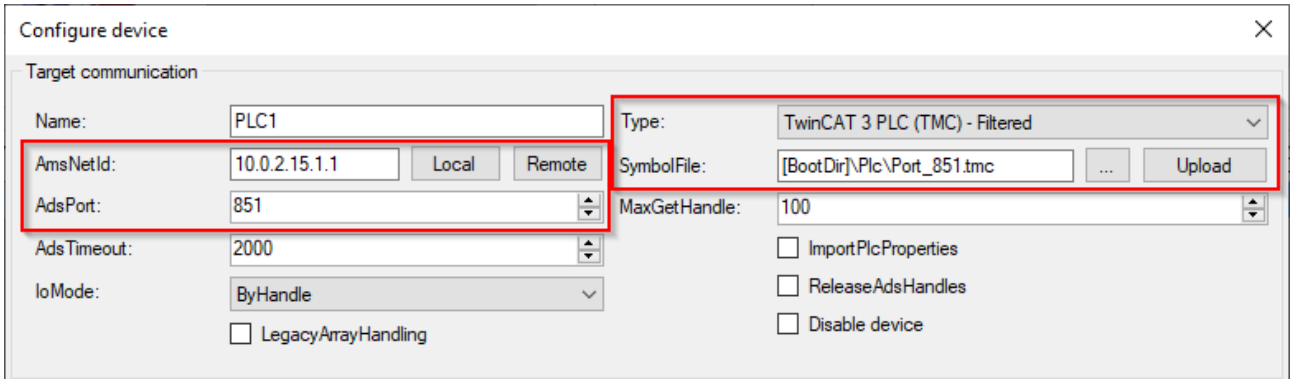
Konfiguration des Servers

Über den TwinCAT OPC UA Configurator können Sie nun den TwinCAT OPC UA Server konfigurieren, so daß er die erzeugte Symboldatei einliest und die SPS Laufzeit als Data Access Gerät in seinem Adressraum zur Verfügung stellt.

Fügen Sie hierzu im TwinCAT OPC UA Configurator in der Registerkarte Data Access ein neues Gerät hinzu. Selektieren Sie die AMS Net ID Ihres Geräts und tragen Sie den ADS Port der SPS Laufzeit ein. Den ADS Port finden Sie in den Eigenschaften des SPS Projekts:



In der Auswahlliste des Felds „Type“ wählen Sie nun den Eintrag „TwinCAT 3 SPS (TMC) – Filtered“ aus. In dem Feld „SymbolFile“ wählen Sie die Symboldatei (TMC) aus, welche nach dem Aktivieren des TwinCAT SPS Projekts erzeugt wurde und nun im Bootverzeichnis liegen sollte. Alternativ können Sie auch über die Online-Symbolik [▶ 69] gehen.



Alle weiteren Einstellungen können Sie auf den Standardwerten belassen.

4.10.3.2 C++

In diesem Abschnitt wird beschrieben, wie Sie den Namensraum des TwinCAT OPC UA Servers konfigurieren, um Zugriff auf die Symbole eines TwinCAT 3 C++ Moduls zu erhalten. Dies erfordert die Durchführung der folgenden Schritte:

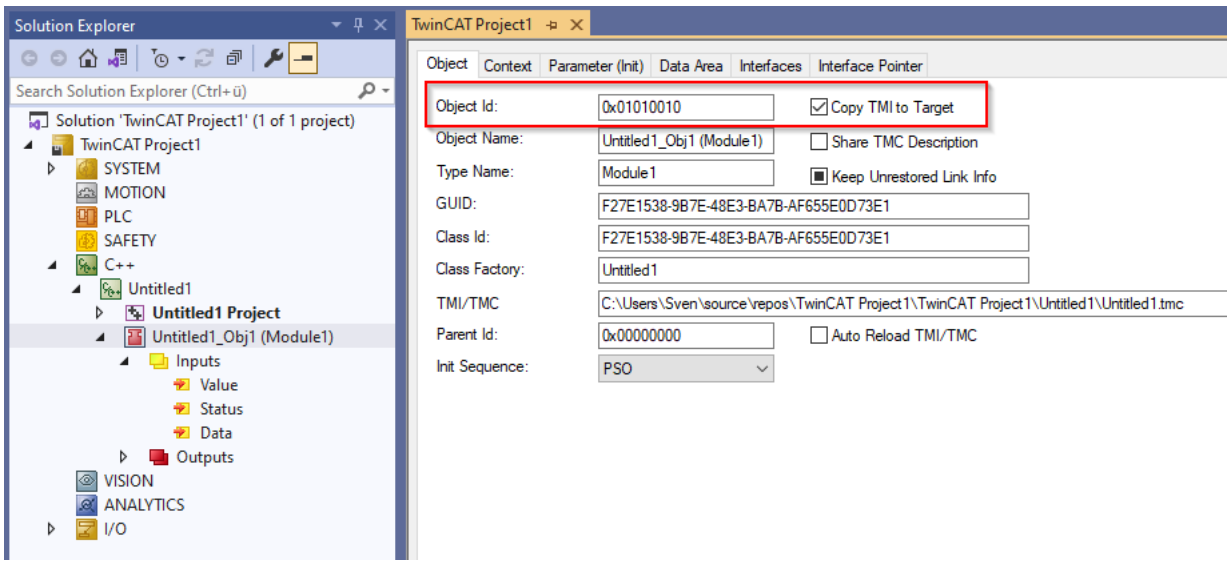
- Aktivieren der Symboldatei
- Freigabe von Symbolen
- Konfiguration des Servers

Diese Schritte sollen nun im Folgenden näher beschrieben werden.

Aktivieren der Symboldatei

Um bestimmte, in einer Instanz eines C++-Moduls enthaltene Symbole so zu konfigurieren, dass sie über OPC UA zugänglich werden, ist es erforderlich, daß Sie zunächst den Download der Symboldatei für die Modulinstanz aktivieren. Hierdurch wird die Symboldatei beim Aktivieren der Konfiguration in das Bootverzeichnis vom Zielgerät kopiert.

Aktivieren Sie den Download der Symboldatei durch Setzen der folgenden Option in den Eigenschaften der Modulinstanz:

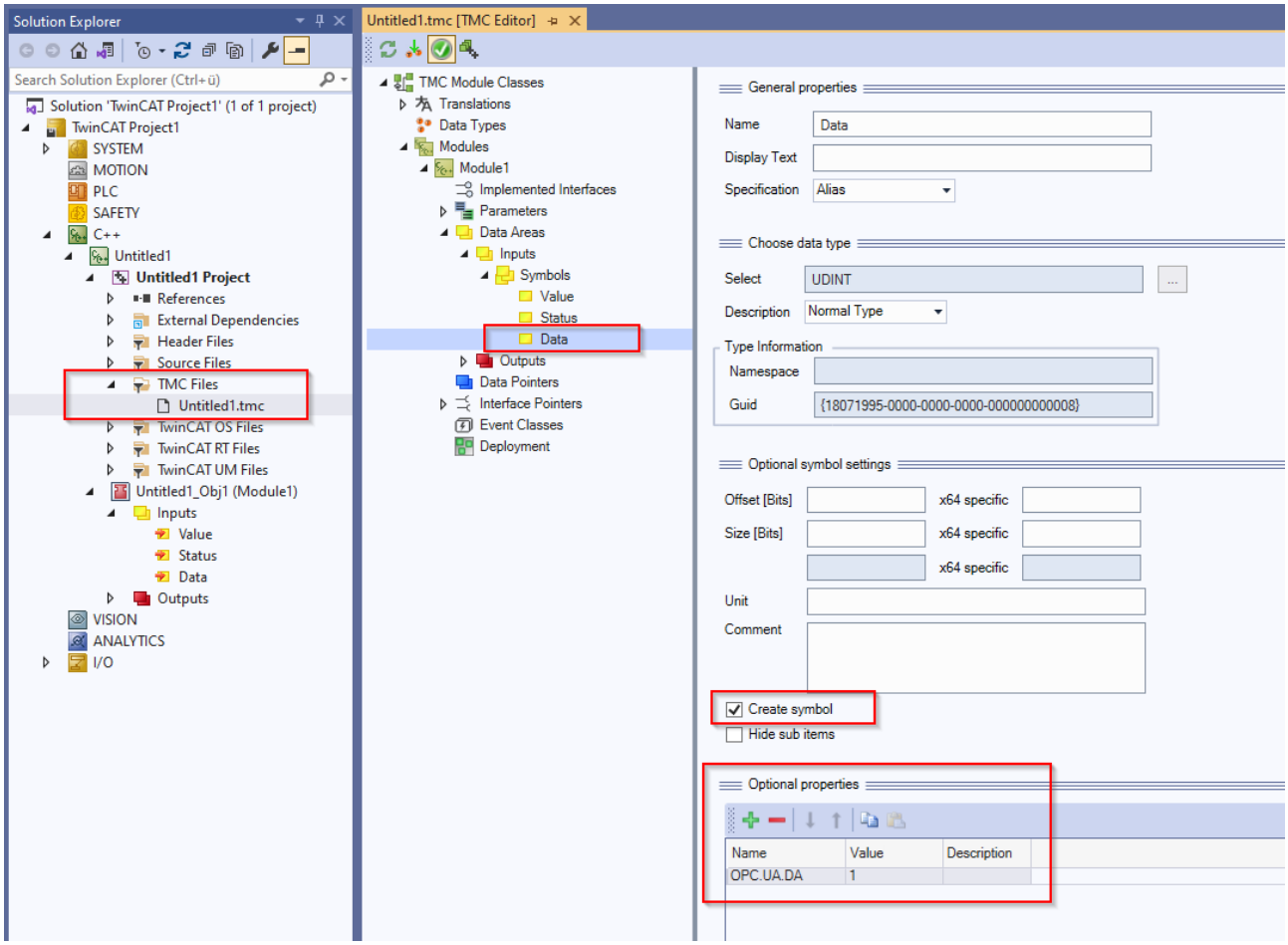


Die erzeugte Symboldatei (TMI) wird immer nach der ObjectId der Modulinstanz benannt nun beim Aktivieren in das Bootverzeichnis vom Zielgerät kopiert, zum Beispiel:

```
%TwinCATInstallDir%\3.1\Boot\Tmi\Obj_01010010.tmi
```

Freigabe von Symbolen

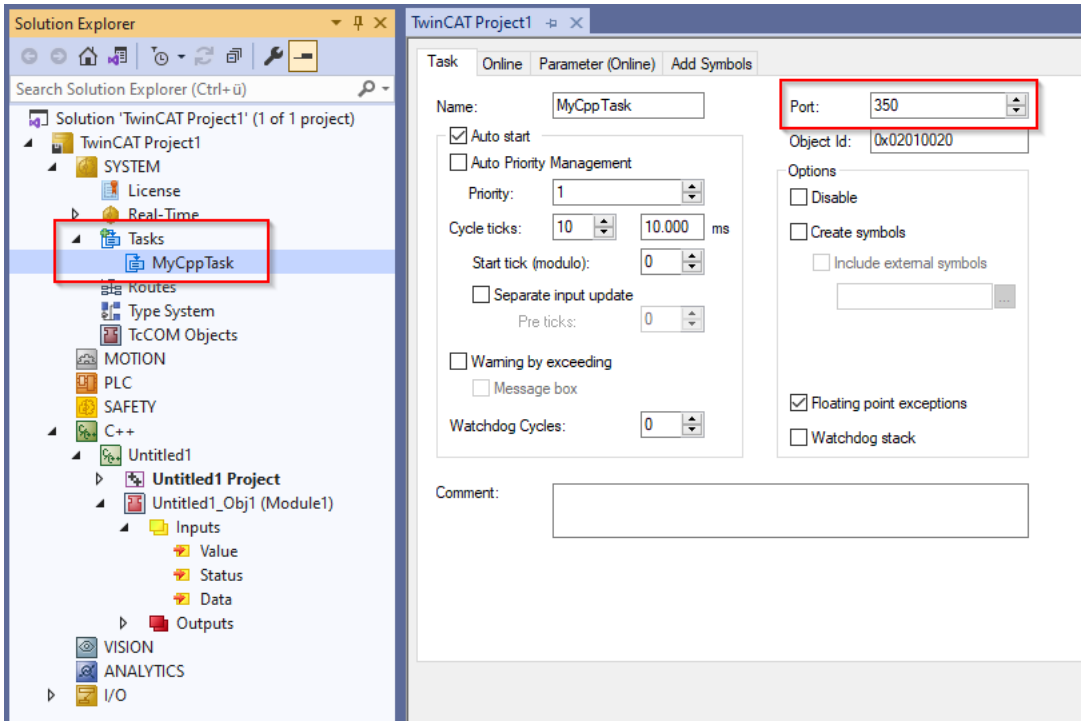
Über den TMC Code Generator können Sie festlegen, welche Symbole über OPC UA freigegeben werden sollen. Hierzu sind zwei Einstellungen notwendig. Aktivieren Sie zunächst an jedem freizugebenden Symbol im TMC Code Generator die Checkbox „Create Symbol“. Fügen Sie anschließend ein optionales Property mit dem Key „OPC.UA.DA“ und dem Value „1“ hinzu.



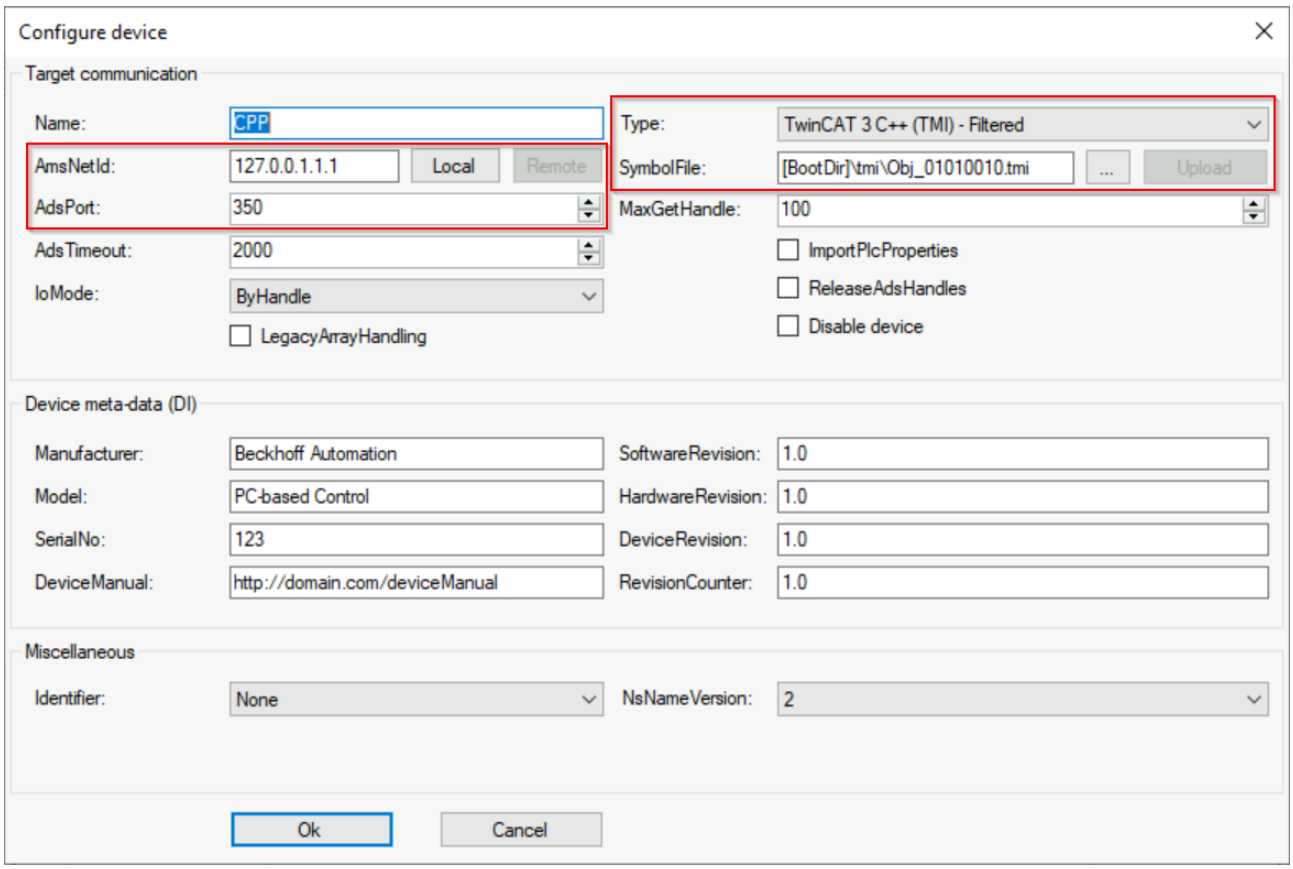
Konfiguration des Servers

Über den TwinCAT OPC UA Configurator können Sie nun den TwinCAT OPC UA Server konfigurieren, so daß er die erzeugte Symboldatei einliest und die C++ Modulinstanz als Data Access Gerät in seinem Adressraum zur Verfügung stellt.

Fügen Sie hierzu im TwinCAT OPC UA Configurator in der Registerkarte Data Access ein neues Gerät hinzu. Selektieren Sie die AMS Net ID Ihres Geräts und tragen Sie den ADS Port der TwinCAT 3 C++ Modulinstanz ein. Den ADS Port finden Sie in den Eigenschaften der mit der C++ Modulinstanz verbundenen Task:



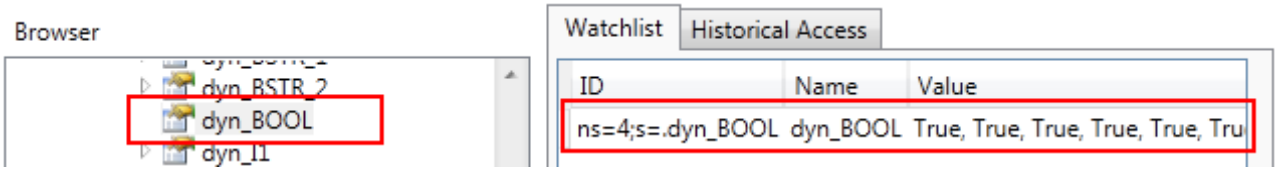
In der Auswahlliste des Felds „Type“ wählen Sie nun den Eintrag „TwinCAT 3 C++ (TMI) - Filtered“ aus. In dem Feld „SymbolFile“ wählen Sie die Symboldatei (TMI) aus, welche nach dem Aktivieren des TwinCAT 3 C++ Projekts erzeugt wurde und nun im Bootverzeichnis liegen sollte. Alternativ können Sie auch über die Online-Symbolik [▶ 69] gehen.



Alle weiteren Einstellungen können Sie auf den Standardwerten belassen.

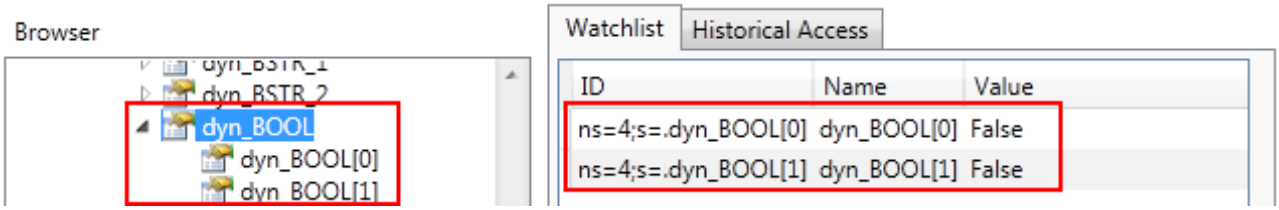
4.10.3.2.1 Arrays

Standardmäßig werden Arrays im UA-Namensraum als einzelner Knoten betrachtet. Dies bedeutet, dass, wenn Sie z. B. in der SPS ein Array dyn_BOOL[10] definieren (und dieses auch für OPC UA freigegeben haben), es anschließend im UA-Namensraum wie folgt auftaucht:



Der Vorteil dieser Vorgehensweise ist eine deutliche Reduzierung der Komplexität des UA-Namensraums und des Speicherverbrauchs, da nicht jede Position eines Arrays als einzelner Knoten im Namensraum zur Verfügung gestellt werden muss. Moderne UA Clients können jedoch weiterhin auf die einzelnen Array-Positionen über den sogenannten „RangeOffset“ zugreifen.

Damit jedoch auch ältere UA Clients unterstützt werden, welche dieses Feature nicht bieten, können Sie die Positionen eines Arrays auch als einzelne Knoten im UA-Namensraum verfügbar machen. Dies stellt sich wie folgt dar:



Diese Einstellung ist durch Aktivierung der Option **Legacy Array Handling** im UA-Konfigurator innerhalb der jeweiligen Namensraum-Konfiguration verfügbar.

Je nach Umfang des SPS-Projekts gewinnt der UA-Namensraum hierdurch deutlich an Komplexität, was sich wiederum in einer erhöhten Speicherauslastung des UA Servers widerspiegelt.

Eine Änderung der oben genannten Einstellungen wird erst aktiv, wenn Sie den UA Server neu starten.

4.10.3.3 Matlab®/Simulink®

In diesem Abschnitt wird beschrieben, wie Sie den Namensraum des TwinCAT OPC UA Servers konfigurieren, um Zugriff auf die Symbole eines TwinCAT 3 Matlab®/Simulink® Moduls zu erhalten. Dies erfordert die Durchführung der folgenden Schritte:

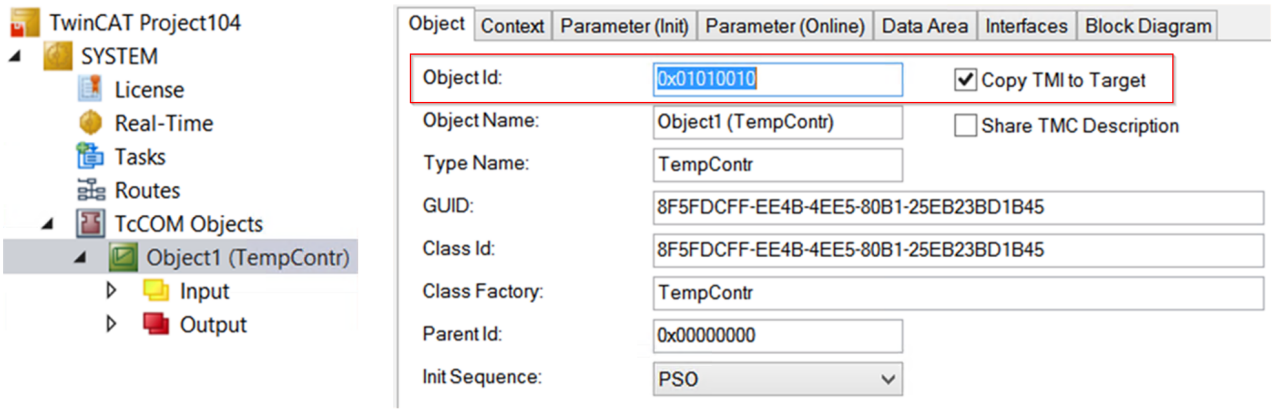
- Aktivieren der Symboldatei
- Freigabe von Symbolen
- Konfiguration des Servers

Diese Schritte sollen nun im Folgenden näher beschrieben werden.

Aktivieren der Symboldatei

Um bestimmte, in einer Instanz eines C++-Moduls enthaltene Symbole so zu konfigurieren, dass sie über OPC UA zugänglich werden, ist es erforderlich, dass Sie zunächst den Download der Symboldatei für die Modulinstanz aktivieren. Hierdurch wird die Symboldatei beim Aktivieren der Konfiguration in das Bootverzeichnis vom Zielgerät kopiert.

Aktivieren Sie den Download der Symboldatei durch Setzen der folgenden Option in den Eigenschaften der Modulinstanz:



Die erzeugte Symboldatei (TMI) wird immer nach der ObjectId der Modulinstanz benannt nun beim Aktivieren in das Bootverzeichnis vom Zielgerät kopiert, zum Beispiel:

```
%TwinCATInstallDir%\3.1\Boot\Tmi\Obj_01010010.tmi
```

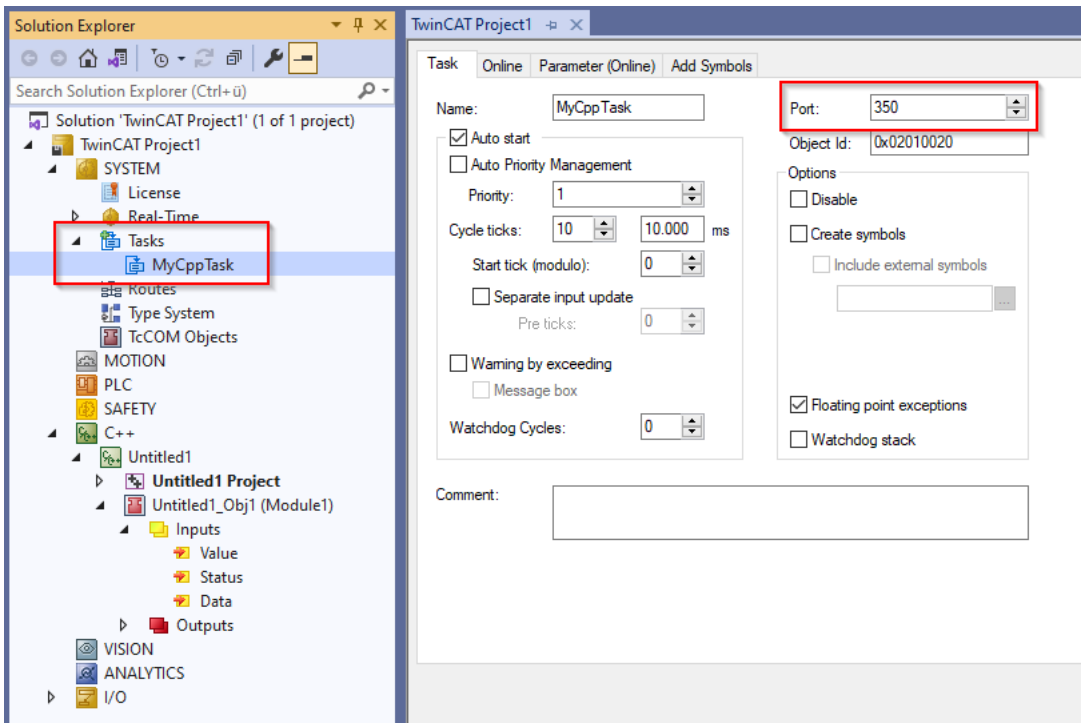
Freigabe von Symbolen

Das Produkt TE1400 Target for Simulink® bietet diverse Einstellmöglichkeiten zur Selektion von Variablen, die über OPC UA freigegeben werden sollen. Diese Einstellungen werden in der entsprechenden TE1400 Produktdokumentation beschrieben.

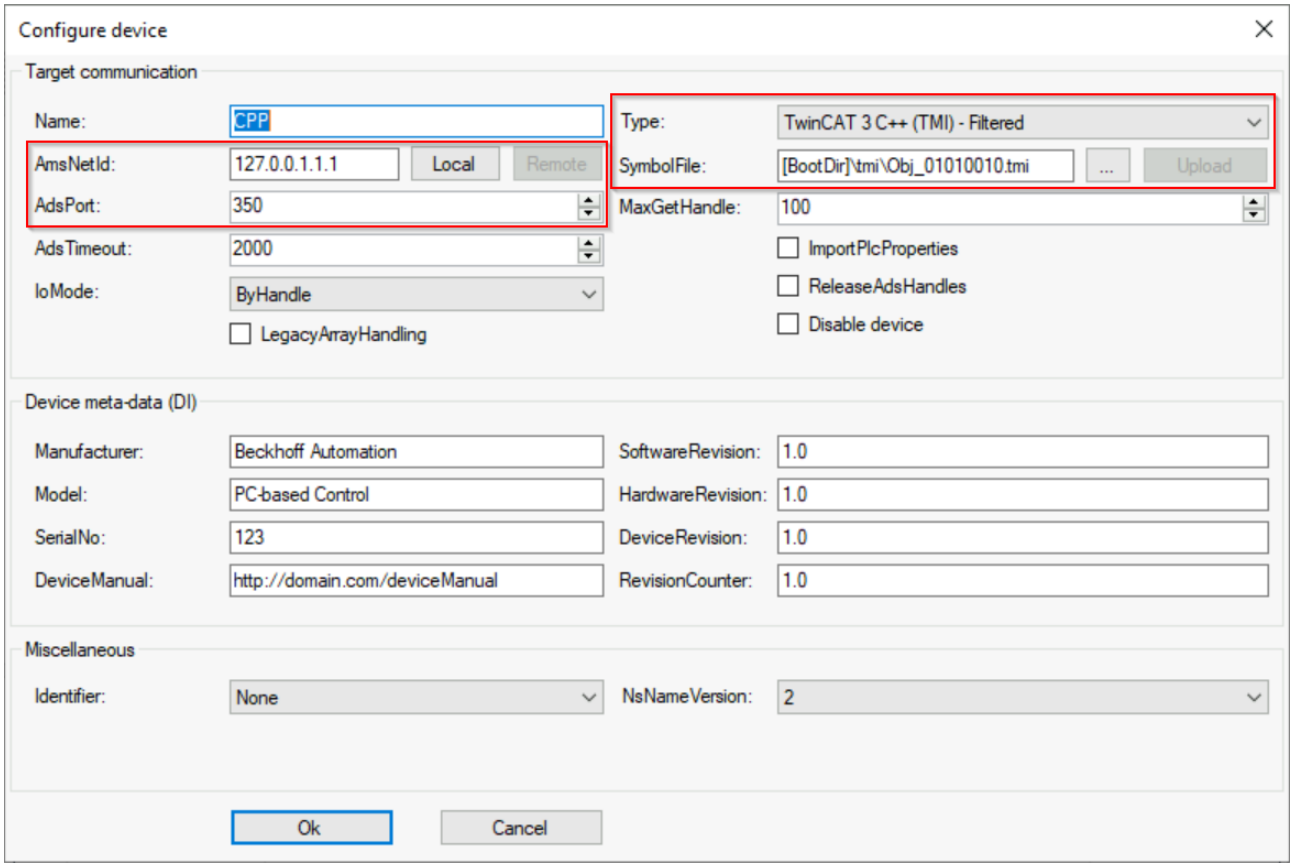
Konfiguration des Servers

Über den TwinCAT OPC UA Configurator können Sie nun den TwinCAT OPC UA Server konfigurieren, so daß er die erzeugte Symboldatei einliest und die Matlab®/Simulink® Modulinstanz als Data Access Gerät in seinem Adressraum zur Verfügung stellt.

Fügen Sie hierzu im TwinCAT OPC UA Configurator in der Registerkarte Data Access ein neues Gerät hinzu. Selektieren Sie die AMS Net ID Ihres Geräts und tragen Sie den ADS Port der TwinCAT 3 Matlab®/Simulink® Modulinstanz ein. Den ADS Port finden Sie in den Eigenschaften der mit der Modulinstanz verbundenen Task:



In der Auswahlliste des Felds „Type“ wählen Sie nun den Eintrag „TwinCAT 3 C++ (TMI) - Filtered“ aus. In dem Feld „SymbolFile“ wählen Sie die Symboldatei (TMI) aus, welche nach dem Aktivieren des TwinCAT 3 C++ Projekts erzeugt wurde und nun im Bootverzeichnis liegen sollte. Alternativ können Sie auch über die Online-Symbolik [▶ 69] gehen.



Alle weiteren Einstellungen können Sie auf den Standardwerten belassen.

4.10.3.4 EtherCAT Master

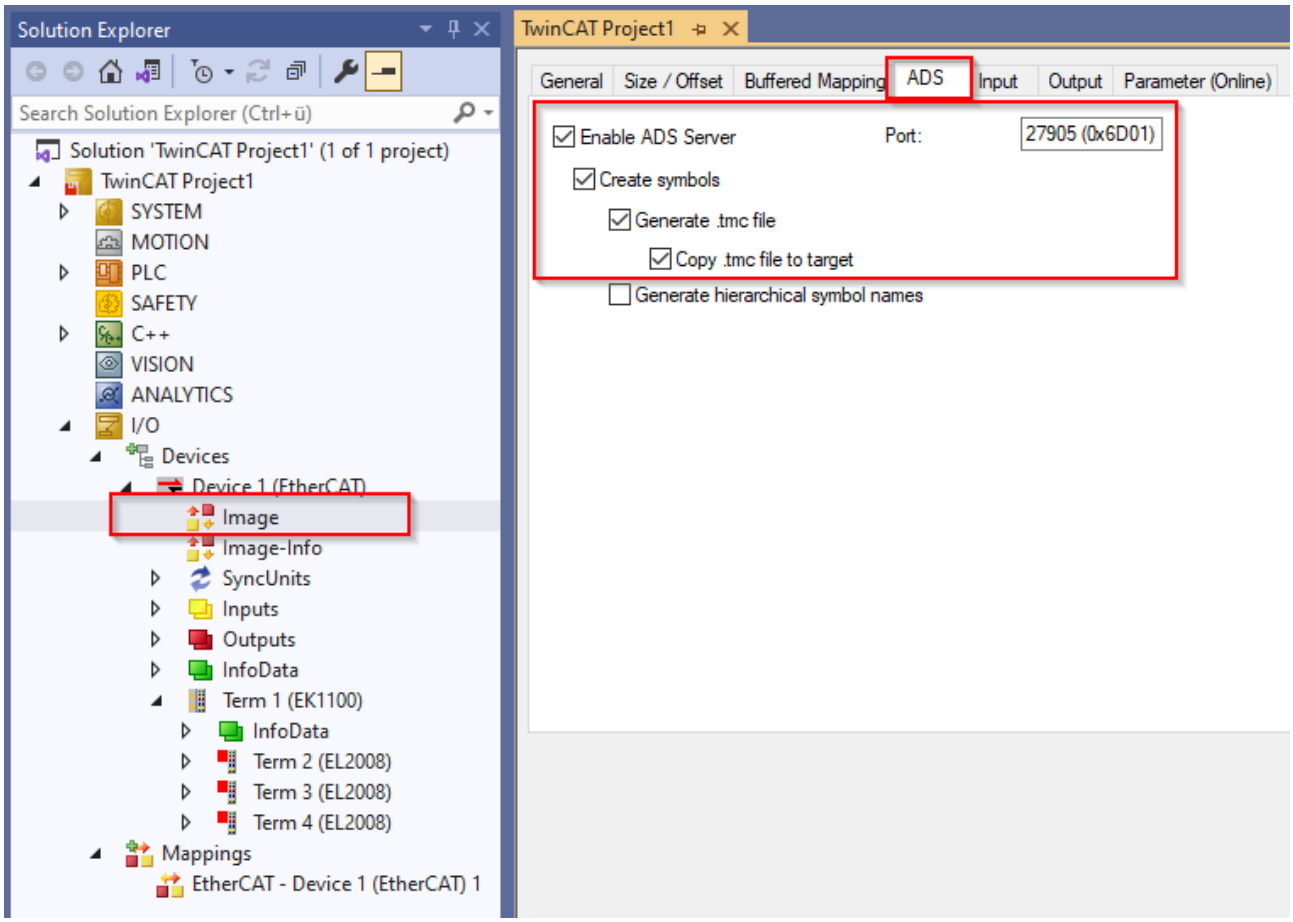
In diesem Abschnitt wird beschrieben, wie Sie den Namensraum des TwinCAT OPC UA Servers konfigurieren, um Zugriff auf die Symbole eines EtherCAT Masters zu erhalten. Dies erfordert die Durchführung der folgenden Schritte:

- Aktivieren der Symboldatei
- Konfiguration des Servers
- [optional] Ändern des Namensraum-Aufbaus

Diese Schritte sollen nun im Folgenden näher beschrieben werden.

Aktivieren der Symboldatei

Um die Symbole eines EtherCAT Masters über OPC UA verfügbar zu machen, ist es erforderlich, daß Sie zunächst den ADS-Server im Image des EtherCAT Masters, sowie die Erzeugung der Symbolik aktivieren. Hierdurch wird ein Zugriff auf die Symbole über ADS ermöglicht. In demselben Dialog finden Sie auch den ADS-Port, welcher im weiteren Verlauf der Konfiguration noch benötigt wird.



Durch das Aktivieren der Parameter „Generate .tmc file“ und „Copy .tmc file to target“ wird eine Symboldatei erzeugt und beim Aktivieren der Konfiguration automatisch auf das Zielgerät heruntergeladen. Diese befindet sich dann im Bootverzeichnis von TwinCAT und trägt zur einfachen Identifizierung den ADS-Port im Dateinamen, zum Beispiel:

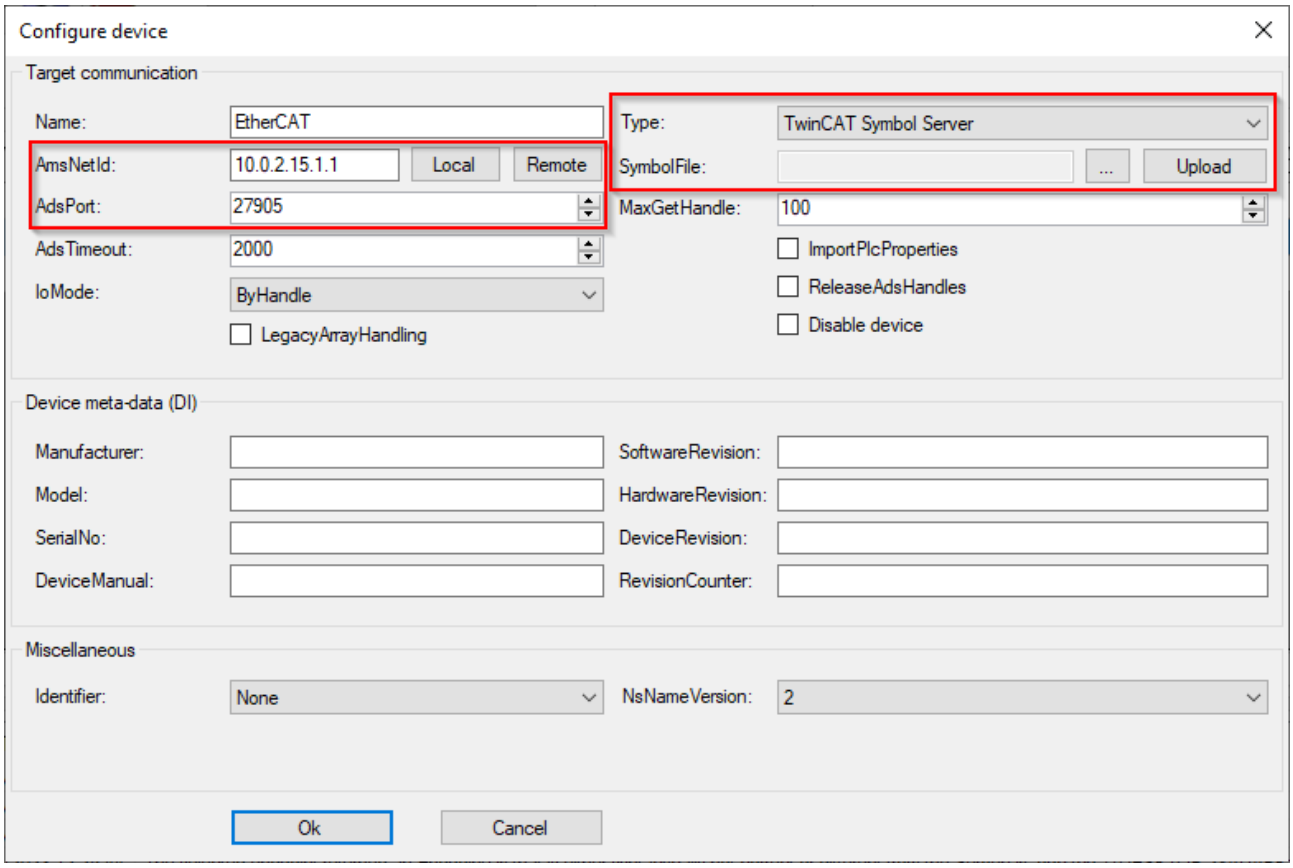
```
%TwinCATInstallDir%\3.1\Boot\Io\Port_27905.tmc
```

Konfiguration des Servers

Über den TwinCAT OPC UA Configurator können Sie nun den TwinCAT OPC UA Server konfigurieren, so daß er die erzeugte Symboldatei einliest und den EtherCAT Master als Data Access Gerät in seinem Adressraum zur Verfügung stellt.

Fügen Sie im TwinCAT OPC UA Configurator in der Registerkarte Data Access ein neues Gerät hinzu. Selektieren Sie die AMS Net ID Ihres Gerätes und tragen Sie den ADS Port des EtherCAT Master Images (siehe oben) ein.

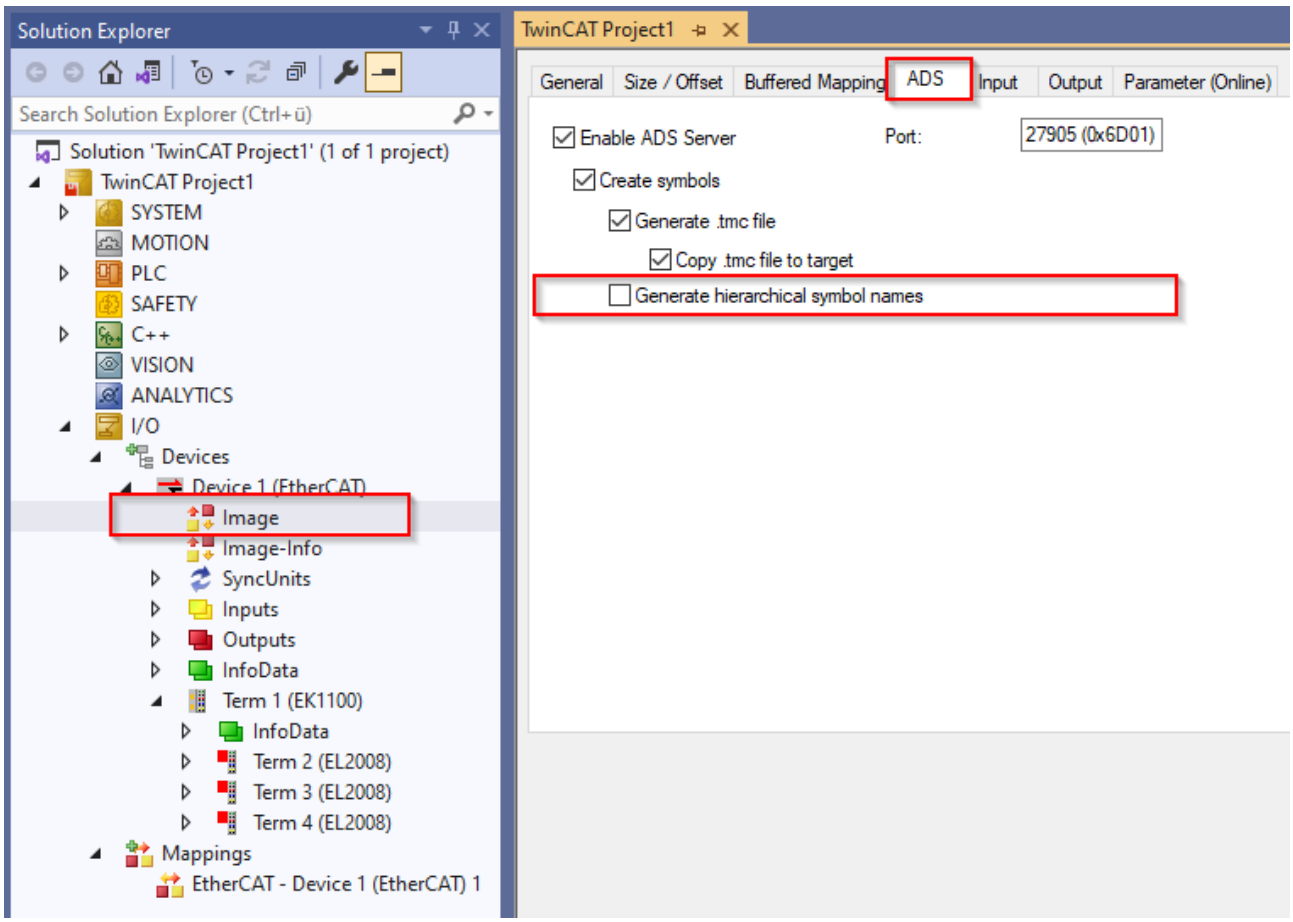
In der Auswahlliste des Felds „Type“ wählen Sie nun den Eintrag „EtherCAT Master (TMC)“ aus. In dem Feld „SymbolFile“ wählen Sie die Symboldatei (TMC) aus, welche nach dem Aktivieren des TwinCAT Projekts erzeugt wurde und nun im Bootverzeichnis liegen sollte. Alternativ können Sie auch die Online-Symbolik [▶ 69] verwenden.



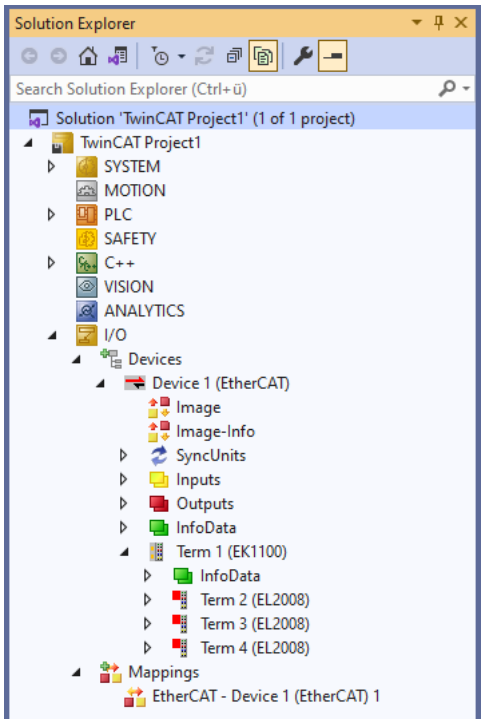
Alle weiteren Einstellungen können Sie auf den Standardwerten belassen.

Ändern des Namensraum-Aufbaus

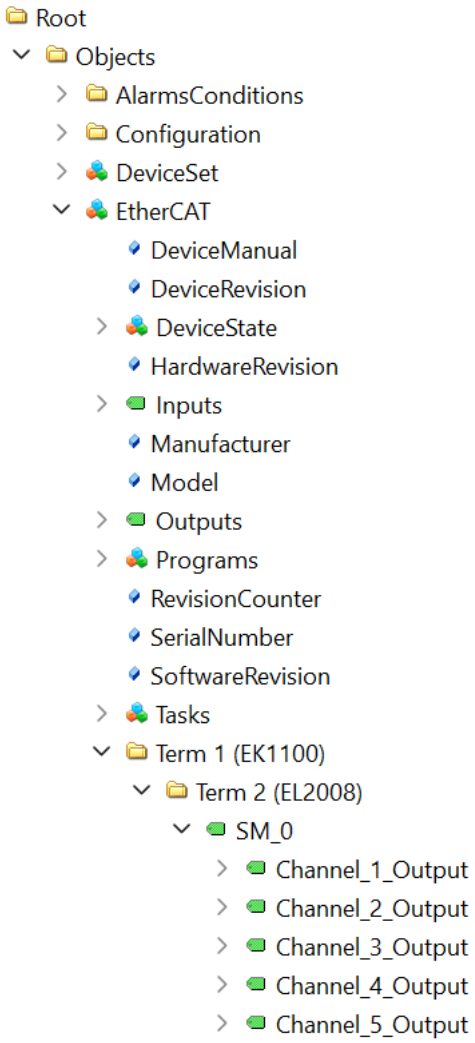
Der Parameter „Generate hierarchical symbol names“ ermöglicht einen anderen Aufbau des Symbolraums, was auch direkte Auswirkungen auf den Namensraum des TwinCAT OPC UA Servers hat.



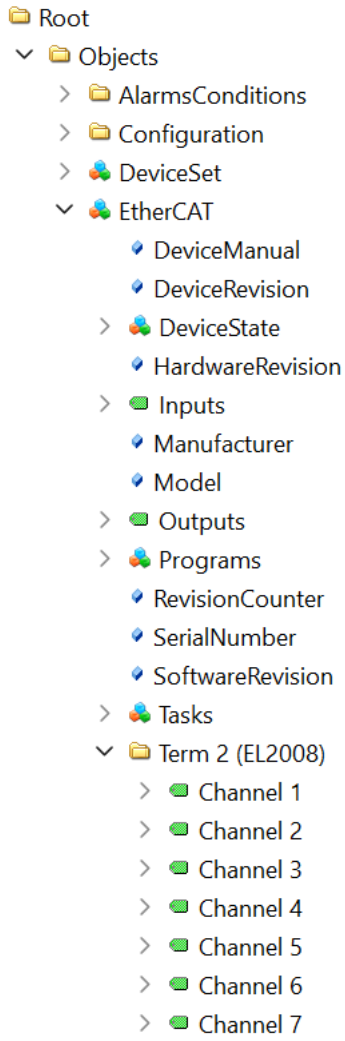
Als Beispiel gehen wir einmal von der folgenden EtherCAT Master Konfiguration im TwinCAT XAE aus:



Ist die genannte Option aktiviert, so spiegelt der Namensraum den Aufbau im TwinCAT XAE wider:



Ist die genannte Option deaktiviert, so wird der Namensraum „flacher“ aufgebaut:



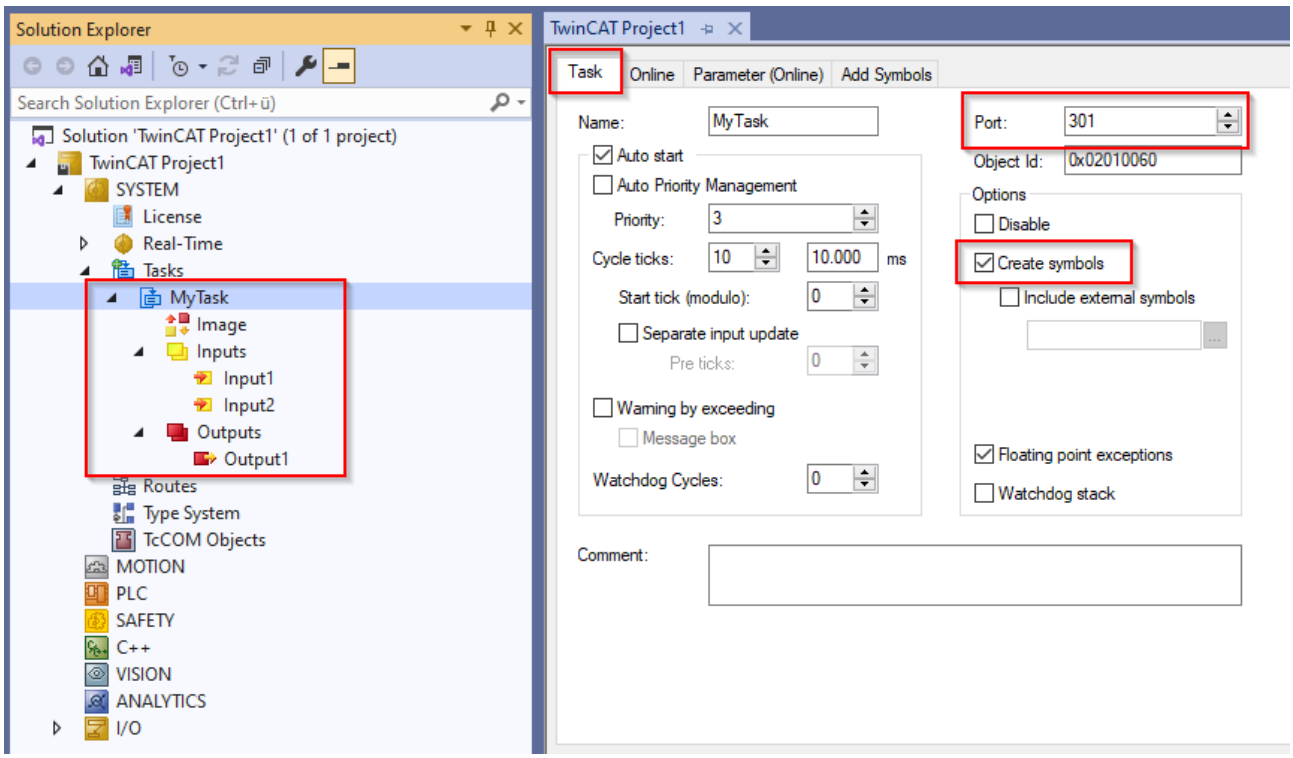
4.10.3.5 I/O Task

In diesem Abschnitt wird beschrieben, wie Sie den Namensraum des TwinCAT OPC UA Servers konfigurieren, um Zugriff auf die Symbole einer Task mit Prozessabbild zu erhalten. Dies erfordert die Durchführung der folgenden Schritte:

- Aktivieren der Symbolik
- Konfiguration des Servers

Aktivieren der Symbolik

Sie können die Variablen einer Task mit Prozessabbild über OPC UA bereitstellen. Hierzu ist es erforderlich, daß Sie in den Einstellungen der Task die Option „Create symbols“ aktivieren. Hierdurch werden Symbole erzeugt und über die Online-Symbolik [▶ 69] bereitgestellt. In demselben Dialog finden Sie auch den ADS-Port, welcher im weiteren Verlauf der Konfiguration noch benötigt wird.

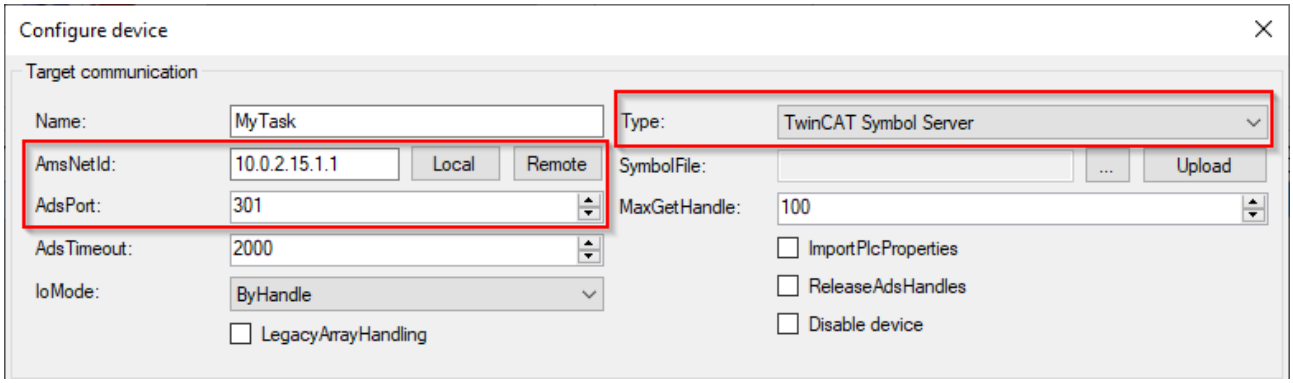


Konfiguration des Servers

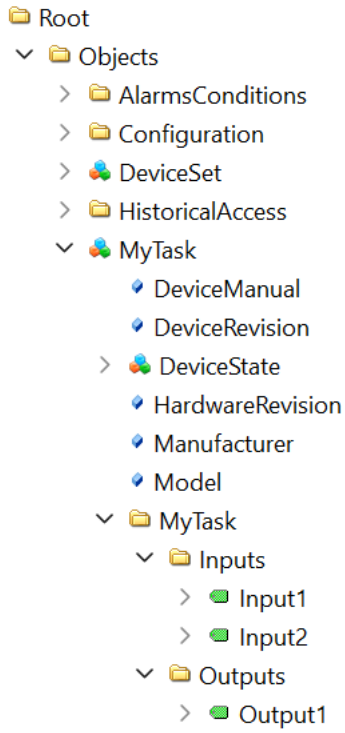
Über den TwinCAT OPC UA Configurator können Sie nun den TwinCAT OPC UA Server konfigurieren, so daß er auf die Online-Symbolik der Task zugreift und diese in seinem Adressraum zur Verfügung stellt.

Fügen Sie im TwinCAT OPC UA Configurator in der Registerkarte Data Access ein neues Gerät hinzu. Selektieren Sie die AMS Net ID Ihres Gerätes und tragen Sie den ADS Port der Task (siehe oben) ein.

In der Auswahlliste des Felds „Type“ wählen Sie nun den Eintrag „TwinCAT Symbol Server“ aus.



Nach dem Aktivieren des TwinCAT Projekts steht die Symbolik bereit und der TwinCAT OPC UA Server kann diese einlesen und seinen Adressraum entsprechend aufbauen.



4.10.3.6 Online-Symbolik

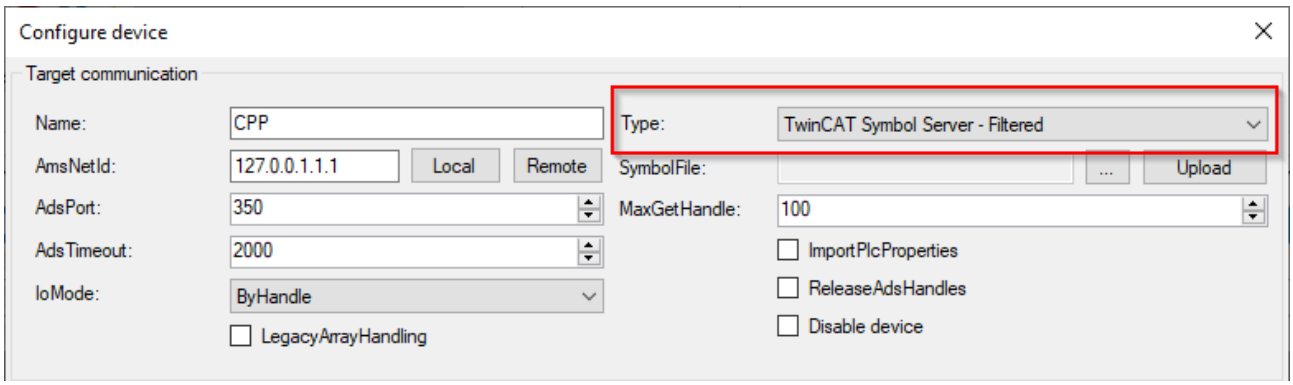
Der TwinCAT OPC UA Server ermöglicht das Auslesen von Symbolinformationen über die sogenannte „Online-Symbolik“. Hierbei handelt es sich um eine standardisierte ADS-Schnittstelle, welche von einigen ADS-Servern im TwinCAT angeboten wird. Oftmals wird diese Schnittstelle auch als „Symbol Upload“ bezeichnet. Die Schnittstelle ermöglicht es einem ADS-Client alle vorhandenen Symbole des ADS-Servers auszulesen und ggf. weiter zu verarbeiten. Zusätzlich zu den Adressinformationen der Symbole werden auch weitere Informationen mitgeliefert, zum Beispiel Datentypbeschreibungen und optionale Attribute.

Über die optionalen Attribute kann eine explizite Freigabe von Symbolen erfolgen oder weitere Einstellungen hierzu getätigt werden – die Handhabung ist hierbei identisch zur Verwendung von Symboldateien und läuft zum Beispiel über Pragmas in der SPS oder dem TMC Code Generator für TwinCAT 3 C++. D.h. daß Sie alle Freigaben, die sie dort gesetzt haben, sowohl über den Symboldateien-Import als auch die Online-Symbolik verwenden können. Bitte beachten Sie, daß nicht alle Echtzeitumgebungen eine explizite Freigabe von Symbolen ermöglichen. In dem Fall werden immer alle Symbole bereitgestellt. Die folgende Tabelle gibt einen Überblick über einige ADS-Server, welche die Schnittstelle zur Online-Symbolik anbieten und ob eine explizite Symbol-Freigabe zur Verfügung steht.

Typ	ADS-Port	Symbol-Freigabe
TwinCAT 2 SPS	801, 811, 821, ...	Ja
TwinCAT 3 SPS	851, 852, 853, ...	Ja
TwinCAT 3 C++ / Matlab®/Simulink®	350, 351, 352, ...	Ja
TwinCAT 3 EtherCAT Master	27905, 27906, ...	Nein (alle Symbole werden freigegeben)
TwinCAT 3 I/O Task with Image	301, 302, 303, ...	Nein (alle Symbole werden freigegeben)

Im TwinCAT OPC UA Configurator steht der Mechanismus der Online-Symbolik an der Konfiguration eines Data Access Geräts in zweierlei Form zur Verfügung:

- TwinCAT Symbol Server: Alle Symbole werden importiert
- TwinCAT Symbol Server – Filtered: Nur freigegebene Symbole werden importiert



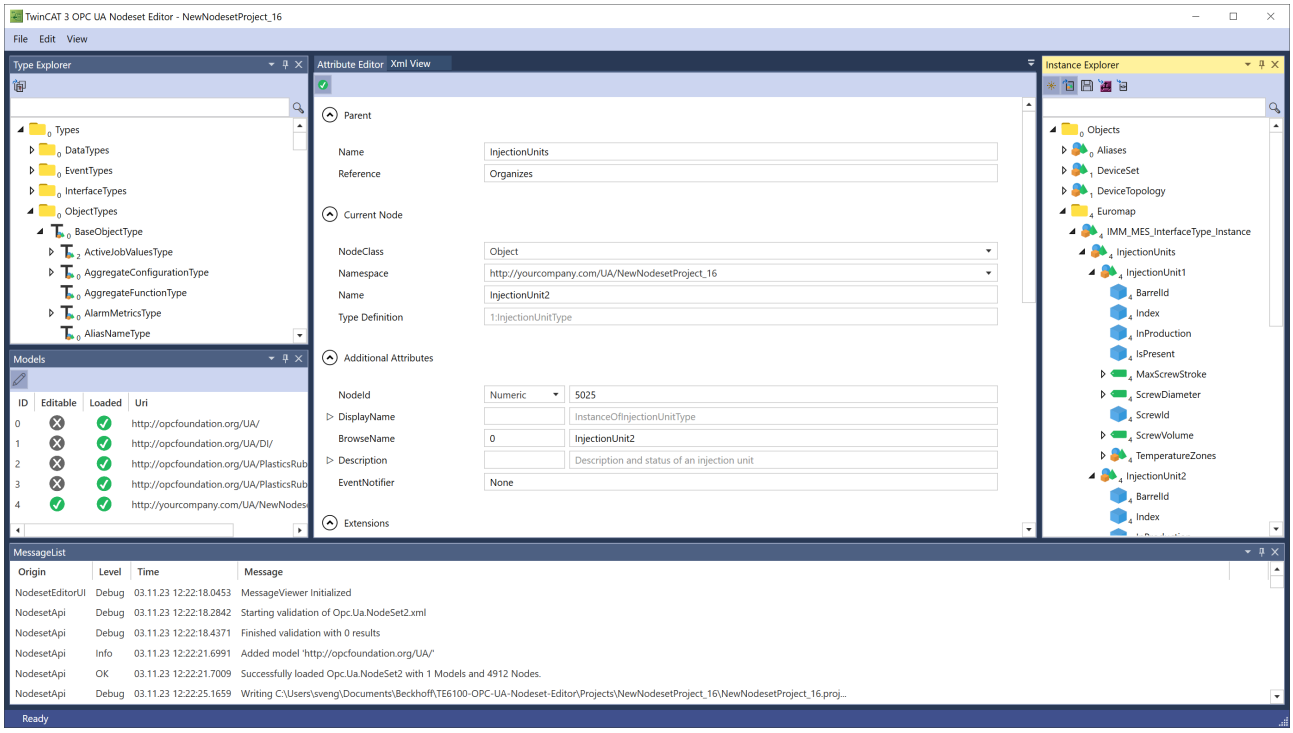
Beim Aufbau seines Namensraums orientiert sich der TwinCAT OPC UA Server am Namensraum der ADS-Symbolik. So wie die Daten hier vorliegen, werden Sie auch über OPC UA bereitgestellt.

4.10.4 Nodesets

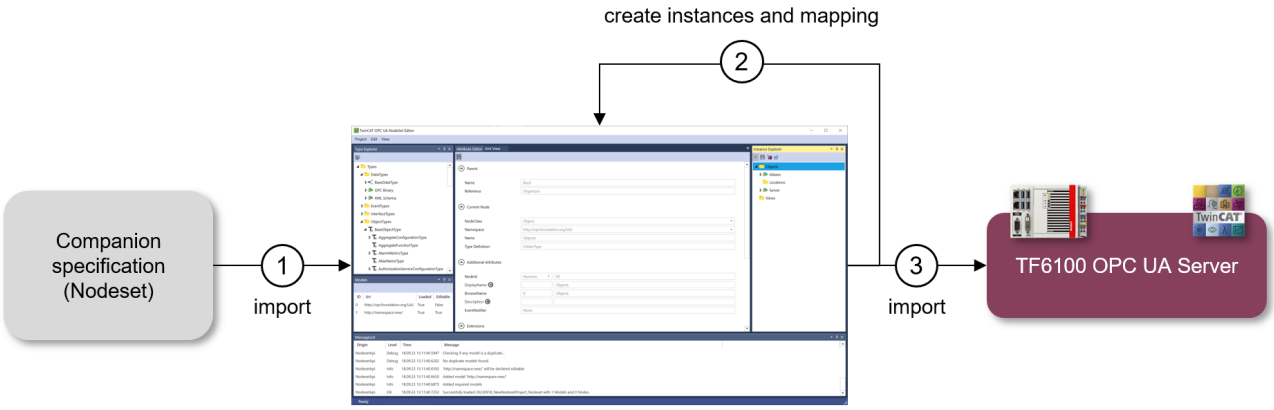
Der TwinCAT OPC UA Server erlaubt den Import von OPC UA Nodeset Dateien. Ein Nodeset kann hierbei, zum Beispiel im Rahmen einer Companion Spezifikation, Datentypen und Instanzen definieren, welche innerhalb des Nodesets in einer (durch die OPC Foundation standardisierten) XML-Struktur vorliegen.

Mit Hilfe der Software TE6100 OPC UA Nodeset Editor können solche Nodeset-Dateien erstellt, modifiziert und mit Symbolen aus einer TwinCAT-Echtzeitumgebung verknüpft werden. Das Nodeset kann anschließend in den TwinCAT OPC UA Server importiert und dort verwendet werden.

Weitere Informationen zu diesem Thema finden Sie in der Produktdokumentation vom TE6100 OPC UA Nodeset Editor.



Der folgende Screenshot zeigt exemplarisch den Workflow zur Nutzung beider TwinCAT Functions.



TE6100 TC3 OPC UA Nodeset Editor

4.10.5 Datentypen

Die IEC 61131-3-Datentypen werden nach Spezifikation des PLCopen-Mappings auf OPC UA-Datentypen gemappt. Die folgende Tabelle zeigt das spezifizierte Mapping.

SPS	OPC UA
BOOL	Boolean
SINT	SByte
INT	Int16
DINT	Int32
LINT	Int64
USINT	Byte
UINT	UInt16
UDINT	UInt32
ULINT	UInt64
REAL	Float
LREAL	Double
TIME	Int64
LTIME	Int64

TIME/LTIME

Bei den Datentypen TIME und LTIME gibt es bezüglich des Mappings zwischen SPS- und OPC UA-Datentyp Besonderheiten zu beachten, siehe nachfolgenden Abschnitt.

TIME und LTIME

Im PLCopen-Mapping der IEC 61131-3 sind sowohl TIME als auch LTIME auf den Datentyp Int64 gemappt. Die Datenbereiche der IEC61131-3-Datentypen UDINT und ULINT entsprechen dagegen UInt32 und UInt64.

Datentyp	Untergrenze	Obergrenze	Abbildung in Zeit
TIME (UDINT)	0	4294967295	49d17h2m47s295ms
LTIME (ULINT)	0	18446744073709551615	213503d23h34m33s709ms551us615ns
Int64	-9223372036854775808	9223372036854775807	-

Das Verhalten des Servers im Umgang mit diesen beiden Datentypen ist in der folgenden Auflistung erläutert:

TIME

- Der mögliche Wertebereich ist in der Tabelle abzulesen.

- Für alle anderen Werte (<0 und >4294967295) gibt der Server beim Schreiben durch einen OPC UA Client den Statuscode **BadOutOfRange** zurück.

LTIME

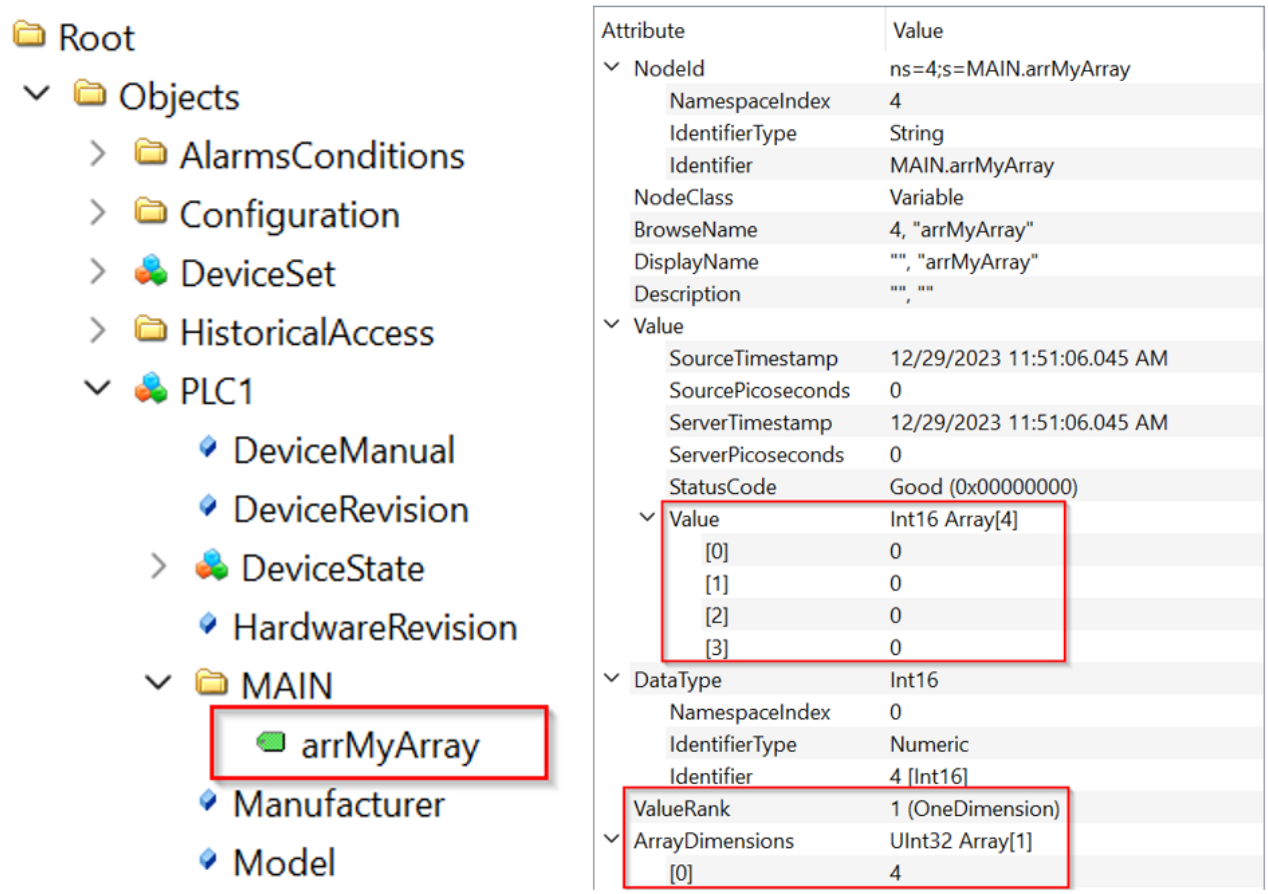
- Der mögliche Wertebereich ist ebenfalls in der Tabelle abzulesen.
- Für alle Werte <0 gibt der Server beim Schreiben durch einen OPC UA Client den Statuscode **BadOutOfRange** zurück.
- Werte >9223372036854775807 können mit dem Wertebereich von Int64 nicht abgebildet und damit auch nicht über OPC UA geschrieben werden. Sie können aber in der SPS geschrieben werden. In diesem Fall wird der Wert der betroffenen OPC UA Variablen im Server **Null.** Zusätzlich wird der StatusCode der Variablen auf **BadOutOfRange** gesetzt.

4.10.6 Arrays

Standardmäßig werden Arrays im Adressraum des Servers als einzelne Node abgebildet. Ein OPC UA Client hat dennoch die Möglichkeit, auf Einzelelemente des Arrays zuzugreifen.

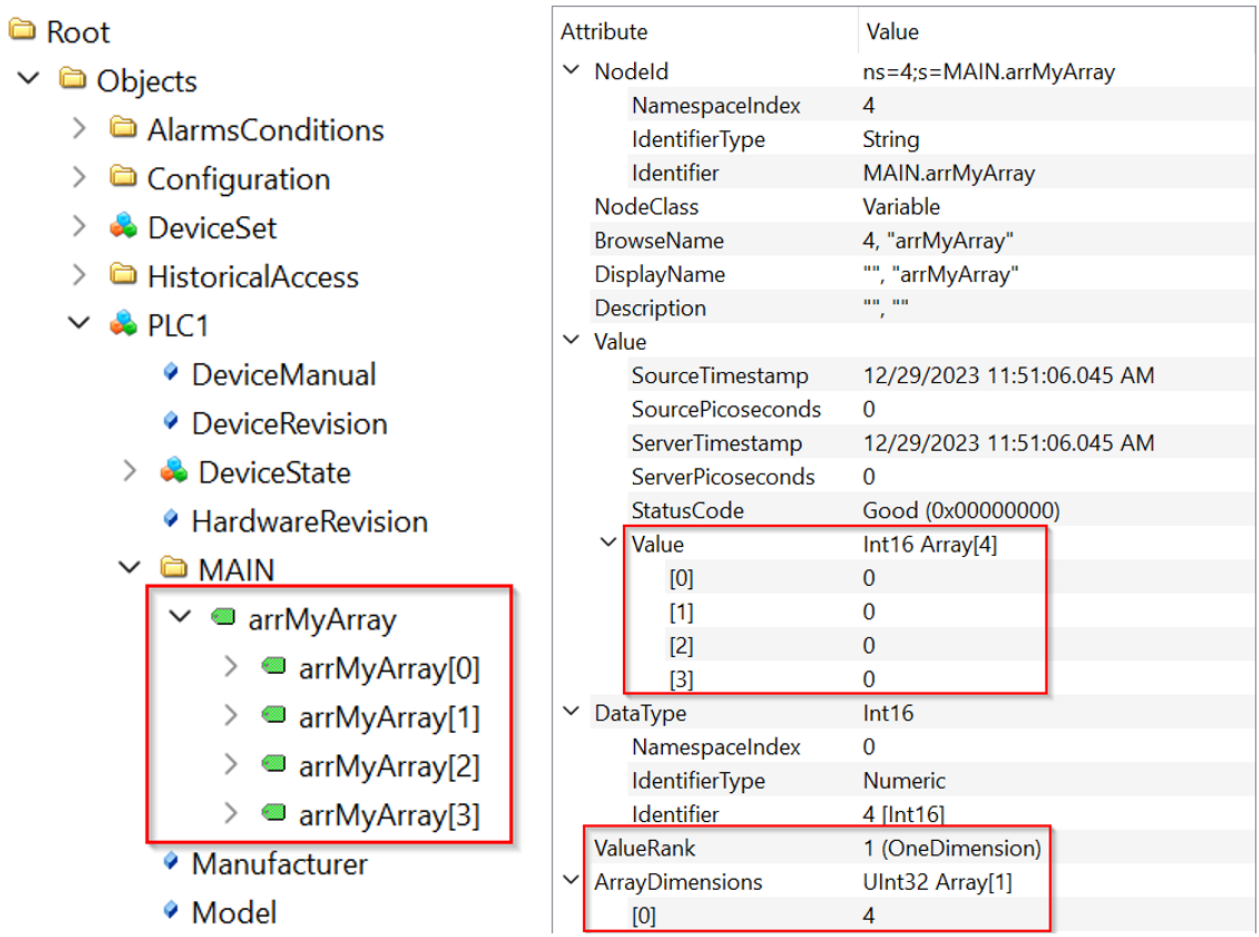
Das folgende Array in der SPS würde somit als einzelne Node dargestellt werden, wobei über die Node-Attribute die Arraygrenzen definiert würden.

```
{attribute 'OPC.UA.DA' := '1'}
arrMyArray : ARRAY[0..3] OF INT;
```

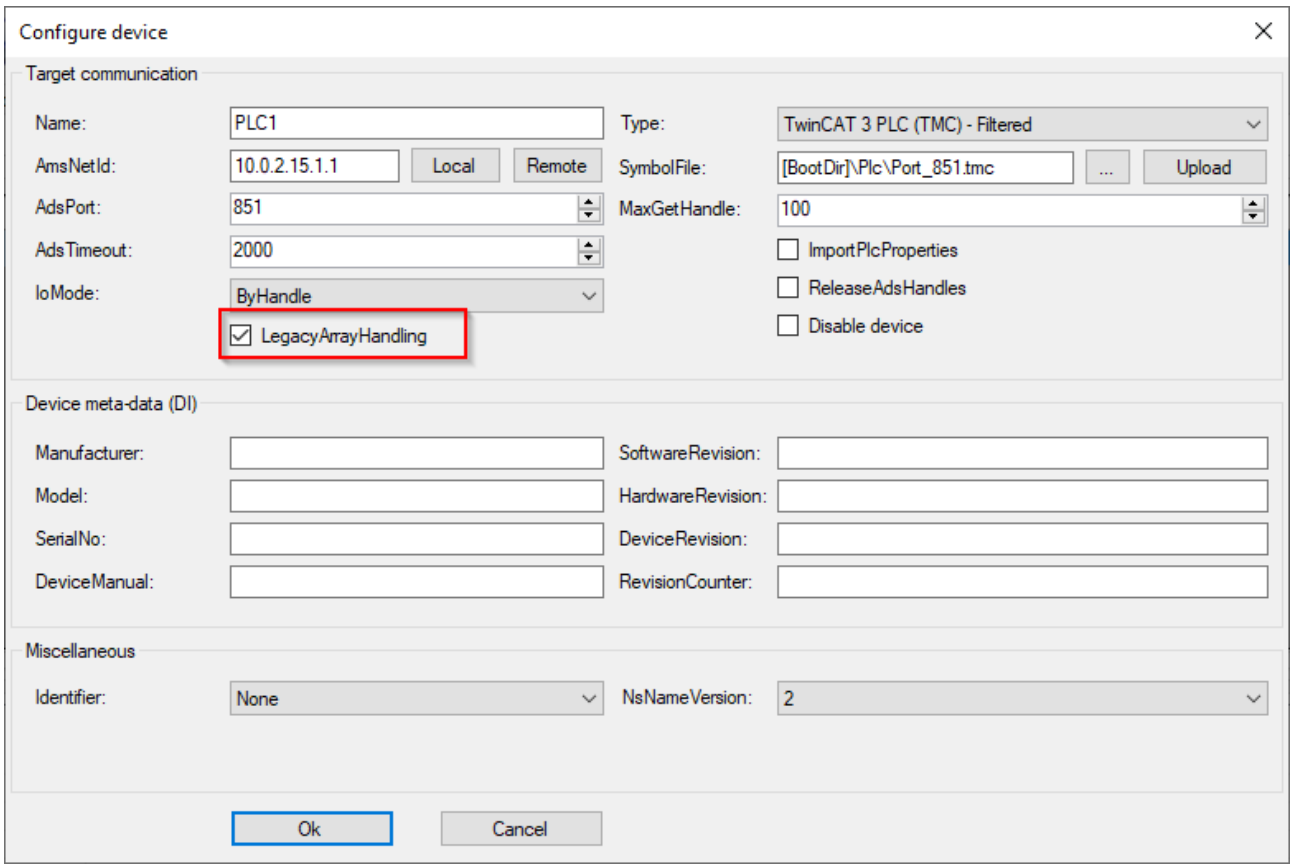


Der Vorteil ist eine deutliche Reduzierung der Komplexität des Server-Adressraums und des Speicherverbrauchs, da nicht jede Position eines Arrays als einzelner Knoten im Namensraum zur Verfügung gestellt werden muss.

Ältere OPC UA Clients unterstützen diese Funktion gegebenenfalls noch nicht. Daher ermöglicht ein spezieller Konfigurationsschalter die Bereitstellung aller Einzelelemente eines Arrays als einzelne Knoten im Namensraum. Dies stellt sich wie folgt dar:



Zusätzlich zu dem Rootelement des Arrays sind auch alle Einzelelemente als separate Nodes verfügbar. Dieser Konfigurationsschalter ist durch Aktivierung der Option „LegacyArrayHandling“ an einem Data-Access-Gerät im TwinCAT OPC UA Configurator verfügbar.



Erhöhte Speicheranforderungen

Durch Aktivierung dieses Konfigurationsschalters vergrößert sich der Adressraum des Servers deutlich. Je nach Umfang des Projekts und gerade auch bei Verwendung von verschachtelten, komplexen Arrays, hat dies eine erhöhte Speicherauslastung des TwinCAT OPC UA Servers zur Folge.
Stellen Sie sicher, dass Ihr Hardwaregerät genügend Arbeitsspeicher aufweist.

4.10.7 Enums

Aufzählungen in OPC UA haben immer den Datentyp Int32. Die Norm IEC 61131-3 ermöglicht jedoch die Definition größerer Datentypen als Int32. Damit diese Aufzählungen ordnungsgemäß behandelt werden, bietet der TwinCAT OPC UA Server die Konfigurationsoption *<ImportBigEnumsNumeric>*, die in seiner Konfigurationsdatei Data Access aktiviert werden kann.

Diese Option ist standardmäßig auf FALSE gesetzt. Das bedeutet, dass eine Statuscode-Ausnahme BadOutOfRange ausgelöst wird, wenn der Aufzählungswert außerhalb des Int32-Bereichs liegt.

Wenn die Option auf TRUE gesetzt wird, werden Aufzählungen mit größeren Datentypen als Int32 als reguläre Variablen mit diesem bestimmten Datentyp behandelt.

Angenommen, wir haben die folgenden Aufzählungsdefinitionen in unserem SPS-Code:

```
TYPE E_Enum_Normal :
(
  enum_member_0 := 0,
  enum_member_1 := 1,
  enum_member_2 := 2
);
END_TYPE

TYPE E_Enum_NotSoBig :
(
  enum_member_0 := 0,
  enum_member_1 := 1,
  enum_member_2 := 2
) UINT;
END_TYPE
```

```
TYPE E_Enum_VeryBig :
(
  enum_member_0 := 0,
  enum_member_1 := 1,
  enum_member_2 := 2
) LINT;
END_TYPE
```

Bei Aktivierung der vorstehenden Konfigurationsoption werden Instanzen von E_Enum_VeryBig als reguläre Variable vom Datentyp Int64 im Namensraum des Servers behandelt, während Instanzen von E_Enum_Normal und E_Enum_NotSoBig als OPC-UA-Aufzählung (mit dem Datentyp Int32) behandelt werden:

Data Access View

#	Server	Node Id	Display Name	Value	Datatype	Source Timestamp	Server Timestamp	Status
1	TcOpcUaServer...	NS4|String|MAI...	eEnumVeryBig	0	Int64	09:05:56.115	09:05:56.115	Good
2	TcOpcUaServer...	NS4|String|MAI...	eEnumNormal	0 (enum_member_0)	Int32	09:05:59.115	09:05:59.115	Good
3	TcOpcUaServer...	NS4|String|MAI...	eEnumNotSoBig	0 (enum_member_0)	Int32	09:07:25.594	09:07:25.594	Good

4.10.8 Strukturen

Voraussetzungen

Diese Funktionalität steht nur für Data-Access [▶ 49]-Geräte basierend auf TwinCAT 3 und dem Import von TMC-Symboldateien [▶ 53] zur Verfügung.

Der TwinCAT OPC UA Server ermöglicht die Verwendung von sogenannten StructuredTypes für Strukturen aus der TwinCAT-3-SPS. StructuredTypes erlauben es, Strukturen datenkonsistent zu lesen oder zu schreiben und die Typbeschreibung der Struktur für den Client bereitzustellen.

Über ein spezielles Pragma können Sie eine Struktur als StructuredType definieren und verfügbar machen. Wenn dieses Pragma nicht gesetzt wird, wird die Struktur nicht als StructuredType in den Adressraum des Servers geladen, sondern als FolderType dargestellt. Die Membervariablen der Struktur werden als separate Nodes dargestellt, auf die zugegriffen werden kann.

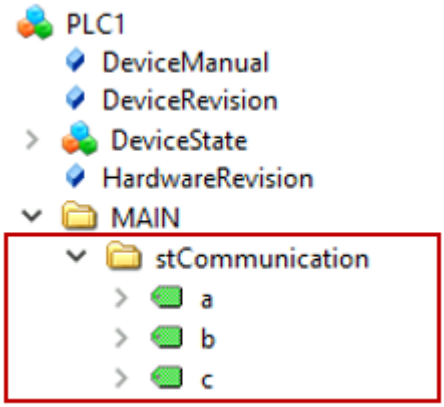
Gegeben sei folgendes Beispiel einer Struktur in der TwinCAT-3-SPS:

```
TYPE ST_Communication :
STRUCT
  a : INT;
  b : INT;
  c : INT;
END_STRUCT
END_TYPE
```

Im Programm MAIN wird diese Struktur nun instanziiert und über ein Pragma für OPC UA freigegeben, wie im Kapitel Freigabe von (SPS-) Symbolen [▶ 55] beschrieben.

```
PROGRAM MAIN
VAR
  {attribute 'OPC.UA.DA' := '1'}
  stCommunication : ST_Communication;
END_VAR
```

Die Struktur-Instanz wird nun standardmäßig wie folgt im Server bereitgestellt:

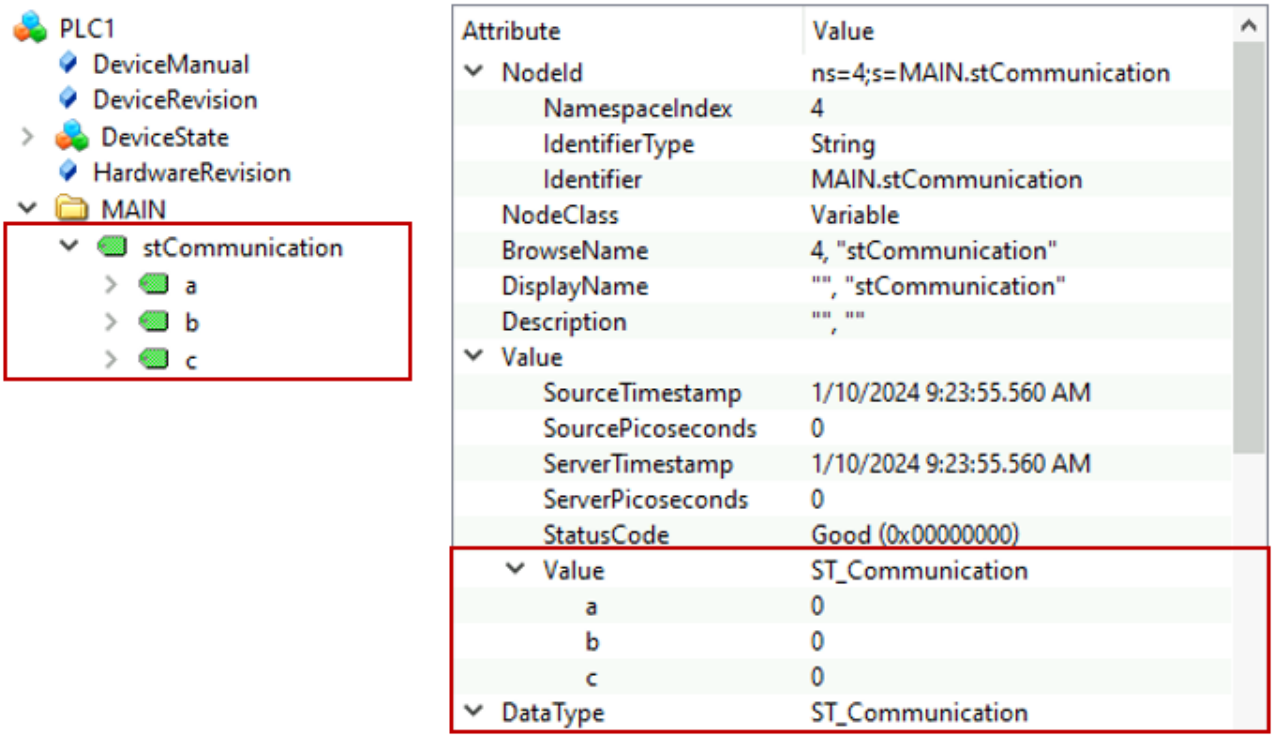


Auf die einzelnen Membervariablen der Struktur kann nun zugegriffen werden, allerdings nicht auf das Rootelement der Struktur. Hierdurch ist gegebenenfalls kein datenkonsistenter Zugriff auf die gesamte Struktur möglich.

Durch die Verwendung eines weiteren Pragmas wird diese Struktur-Instanz nun als StructuredType definiert.

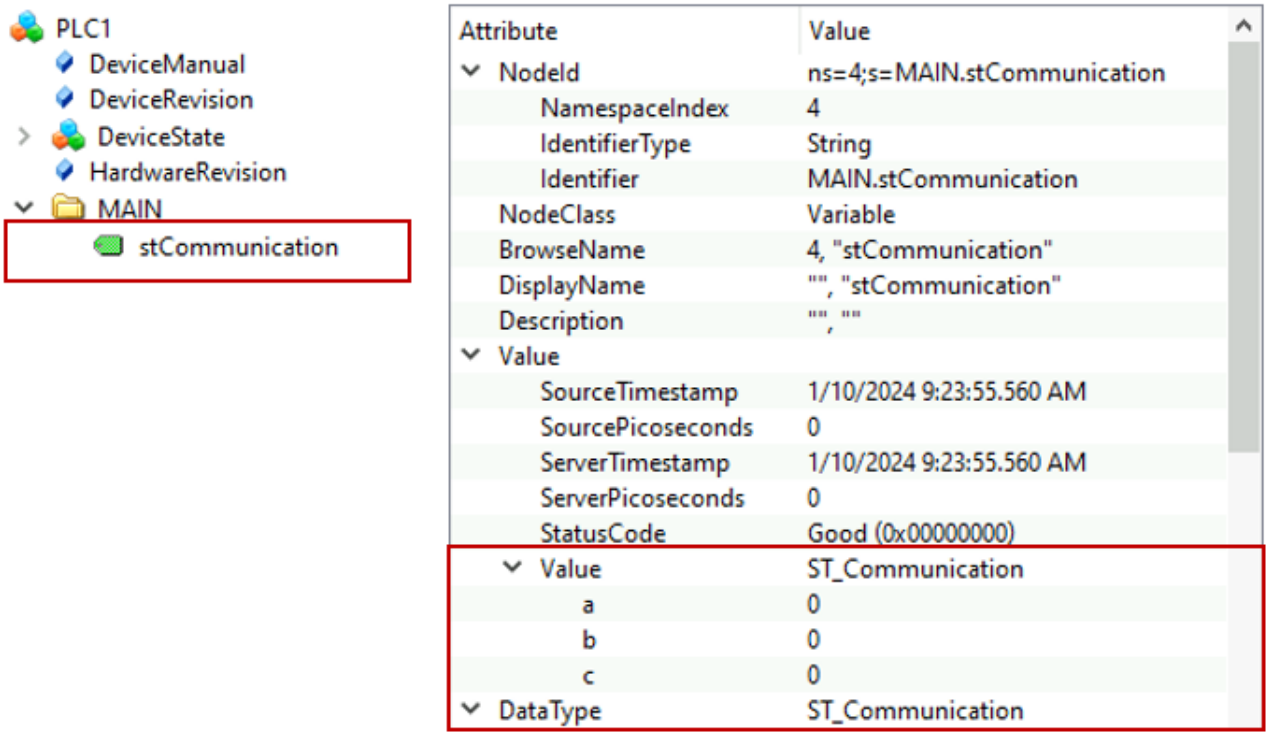
```
PROGRAM MAIN
VAR
  {attribute 'OPC.UA.DA' := '1'}
  {attribute 'OPC.UA.DA.StructuredType' := '1'}
  stCommunication : ST_Communication;
END_VAR
```

Im TwinCAT OPC UA Server wird die Instanz dann wie folgt bereitgestellt:



Beim Auslesen des Rootelements werden die Strukturinformationen datenkonsistent über OPC UA bereitgestellt und können von einem Client verarbeitet werden. Zusätzlich werden auch die Membervariablen als separate Nodes dargestellt und es kann auf sie zugegriffen werden. Über ein spezielles Attribut können Sie dies deaktivieren. Die Membervariablen sind dann nur noch über den StructuredType zugreifbar, wie das folgende Beispiel zeigt:

```
PROGRAM MAIN
VAR
  {attribute 'OPC.UA.DA' := '2'}
  {attribute 'OPC.UA.DA.StructuredType' := '1'}
  stCommunication : ST_Communication;
END_VAR
```



Hierbei handelt es sich vor allem um eine Speicheroptimierung des Servers, wie auch im Kapitel Optimierungen [▶ 42] beschrieben, denn für jede Node im Adressraum muss Arbeitsspeicher allokiert werden.

Sie müssen die Platzierung der Pragmas nicht zwingend an jeder Instanz einer Struktur vornehmen, sondern können diese auch einmalig an der Struktur-Definition durchführen. In diesem Fall werden alle Instanzen von dieser Struktur automatisch für OPC UA als StructuredType freigegeben.

```
{attribute 'OPC.UA.DA' := '1'}
{attribute 'OPC.UA.DA.StructuredType' := '1'}
TYPE ST_Communication :
STRUCT
  a : INT;
  b : INT;
  c : INT;
END_STRUCT
END_TYPE
```

Sie können in diesem Fall die StructuredType Definition auch explizit für bestimmte Instanzen deaktivieren, hierzu verwenden Sie das folgende Attribut:

```
PROGRAM MAIN
VAR
  {attribute 'OPC.UA.DA.StructuredType' := '0'}
  stCommunication : ST_Communication;
END_VAR
```

StructuredType für Funktionsbausteine

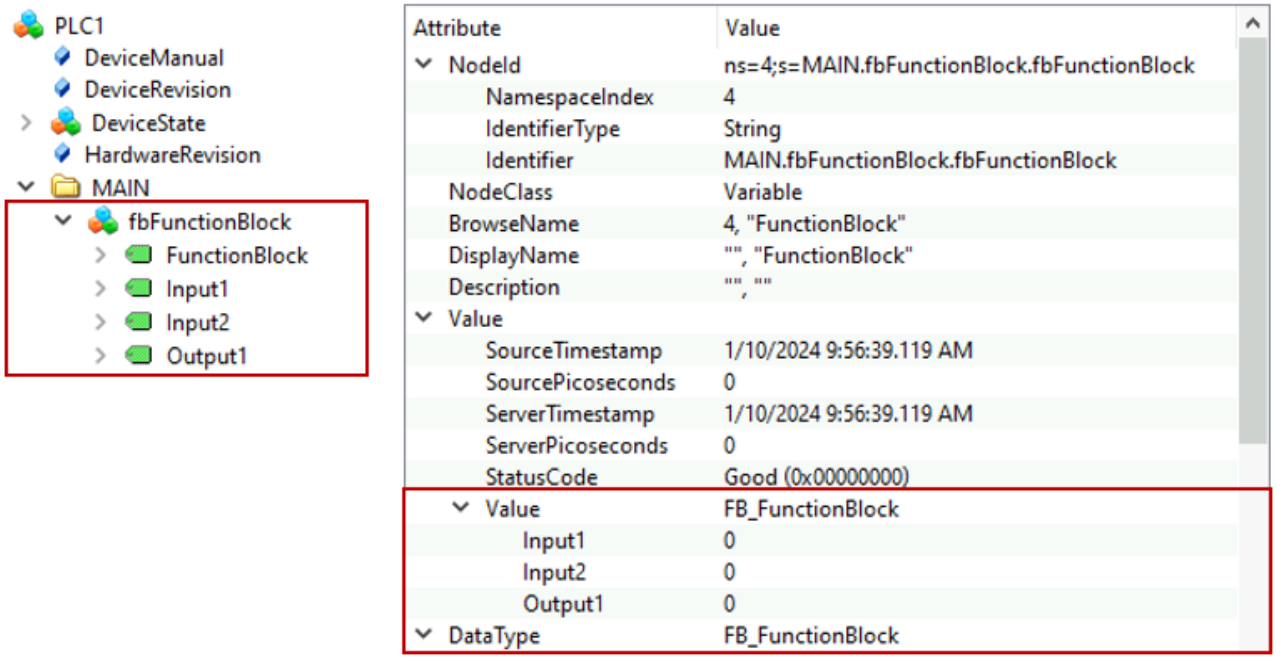
Sie können StructuredTypes auch bei SPS-Funktionsbausteinen verwenden. In diesem Fall erhält eine Funktionsbaustein-Instanz einen Kindknoten namens „FunctionBlock“, welcher den gesamten Funktionsbaustein als StructuredType darstellt.
Betrachten Sie folgendes Beispiel:

```
FUNCTION_BLOCK FB_FunctionBlock
VAR_INPUT
  Input1 : INT;
  Input2 : LREAL;
END_VAR
VAR_OUTPUT
  Output1 : LREAL;
END_VAR
```

Im MAIN Programm wird eine Instanz des Funktionsbausteins angelegt und über Pragmas sowohl für OPC UA freigegeben als auch als StructuredType definiert.

```
PROGRAM MAIN
VAR
  {attribute 'OPC.UA.DA' := '1'}
  {attribute 'OPC.UA.DA.StructuredType' := '1'}
  fbFunctionBlock : FB_FunctionBlock;
END_VAR
```

Der TwinCAT OPC UA stellt die Funktionsbaustein-Instanz dann wie folgt bereit:



Die oben für Strukturen genannten Definitionsorte der Pragmas gelten ebenfalls für Funktionsbausteine. Desweiteren können Sie auch hier die Membervariablen eines Funktionsbausteins explizit ausblenden.

Einschränkungen

Es gelten die folgenden Einschränkungen bei der Verwendung von StructuredTypes.

Pointer und Referenzen

Wenn in der Struktur Pointer- und Referenztypen verwendet werden, dann können diese nicht in einen StructuredType überführt werden. Der OPC UA Server stellt diese Strukturen als reguläre FolderTypes mit der entsprechenden Membervariablen dar.

Maximale Größe der Struktur

Die maximale Größe einer Struktur ist standardmäßig auf 16 kByte beschränkt. Überschreitet die Größe der Struktur dieses Limit, werden Struktur-Instanzen als FolderType dargestellt. Sie können dieses Limit bei Bedarf erhöhen. Die Hintergründe sind in dem Kapitel Optimierungen [▶ 42] näher beschrieben. Jedes STRUCT tauscht ständig Daten mit dem Basis-ADS-Gerät aus, d. h. mit jedem Lese-/Schreibbefehl eines StructuredTypes wird eine große ADS-Meldung verschickt. Damit der ADS-Router nicht mit großen Meldungen überflutet wird, ist die maximale Größe beschränkt.

4.10.9 Properties

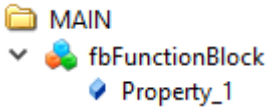
Voraussetzungen

Diese Funktionalität steht nur für Data-Access [▶ 49]-Geräte basierend auf TwinCAT 3 und dem Import von TMC-Symboldateien [▶ 53], sowie der Online-Symbolik zur Verfügung.

Die Anzeige von SPS-Properties im Namensraum des Servers wird seit TwinCAT 3.1 Build 4024 unterstützt. Hierzu müssen am Property lediglich zwei spezielle SPS-Attribute gesetzt werden, damit das Property in den Namensraum importiert und dort als UA-Property dargestellt wird. Beispiel:

```
{attribute 'OPC.UA.DA.Property' := '1'}
{attribute 'monitoring' := 'call'}
PROPERTY Property_1 : BOOL
```

Der Funktionsbaustein, an dem das Property definiert ist, muss das normale SPS-Attribut zur Freigabe von Symbolen enthalten.



In der Konfigurationsdatei TcUaDaConfig.xml kann global das Handling von SPS-Properties festgelegt werden. Hierzu dient das Flag „ImportPlcProperties“.

Wert	Beschreibung
false	Zeigt alle Properties im OPC UA Namensraum an, bei denen sowohl das OPC.UA.DA.Property als auch monitoring-Attribut gesetzt wurden.
true	Zeigt alle Properties im OPC UA Namensraum an, bei denen das monitoring-Attribut gesetzt wurde.

4.10.10 StatusCode

Voraussetzungen

Diese Funktionalität steht nur für Data-Access [▶ 49]-Geräte basierend auf TwinCAT 3 und dem Import von TMC-Symboldateien [▶ 53] zur Verfügung.

Neuimplementierung in Setup-Version 5.x

Im Folgenden wird die Funktionalität für den TwinCAT OPC UA Server in Setup-Version 5.x beschrieben. Anschließend folgt eine Beschreibung [▶ 82] der Funktionalität für den TwinCAT OPC UA Server in Setup-Version 4.x..

Der TwinCAT OPC UA Server ermöglicht die Anpassung des OPC UA Status Codes für ein bestimmtes SPS-Symbol. Führen Sie die folgenden Schritte aus, um den StatusCode eines Symbols zur Laufzeit ändern zu können.

Struktur erstellen

Damit die Datenkonsistenz gewährleistet ist, basiert das Konzept auf einer Struktur (siehe Strukturierte Typen). Jedes Symbol, für das der StatusCode geändert werden können soll, muss in eine Struktur verpackt werden.

Erzeugen Sie eine neue Struktur und fügen Sie das folgende Pragma vor der Struktur-Definition hinzu. Die Struktur enthält das Symbol selbst sowie eine Variable vom Datentyp DINT, welche den StatusCode in Dezimalform repräsentiert.

```
{attribute 'OPC.UA.DA.StructuredType' := '1'}
{attribute 'OPC.UA.DA.StatusAndValue' := '1'}
TYPE ST_StatusCodeSimple :
STRUCT
  value : REAL;
  status : DINT := -2147155968; // equals 'BadCommunicationError'
END_STRUCT
END_TYPE
```

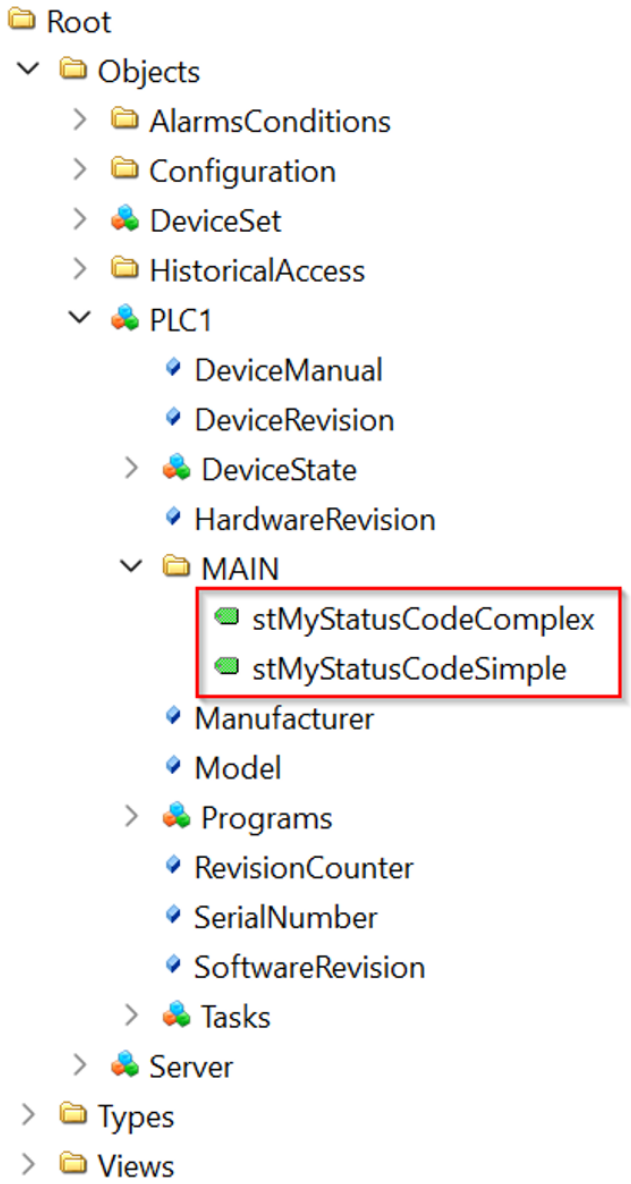
Sie können hierbei auch komplexe Symbole verwenden, zum Beispiel:

```
{attribute 'OPC.UA.DA.StructuredType' := '1'}
{attribute 'OPC.UA.DA.StatusAndValue' := '1'}
TYPE ST_StatusCodeComplex :
STRUCT
  {attribute 'OPC.UA.DA.StructuredType' := '1'}
  value : ST_Complex_1;
  status : DINT := -2146762752; // equals 'BadServiceUnsupported'
END_STRUCT
END_TYPE
```

Erstellen Sie nun eine Instanz von dieser Struktur, z. B. im Programm MAIN, und fügen Sie das Pragma zur Freigabe der Instanz über OPC UA hinzu.

```
PROGRAM MAIN
VAR
  {attribute 'OPC.UA.DA' := '1'}
  stStatusCodeSimple : ST_StatusCodeSimple;
  {attribute 'OPC.UA.DA' := '1'}
  stStatusCodeComplex : ST_StatusCodeComplex;
END_VAR
```

Die definierten Struktur-Instanzen werden nun mit dem Datentyp der als „value“ definierten Membervariablen im TwinCAT OPC UA Server bereitgestellt.

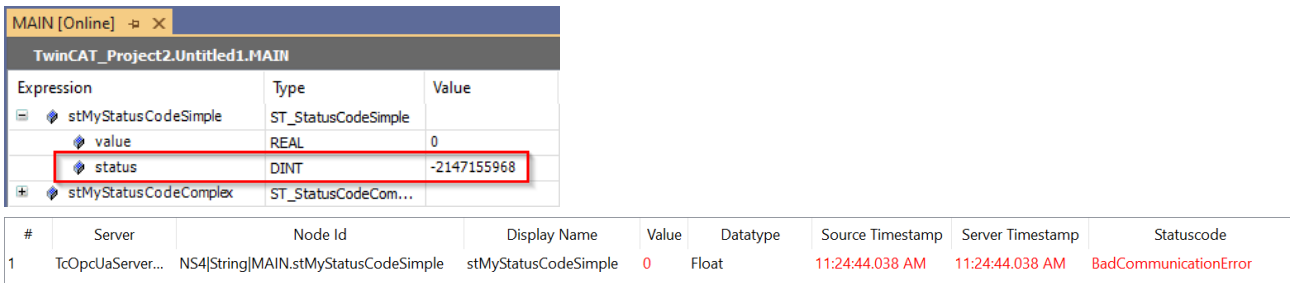


Attribute	Value
NodeId	ns=4;s=MAIN.stMyStatusCodeComplex
NamespaceIndex	4
IdentifierType	String
Identifier	MAIN.stMyStatusCodeComplex
NodeClass	Variable
BrowseName	4, "stMyStatusCodeComplex"
DisplayName	"", "stMyStatusCodeComplex"
Description	"", ""
Value	
SourceTimestamp	12/29/2023 11:19:59.108 AM
SourcePicoseconds	0
ServerTimestamp	12/29/2023 11:19:59.108 AM
ServerPicoseconds	0
StatusCode	BadServiceUnsupported (0x800b0000)
Value	ST_Complex_1
x	0 (ZERO)
a	0
str	
b	0
DataType	ST_Complex_1
NamespaceIndex	4
IdentifierType	String
Identifier	<StructuredDataType>:ST_Complex_1

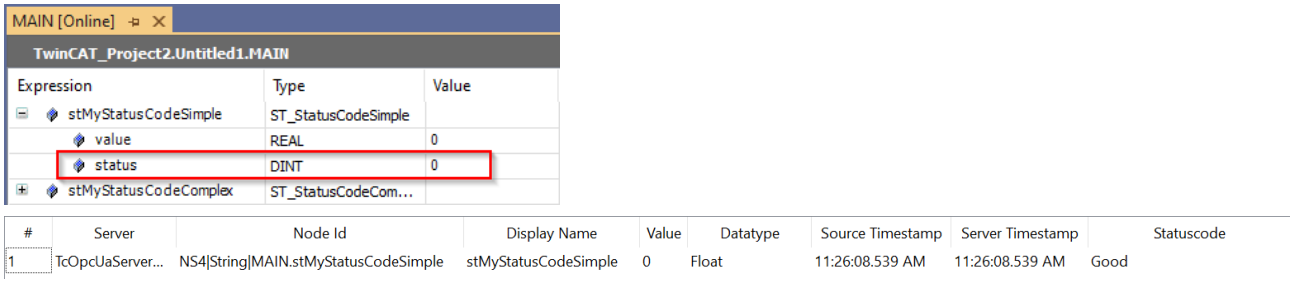
Attribute	Value
NodeId	ns=4;s=MAIN.stMyStatusCodeSimple
NamespaceIndex	4
IdentifierType	String
Identifier	MAIN.stMyStatusCodeSimple
NodeClass	Variable
BrowseName	4, "stMyStatusCodeSimple"
DisplayName	"", "stMyStatusCodeSimple"
Description	"", ""
Value	
SourceTimestamp	12/29/2023 11:18:45.008 AM
SourcePicoseconds	0
ServerTimestamp	12/29/2023 11:18:45.008 AM
ServerPicoseconds	0
StatusCode	BadCommunicationError (0x80050000)
Value	0
DataType	Float
NamespaceIndex	0
IdentifierType	Numeric
Identifier	10 [Float]

StatusCode zur Laufzeit ändern

Um den StatusCode zur Laufzeit zu ändern, bearbeiten Sie einfach den Wert der Membervariablen „status“. Wenn Sie diesen, wie in obigem Beispiel auf „-2147155968“ setzen, so ändert sich der StatusCode zu „BadCommunicationError“.



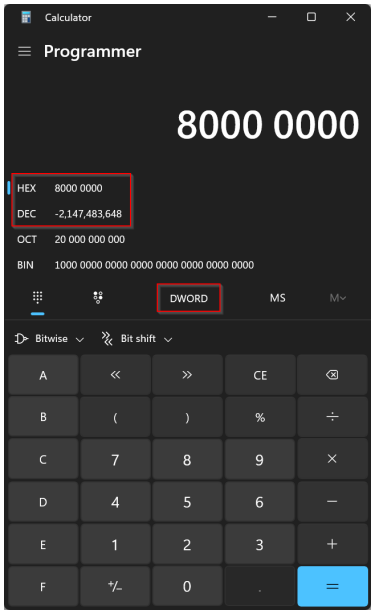
Setzen Sie den Wert hingegen auf „0“, so ändert sich der StatusCode zu „Good“.



Der hierbei verwendete Dezimalwert ist durch die OPC UA Spezifikation festgelegt, hier die aktuelle Version des StatusCode-Mappings. In der nachfolgenden Tabelle werden einige gängige StatusCodes und deren numerische Darstellung aufgelistet.

StatusCode	Hex	Dezimal
Good	0x00000000	0
Uncertain	0x40000000	1073741824
Bad	0x80000000	-2147483648
BadUnexpectedError	0x80010000	-2147418112
BadInternalError	0x80020000	-2147352576
BadCommunicationError	0x80050000	-2147155968
BadTimeout	0x800A0000	-2146828288
BadServiceNotSupported	0x800B0000	-2146762752

Bitte beachten Sie bei der Berechnung der dezimalen Darstellung eines StatusCodes auf der Grundlage von dessen hexadezimaler Darstellung, dass Ihr Rechner auf DWORD eingestellt ist, zum Beispiel der Windows-Rechner („Programmierer“-Ansicht).



Verwendung in Setup-Version 4.x

In der „älteren“ Implementierung in Setup-Version 4.x wird die Variable auf OPC UA-Seite nicht als Einzelvariable dargestellt, sondern genauso wie auf SPS-Seite als Struktur. Auch in der Konfiguration gibt es Unterschiede bei den Attributen.

```
{attribute 'OPC.UA.DA.StructuredType' := '1'}
{attribute 'OPC.UA.DA.STATUS' := 'Quality'}
TYPE ST_StatusCodeSimple_V4 :
STRUCT
  Value : REAL;
  Quality : DINT := -2147155968; // equals 'BadCommunicationError'
END_STRUCT
END_TYPE
```

Der StatusCode wird nur an der Strukturvariable über OPC UA zur Verfügung gestellt. Die Werte Value und Quality enthalten den Wert und den Wert des StatusCodes.

Data Access View

#	Server	Node Id	Display Name	Value	Datatype	Source Timestamp	Server Timestamp	Statuscode
1	TcOpcUaServer...	NS4|String|...	stStatusCodeSimpleV4		Null	11:38:00.579 AM	11:38:00.579 AM	BadCommunicationError
2	TcOpcUaServer...	NS4|String|...	Quality	-2147155968	Int32	11:38:08.830 AM	11:38:08.830 AM	Good
3	TcOpcUaServer...	NS4|String|...	Value	0	Float	11:38:11.581 AM	11:38:11.581 AM	Good

4.10.11 AnalogItemType

Voraussetzungen

Diese Funktionalität steht nur für Data-Access [▶ 49]-Geräte basierend auf TwinCAT 3 und dem Import von TMC-Symboldateien [▶ 53] zur Verfügung.

AnalogItemTypes sind ein Bestandteil der OPC-UA-Spezifikation und ermöglichen es, Metainformationen, wie z. B. Einheiten an eine Variable zu heften. In der TwinCAT-3-SPS können Sie diese Metainformationen in Form von SPS-Attributen definieren.

Folgende Einstellungen sind möglich:

- EngineeringUnits: Einheiten, definiert über die OPC-UA-Spezifikation
- EURange: Maximaler Wertebereich der Variablen
- InstrumentRange: Normaler Wertebereich der Variablen
- WriteBehavior: Verhalten, wenn bei einem Schreibvorgang der Wertebereich überschritten wird.

Das nachfolgende Beispiel zeigt, wie die Variable fillLevel als AnalogItemType konfiguriert wird. Folgenden Parameter werden hierbei gesetzt:

- Einheit: 20529 („Prozent“, definiert in der OPC-UA-Spezifikation)
- Max. Wertebereich: 0 bis 100
- Normaler Wertebereich: 10 bis 90
- Schreibverhalten: 1 (Clamping)

```
{attribute 'OPC.UA.DA' := '1'}
{attribute 'OPC.UA.DA.AnalogItemType' := '1'}
{attribute 'OPC.UA.DA.AnalogItemType.EngineeringUnits' := '20529'}
{attribute 'OPC.UA.DA.AnalogItemType.EURange' := '0:100'}
{attribute 'OPC.UA.DA.AnalogItemType.InstrumentRange' := '10:90'}
{attribute 'OPC.UA.DA.AnalogItemType.WriteBehavior' := '1'}
fillLevel : UINT;
```

EngineeringUnits können anhand der in OPC UA spezifizierten IDs (Teil 8 der OPC-UA-Spezifikation) konfiguriert werden. Die IDs orientieren sich nach den weit verbreiteten und akzeptierten „Codes for Units of Measurement (Recommendation N.20)“, die vom „United Nations Centre for Trade Facilitation and Electronic Business“ veröffentlicht wurden. CommonCode, der die dreistellige, alphanumerische ID angibt, wird von OPC UA laut Spezifikation in einen Int32-Wert konvertiert und referenziert (Auszug aus OPC-UA-Spezifikation v1.02, Pseudo-Code):

```
Int32 unitId = 0;
Int32 c;
for (i=0; i<=3;i++)
{
```

```
  c = CommonCode[i];
  if (c == 0)
    break; // end of Common Code
  unitId = unitId << 8; // shift left
  unitId = unitId | c; // OR operation
}
```

Schreibverhalten

Beim Schreiben einer AnalogItemType-Variablen können Sie definieren, wie der OPC UA Server mit dem neuen Wert in Bezug auf den Wertebereich umgehen soll. Hierbei gibt es die folgenden Möglichkeiten:

- 0: Alle Werte sind zugelassen und werden bei einem Schreibvorgang übernommen.
- 1: Der zu schreibende Wert wird passend zum Wertebereich abgeschnitten.
- 2: Der zu schreibende Wert wird abgewiesen, falls er den Wertebereich überschreitet.

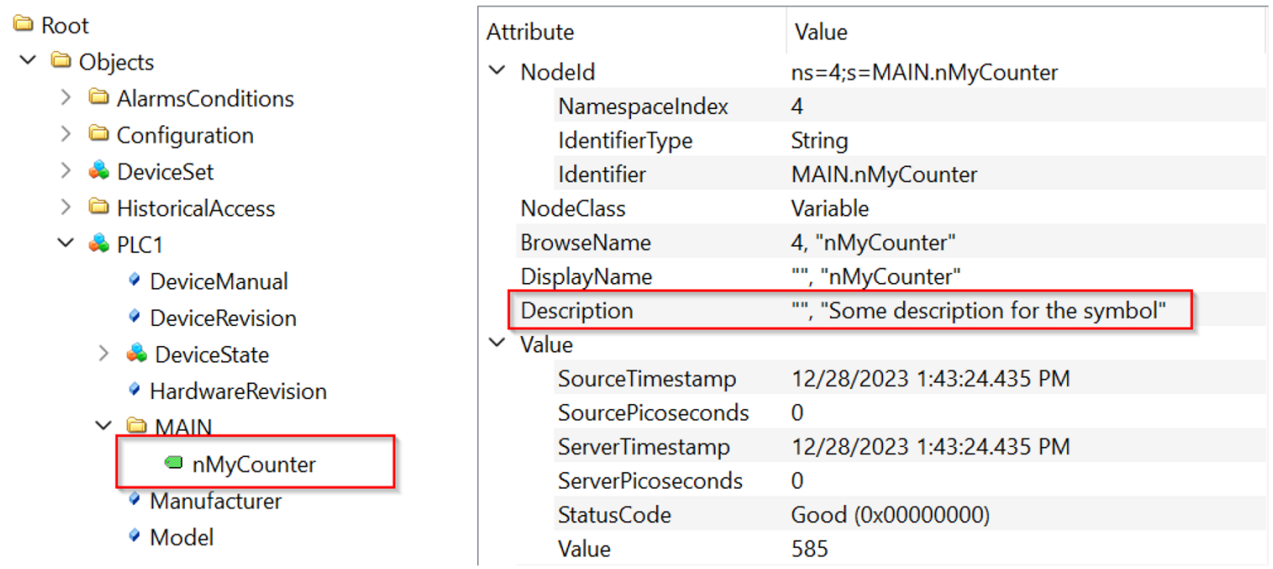
4.10.12 Description

Voraussetzungen

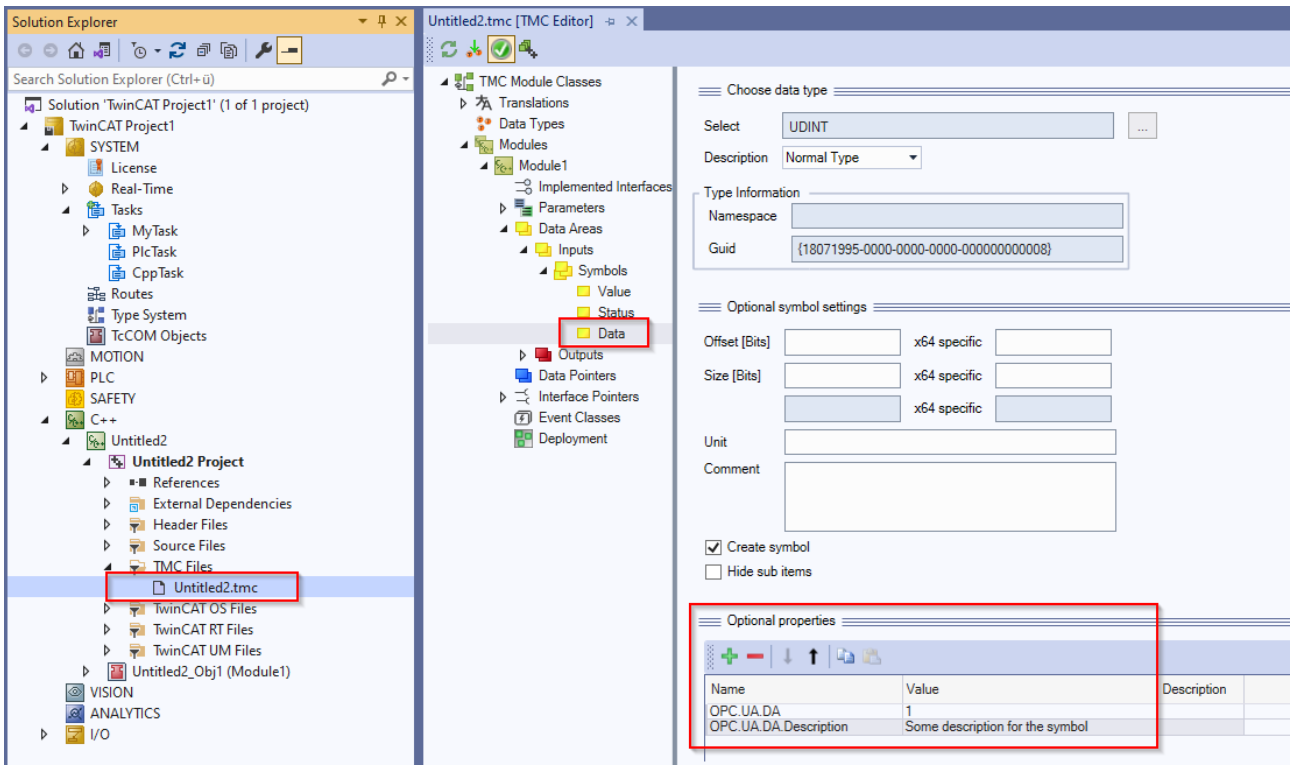
Diese Funktionalität steht nur für Data Access [▶ 49] Geräte basierend auf TwinCAT 3 und dem Import von TMC- sowie TMI-Symboldateien [▶ 53], sowie der Online-Symbolik zur Verfügung.

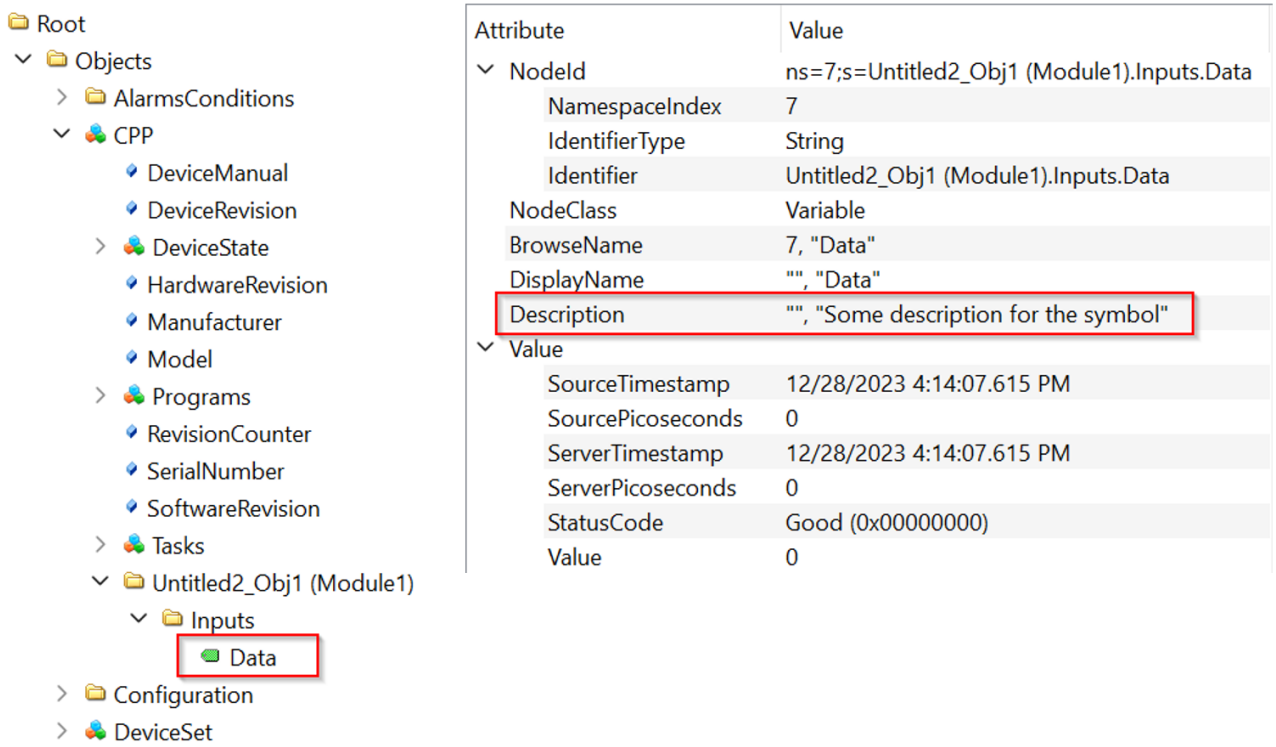
Mit dem folgenden TwinCAT-3-SPS [▶ 55]-Pragma kann das OPC UA Attribut „Description“ für ein Symbol gesetzt werden.

```
{attribute 'OPC.UA.DA' := '1'}
{attribute 'OPC.UA.DA.Description' := 'Some description for the symbol'}
nMyCounter : INT;
```



Im Falle von TwinCAT 3 C++ [▶ 57] können Sie dieses Pragma auch im TMC-Code-Generator verwenden, um eine Description für die Node zu setzen. Bitte beachten Sie hierbei, dass dieses Feature unter TwinCAT 3 C++ nur bei Verwendung der Online-Symbolik [▶ 69] zur Verfügung steht.





4.10.13 ReadOnly

Voraussetzungen

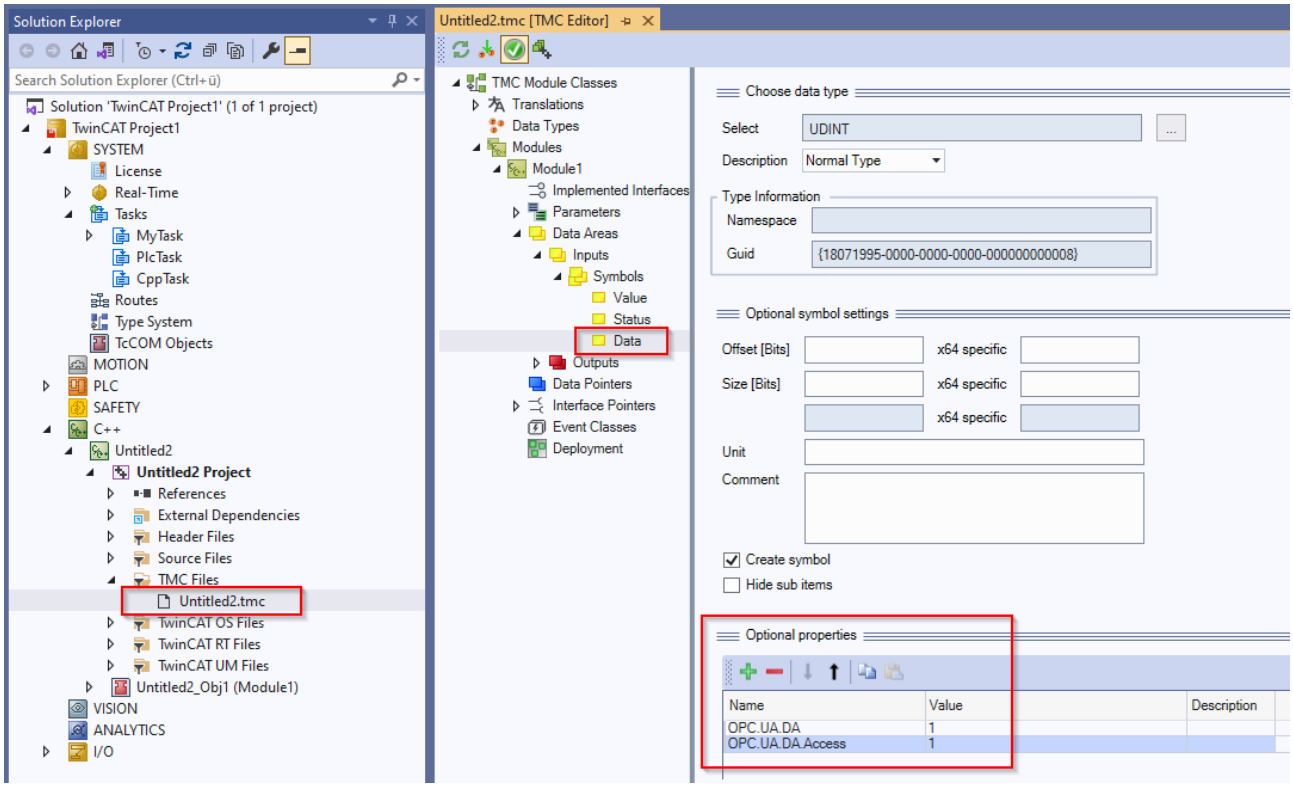
Diese Funktionalität steht nur für Data-Access [▶ 49]-Geräte basierend auf TwinCAT 3 und dem Import von TMC-Symboldateien [▶ 53] zur Verfügung.

Mit dem folgenden Pragma kann ein Symbol read-only gesetzt werden. Schreibvorgänge seitens eines OPC UA Clients quittiert der Server dann mit dem StatusCode BadNotWriteable.

```
{attribute 'OPC.UA.DA' := '1'}
{attribute 'OPC.UA.DA.Access' := '1'}
nMyCounter : INT;
```

Timestamp	Source	Server	Message
12/28/2023 4:21:21.953 PM	DA Plugin	TcOpcUaServer@DESKTOP-28AUAPK	Write to node 'NS7|String|Untitled2_Obj1 (Module1).Inputs.Data' failed [ret = BadNotWritable]

Im Fall von TwinCAT 3 C++ [▶ 57] können Sie dieses Pragma auch im TMC Code Generator verwenden, um eine Node als Read-Only zu markieren. Bitte beachten Sie hierbei, dass dieses Feature unter TwinCAT 3 C++ nur bei Verwendung der Online-Symbolik [▶ 69] zur Verfügung steht.



4.10.14 Alias

Voraussetzungen

Diese Funktionalität steht nur für Data-Access [▶ 49]-Geräte basierend auf TwinCAT 3 und dem Import von TMC-Symboldateien [▶ 53], sowie der Online-Symbolik zur Verfügung.

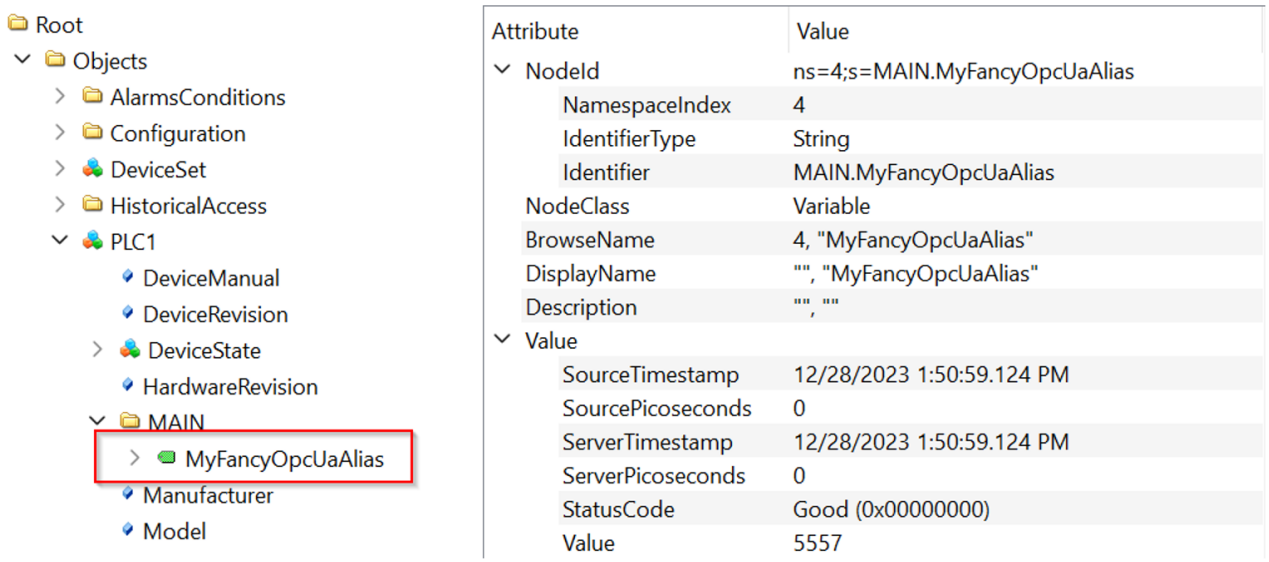
Veraltetes Feature

Dieses Feature wurde als „deprecated“ eingestuft und kann in zukünftigen Serverversionen entfernt werden. Als bessere Alternative steht Ihnen unser Produkt TE6100 TC3 OPC UA Nodeset Editor zur Verfügung, welches dieselbe Funktionalität (und mehr) bereitstellt.

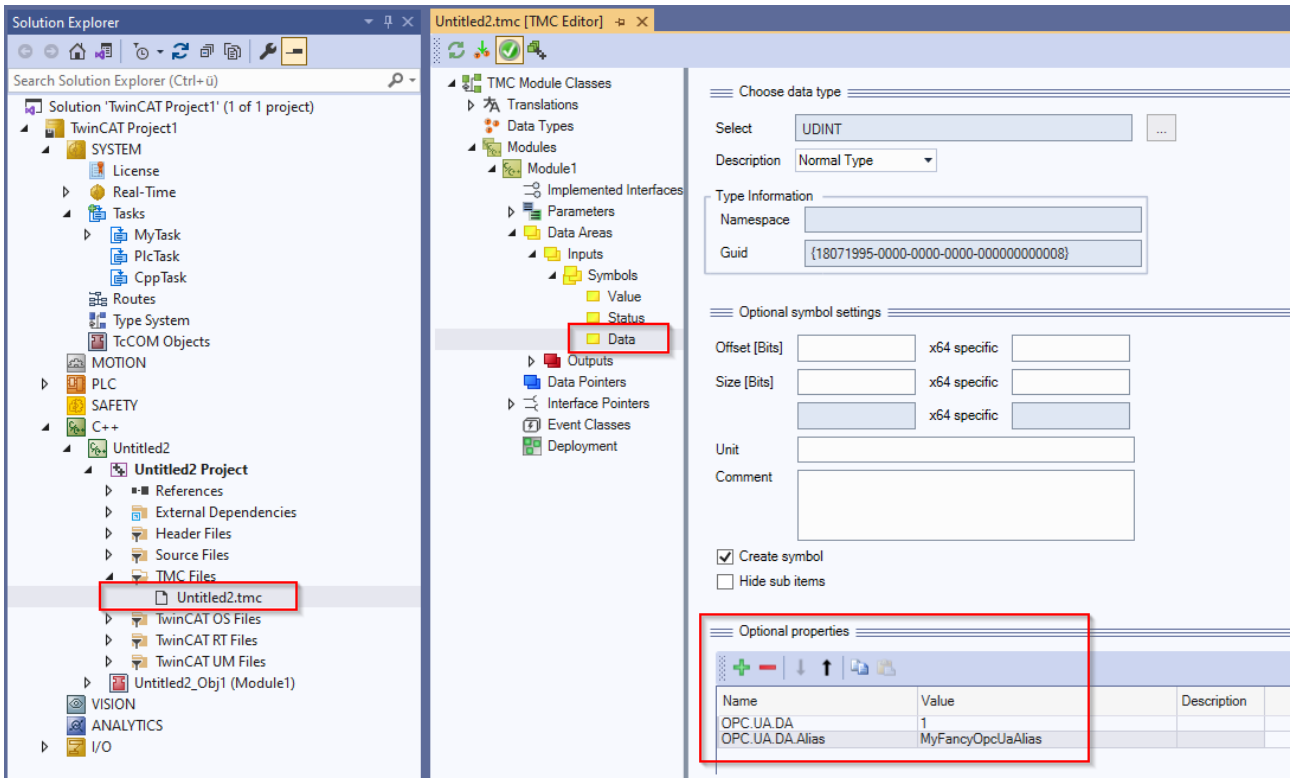
Mit dem folgenden Pragma kann ein Symbol einen anderen Namen im OPC-UA-Adressraum tragen. Bitte beachten Sie, dass diese Funktionalität nur für einfache Datentypen zur Verfügung steht, nicht jedoch für Arrays oder Strukturen.

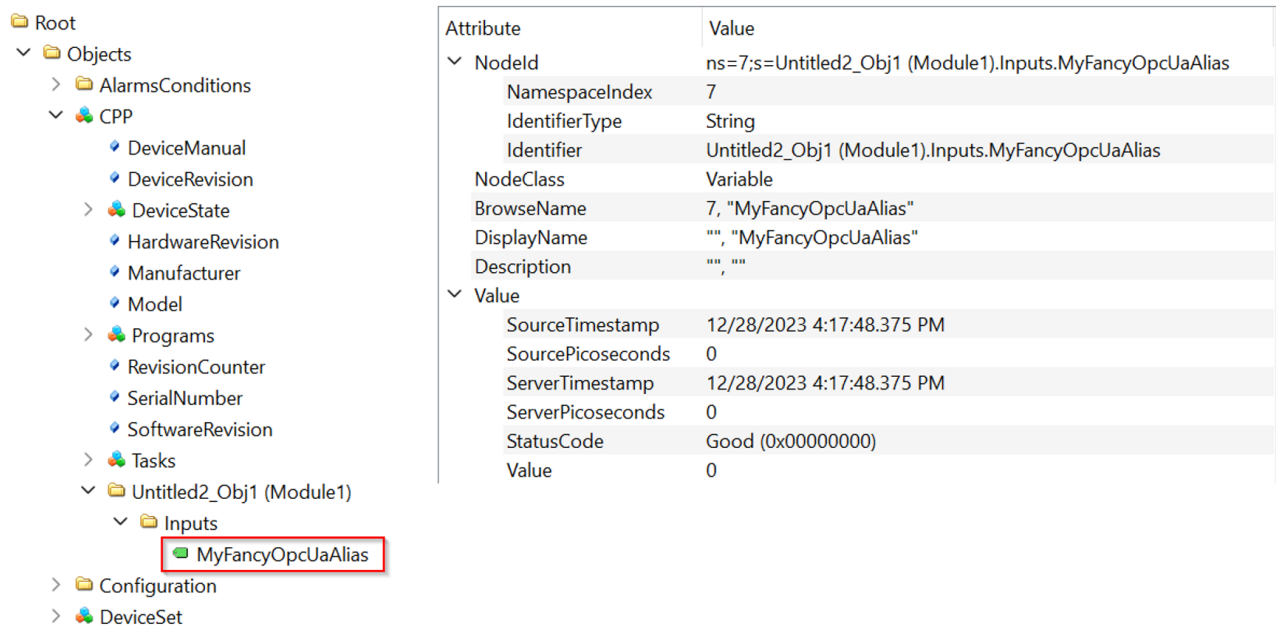
```
{attribute 'OPC.UA.DA' := '1'}
{attribute 'OPC.UA.DA.Alias' := 'MyFancyOpcUaAlias'}
nMyCounter : INT;
```

Hierbei werden sowohl die NodeID als auch der BrowseName und DisplayName entsprechend auf den Alias-Wert gesetzt.



Im Fall von TwinCAT 3 C++ [▶ 57] können Sie dieses Pragma auch im TMC-Code-Generator verwenden, um ein Alias für die Node zu setzen. Bitte beachten Sie hierbei, dass dieses Feature unter TwinCAT 3 C++ nur bei Verwendung der Online-Symbolik [▶ 69] zur Verfügung steht.





4.10.15 Pointer und Referenzen

Pointer

Pointervariablen (z. B. POINTER TO) werden grundsätzlich nicht vom Server im Namensraum dargestellt. Falls sich eine Pointervariable in einer Struktur befindet und diese als Structured Data Type konfiguriert wurde, so wird die Struktur nicht als Structured Data Type sondern als FolderType dargestellt.

Referenzen

Referenzvariablen (REFERENCE TO) werden als Einzelvariablen vom Server im Namensraum dargestellt und können ohne Einschränkungen gelesen werden. Befindet sich eine Referenz innerhalb einer Struktur, kann diese Struktur nicht mehr als StructuredTypes im Server zur Verfügung gestellt werden, sondern nur als FolderType. Der Zugriff auf die einzelnen Referenzvariablen innerhalb der Struktur funktioniert hingegen.

4.10.16 Typsystem

Voraussetzungen

Diese Funktionalität steht nur für Data-Access [▶ 49]-Geräte basierend auf TwinCAT 3 und dem Import von TMC-Symboldateien [▶ 53], sowie der Online-Symbolik zur Verfügung.

Einer der größten Vorteile von OPC UA ist das Meta-Modell, das zur Bereitstellung von Basistypen genauso verwendet werden kann, wie zur Erweiterung des Typsystems durch eigene Modelle. Derselbe Mechanismus wird zur Darstellung realer Objekte (Nodes) verwendet, sodass OPC UA Clients einen Objekttyp bestimmen können.

Der OPC UA Server veröffentlicht Typinformationen von der IEC61131-Welt in seinem Namensraum. Dies umfasst nicht nur Basistypen, wie z. B. BOOL, INT, DINT oder REAL, sondern auch erweiterte Typinformationen, wie z. B. die aktuelle Klasse (Funktionsbaustein) oder Struktur, die ein Objekt repräsentiert.

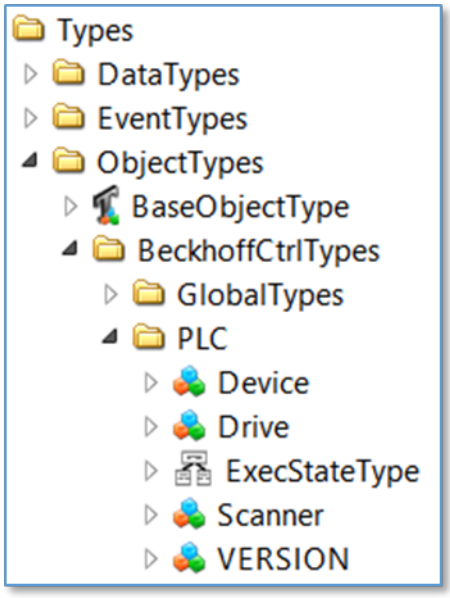
Typinformationen

Typinformationen sind Bestandteil des UA-Namensraums. Der OPC UA Server erweitert die Basistypinformationen wie folgt:

- Lokale Typinformationen, die nur für eine Laufzeit gültig ist, werden in demselben Namensraum, wie die Laufzeitsymbole, gespeichert.

- Globale Typinformationen, die über verschiedene Laufzeiten gültig sein können, werden in einem eigenen globalen Namensraum gespeichert.

Das Typsystem ist ebenfalls virtuell verfügbar und kann im Bereich Types des OPC UA Servers eingesehen werden:

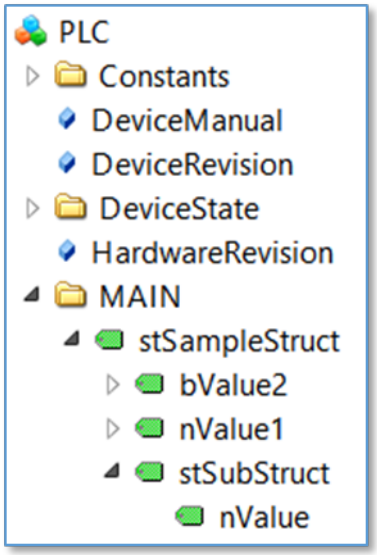


Jeder Nicht-Standarddatentyp wird im Bereich BeckhoffCtrlTypes eingetragen.

Grundlagen

Angenommen, die TwinCAT-3-SPS besteht aus einem SPS-Programm mit verschiedenen STRUCTs. Jeder STRUCT wird als Knoten in einem UA-Namensraum repräsentiert, mit jedem Element der Struktur als untergeordneten Knoten.

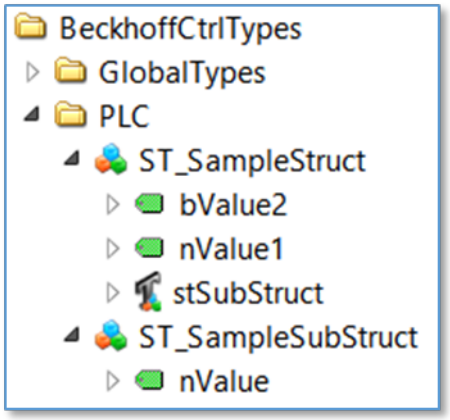
In dem Beispiel besteht der STRUCT stSampleStruct aus drei untergeordneten Elementen: einer Variable nValue1 vom Typ INT, einer Variable bValue2 vom Typ BOOL und einem weiteren STRUCT stSubStruct, das mit der Variablen nValue vom Typ INT lediglich ein untergeordnetes Element enthält.



Die Struktur selbst ist eine reguläre Variable im UA-Namensraum, die den Datentyp ByteString hat. Die Clients können daher einfach mit dem Wurzelelement (der Struktur selbst) verbunden werden und dessen Werte durch Interpretation des ByteString gelesen/geschrieben werden. Damit die Interpretation jedes untergeordneten Elements vereinfacht wird, enthält das Typsystem mehr Informationen über die Struktur selbst; in erster Linie in der Referenz zur Instanz:

Reference	Target DisplayName
HasTypeDe...	ST_SampleStruct
HasCompo...	nValue1
HasCompo...	bValue2
HasCompo...	stSubStruct

Außerdem enthält das Typsystem mehr Informationen über ST_SampleStruct:



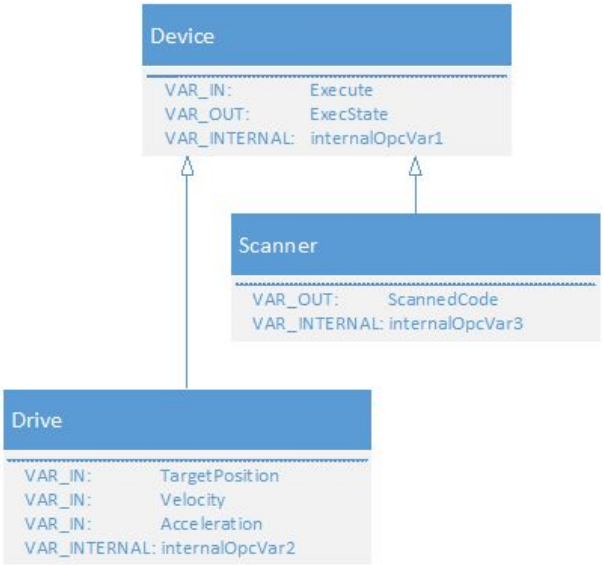
Und in den Referenzen von ST_SampleStruct:

Reference	Target DisplayName
HasTypeDe...	DataItemType
HasCompo...	nValue1
HasCompo...	bValue2
HasCompo...	stSubStruct

Insgesamt kann das Typsystem sehr nützliche Informationen bieten, wenn ein Client die Struktur weiter interpretieren will.

Objektorientierte Erweiterungen

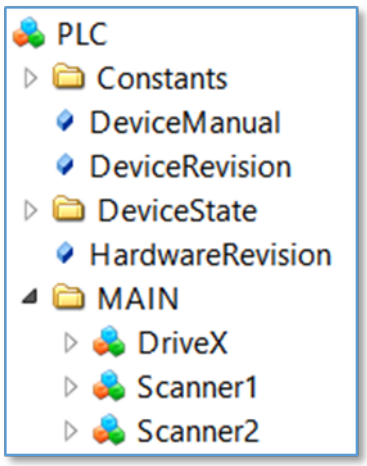
Angenommen, die TwinCAT-3-SPS-Laufzeit enthält ein SPS-Programm, dessen Struktur wie folgt visualisiert werden kann:



Die zwei Funktionsbausteine Scanner und Drive sind von der Basisklasse Device abgeleitet, indem objektorientierte Erweiterungen der IEC61131-3 verwendet werden. Das Programm MAIN enthält nun die folgenden Deklarationen:

```
PROGRAM MAIN
VAR
  {attribute 'OPC.UA.DA':='1'}
  Scanner1 : Scanner;
  {attribute 'OPC.UA.DA':='1'}
  Scanner2 : Scanner;
  {attribute 'OPC.UA.DA':='1'}
  DriveX : Drive;
END_VAR
```

Alle drei Objekte sind vom Typ Device, aber auch von ihrem speziellen Datentyp. Der OPC UA Server importiert die Objekte wie folgt:



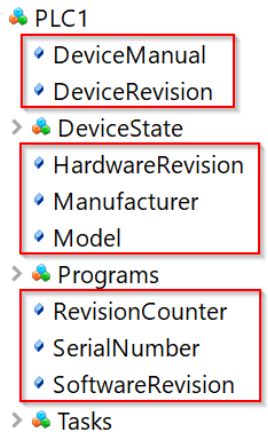
Der zugrunde liegende Datentyp kann nun in der Referenz jedes Objekts bestimmt werden, z. B. Objekt Scanner1:

Reference	Target DisplayName
HasTypeDe...	Scanner
HasInputVar	Execute
HasOutputV...	ExecState
HasLocalVar	internalOpcVar1
HasOutputV...	ScannedCode
HasLocalVar	internalOpcVar3

Entsprechend des zugrundeliegenden IEC61131-Programms ist das Objekt Scanner1 vom Datentyp Scanner und zeigt ebenfalls, welche Variablen enthalten sind und welcher Art die Variable ist: Eingangs-, Ausgangs- oder interne (lokale) Variable. Das Diagramm oben zeigt, dass nicht nur die Variablen des tatsächlichen Funktionsbausteins hier wiedergegeben werden, sondern auch die abgeleiteten Variablen der Basisklasse. Die gesamte IEC61131-3-Vererbungskette wird im UA-Namensraum repräsentiert.

4.10.17 DI Components

Jeder SPS-Namensraum auf dem Server beinhaltet eine Anzahl an Nodes, über welche sich statische Metainformationen zur SPS angeben lassen. Diese optionalen Informationen können in der TcUaDaConfig.xml für jeden Namenraum angegeben werden.

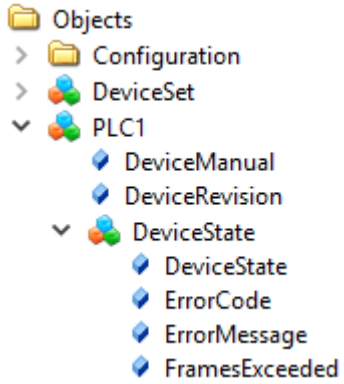


Die folgende Tabelle gibt weitere Informationen zu diesen Nodes. Die konkreten Wertebelegungen der einzelnen Nodes können zu einem großen Teil anwendungsspezifisch sein, daher werden im Auslieferungszustand des Servers nur Beispielwerte verwendet. Der Server selbst stellt diese Nodes in seinem Namensraum zur Verfügung, ändert jedoch nicht selbstständig deren Wertebelegungen.

Node	Beschreibung
DeviceManual	Erlaubt die Angabe einer Adresse unter der sich das Gerätehandbuch finden lässt, z. B. ein Pfad im Dateisystem oder eine Webadresse.
DeviceRevision	Beinhaltet den Revisions-Level einer Hardwarekomponente oder des gesamten Geräts.
HardwareRevision	Beinhaltet den Revisions-Level der Hardware.
Manufacturer	Beinhaltet den Namen des Geräteherstellers, üblicherweise als FQDN (Fully Qualified Domain Name), z. B. beckhoff.com.
Model	Beinhaltet den Namen des „Produkts“ (falls anwendbar).
RevisionCounter	Kann einen Zähler beinhalten, wie oft die Konfiguration des Geräts aktualisiert wurde.
SerialNumber	Eindeutige Seriennummer des Geräts, wie vom Gerätehersteller vergeben.
SoftwareRevision	Beinhaltet die Version oder den Revisions-Level der Softwarekomponente, der Firmware einer Hardwarekomponente oder auch der Firmware des Geräts.

4.10.18 DeviceState

Jeder Namespace im TwinCAT OPC UA Server enthält ein sogenanntes DeviceState-Objekt.



Dieses Objekt zeigt über diverse Properties den Zustand des unterlagerten ADS-Geräts an.

```
typedef enum
{
  UADEV_NOTINIT = 0x0100,
  UADEV_STARTING = 0x0110,
  UADEV_CONNECTED = 0x0120,
  UADEV_SHUTDOWN = 0x0130,
  UADEV_ERROR = 0xF000
}UaDeviceState;
```

Wenn sich das Gerät in einem ERROR Zustand befindet, dann liefert das ErrorCode Property folgende Werte:

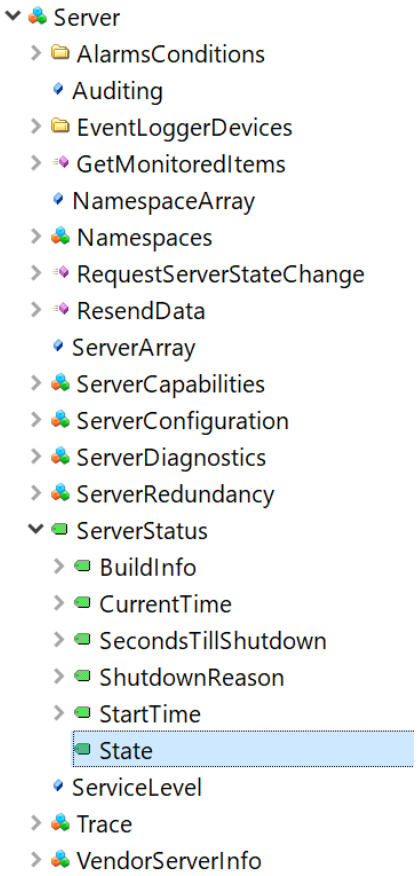
```
#define UA_DEVSTATE_INVALID_STATE 0x80EB0010
#define UA_DEVSTATE_CREATE_NS_FAILED 0x80EB0011
#define UA_DEVSTATE_LOAD_NS_FAILED 0x80EB0012
#define UA_DEVSTATE_INVALID_IO_SETTING 0x80EB0100
```

Eine entsprechende Fehlermeldung wird im ErrorMessage Property dargestellt.

4.10.19 ServerState

Die ServerState Variable im Server-Namensraum gibt den aktuellen Zustand des Servers an. Die folgende Tabelle gibt einen Überblick über die möglichen Variablenwerte.

Variablenwert	Beschreibung
Running	Der Server wurde erfolgreich hochgefahren.
Failed	Es wurde ein Problem in einer der Server Konfigurationsdateien gefunden, z. B. eine ungültige Konfiguration in der TcUaSecurityConfig.xml.
NoConfiguration	Der Server wurde noch nicht initialisiert.
Suspended	Der Server wurde noch nicht komplett hochgefahren, d. h. es sind unter Umständen noch nicht alle Funktionen verfügbar.



4.11 Historical Access

4.11.1 Überblick

In diesem Kapitel werden die notwendigen Schritte zur Konfiguration der Variablen im Namensraum des TwinCAT OPC UA Servers für **H**istorical **A**ccess (HA) beschrieben.

Historical Access ist eine Funktion von OPC UA, bei der (historische) Variablenwerte in einem Speicher (zum Beispiel einer Datei oder einer Datenbank) vorgehalten und über eine standardisierte Schnittstelle Clients zugänglich gemacht werden. Hierbei kann konfiguriert werden, wie der TwinCAT OPC UA Server die Variablenwerte ausliest und abspeichert.

Voraussetzungen

Diese Konfiguration kann für Variablen aus beliebigen Laufzeitsystemen [▶ 50] (zum Beispiel der TwinCAT SPS oder TwinCAT 3 C++, ...) durchgeführt werden. Als Voraussetzung muss die jeweilige Variable zunächst für OPC UA freigegeben werden. Wie dies geschieht erfahren Sie in unserem Quick Start [▶ 21] Tutorial.

Speichertypen

Die einzelnen Speichertypen werden als sogenannte „HistoryAdapter“ im TwinCAT OPC UA Server konfiguriert. Dies geschieht über den TwinCAT OPC UA Server Konfigurator. Jeder HistoryAdapter (außer dem Adapter-Typ „Arbeitsspeicher“ (Volatile)) kann hierbei mehrfach verwendet werden, z. B., wenn die historischen Werte für einzelne Variablen in unterschiedlichen Datenspeichern abgelegt werden sollen.

Sampling der Variablenwerte

Der TwinCAT OPC UA Server kann Variablenwerte auf zwei verschiedene Arten erhalten:

1. Über die eigene Sampling Engine und
2. Von einem OPC UA Client (über die sogenannte „HistoryUpdate“ Funktion)

Bei der eigenen Sampling Engine wird für jede Variable konfiguriert, mit welcher Samplingrate die Variable aus dem unterlagerten Echtzeitsystem gesampled und in dem Speicher abgelegt werden soll. Als Zeitstempel wird die Zeit verwendet, zu der der Server den Variablenwert aus der Echtzeit ausgelesen hat.

Bei der HistoryUpdate Funktion hingegen erhält der Server sowohl den Variablenwert als auch den Zeitstempel von einem verbundenen OPC UA Client und führt somit kein eigenes Sampling durch.

4.11.2 Unterstützte Funktionen

Es gelten die folgenden Voraussetzungen zur Nutzung von Historical Access:

- Das Laufzeitsystem, dessen Symbole für Historical Access gespeichert werden sollen, muss für Data Access konfiguriert sein (ebenso müssen die jeweiligen Variablen freigegeben sein).
- Um eine SQL-Compact-Datenbank als Speichermedium nutzen zu können, muss auf dem Rechner, auf dem der OPC UA Server läuft, SQL Compact Runtime 3.5 SP2 installiert sein.
- SQL Compact Datenbanken werden auch unter Windows CE unterstützt.
- MS SQL Server Datenbanken werden unter Windows CE nicht unterstützt.
- Es werden die folgenden MS SQL Server Versionen unterstützt: 2017, 2019
- Es gelten die folgenden Empfehlungen bei Verwendung des jeweiligen Speichertyps:

  Arbeitsspeicher: Anzahl der Einträge < 5000

  Dateisystem: Anzahl der Einträge < 10000

  Datenbank: Anzahl der Einträge >= 10000

Speichertypen

Es werden die folgenden Speichertypen unterstützt:

Speichertyp	TF6100 Server Setup Version	Betriebssysteme	Beschreibung
Arbeitsspeicher	4.x und 5.x	Windows, Windows CE, TwinCAT/BSD	Speichert die Werte im Arbeitsspeicher des Geräts, auf dem der OPC UA Server läuft. Es sind keine weiteren Parameter erforderlich. Nach einem Neustart des Geräts sind die gespeicherten Werte nicht mehr verfügbar. Das Speichermedium ist demzufolge nicht persistent. Es gelten die oben aufgeführten Anforderungen und Empfehlungen.
Dateisystem	4.x	Windows, Windows CE	Speichert die Werte in mehreren Dateien, deren Speicherort festgelegt werden kann. Jedem für dieses Speichermedium konfigurierten Symbol wird eine eigene Datei in diesem Verzeichnis zugeordnet. Darüber hinaus besteht eine Sicherungskopie der Datei für jedes Symbol, die dann erzeugt wird, sobald die Puffergröße erreicht ist. Der Inhalt der Datendatei wird dann als Sicherungskopie gespeichert und es wird eine neue Datei erzeugt. Es gelten die oben aufgeführten Anforderungen und Empfehlungen.
SQL Compact Datenbank	4.x	Windows, Windows CE	Speichert die Werte in eine SQL Compact Database, dessen Speicherort festgelegt werden kann. Es gelten die oben aufgeführten Anforderungen und Empfehlungen.
MS SQL Server Datenbank	4.x	Windows	Speichert die Werte in einer SQL Server Database, die mit verschiedenen Parametern referenziert wird. Es gelten die oben aufgeführten Anforderungen und Empfehlungen.
TwinCAT Analytics [▶ 97]	5.x	Windows, TwinCAT/BSD	Speichert die Werte in einem Dateiformat, welches dem TwinCAT Analytics Speicherformat entspricht. Die Daten können somit mit der TwinCAT Analytics Toolchain weiter verwendet werden. Dieses Format ersetzt seit TF6100 Version 5.x das alte Datei-basierte Speicherformat (s.o.).

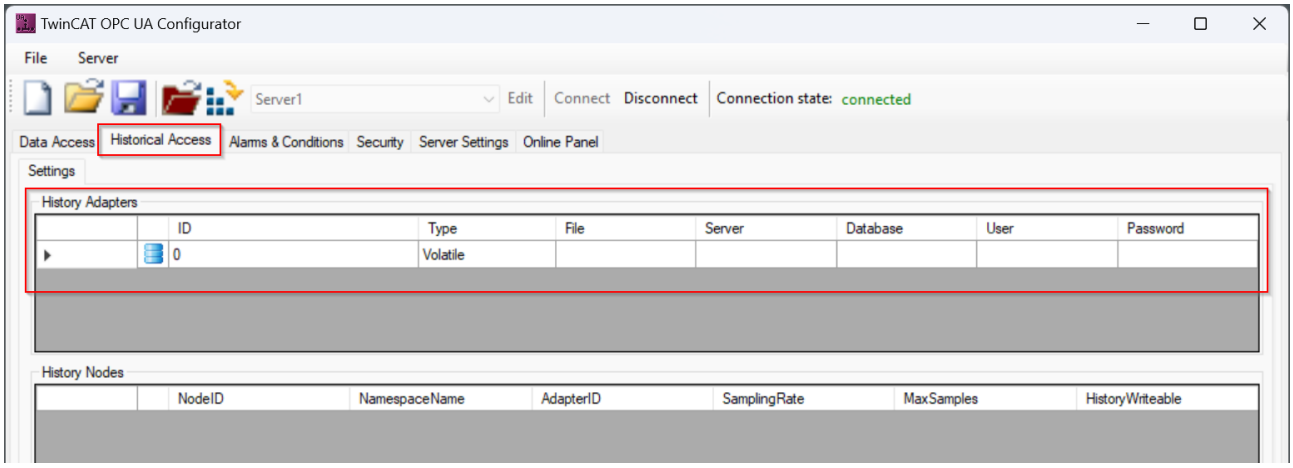
4.11.3 Konfiguration

Zur Konfiguration der Historical-Access-Funktionalität können Sie den TwinCAT OPC UA Configurator verwenden.
Zur Konfiguration eines Symbols für Historical Access müssen Sie drei Schritte durchführen:

1. Das Symbol muss für Data Access freigegeben worden sein (siehe Kapitel Freigabe von Symbolen [▶ 53]).
2. Ein Speichertyp muss angelegt werden, in welchem die Symbolwerte abgespeichert werden sollen.
3. Konfigurationsparameter für das Symbol müssen festgelegt werden.

Anlegen eines Speichertyps

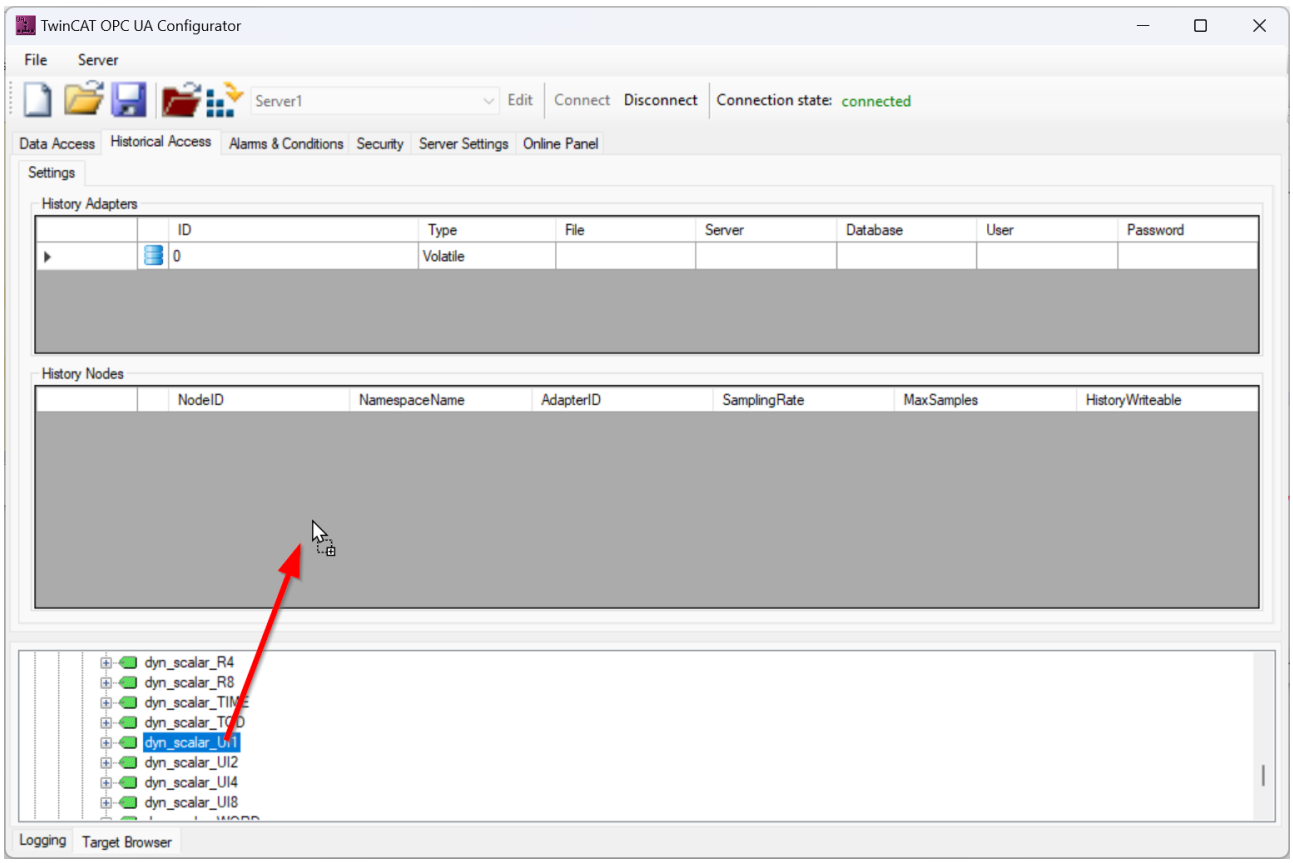
In der Standalone-Variante des TwinCAT OPC UA Configurators können Sie über die Registerkarte **Historical Access** die Konfiguration der Speichertypen (sogenannte „History Adapter“) durchführen. In dem folgenden Screenshot ist beispielsweise der Speichertyp „Arbeitsspeicher“ („Volatile“) konfiguriert worden.



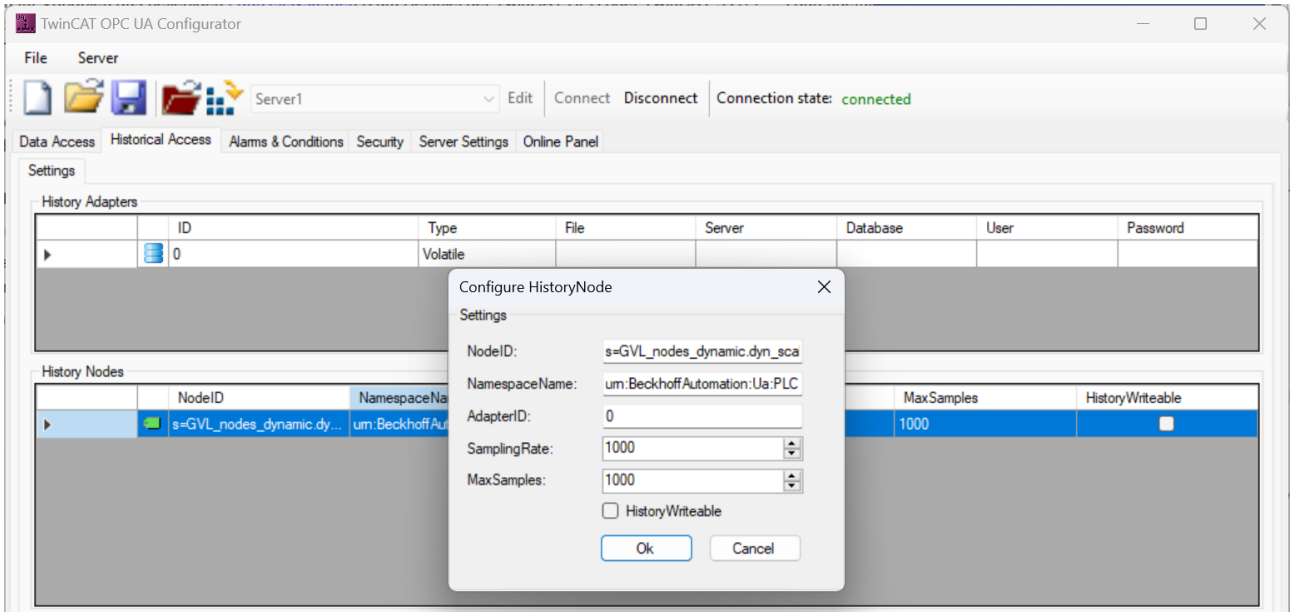
Über das Kontextmenü lassen sich weitere Speichertypen anlegen oder existierende Speichertypen editieren bzw. aus der Konfiguration entfernen.

Konfigurieren eines Symbols

In dem Bereich „History Nodes“ desselben Dialogs können Sie die Konfiguration der Symbole durchführen und diese mit einem Speichertyp verknüpfen. Sie können entweder ein Symbol manuell anlegen (sofern Sie dessen „Adresse“ bzw. NodeID kennen) oder einfach ein existierendes Symbol über den integrierten Target Browser per Drag-and-Drop in den Konfigurationsbereich ziehen.



Das Symbol wird dann mit Standard-Konfigurationswerten hinzugefügt. Durch einen Doppelklick auf den Symboleintrag können Sie diese Konfigurationswerte modifizieren. Hier können Sie dann auch die Verknüpfung zu einem Speichertyp vornehmen, welcher über dessen ID referenziert wird.



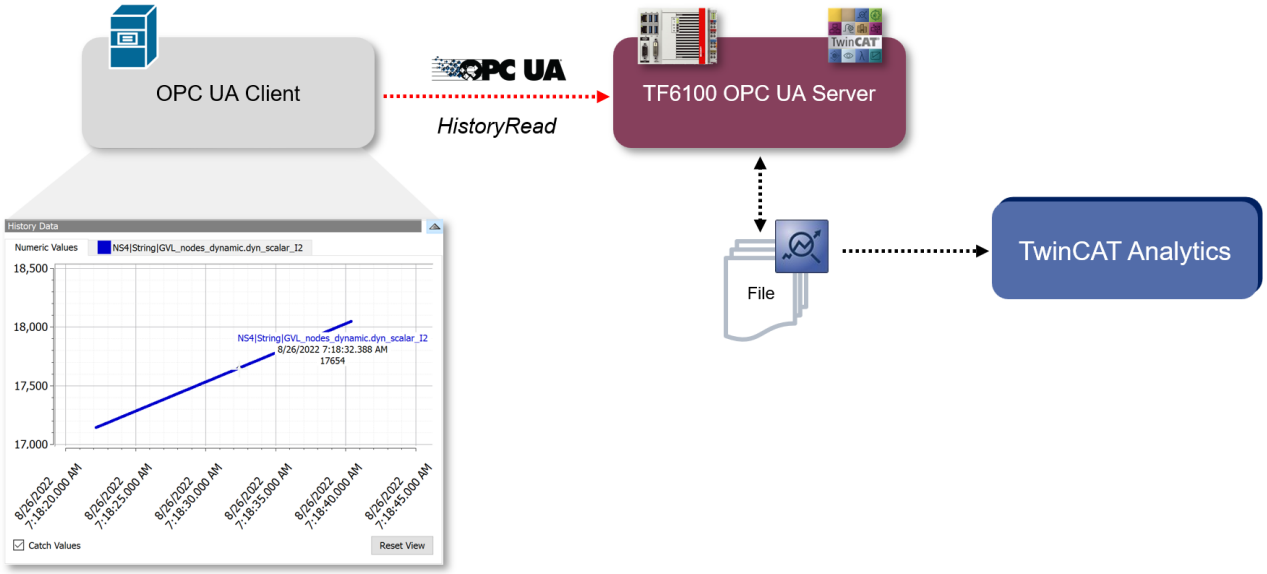
4.11.4 HistoryUpdate

Bei der HistoryUpdate-Funktion sampled der Server nicht eigenständig Variablenwerte und speichert diese für Historical Access ab. Anstelle dessen nimmt der Server sowohl die Variablenwerte als auch die Zeitstempel von einem OPC UA Client entgegen.

Eine Variable kann für diese Funktion aktiviert werden, indem Sie im TwinCAT OPC UA Configurator das Attribut `HistoryWriteable="true"` setzen. In diesem Fall führt der Server kein eigenes Sampling mehr für diese Variable durch und „wartet“ darauf, dass ein OPC UA Client die Werte plus Zeitstempel liefert. Ein solcher Aufruf kann z. B. vom TwinCAT OPC UA Client über den Funktionsbaustein UA_HistoryUpdate erfolgen.

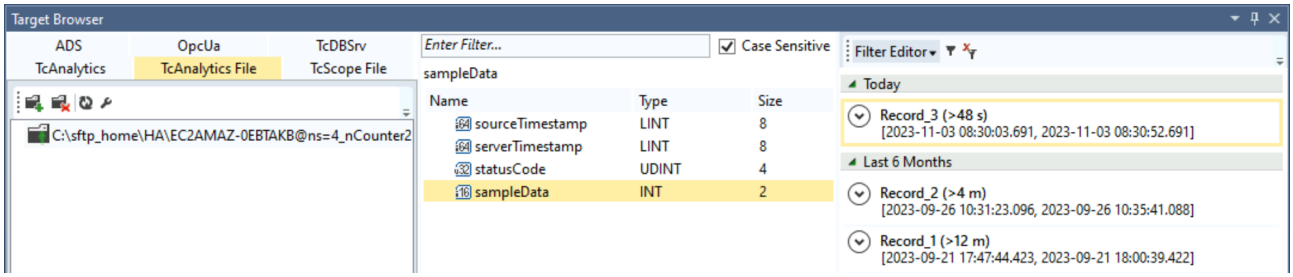
4.11.5 TwinCAT Analytics

Bei der Verwendung des Speichertyps „TwinCAT Analytics“ wird als Speicher ein Binärfile geschrieben, dessen Format dem TwinCAT Analytics Datenformat entspricht. Obwohl dies für den OPC UA Client keinerlei Relevanz hat (denn dieser greift ja über OPC UA auf die historischen Daten zu), so ergeben sich hieraus jedoch interessante Möglichkeiten zur weiteren Nutzung der historischen Daten, denn Sie können das Binärfile in die TwinCAT Analytics Toolchain einlesen und dort weiter verarbeiten.
Das folgende Schaubild verdeutlich diesen Zusammenhang.



TwinCAT Target Browser

Wenn Sie in Ihrer Historical-Access-Konfiguration das TwinCAT-Analytics-Speicherformat verwenden, können Sie die gespeicherten Werte auch über die TwinCAT Analytics Toolchain abrufen, zum Beispiel dem TwinCAT Target Browser. Fügen Sie hierzu das Verzeichnis, welches Sie in Ihrer Konfiguration zur Speicherung der historischen Werte festgelegt haben, in der Registerkarte **TcAnalytics File** vom TwinCAT Target Browser hinzu. Die gespeicherten Werte werden dann in der gewohnten TwinCAT Analytics Form als Records dargestellt und können weiterverarbeitet werden.



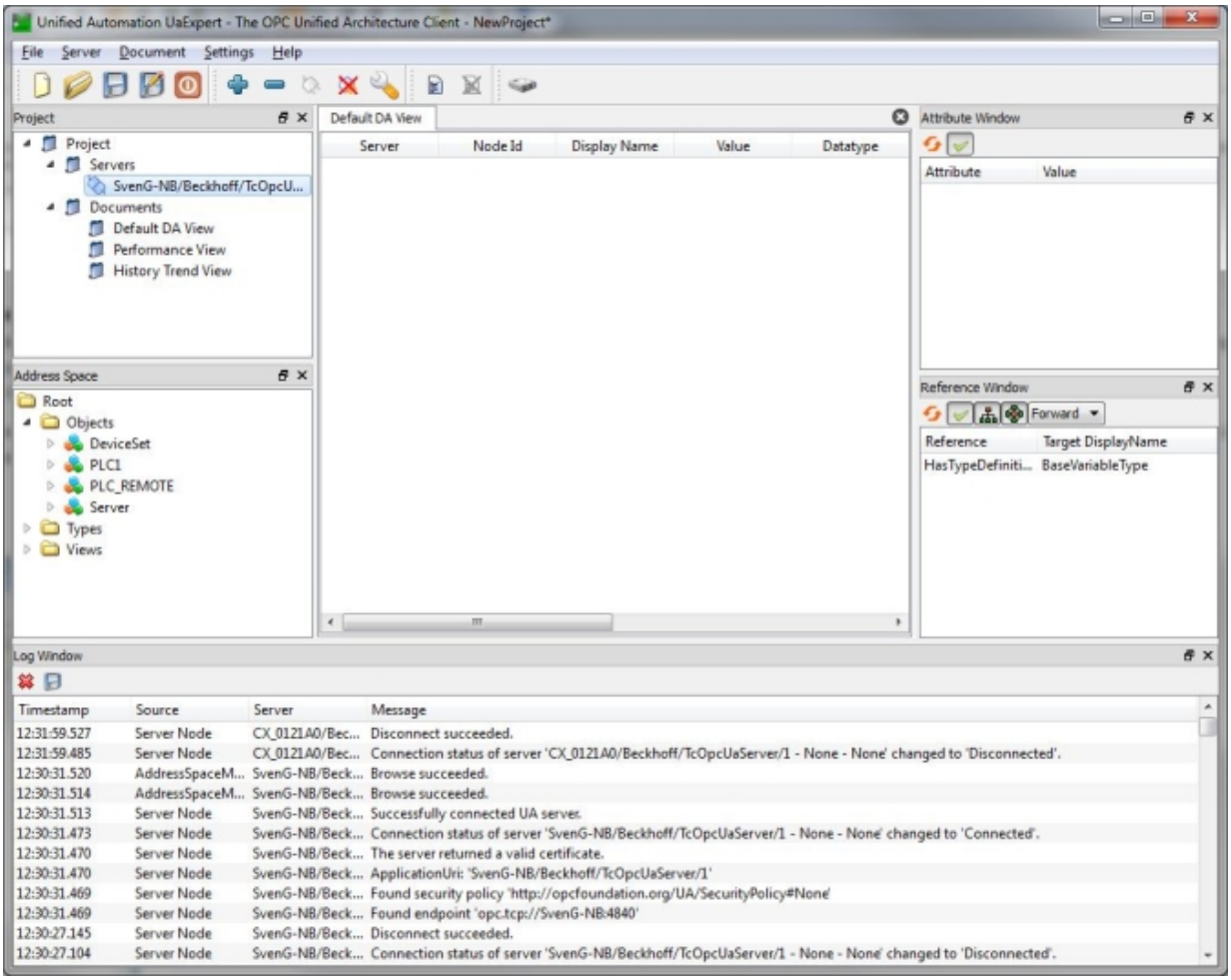
4.11.6 Zugriff auf historische Daten

Sie können zum Abrufen der historischen Daten jeden OPC UA Client verwenden, der die OPC UA Historical Access Funktion unterstützt. Im folgenden Beispiel wird die Software „UA Expert“ von Unified Automation verwendet, um historische Daten aus dem TwinCAT OPC UA Server auszulesen und visuell darzustellen.

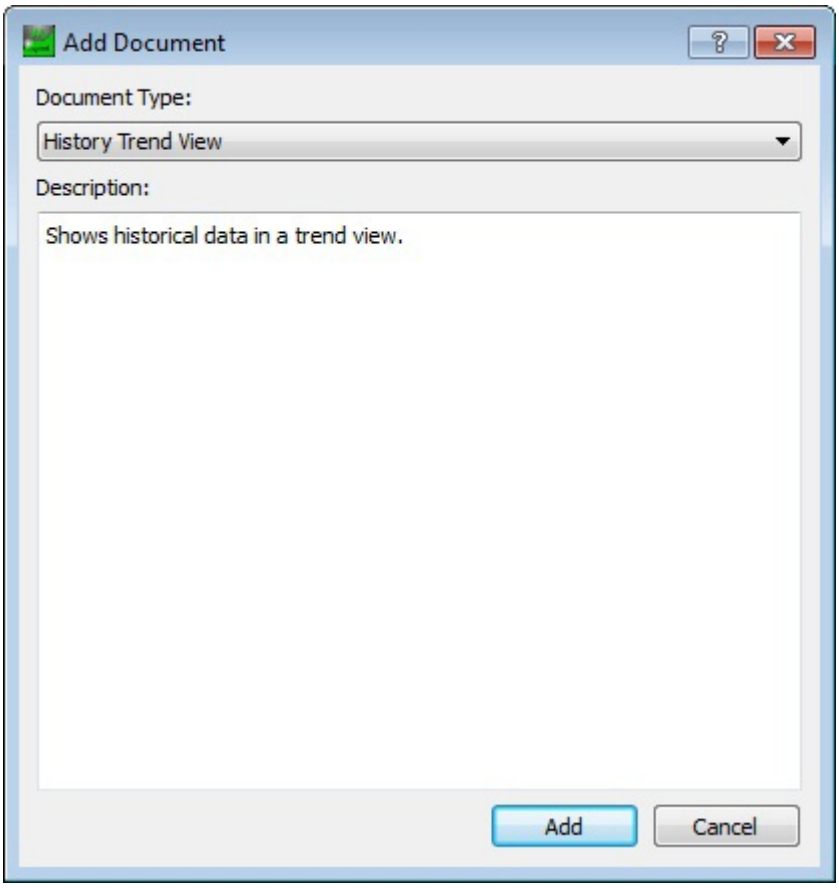
Historical-Access-Werte in einem OPC UA Client anzeigen

Die folgende Schritt-für-Schritt-Anleitung beschreibt, wie Sie die UA-Expert-Software konfigurieren, um auf historische Daten zuzugreifen.

1. Starten Sie die UA-Expert-Software und stellen Sie eine Verbindung mit dem OPC UA Server her.

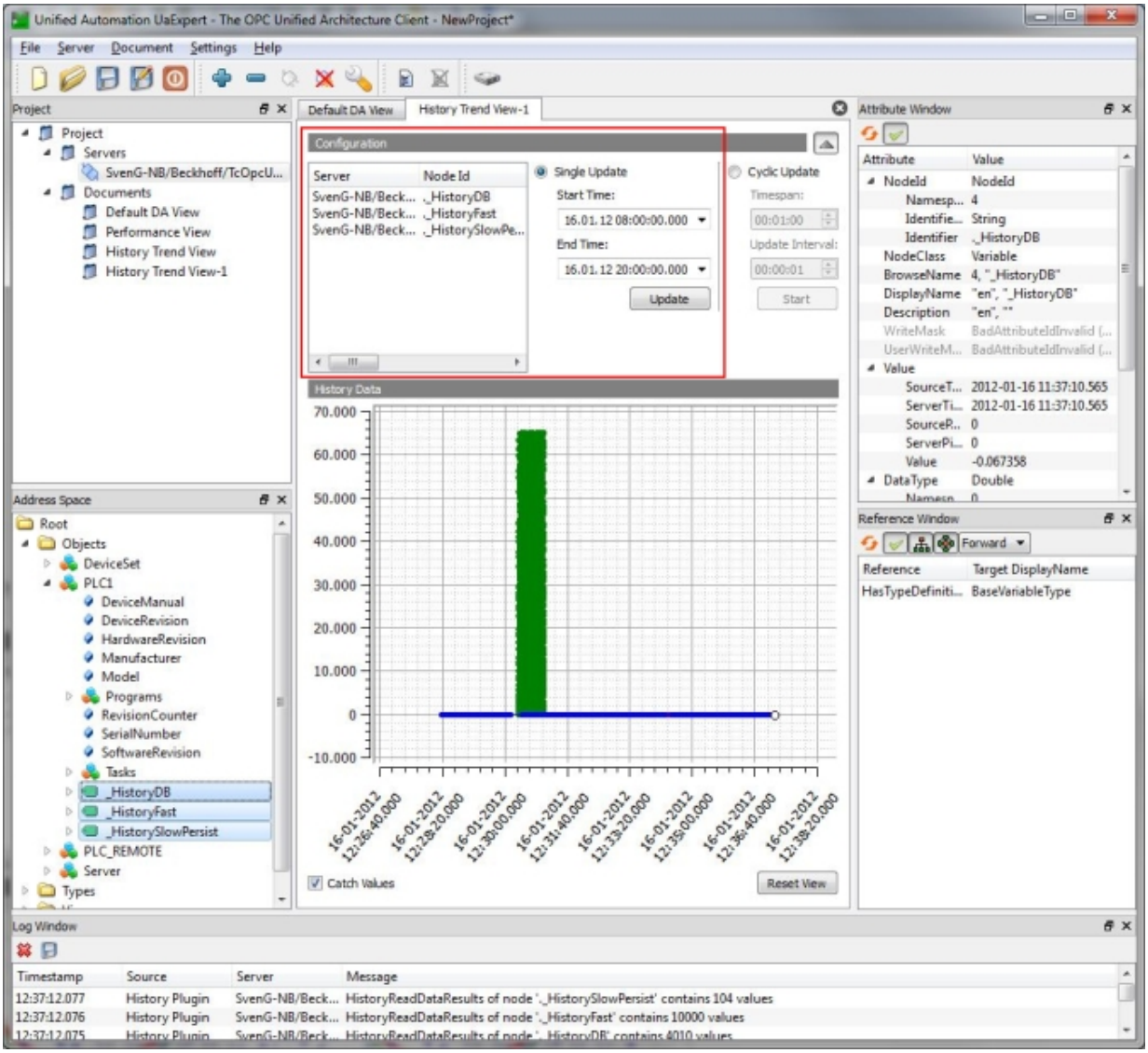


2. Fügen Sie eine neue **History Trend Ansicht** hinzu.



3. Durchsuchen Sie den PLC1-Namensraum und fügen Sie die SPS-Variablen _HistoryDB, _HistoryDBcompact, _HistoryFast und _HistorySlowPersist per Drag-and-drop hinzu.

⇨ Sie können nun den gewünschten Zeitraum, für den die Symbolwerte angezeigt werden sollen, mithilfe der Steuerelemente **Start Time** und **End Time** festlegen, oder, falls erforderlich, ein **Cyclic Update** für diese Variablen starten.



4.12 Alarms and Conditions

4.12.1 Überblick

In diesem Kapitel werden die Konfigurationsschritte zur Verwendung von OPC UA **A**larms and **C**onditions (A&C) auf dem TwinCAT OPC UA Server beschrieben. OPC UA A&C beschreibt ein Modell für die Überwachung von Symbolwerten und das Ausgeben von Alarmen und Ereignissen bei entsprechenden Wertänderungen bzw. Überschreiten von Schwellenwerten eines Symbols.

Voraussetzungen

Diese Konfiguration kann für Variablen aus beliebigen Laufzeitsystemen [▶ 50] (zum Beispiel der TwinCAT SPS oder TwinCAT 3 C++, ...) durchgeführt werden. Als Voraussetzung muss die jeweilige Variable zunächst für OPC UA freigegeben werden. Wie dies geschieht erfahren Sie in unserem Quick Start [▶ 21] Tutorial.

4.12.2 Unterstützte Funktionen

Die folgende Tabelle listet die unterstützten Standard AlarmTypes der Alarms & Conditions Funktion des TwinCAT OPC UA Server auf.

AlarmType	Beschreibung
LimitAlarmType	Ermöglicht die Festlegung von verschiedenen Schwellenwerten für ein Symbol. Bei Überschreitung eines Schwellenwerts wird der Alarm mit einem konfigurierbaren Alarmtext und Schweregrad ausgelöst.
OffNormalAlarmType	Ermöglicht die Definition von „normal“- und „nicht-normal“-Schwellenwerten für ein Symbol. Weicht der Wert des Symbols von dem jeweils definierten Wert ab, wird der Alarm ausgelöst.

Zusätzlich zur Alarms & Conditions Funktion bietet der TwinCAT OPC UA Server auch die Möglichkeit, sich mit dem TwinCAT Eventlogger [▶ 114] zu verbinden und die dort empfangenen Alarme oder Events entsprechend als OPC UA Alarm oder Event weiterzureichen.

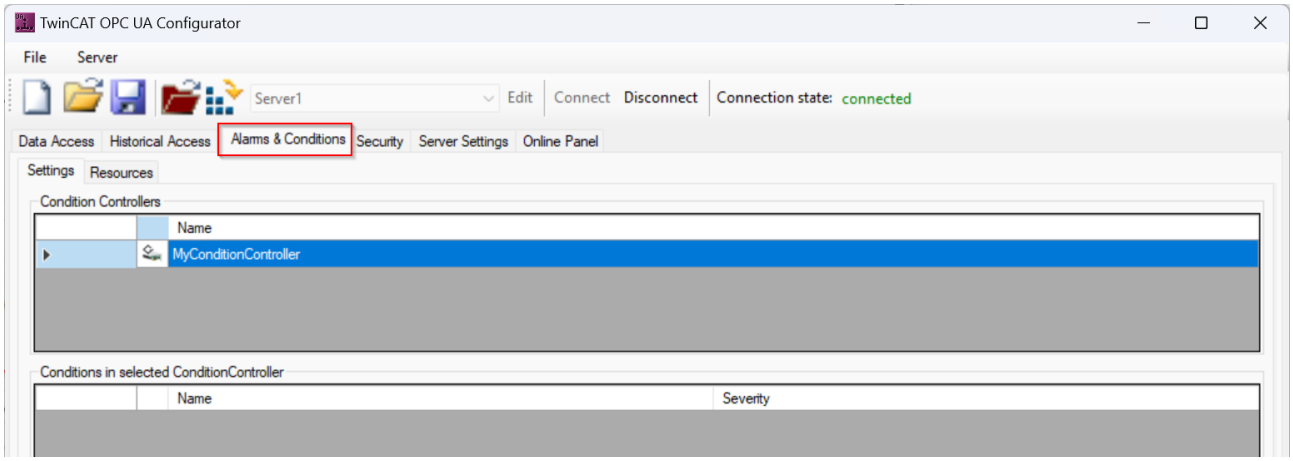
4.12.3 Konfiguration

Zur Konfiguration der Alarms & Conditions Funktionalität können Sie den TwinCAT OPC UA Configurator verwenden. Zur Konfiguration eines Symbols für Alarms & Conditions müssen Sie zwei Schritte durchführen:

1. Das Symbol muss für Data Access freigegeben worden sein (siehe Kapitel Freigabe von Symbolen [▶ 53]).
2. Ein ConditionController muss angelegt werden, welcher eine Gruppierung von Alarmen und Events ermöglicht.
3. Alarm- und Eventtexte müssen angelegt werden.
4. Konfigurationsparameter für das Symbol müssen festgelegt werden.

Anlegen eines ConditionController

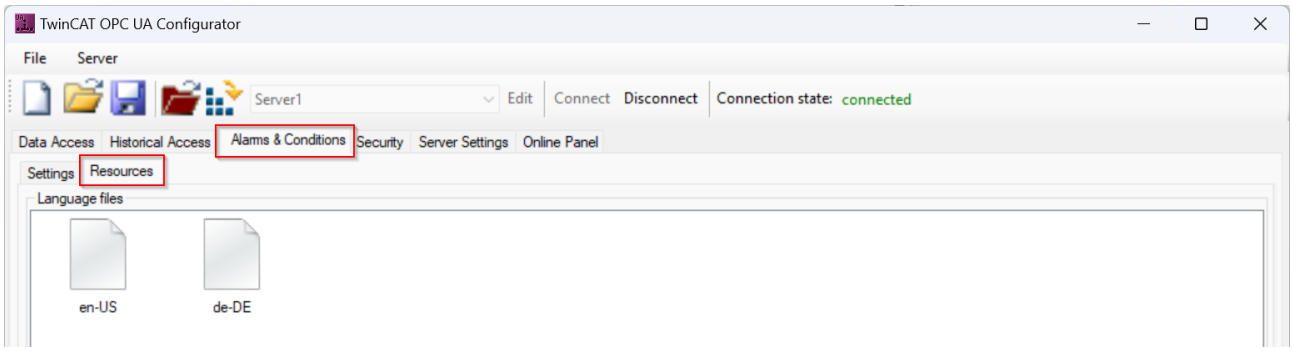
In der Stand-alone Variante des TwinCAT OPC UA Configurators können Sie über die Registerkarte **Alarms & Conditions** die Konfiguration der ConditionController durchführen. Im folgenden Screenshot ist zum Beispiel ein ConditionController mit dem Namen „MyConditionController“ konfiguriert worden.



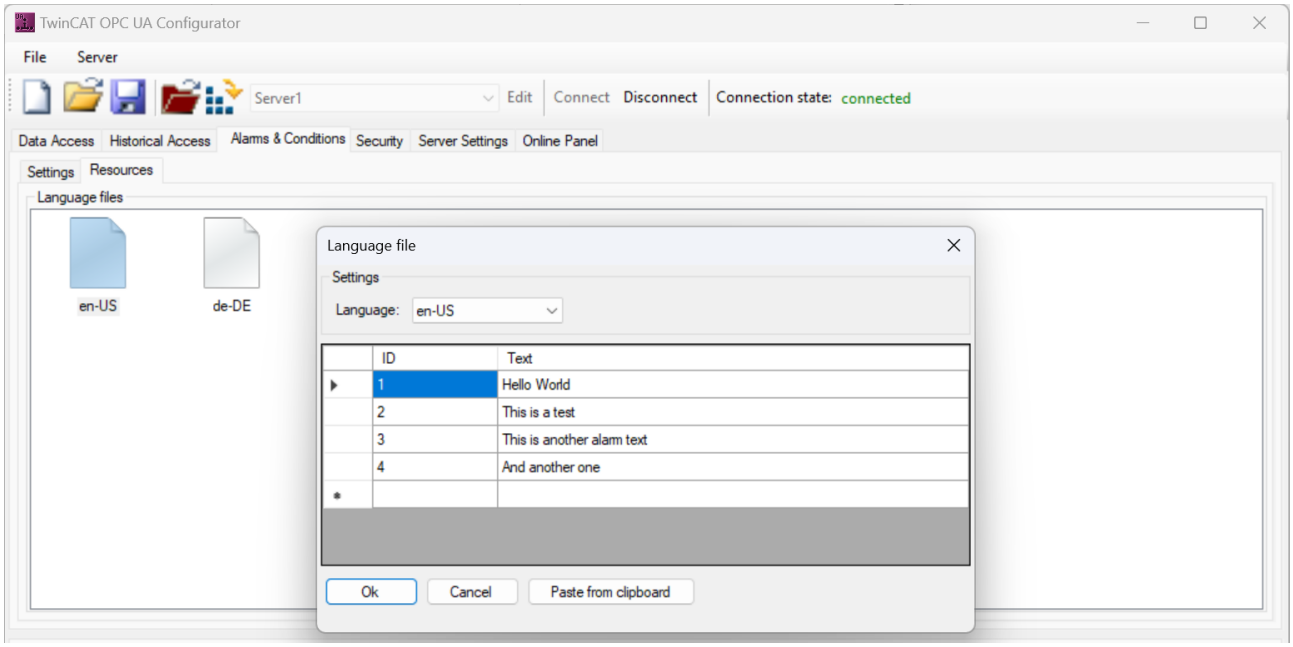
Über das Kontextmenü lassen sich weitere ConditionController anlegen oder existierende ConditionController editieren bzw. aus der Konfiguration entfernen.

Anlegen von Alarm- und Eventtexten

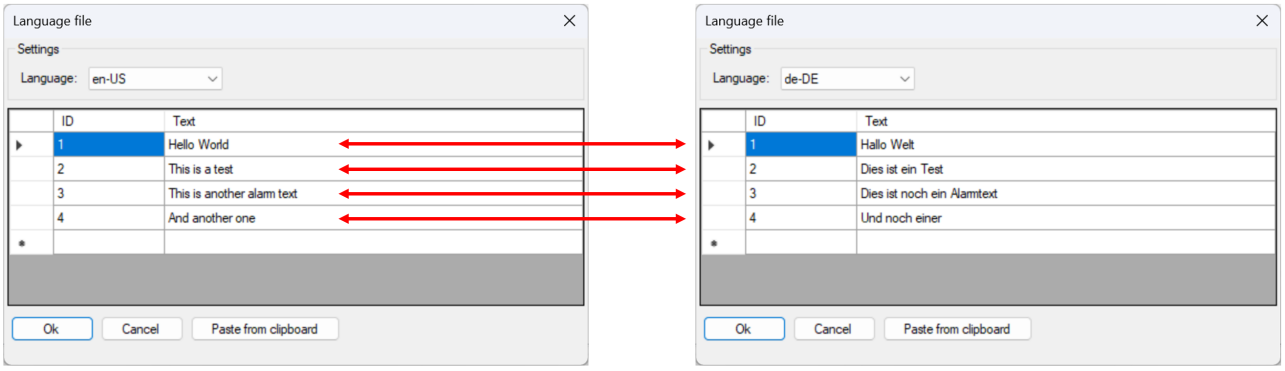
Die Texte, welche beim Auslsen eines Alarms oder Events verwendet werden sollen, können als sogenannte „Ressourcen“ im TwinCAT OPC UA Server hinterlegt werden. Diese Texte können mehrsprachig definiert werden, wobei die Sprache durch das jeweilige Landeskennzeichen (engl. „Country Code“) identifiziert wird. Im TwinCAT OPC UA Configurator werden diese Ressourcen über die Registerkarte **Resources** in der Alarms & Conditions Konfiguration hinzugefügt.



Über das Kontextmenü können Sie weitere Sprachen als Ressource hinzufügen, löschen oder editieren. Im obigen Screenshot sind zwei Sprachen als Ressource konfiguriert worden: de-DE (Deutsch/Deutschland) und en-US (Englisch/USA). Die Texte werden innerhalb einer Sprache tabellarisch gepflegt und können mittels Copy-and-Paste auch aus Microsoft Excel übernommen werden.



Die konfigurierten Texte werden sprachübergreifend über deren ID zugeordnet, zum Beispiel:

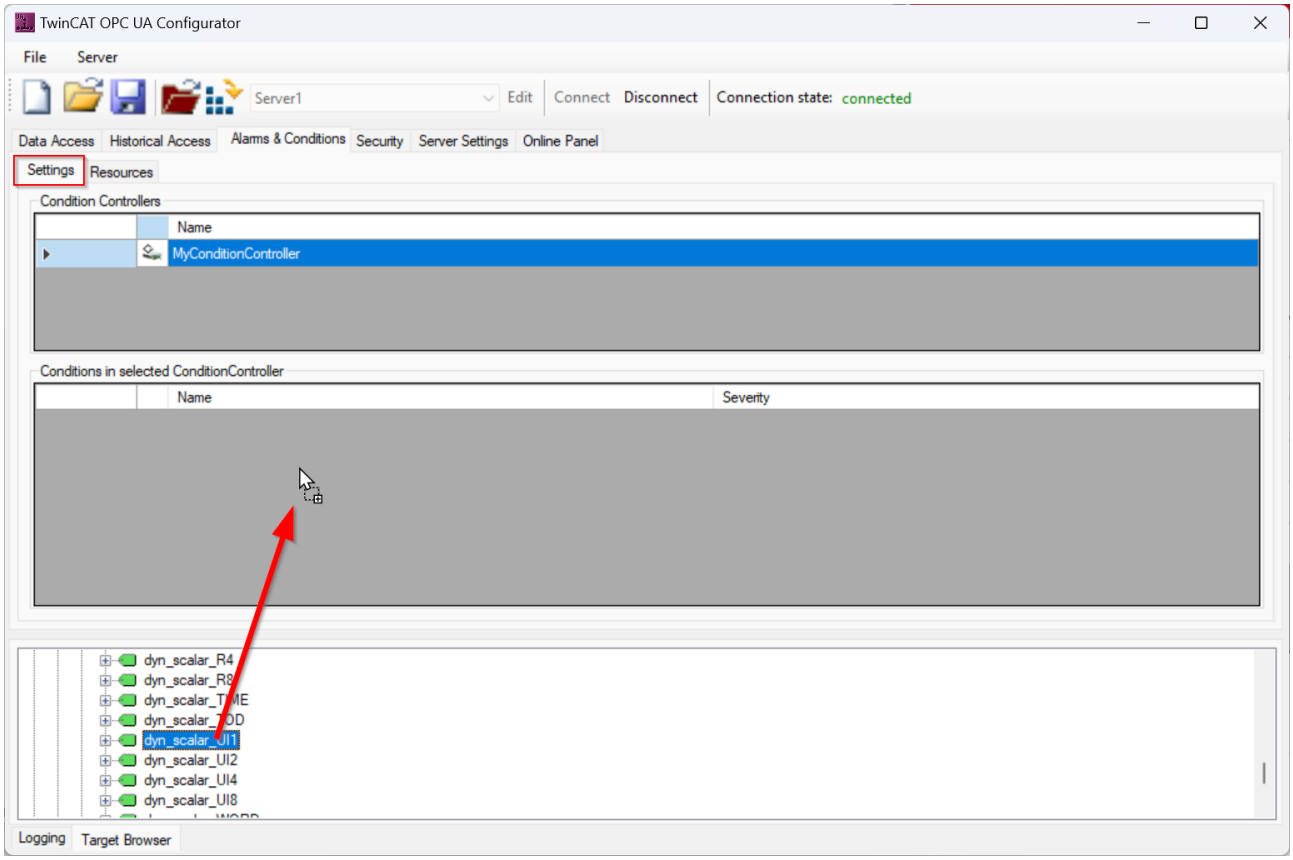


Verwendung der Landessprache bei einem Alarm/Event

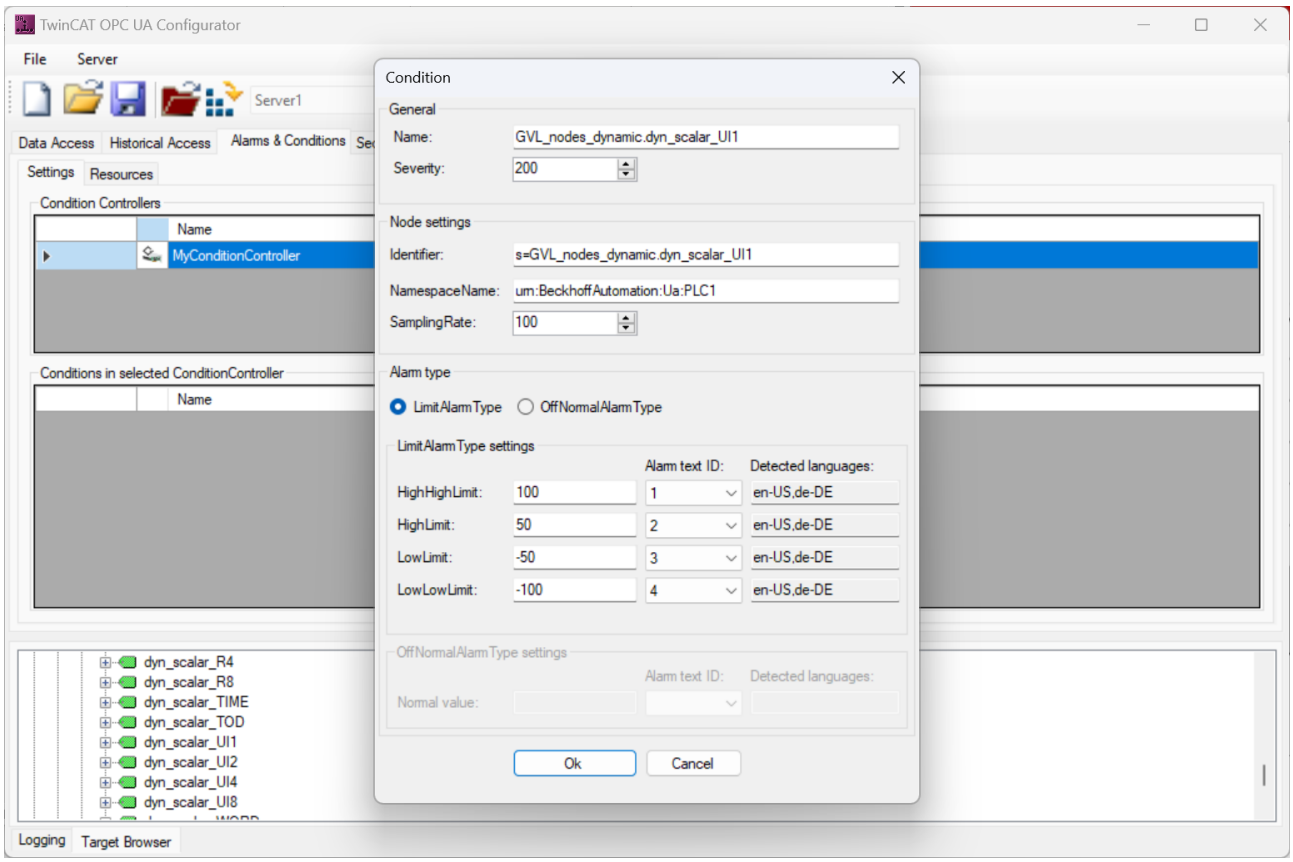
Welcher Sprachtext beim Auslösen eines Alarms oder Events verwendet wird, hängt von der verwendeten Landeskennung ab, mit welcher sich ein OPC UA Client beim Server verbindet.

Konfigurieren eines Symbols

Nachdem die Alarm- und Eventtexte konfiguriert wurden, können Sie diese auch für die Konfiguration eines Symbols verwenden. Die Symbolkonfiguration wird in der Registerkarte **Settings** durchgeführt. Sie können ein Symbol manuell konfigurieren oder einfach über den integrierten Target Browser per Drag-and-Drop auf den Konfigurationsbereich ziehen.



Das Symbol wird dann zu dem selektierten ConditionController hinzugefügt. In dem darauf folgenden Konfigurationsdialog können Sie den zu konfigurierenden AlarmTyp sowie die Schwellenwerte definieren und auch die Sprachtexte selektieren.

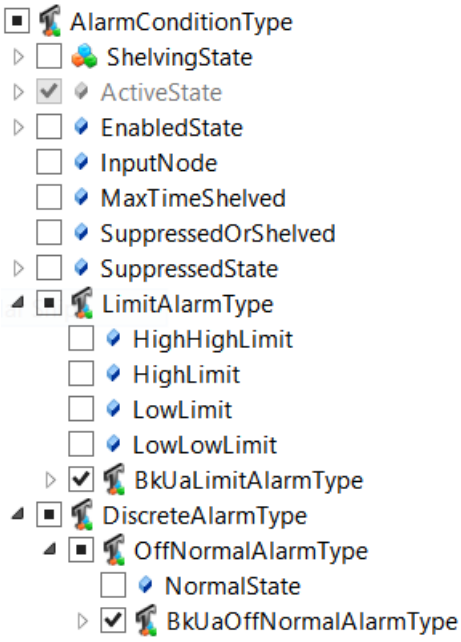


4.12.4 Zusätzliche Applikationsdaten

Möchten Sie zusätzliche Daten aus Ihrer Applikation an einen Alarm anhängen, so können Sie dies über die folgenden, speziellen AlarmTypes konfigurieren:

AlarmType	Abgeleitet von
BkUaLimitAlarmType	LimitAlarmType
BkUaOffNormalAlarmType	OffNormalAlarmType

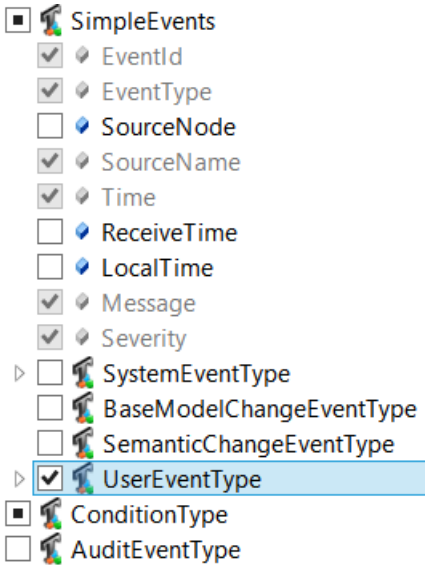
In diesen AlarmTypes sind zusätzliche Felder definiert, welche Sie mit Werten aus Ihrer Applikation füllen können. Soll ein OPC UA Client diese zusätzlichen Werte verwenden können, so muss dieser die entsprechenden AlarmTypes abonnieren und interpretieren.



Der OPC UA Client empfängt dann die zusätzlichen Applikationsdaten in den Feldern BkUaEventData und BkUaEventValue des eingehenden Alarms, zum Beispiel:

ConditionId	NodeId
NamespaceIndex	5
IdentifierType	String
Identifier	A&C|ConditionController1.CustomStruct
5:BkUaEventData	NodeId
SourceTimestamp	19.10.2015 15:42:20.461
SourcePicoseconds	0
ServerTimestamp	19.10.2015 15:42:20.461
ServerPicoseconds	0
StatusCode	Good
Value	ST_SomeStruct
Data1	1
Data2	2
Data3	3
5:BkUaEventValue	NodeId
SourceTimestamp	19.10.2015 15:42:20.461
SourcePicoseconds	0
ServerTimestamp	19.10.2015 15:42:20.461
ServerPicoseconds	0
StatusCode	Good
Value	12

Die benutzerdefinierten EventFields werden als „UserEventData“ angehängt. Diese Daten können von OPC UA Clients empfangen werden, die bei dem SimpleEventType „UserEventType“ angemeldet sind.



In der SPS müssen Sie zur Verwendung dieser Funktion eine Struktur definieren, welche sowohl den zu überwachenden Symbolwert enthält als auch die zusätzlichen Werte, die beim Auslösen des Alarms mit verschickt werden sollen. Diese Struktur muss wie folgt definiert werden:

```
TYPE ST_CustomStruct :
STRUCT
  value : INT;
  data  : ST_SomeStruct;
END_STRUCT
END_TYPE
```

```
TYPE ST_SomeStruct :
STRUCT
  Data1 : INT;
  Data2 : REAL;
  Data3 : LREAL;
END_STRUCT
END_TYPE
```

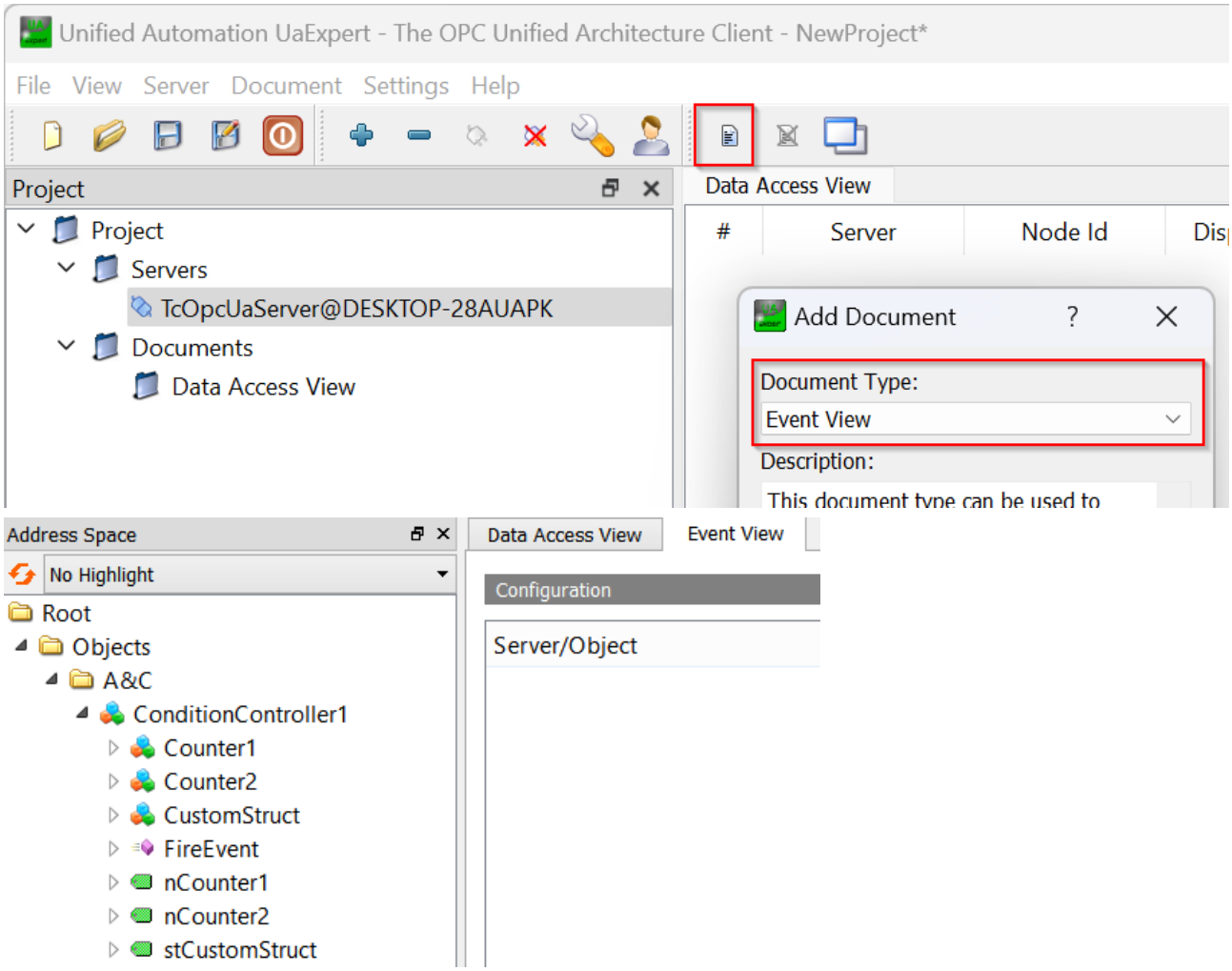
Die Instanz der Struktur ST_CustomStruct wird dann als Symbol für Data Access freigegeben [▶ 53]. Zusätzlich muss die Struktur als StructuredType [▶ 75] aktiviert werden.

4.12.5 Zugriff auf Alarme und Events

Zum Empfangen von Alarmen und Events muss sich ein OPC UA Client auf dem TwinCAT OPC UA Server bei einem der konfigurierten ConditionController anmelden. Der Client empfängt dann die Alarme und Events für Symbole, die in diesem ConditionController konfiguriert wurden.

In dem folgenden Beispiel sehen Sie, wie Sie mit Hilfe der Software UA Expert von Unified Automation einen ConditionController abonnieren können, um von diesem Alarme und Events zu empfangen.

Nachdem Sie den UA Expert gestartet und eine Verbindung zum TwinCAT OPC UA Server hergestellt haben, fügen Sie Ihrem Arbeitsbereich eine neue Dokumentenansicht vom Typ „Event View“ hinzu.



Im Adressraum des Servers finden Sie alle konfigurierten ConditionController unterhalb des Knotens „A&C“. Sie können nun per Drag-and-Drop einen ConditionController aus dem Arbeitsbereich des „Event View“ ziehen, um diesen zu abonnieren. Die Alarme und Events für diesen ConditionController werden dann im Event Window angezeigt.

Events | Alarms | Event History

A	C	Time	Severity	Server/Objec	SourceName	Message	EventType	Active
⚠		11:41:54.553	300	TcOpcUaSe...	ConditionC...	Value is High	LimitAlarm...	
		11:58:04.415	500	TcOpcUaSe...	Server		RefreshStart...	
⚠		11:41:54.553	300	TcOpcUaSe...	ConditionC...	Value is High	LimitAlarm...	
		11:58:04.415	500	TcOpcUaSe...	Server		RefreshEnd...	
		12:44:09.875	500	TcOpcUaSe...	Server		RefreshStart...	
⚠		11:41:54.553	300	TcOpcUaSe...	ConditionC...	Value is High	LimitAlarm...	
		12:44:09.875	500	TcOpcUaSe...	Server		RefreshEnd...	

4.13 Methodenaufrufe

4.13.1 Überblick

Für den Aufruf von OPC-UA-Methoden stehen zwei verschiedene Konzepte in der SPS zur Verfügung:

1. RPC-Methoden [▶ 110] und
2. Job-Methoden [▶ 107]

Der grundlegende Unterschied zwischen beiden Konzepten ist der Kontext der Abarbeitung in der SPS, sowie die Implementierung im SPS-Code.
Die folgende Tabelle veranschaulicht diesen Zusammenhang.

Typ	Abarbeitung	SPS-Code
RPC-Methode	Innerhalb eines SPS-Zyklus	OPC-UA-Methode wird durch eine SPS-Methode repräsentiert.
Job-Methode	Über mehrere SPS-Zyklen	OPC-UA-Methode wird durch einen SPS-Funktionsbaustein repräsentiert.

Allgemein gesprochen kann man definieren: Job-Methoden sind das Mittel der Wahl, wenn ein Methodenaufruf „lange“ dauert, d. h. mehrere SPS-Zyklen umspannen kann. Die Abarbeitung von RPC-Methoden muss hingegen „schnell“ erfolgen, ansonsten drohen Zyklusüberschreitungen. **Üblicherweise sind Job-Methoden somit das Mittel der Wahl.** Die Programmierung in der SPS ist etwas aufwändiger, jedoch ist die Entkopplung vom SPS-Zyklus ein großer Vorteil der diesen Aufwand durchaus rechtfertigt.

4.13.2 Job-Methoden

Das Konzept der Job-Methoden hat im Vergleich zu den regulären Methodenaufrufen einen grundlegenden Unterschied: Die OPC-UA-Methoden werden nicht mehr 1:1 auf eine SPS-Methode abgebildet, sondern anstelle dessen auf einen Funktionsbaustein mit einer bestimmten Signatur. Hierdurch können auch Methodenaufrufe realisiert werden, welche aus Sicht einer SPS-Anwendung länger als einen Zyklus dauern.

Voraussetzungen

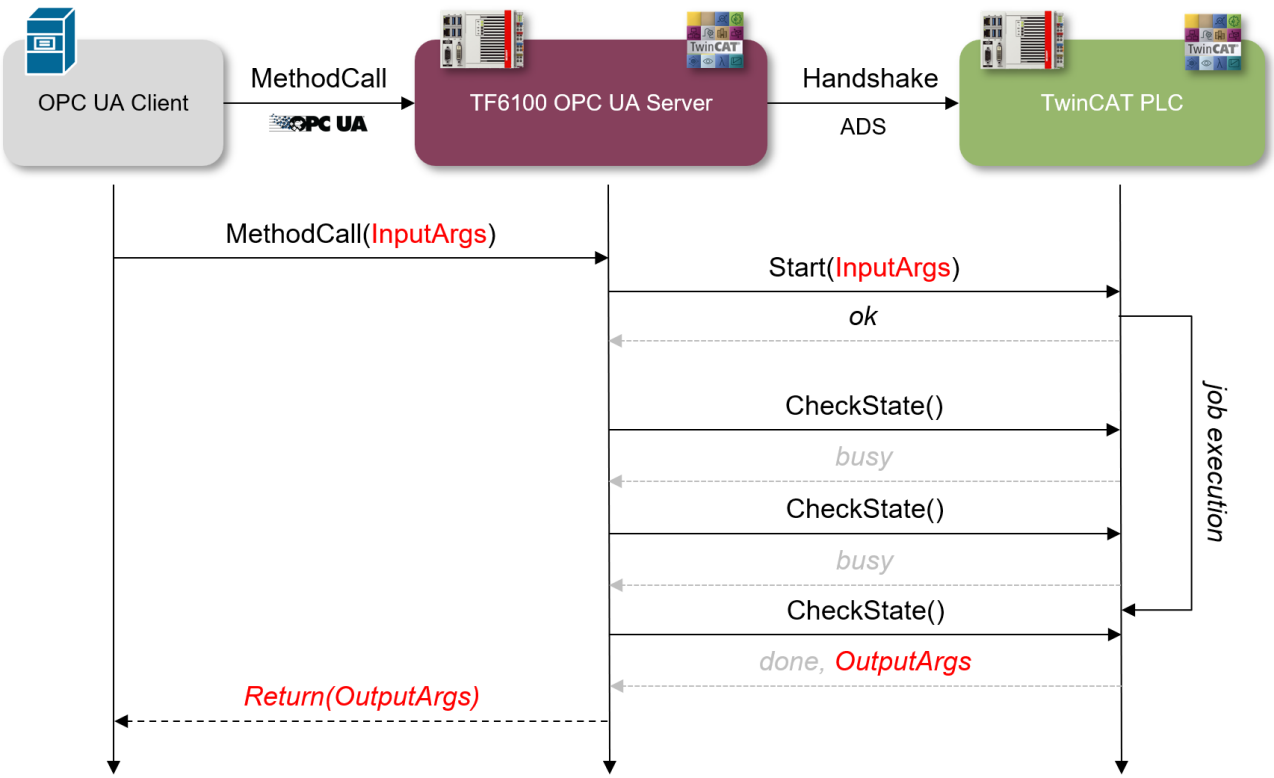
Diese Funktionalität steht nur für Data-Access [▶ 49]-Geräte basierend auf TwinCAT 3 und dem Import von TMC-Symboldateien [▶ 53], sowie der Online-Symbolik zur Verfügung.

Der SPS-seitige Aufbau einer solchen Job-Methode ist hierbei wie folgt definiert: Es existiert ein Funktionsbaustein, welcher über ein SPS-Attribut als Job-Methode definiert wird. Der Funktionsbaustein beinhaltet dann verschiedene SPS-Methoden, auf welche vom TwinCAT OPC UA Server in Form eines Handshake-Mechanismus zugegriffen wird, um diese als OPC-UA-Methode bereitstellen zu können.

Methode	Beschreibung
Start	Wird vom Server aufgerufen, sobald ein OPC UA Client die OPC-UA-Methode aufruft. Beinhaltet die Eingabeparameter des OPC-UA-Methodenaufrufs als VAR_INPUT. Über den HRESULT Rückgabewert der Methode kann direkt ein OPC UA Status Code in dessen Dezimalrepräsentation zurückgegeben werden, z. B. „0“ für den Status Code „Good“. Als Wert wird hierbei der entsprechend in der OPC-UA-Spezifikation definierte numerische Wert eines Status Codes verwendet. Eine Definition aller verfügbaren Status Codes kann hier eingesehen werden: http://www.opcfoundation.org/UA/schemas/StatusCode.csv Typischerweise returniert diese Methode den Wert „0“ (Good). Der SPS-Entwickler kann sich jedoch auch dazu entscheiden, dass zum Beispiel die Eingabeparameter validiert werden sollen. Im Fehlerfall könnte man den OPC-UA-Methodenaufruf dann fehlschlagen lassen, z. B. mit dem Rückgabewert „2158690304“ (BadInvalidArguments).
CheckState	Wird zyklisch vom Server aufgerufen, um zu überprüfen, ob der Job noch in Bearbeitung ist oder nicht. Solange der Job noch in Bearbeitung ist, gibt diese Methode den Wert „Busy“ zurück, andernfalls „Done“. Ausgabeparameter für die OPC-UA-Methode werden hier als VAR_OUTPUT deklariert. Über den HRESULT Rückgabewert der Methode kann direkt ein OPC UA Status Code in dessen Dezimalrepräsentation zurückgegeben werden, z. B. „0“ für den Status Code „Good“. Als Wert wird hierbei der entsprechend in der OPC-UA-Spezifikation definierte numerische Wert eines Status Codes verwendet. Eine Definition aller verfügbaren Status Codes kann hier eingesehen werden: http://www.opcfoundation.org/UA/schemas/StatusCode.csv Bei einer erfolgreichen Abarbeitung des Jobs returniert diese Methode typischerweise den Wert „0“ (Status Code „Good“). Der SPS-Entwickler kann sich jedoch auch dazu entscheiden, dass im Falle eines Fehlers der OPC-UA-Methodenaufruf fehlschlagen soll, z. B. mit dem Rückgabewert „2151415808“ (BadOutOfRange) oder dem allgemeineren „2147483648“ (Bad).
Abort	Diese Methode wird vom Server aufgerufen, falls ein Job abgebrochen werden muss, zum Beispiel, wenn der Server heruntergefahren wird oder neu startet. Der SPS-Entwickler hat dann die Möglichkeit seinen SPS-Code entsprechend aufzuräumen.

Workflow

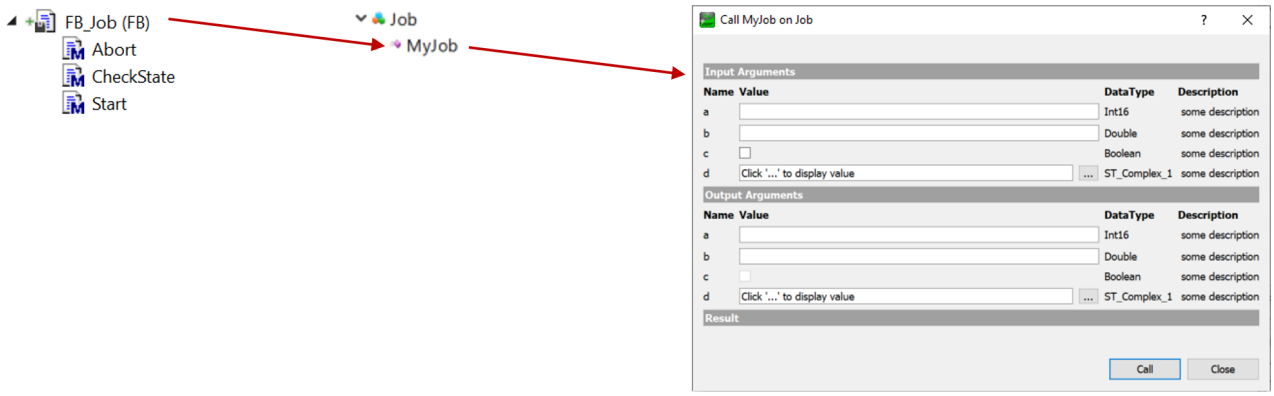
Der Handshake-Mechanismus zwischen Server und SPS kann vereinfacht wie folgt dargestellt werden:



Beispiel

Das folgende Beispiel ist in ausführbarer Form auch in den Samples enthalten. Dieser Teil der Doku soll noch einmal die grundlegenden Zusammenhänge zur Funktionsweise erläutern. Das Beispiel „simuliert“ einen Job-Methodenaufruf, dessen Abarbeitung in der SPS circa vier Sekunden dauert. Dies wurde mit Hilfe eines Timers realisiert, welcher im Funktionsbaustein-Teil des Jobs deklariert und verwendet wurde. Über die State Machine des Funktionsbausteins wird hierbei der Abschluss des Jobs verzögert (die Handshake-Methode CheckState returniert „busy“) und erst nach Ablauf des Timers wird der Job beendet (die Handshake-Methode CheckState returniert „done“).

In diesem Beispiel wurde für die OPC-UA-Methode mit dem Namen „MyJob“ ein Funktionsbaustein namens FB_Job erzeugt, welcher die oben genannte, benötigte Signatur aufweist.



Die Funktionsbaustein-Deklaration enthält das Attribut OPC.UA.DA.JobMethod und als dessen Wert den zu verwendenden Namen der OPC-UA-Methode.

```
{attribute 'OPC.UA.DA.JobMethod' := 'MyJob'}
FUNCTION_BLOCK FB_Job
VAR
END_VAR
```

Die drei Methoden Abort, CheckState und Start enthalten in ihrer Deklaration das Attribut TcRpcEnable (damit die Methoden auch über ADS aufgerufen werden können). Beispiel:

```
{attribute 'TcRpcEnable' := '1'}
METHOD Start : HRESULT

{attribute 'TcRpcEnable' := '1'}
METHOD Abort : HRESULT

{attribute 'TcRpcEnable' := '1'}
METHOD CheckState : HRESULT
```

Die Eingabeparameter der OPC-UA-Methode werden in der SPS-Methode Start() als VAR_INPUT deklariert. Kommentare hinter den Variablen werden als Description des jeweiligen OPC-UA-Eingabeparameters verwendet. Beispiel:

```
{attribute 'TcRpcEnable' := '1'}
METHOD Start : HRESULT
VAR_INPUT
  a : INT; // some description
  b : LREAL; // some description
  c : BOOL; // some description
  d : ST_Complex_1; // some description
END_VAR
VAR_OUTPUT
  hdl : UDINT; // handle, can be used for concurrent calls
END_VAR
```

Die Ausgabeparameter der OPC-UA-Methode werden in der SPS-Methode CheckState() als VAR_OUTPUT deklariert. Kommentare hinter den Variablen werden als Description des jeweiligen OPC-UA-Ausgabeparameters verwendet. Beispiel:

```
{attribute 'TcRpcEnable' := '1'}
METHOD CheckState : HRESULT
VAR_INPUT
  hdl : UDINT; // handle, can be used for concurrent calls
END_VAR
VAR_OUTPUT
  bBusy : BOOL; // required
  a : INT; // some description
  b : LREAL; // some description
  c : BOOL; // some description
  d : ST_Complex_1; // some description
END_VAR
```

(In diesem Beispiel werden zu Zwecken der Veranschaulichung die Werte der Eingabeparameter 1:1 auf die Ausgabeparameter gelegt. Daher sind die Eingabe- und Ausgabeparameter identisch.)

Gleichzeitige Methodenaufrufe

Gleichzeitige Methodenaufrufe können über die SPS-Applikation verwaltet werden. Hierfür kann über einen Rückgabewert der Start() Methode ein Handle (Ausgangsvariable hdl) an den Server zurückgegeben werden, welches dann bei allen weiteren Aufrufen der CheckState() Methode als Eingangsvariable verwendet wird. Hierdurch wird es der SPS-Applikation ermöglicht, einen Ausführungskontext für den jeweiligen Methodenaufruf zu bestimmen. Zusammengefasst ist der Workflow somit wie folgt:

- Start() Methode bestimmt den Ausführungskontext und liefert über die Ausgangsvariable „hdl“ ein Handle an den Server zurück.
- Der Server verwendet nun bei den zyklischen CheckState()-Aufrufen dieses Handle und lässt die SPS-Applikation hierüber wissen, um welchen Ausführungskontext es sich handelt.

4.13.3 RPC-Methoden

Methodenaufrufe sind ein grundlegender Teil der OPC-UA-Spezifikation. Mit der Einführung dieser Funktionalitäten in die SPS-Welt bietet TwinCAT 3 die Möglichkeit zur effizienten Ausführung von RPC-Aufrufen in der IEC61131-Welt und verringert somit die klassischen Handshake-Pattern bei der Kommunikation zwischen den Geräten.

Beim Konzept der RPC-Methoden, importiert der TwinCAT OPC UA Server eine Methode aus der TwinCAT SPS/C++ direkt als OPC-UA-Methode.

Voraussetzungen

Diese Funktionalität steht nur für Data Access [▶ 49] Geräte basierend auf TwinCAT 3 und dem Import von TMC- sowie TMI-Symboldateien [▶ 53], sowie der Online-Symbolik zur Verfügung.

Das Handling eines RPC-Methodenaufrufs wird auf OPC-UA-Ebene wie folgt durchgeführt:

- Wenn eine RPC-Methode erfolgreich ausgeführt wurde, gibt der Server als Rückmeldung für den OPC-UA-Methodenaufruf den Statuscode „Good“ zurück.
- Wenn die RPC-Methode nicht aufgerufen werden konnte, gibt der Server eine Fehlermeldung im Format „Bad_***“ zurück, abhängig vom aufgetretenen Fehler.
- Wenn die RPC-Methode erfolgreich aufgerufen wurde, aber die Antwort nicht im Server gelesen werden konnte, wird der Statuscode „OpcUa_GoodPostActionFailed“ vom Server zurückgegeben.

Abarbeitungskontext

Die TwinCAT SPS/C++ Methode wird innerhalb des Echtzeit-Kontexts ausgeführt und fällt somit in den Abarbeitungskontext einer Echtzeit-Task. Der TwinCAT-Entwickler muss daher Vorsichtsmaßnahmen treffen, sodass die Ausführungszeit der Methode in die Task-Zykluszeit passt.

Methoden in der TwinCAT 3 SPS

Methoden in der IEC61131-Welt werden immer unterhalb eines Funktionsbausteins konfiguriert. Auf Hochsprachenebene kann der Funktionsbaustein als umgebende Klasse der Methode betrachtet werden. Die Methode selbst müssen Sie mit einem speziellen SPS-Attribut deklarieren, sodass das TwinCAT-System weiß, dass die Methode für einen remote Methodenaufruf aktiviert werden soll.

```
{attribute 'TcRpcEnable':='1'}
METHOD M_Sum : INT
VAR_INPUT
  a : INT;
  b : INT;
END_VAR
```

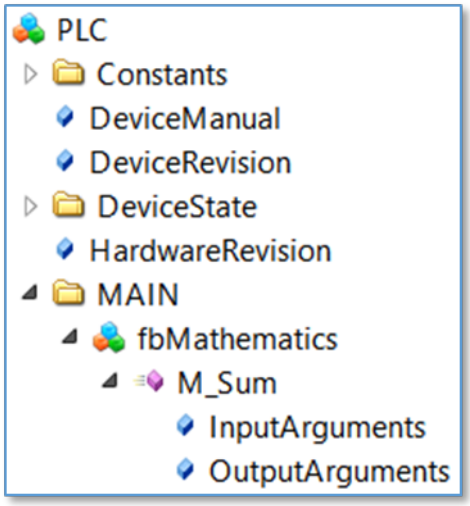
In Abhängigkeit davon, welcher Importmodus verwendet wird, müssen Sie den umgebenden Funktionsbaustein ebenfalls für den OPC-UA-Zugriff aktivieren. Dies kann durch Verwendung der normalen SPS-Attribute für den OPC-UA-Zugriff erfolgen.

Beachten Sie dabei, dass Sie nur dann die SPS-Attribute verwenden müssen, wenn der gefilterte Modus verwendet wird, um Symbole aus der SPS für den Zugriff über OPC UA freizugeben. Dies ist die Standardeinstellung des OPC UA Servers.

Beispiel:

Die Methode M_Sum befindet sich im Funktionsbaustein FB_Mathematics. Die Deklaration der Funktionsbausteininstanz verwendet das SPS-Attribut, das den Funktionsbaustein und damit die Methode ebenfalls für den OPC-UA-Zugriff aktiviert hat.

```
PROGRAM MAIN
VAR
  {attribute 'OPC.UA.DA':='1'}
  fbMathematics : FB_Mathematics;
END_VAR
```



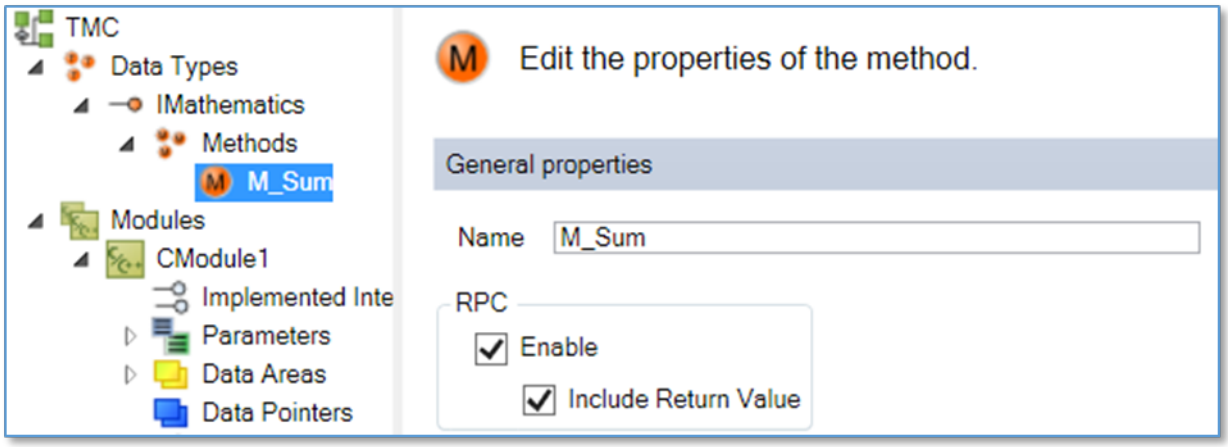
Pointervariablen als VAR_IN_OUT

Pointer-Variablen, die als VAR_IN_OUT definiert wurden, werden weder von der SPS noch vom TwinCAT OPC UA Server behandelt. Ein entsprechendes Trace-Ereignis wird in den Server-Trace geschrieben.

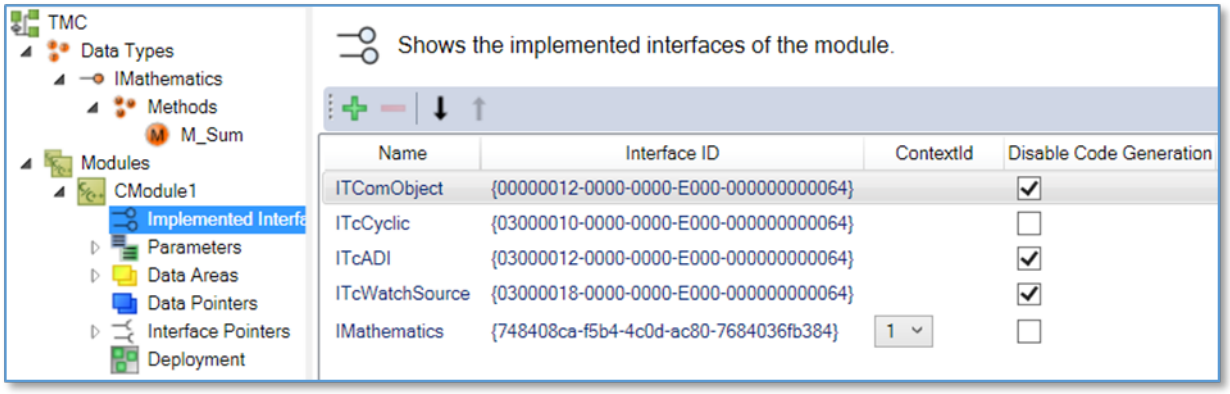
08:47:37.677Z|1|11A0* Fehler beim Importieren der Methode 'METH_PArray': VAR_IN_OUT Zeigervariablen sind nicht erlaubt!

Methoden in TwinCAT 3 C++

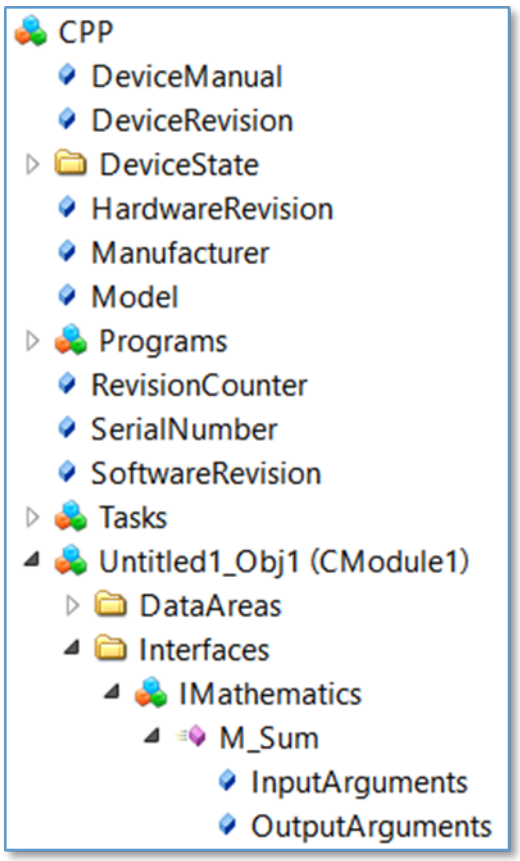
TwinCAT-Module können Schnittstellen implementieren, die über vordefinierte Methoden verfügen (siehe TcCOM-Module). Die Methode selbst muss für RPC-Aufrufe während ihrer Definition aktiviert sein (siehe Dokumentation TwinCAT Module Class Wizard), sodass der OPC UA Server weiß, dass sie zur Ausführung bereitsteht.



Damit auch der Rückgabewert der Methode zur Verfügung steht, muss die entsprechende Option **Include Return Value** aktiviert sein. Beachten Sie, dass die Schnittstelle, unter der die Methode erstellt wurde, implementiert werden muss.



In Abhängigkeit davon, welcher Importmodus verwendet wird, müssen Sie die Methode für den Zugriff über OPC UA deklarieren. Dies kann durch Verwendung des TMC-Code-Editors und der üblichen OPC-UA-Attribute als optionale Eigenschaften erfolgen.



4.14 TwinCAT Eventlogger

4.14.1 Überblick

Der TwinCAT OPC UA Server ermöglicht eine Integration des TwinCAT Eventloggers, um Alarm und Events zu Versenden. Hierbei wird ein Alarm/Event vom TwinCAT Eventlogger in einen OPC UA Alarm bzw. Event umgewandelt. Es können Eventlogger-Nachrichten von mehr als einem TwinCAT-Gerät im TwinCAT OPC UA Server konfiguriert werden. Jedes Gerät wird hierbei durch dessen AMS Net ID identifiziert.

Die Integration des TwinCAT Eventloggers ist für den TwinCAT OPC UA Server nicht unter Windows CE verfügbar. Sie können allerdings den TwinCAT OPC UA Server auf einem Nicht-CE-Gerät installieren und dann remote TwinCAT-Eventlogger-Nachrichten von einem Windows-CE-Gerät empfangen.

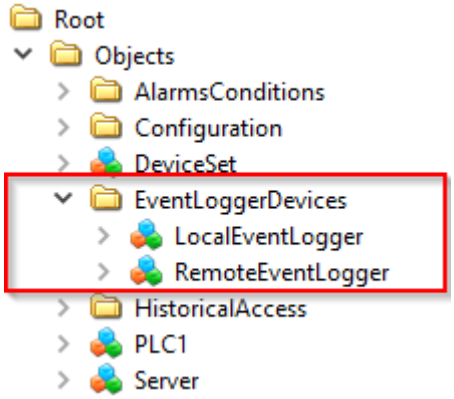
Die Integration des TwinCAT Eventloggers ist nur in Verbindung mit dem TwinCAT 3 Eventlogger verfügbar. Der TwinCAT 2 Eventlogger wird nicht unterstützt.

4.14.2 Konfiguration

Zur Konfiguration des Servers für eine Verbindung mit dem Eventlogger, müssen Sie im Serververzeichnis eine neue Konfigurationsdatei mit dem Namen „TcUaEventLogConfig.xml“ anlegen. Diese Datei beinhaltet eine Liste mit TwinCAT-Eventlogger-Geräten, welche anhand ihrer AMS NetID identifiziert werden. Die Konfiguration hat hierbei den folgenden Aufbau:

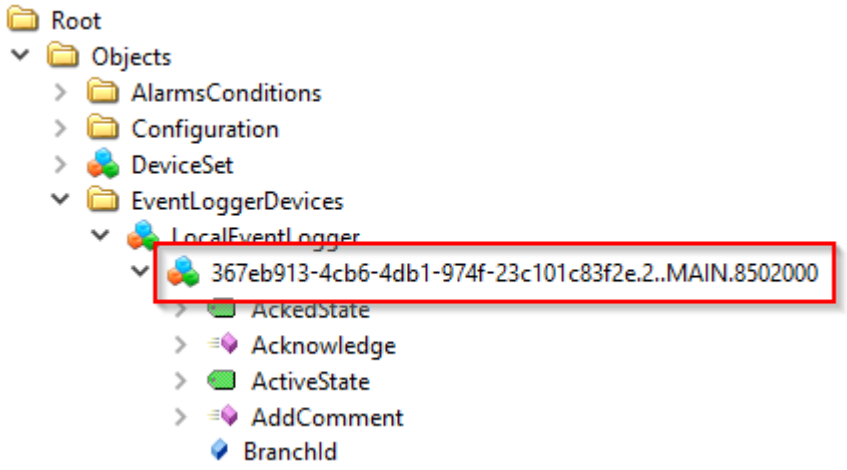
```
<TcUaEventLogConfig>
  <EventLoggerDevice Name="LocalEventLogger" AmsAddr="127.0.0.1.1.1"/>
  <EventLoggerDevice Name="RemoteEventLogger" AmsAddr="192.168.0.56.1.1"/>
</TcUaEventLogConfig>
```

Die einzelnen Geräteeinträge werden anschließend im Folder „EventLoggerDevices“ im Namensraum des Servers angezeigt.



Ein OPC UA Client kann sich nun auf das entsprechende Objekt subscriben um Events und/oder Alarme von dem jeweiligen TwinCAT-Eventlogger-Gerät zu empfangen.

Im Gegensatz zu einem Event wird ein Alarm zusätzlich als Kindelement von dem jeweiligen Eventlogger-Gerät angezeigt.

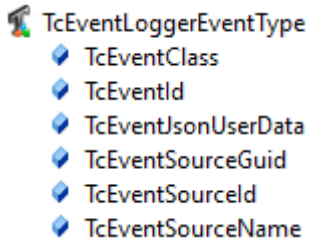


4.14.3 Zugriff auf Alarme und Events

Beim Subscriben auf das Objekt und den Empfang von Alarmen oder Events muss ein Client berücksichtigen, dass hierbei spezielle Objekttypen verwendet werden. Die jeweiligen Typen sind wie folgt definiert:

TcEventLoggerEventType

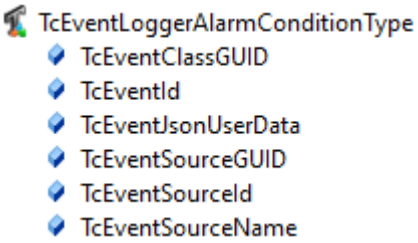
Der TcEventLoggerEventType ist abgeleitet vom BaseEventType und erweitert diesen mit TwinCAT-Eventlogger-spezifischen Properties:




```
NodeID: i=4200
NamespaceName: urn:BeckhoffAutomation:Ua:Types:GlobalTypes
```

TcEventLoggerAlarmConditionType

Der TcEventLoggerAlarmConditionType ist abgeleitet vom AlarmConditionType und erweitert diesen mit TwinCAT-Eventlogger-spezifischen Properties:




```
NodeID: i=4000
NamespaceName: urn:BeckhoffAutomation:Ua:Types:GlobalTypes
```

Beispiel

Das folgende Beispiel basiert auf dem Standard-Code-Sample vom TwinCAT Eventlogger, welches aus dem Beckhoff Information System bezogen werden kann. Dieses Sample beinhaltet Code Snippets für die SPS, mit welchen man sowohl ein Event als auch einen Alarm feuern kann.

Schritt 1: Konfiguration des Servers

TwinCAT OPC UA Server und die SPS laufen in diesem Beispiel nun lokal auf demselben System. Dementsprechend wurde auch die TcEventLogConfig.xml erstellt:

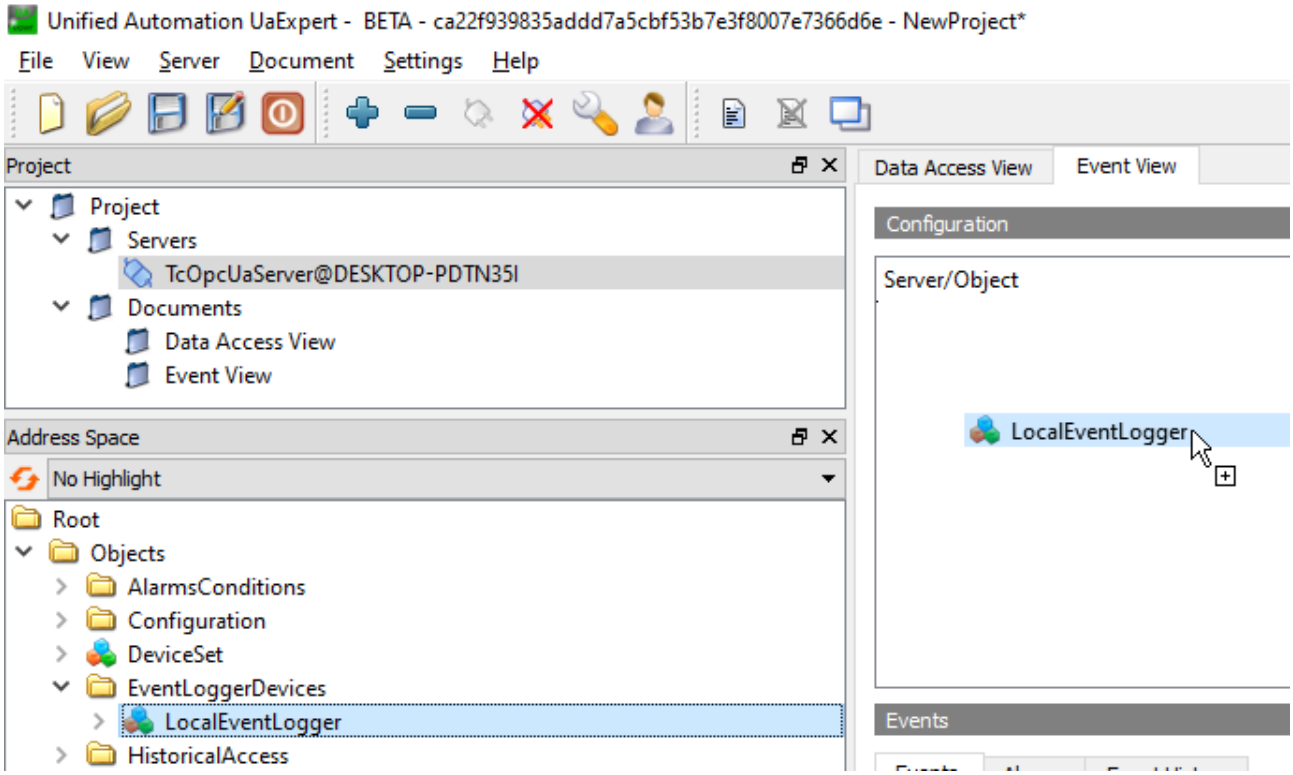
```
<TcUaEventLogConfig>
  <EventLoggerDevice Name="LocalEventLogger" AmsAddr="127.0.0.1.1.1"/>
</TcUaEventLogConfig>
```

Schritt 2: Aktivieren des TwinCAT Eventlogger Samples

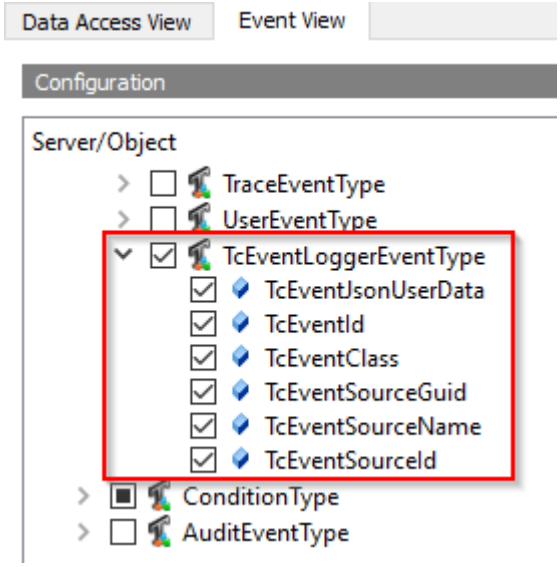
Vor dem Aktivieren des TwinCAT Eventlogger Samples wurde das automatische Feuern eines Events bzw. Alarms zunächst deaktiviert. In der vorliegenden Sample-Version erfolgt dies durch das Initialisieren von bSend und bAlmRaise mit dem Wert FALSE. Anschließend wird das Projekt aktiviert und in der lokalen SPS-Laufzeit ausgeführt.

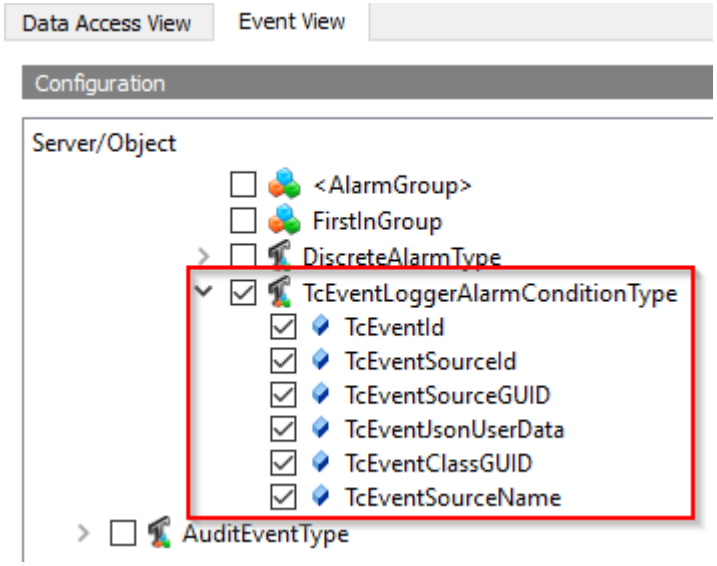
Schritt 3: Verbinden eines OPC UA Clients mit dem Server

Als OPC UA Client wurde der UA Expert gewählt. Nach dem Aufbau einer Verbindung mit dem Server fügen Sie im UA Expert ein neues „Event View“ Dokument hinzu. Anschließend ziehen Sie per Drag-and-Drop das konfigurierte TwinCAT-Eventlogger-Gerät in das Event View.



Damit die TwinCAT-Eventlogger-spezifischen Properties empfangen werden können, muss der UA Expert den entsprechend Event- bzw. AlarmType filtern. Sie finden den TcEventLoggerEventType bzw. TcEventLoggerAlarmConditionType in der Typliste des Event Views. Beispiel:

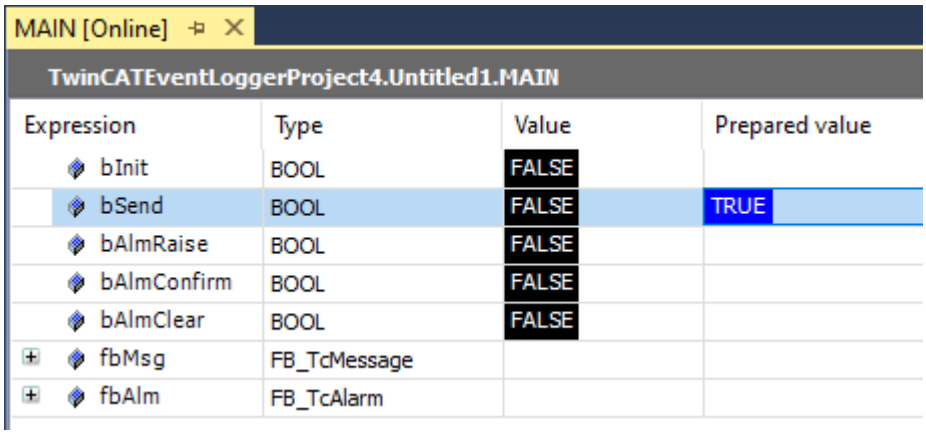


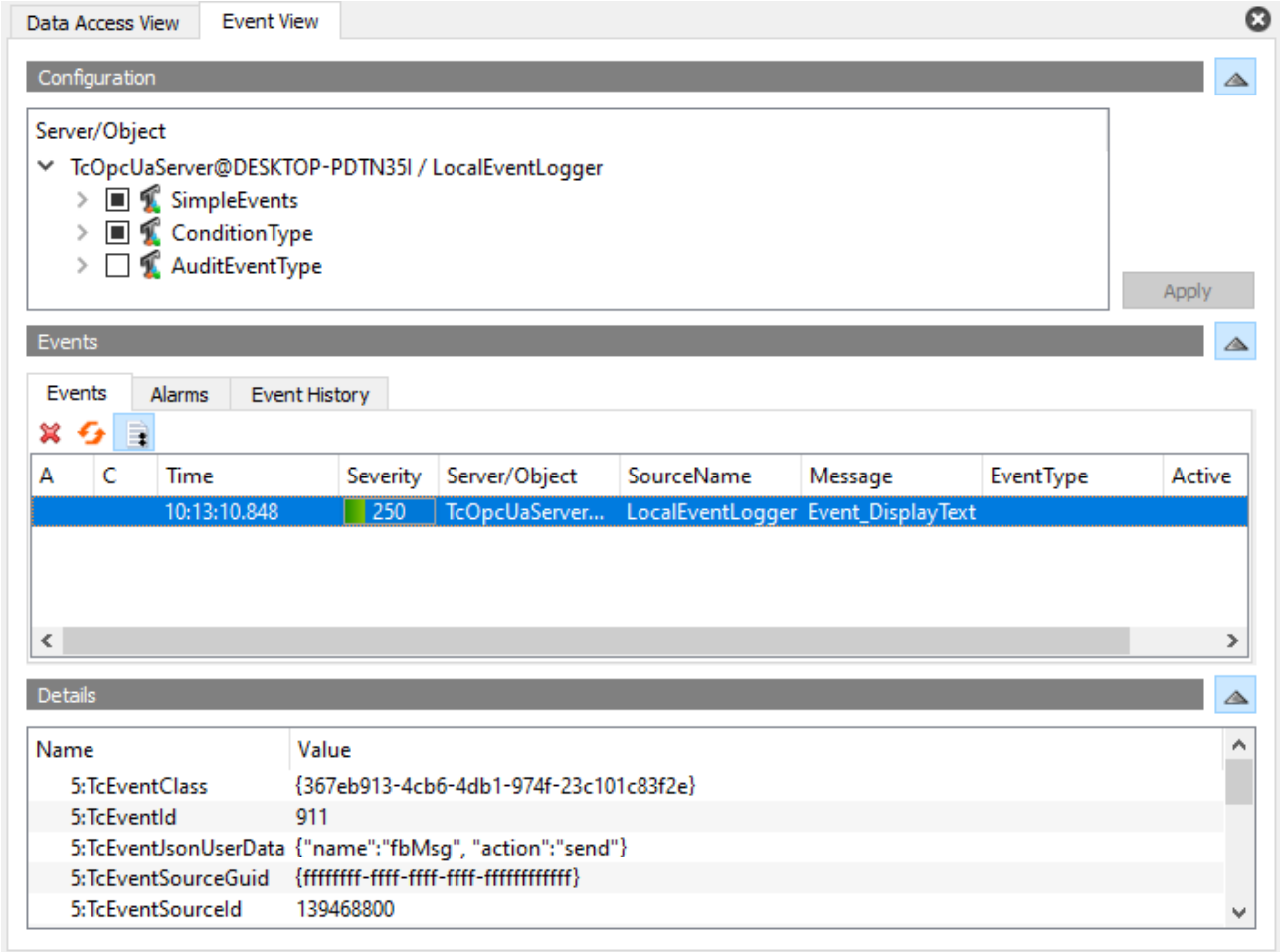


Durch einen Klick auf den **Apply**-Button werden diese Filter übernommen.

Schritt 4: Feuern eines Events

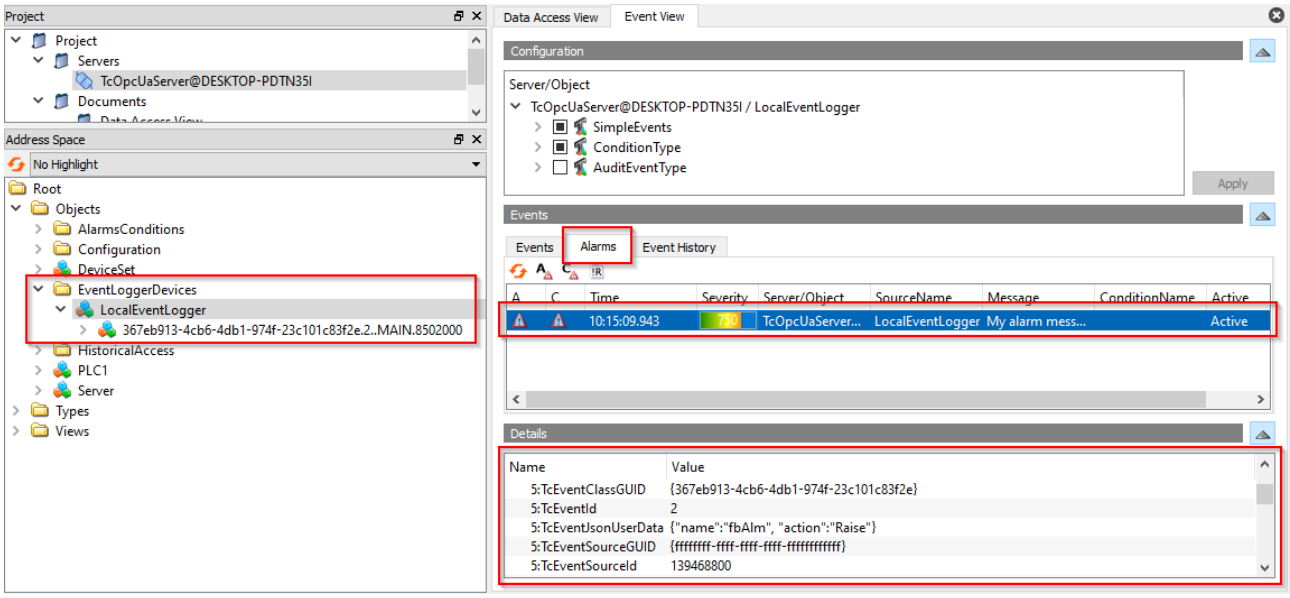
Im TwinCAT Eventlogger Sample setzen wir die Variable bSend auf den Wert TRUE, um ein Event abzufeuern. Der UA Expert empfängt dieses Event automatisch und zeigt es im Event View mitsamt der TwinCAT-Eventlogger-spezifischen Properties an.





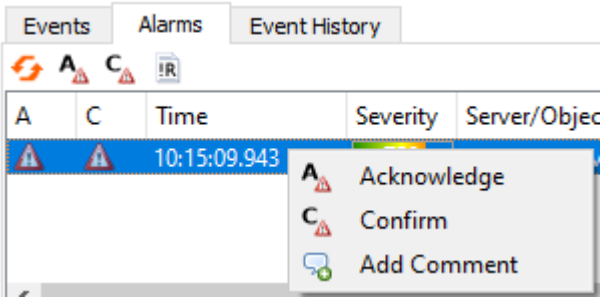
Schritt 5: Feuern eines Alarms

Im TwinCAT Eventlogger Sample setzen wir die Variable bAlmRaise auf den Wert TRUE, um einen Alarm abzufeuern. Der UA Expert empfängt diesen Alarm automatisch und zeigt ihn im Event View mitsamt der TwinCAT-Eventlogger-spezifischen Properties an. Zusätzlich wird der Alarm als eigenes Objekt im Namespace unterhalb des Eventlogger-Geräts angezeigt.



Schritt 6: Quittieren eines Alarms

Zusätzlich zum Empfang eines Alarms kann dieser auch quittiert werden. Die Quittierung wird hierbei vom Server auch an den TwinCAT Eventlogger zurückgemeldet. Im UA Expert lässt sich ein Alarm über das Kontextmenü im Event View quittieren.



Acknowledge und Confirm

Bitte beachten Sie, dass nur ein Confirm zurück an den TwinCAT Eventlogger gemeldet wird. Die Information zu einem Acknowledge bleibt hierbei im Server, da der TwinCAT Eventlogger aktuell nur das Konzept eines Confirms vorsieht.

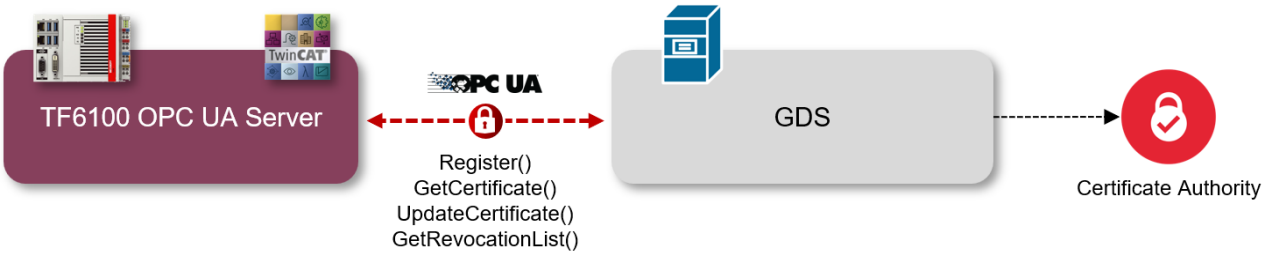
Nach einem Confirm wird die entsprechende Variable eConfirmationState in der TwinCAT-Eventlogger-Instanz vom Typ FB_TcAlarm gesetzt.

bRaised	BOOL	TRUE
eConfirmationState	TCEVENTCONFIRMA...	Confirmed

4.15 Global Discovery Server

4.15.1 Überblick

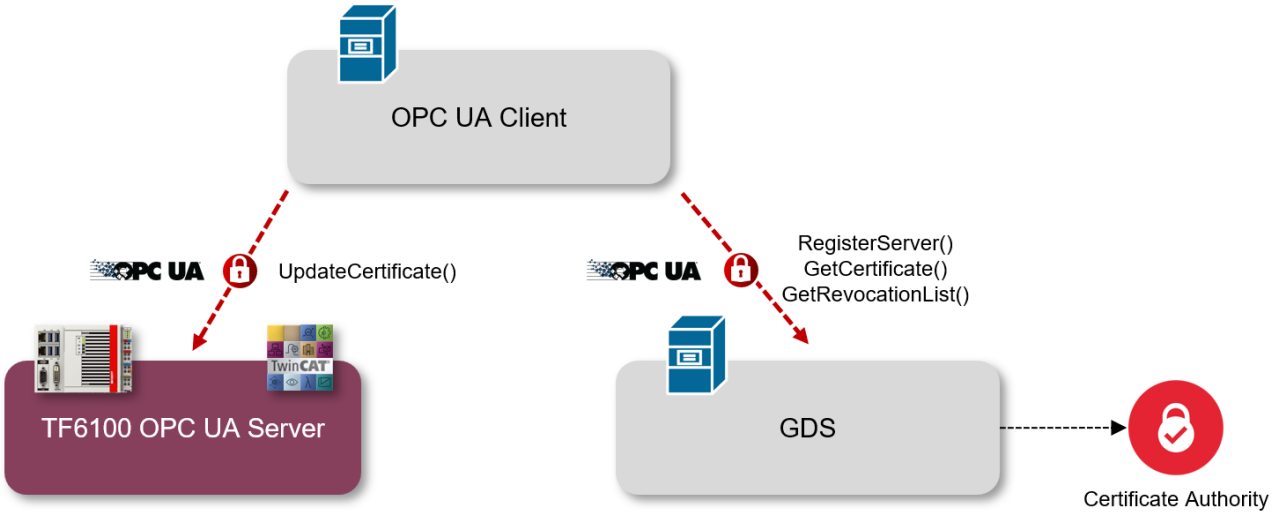
Der TwinCAT OPC UA Server ermöglicht die Registrierung an einem **G**lobal **D**iscovery **S**erver (GDS). Ein GDS bietet eine einheitliche, auf OPC UA basierte Schnittstelle zur Registrierung von OPC-UA-Applikationen und der Ausstellung von Applikationszertifikaten sowie den damit verbundenen Sperrlisteninformationen. Ein GDS hat hierfür eine Verbindung zu einer Zertifizierungsstelle (engl. **C**ertificate **A**uthority (CA)). Die Registrierung der Applikationen sowie die Ausstellung (und auch die Aktualisierung) von Zertifikaten erfolgt über ein einheitliches OPC-UA-Informationsmodell.



Beim Zugriff auf den GDS unterstützt der TwinCAT OPC UA Server die beiden Modelle Push [▶ 119] und Pull [▶ 120].

4.15.2 Push

Bei diesem Modell beinhaltet der Server ein standardisiertes Interface, welches ein (*das Push-Modell unterstützender*) OPC UA Client verwenden kann, um sich im Auftrag des Servers mit einem Global Discovery Server zu verbinden, dort die Serverapplikation zu registrieren und ein Server-Zertifikat inklusive der aktuellen **C**ertificate **R**evocation **L**ist (CRL) anzufordern.



Als Voraussetzung zur Nutzung dieser Funktionalität muss sich der OPC UA Client am Server mit einem Benutzerkonto authentifizieren, welches über Administrator-Berechtigung verfügt.

OPC UA Client

Als OPC UA Client kann jeder Client verwendet werden, der das Push-Modell unterstützt. Diverse OPC UA Toolkithersteller bieten hier entsprechende Softwarepakete an. Alternativ stellt auch die OPC Foundation einen GDS Sample Client auf Github zur Verfügung.

GDS Server

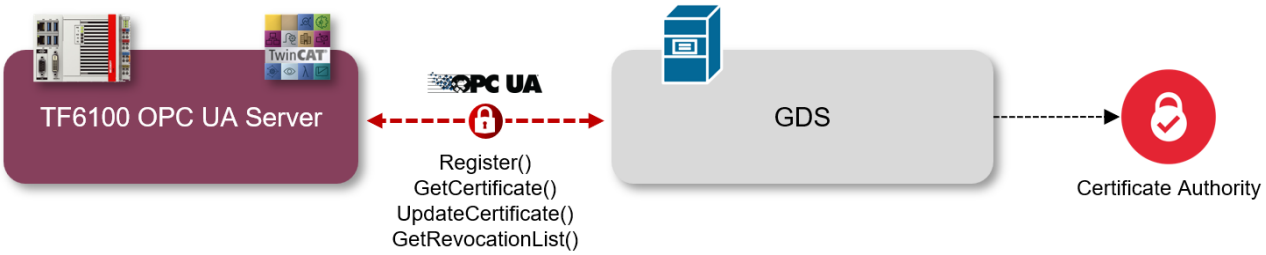
Als Global Discovery Service kommt zur Verwendung jeder GDS in Frage. Diverse OPC UA Toolkithersteller bieten hier entsprechende Softwarepakete an. Alternativ stellt auch die OPC Foundation einen GDS Sample Server auf Github zur Verfügung.

Konfiguration im Server

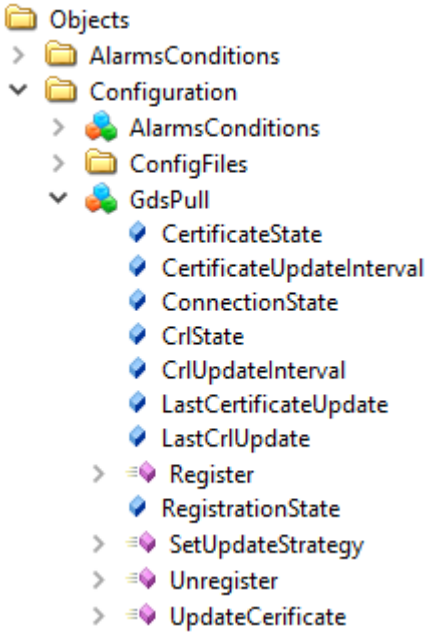
Zur Verwendung dieser Funktionalität sind keine weiteren speziellen Konfigurationsschritte im TwinCAT OPC UA Server notwendig. Die entsprechende Schnittstelle ist standardmäßig aktiviert und kann von einem Benutzer mit Administratorrechten verwendet werden. Der während der Initialisierung [▶ 28] konfigurierte Benutzer hat hierfür alle notwendigen Berechtigungen.

4.15.3 Pull

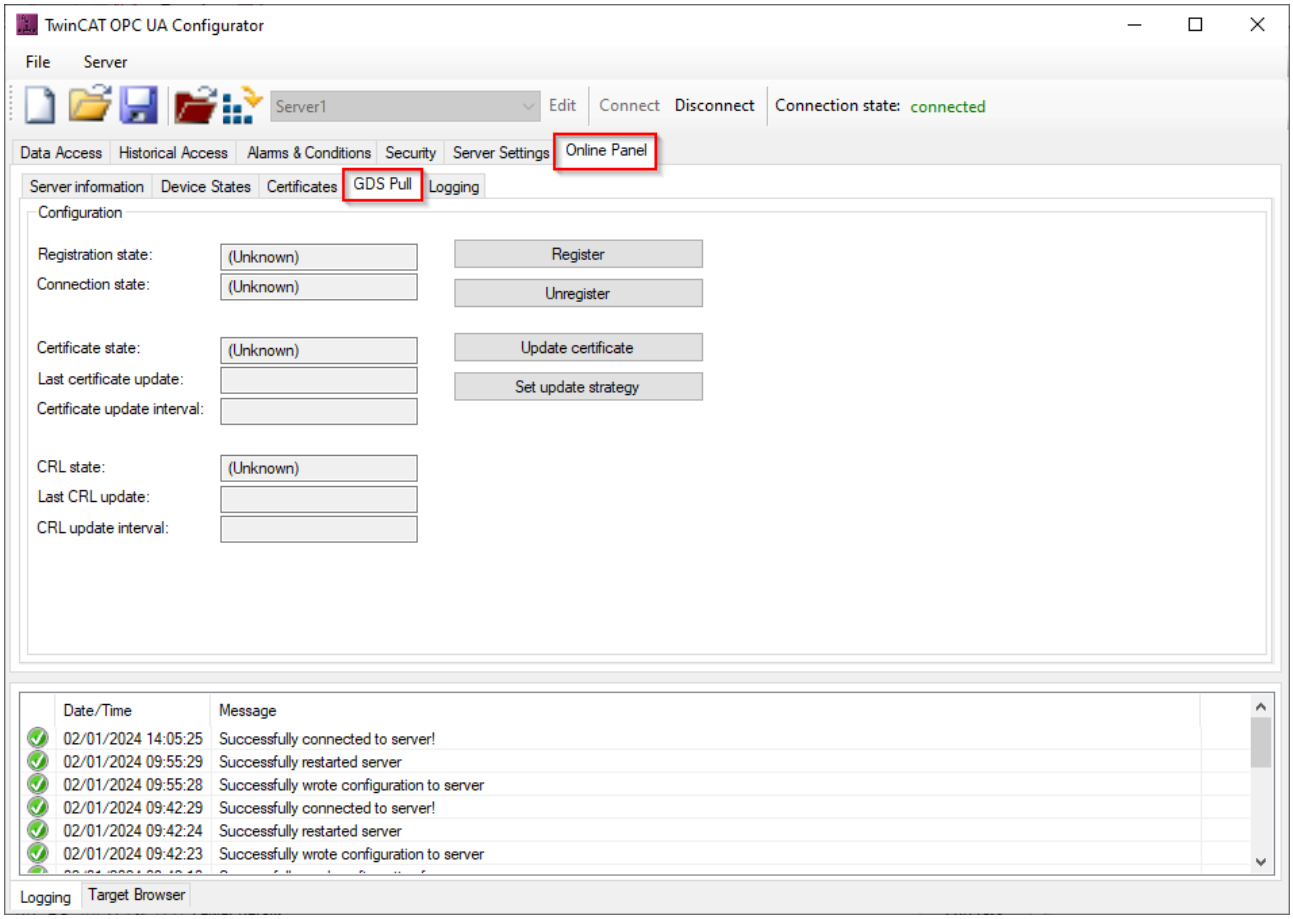
Bei diesem Modell stellt der Server eigenständig eine Verbindung zum Global Discovery Server her, registriert sich dort als Serverapplikation und bezieht ein passendes Zertifikat.



Der TwinCAT OPC UA Server bietet über seinen Konfigurations-Namensraum die Möglichkeit, die GDS-Pull-Funktionen zu aktivieren und konfigurieren.



Alternativ kann auch der TwinCAT OPC UA Configurator verwendet werden, um diese Funktion im Server zu konfigurieren. Hierfür steht eine entsprechende Benutzeroberfläche zur Verfügung.



Konfiguration über OPC UA Client

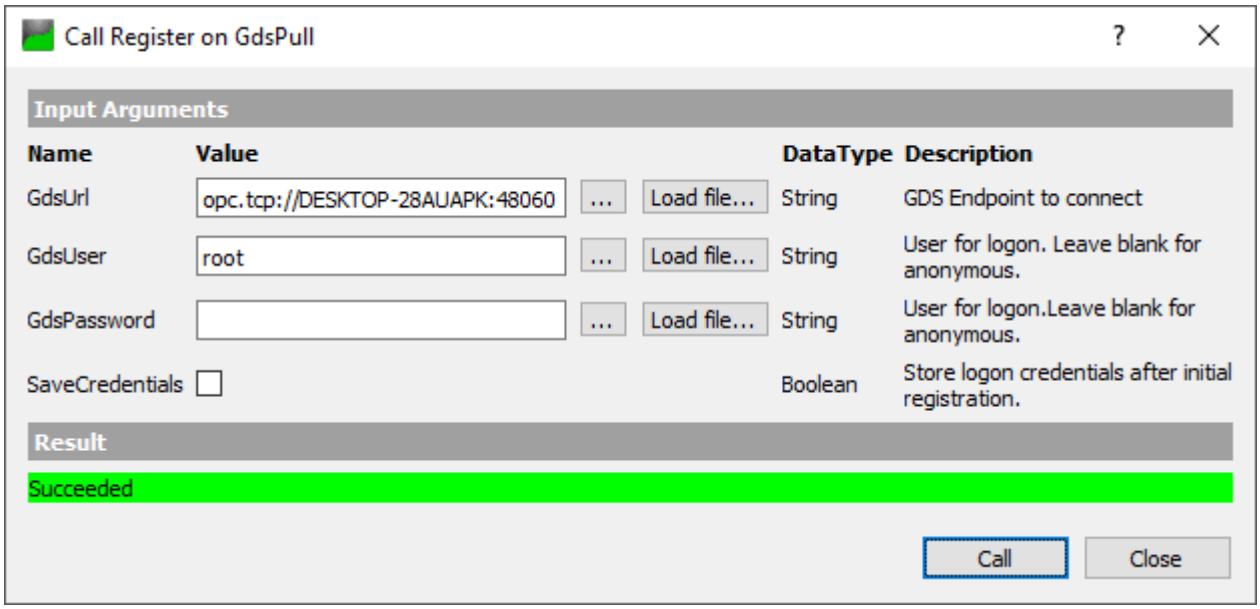
Der folgende Abschnitt beschreibt die Konfiguration über den Adressraum des Servers durch einen generischen OPC UA Client. Als Clientsoftware wird hierbei der UA Expert von Unified Automation verwendet.

Im ersten Schritt muss der TwinCAT OPC UA Server am GDS als Applikation registriert werden. Dies geschieht über die Methode Register(). Durch die Registrierung am GDS wird automatisch auch ein Serverzertifikat für die Serverapplikation beantragt. Je nach Implementierung der GDS-Applikation wird ein

solches Zertifikat entweder automatisch oder nach manueller Freigabe durch einen Administrator ausgestellt. Anhand der Variablen RegistrationState und CertificateState lässt sich überprüfen, ob der Server bereits an einem Global Discovery Service registriert wurde und ein Zertifikat von diesem erhalten hat. Die Variable CrlState gibt den Status der Certificate Revocation List an und ob diese vom GDS bezogen werden konnte.

#	Server	Node Id	Display Name	Value	Datatype	Source Timestamp	Server Timestamp	Statuscode
1	TcOpcUaServer...	NS13|Numeric|...	RegistrationState	5 (Registered)	Int32	2:06:47.020 PM	2:07:42.248 PM	Good
2	TcOpcUaServer...	NS13|Numeric|...	CertificateState	10 (Certificate updated)	Int32	2:06:47.020 PM	2:11:07.041 PM	Good
3	TcOpcUaServer...	NS13|Numeric|...	CrlState	12 (Crl updated)	Int32	2:06:47.020 PM	2:11:10.329 PM	Good

Die Methode erwartet die folgenden Eingabeparameter:



Eingabeparameter	Bedeutung
GdsUrl	Die URL des Global Discovery Service im Format opc.tcp://hostnameOrIpAddress:port
GdsUser	Benutzername eines GDS-Benutzers mit der Berechtigung, neue Applikationen registrieren zu dürfen.
GdsPassword	Passwort des Benutzers
SaveCredentials	Speichert das Passwort des Benutzers in einer Konfigurationsdatei des TwinCAT OPC UA Servers. Aus Sicherheitsgründen wird nicht empfohlen, diese Einstellung zu setzen. Sie wurde ausschliesslich für Global Discovery Services entwickelt, welche eine Benutzername-/Passwort- Authentifizierung zwingend benötigen. Üblicherweise wird die Benutzername-/Passwort-Authentifizierung nur einmalig bei der Registrierung der Applikation benötigt. Bei allen darauf folgenden Verbindungen zum GDS wird dann das ausgestellte Server-Zertifikat verwendet.

Reinitialisierung der Server-Endpunkte

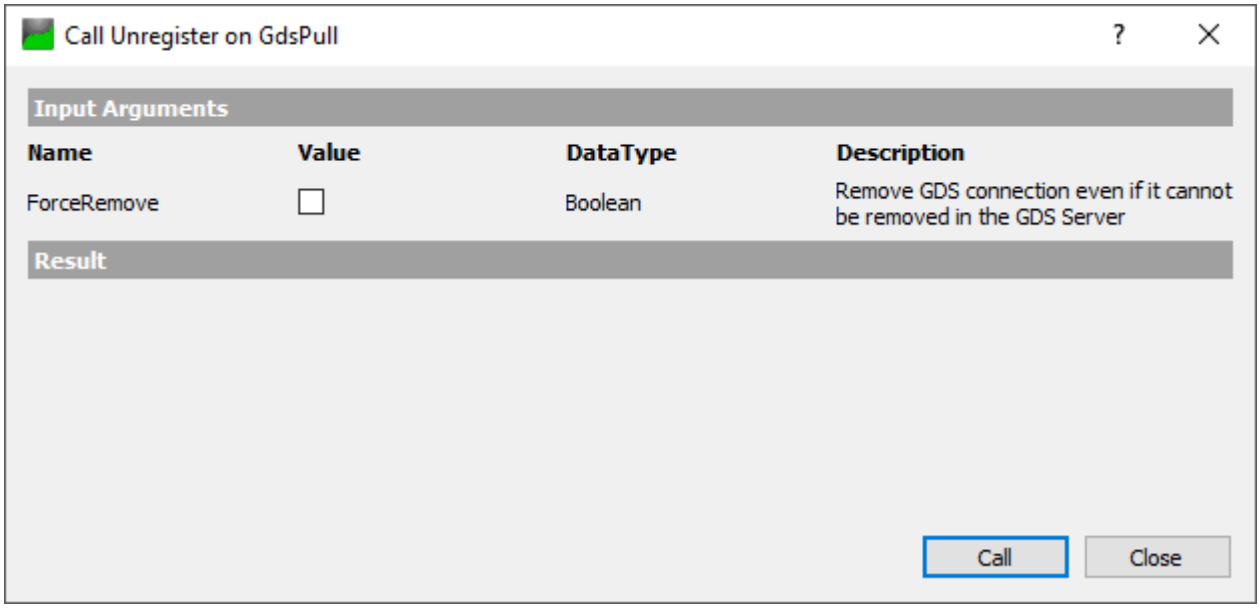
Nach der Registrierung der Serverapplikation am Global Discovery Service und dem Erhalt eines Serverzertifikats, initialisiert der Server einmal seine Endpunkte neu, wodurch verbundene Clients kurzzeitig die Verbindung verlieren.

Nachdem die Serverapplikation bei einem Global Discovery Service registriert wurde, wird im Installationsverzeichnis des TwinCAT OPC UA Servers eine neue Datei namens "TcUaGdsClientConfig.xml" erzeugt. Diese beinhaltet sowohl die Verbindungsinformationen des konfigurierten GDS als auch die von dort bezogenen Registrierungsinformationen und Zeitstempelinformationen für die Serverapplikation, z. B., wann das Zertifikat und die CRL das letzte Mal aktualisiert wurden.

De-Registrierung am Global Discovery Service

Um die TwinCAT OPC UA Server Applikation beim GDS zu de-registrieren, kann die Methode Unregister() verwendet werden. Die erfolgreiche Ausführung der Methode bewirkt, dass die Serverapplikation auf dem GDS de-registriert und der Inhalt der Datei TcUaGdsClientConfig.xml gelöscht wird.

Die Methode erwartet die folgenden Eingabeparameter:



Eingabeparameter	Bedeutung
ForceRemove	Falls die Verbindung zum Global Discovery Service nicht mehr zur Verfügung steht, kann durch das Setzen dieses Eingabeparameters die Verbindung zum GDS entfernt werden. Der TwinCAT OPC UA Server entfernt hierbei den GDS aus seiner Konfiguration.

Nachdem der TwinCAT OPC UA Server vom GDS entkoppelt wurde, bleiben das ausgestellte Serverzertifikat und die CRL bestehen. Falls Sie diese löschen und den Server mit einem self-signed Zertifikat betreiben möchten, müssen Sie die entsprechenden Dateien im PKI-Verzeichnis des Servers entfernen und den Server anschließend neu starten. Der Server erstellt sich daraufhin wieder ein self-signed Zertifikat.

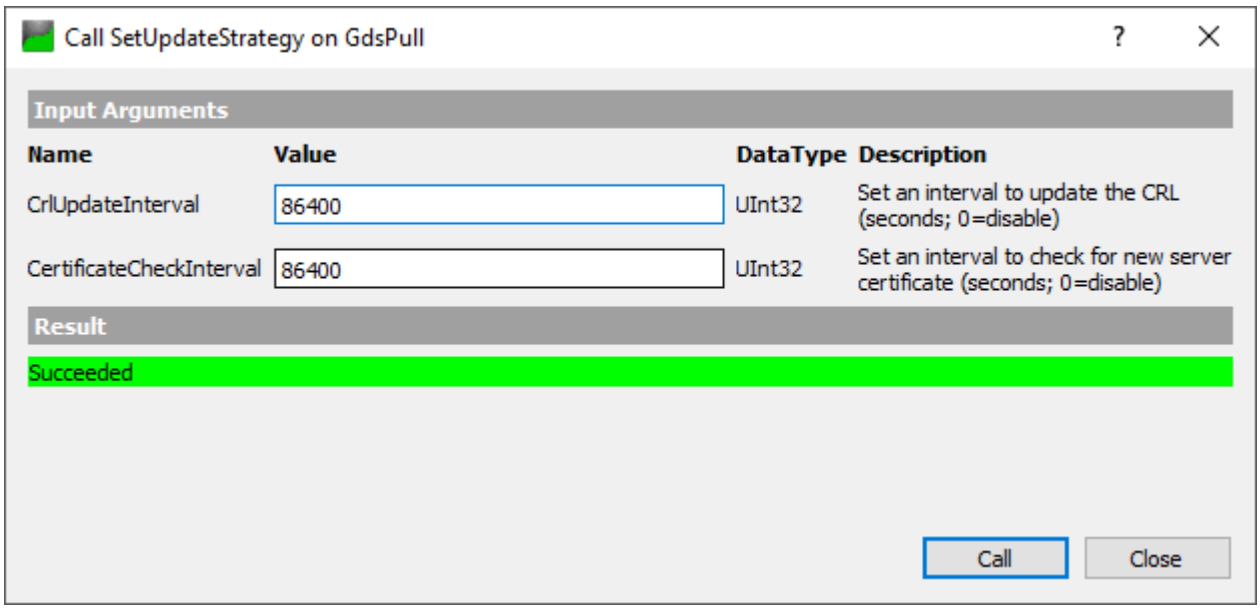
Aktualisierung des Serverzertifikats

Durch Ausführen der Methode UpdateCertificate() kann eine Aktualisierung des Serverzertifikats außerhalb des regulären Updateintervalls beim GDS angefragt werden. Die Methode erwartet keine weiteren Eingabeparameter.

Setzen der Updateintervalle für Serverzertifikat und CRL

Durch Ausführen der Methode SetUpdateStrategy() können die Updateintervalle für das Serverzertifikat und die Zertifikatssperrliste gesetzt werden.

Die Methode erwartet die folgenden Eingabeparameter:



Eingabeparameter	Bedeutung
CrlUpdateInterval	Setzt das Aktualisierungsintervall für die Zertifikatssperrliste (Sekunden).
CertificateCheckInterval	Setzt das Aktualisierungsintervall für das Serverzertifikat (Sekunden).

4.16 Security

4.16.1 Übersicht

OPC UA ist als Kommunikationstechnologie unter anderem aufgrund der integrierten Sicherheitsmechanismen so erfolgreich. Eine auf OPC UA basierte Datenkommunikation lässt sich dabei auf zwei Ebenen absichern: Auf Transport- *und* Applikationsebene. Beim Verbindungsaufbau mit dem Server wählt der Client zunächst einen sogenannten Endpunkt aus, welcher unter anderem die zu verwendenden Sicherheitsfunktionen angibt.

Endpunkte

Ein Server bietet dem Client eine Liste mit verschiedenen Endpunkten [▶ 125] an, mit denen sich der Client verbinden kann. Ein Endpunkt beschreibt hierbei unter anderem, welche Sicherheitsfunktionen (z. B. Message Security Mode, Security Policy und zur Verfügung stehende Identity Token) die Kommunikationsverbindung über diesen Endpunkt erfüllen soll. So kann ein Endpunkt z. B. eine Signierung und Verschlüsselung der Datenpakete erfordern (Transportebene) sowie eine zusätzliche Authentifizierung des Clients auf Basis von Benutzername/Passwort (Applikationsebene).

Transportebene

Eine auf OPC UA basierte Kommunikationsverbindung kann auf Transportebene abgesichert werden. Dies geschieht durch die Verwendung von Client-/Server-Zertifikaten und eine gegenseitige Vertrauensstellung zwischen Client- und Serverapplikation. Hierbei muss der Client dem Server-Zertifikat vertrauen und umgekehrt, damit eine Kommunikationsverbindung hergestellt werden kann. Hierfür ist ein gegenseitiger Zertifikatsaustausch [▶ 126] notwendig.

Applikationsebene

Zusätzlich zur Transportebene lässt sich eine Kommunikationsverbindung auch auf Applikationsebene absichern. Hierfür stehen verschiedene Authentifizierungsmechanismen [▶ 128] zur Verfügung, die vom Serverendpunkt angeboten werden.

4.16.2 Endpunkte

Der TwinCAT OPC UA Server stellt verschiedene Endpunkte über den Standard-Port 4840/tcp für OPC UA Clients zur Verfügung. Die Endpunkte definieren hierbei die Art der Verbindung zwischen Client und Server, und, ob diese gesichert oder ungesichert erfolgen soll.

Standard-Port

Beachten Sie, dass der Standard-Port 4840 eventuell von anderen OPC UA Servern verwendet wird, z. B. dem **L**ocal **D**iscovery **S**erver (LDS) von der OPC Foundation, die von manchen Anbietern mit OPC-UA-Softwarepaketen eingesetzt wird.

Vertrauensverhältnis

Beachten Sie, dass zur Verwendung der sicheren Endpunkte ein Vertrauensverhältnis zwischen Server und Client hergestellt werden muss, was üblicherweise über deren Zertifikate erfolgt. Hier erfahren Sie, wie Sie ein solches Vertrauensverhältnis Server-seitig konfigurieren [▶ 126].

Deprecated Endpunkte

Bitte beachten Sie, dass die aktuell in den Endpunkten zur Verfügung stehenden Securityprofile im Laufe der Zeit gegebenenfalls als potenziell unsicher eingestuft werden könnten und durch neuere ersetzt werden. In diesem Fall ist ein Update des TwinCAT OPC UA Servers empfehlenswert. Über einen Konfigurationsschalter (<AllowDeprecatedSecurityPolicies>) in der TcUaServerConfig.xml lassen sich veraltete und als unsicher eingestufte Security Policies wieder aktivieren.
Beckhoff empfiehlt aus Sicherheitsgründen, diesen Konfigurationsschalter deaktiviert zu lassen.

Liste der Endpunkte

Die folgende Liste fasst die Endpunkte des TwinCAT OPC UA Servers zusammen. Dabei sind auch bereits abgekündigte Endpunkte enthalten. Im Auslieferungszustand bietet der TwinCAT OPC UA Server nur aktuell als sicher geltende Endpunkte an. Seit Setupversion 4.3.28 ist außerdem der unverschlüsselte Endpunkt aus Security- und Zertifizierungsgründen standardmäßig deaktiviert.

Security-Profil	Security-Modus	Kurzbeschreibung
None	None	Bei diesem Endpunkt wird keinerlei Verschlüsselung oder Signierung der Nachrichten durchgeführt. Eine Authentifizierung hingegen ist möglich.
Basic128Rsa15 (veraltet)	Sign Sign & Encrypt	Dieser Endpunkt ist aus Security-Sicht als veraltet eingestuft worden und ist standardmäßig deaktiviert. Bei Bedarf kann der Endpunkt wieder freigeschaltet werden.
Basic256 (veraltet)	Sign Sign & Encrypt	Dieser Endpunkt ist aus Security-Sicht als veraltet eingestuft worden und ist standardmäßig deaktiviert. Bei Bedarf kann der Endpunkt wieder freigeschaltet werden.
Basic256Sha256	Sign Sign & Encrypt	Aktuell im Server vorhandener Endpunkt für sichere Signierung und Verschlüsselung. Eine zusätzliche Authentifizierung ist möglich.
Aes256_Sha256_RsaPss	Sign Sign & Encrypt	Aktuell im Server vorhandener Endpunkt für sichere Signierung und Verschlüsselung. Eine zusätzliche Authentifizierung ist möglich.
Aes256_Sha256_RsaOaep	Sign Sign & Encrypt	Aktuell im Server vorhandener Endpunkt für sichere Signierung und Verschlüsselung. Eine zusätzliche Authentifizierung ist möglich.

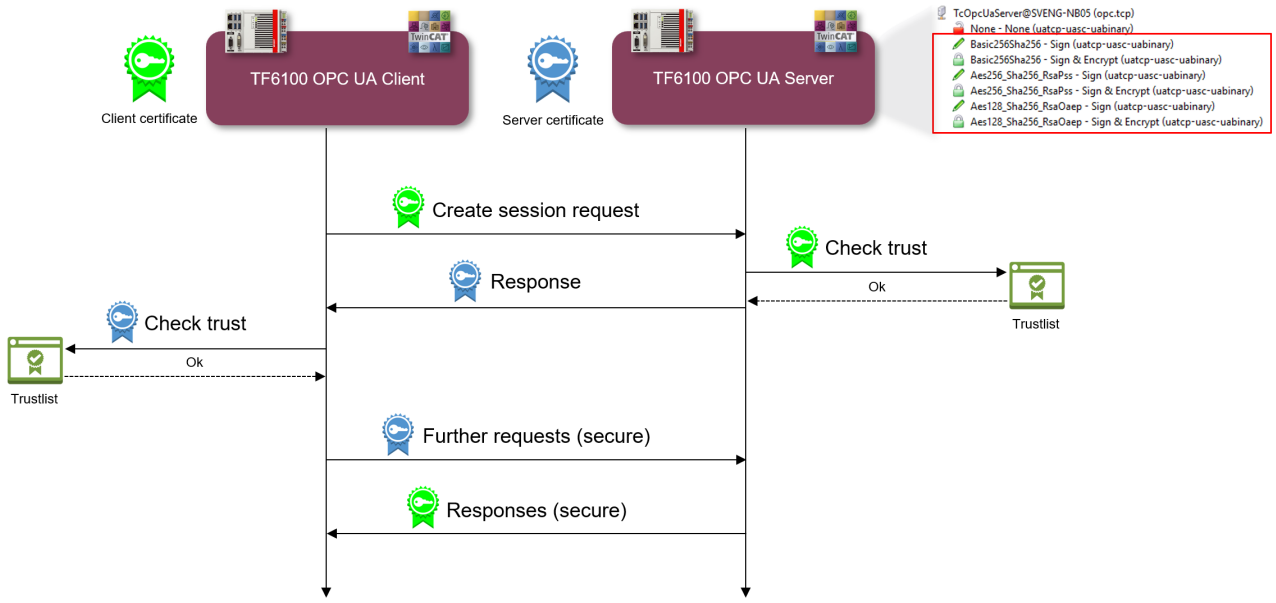
Alle in der Liste aufgeführten Endpunkte können über die Konfiguration des Servers aktiviert oder deaktiviert werden. In der folgenden Abbildung sind alle Endpunkte aktiviert.

4.16.3 Zertifikatsaustausch

Für eine Absicherung der Kommunikationsverbindung auf Transportebene über einen sicheren Endpunkt [▶ 125] ist die Herstellung einer gegenseitigen Vertrauensstellung zwischen Client und Server notwendig. Standardmäßig generieren hierzu sowohl der TwinCAT OPC UA Server als auch der TwinCAT OPC UA Client beim ersten Start ein maschinenspezifisches, selbstsigniertes Schlüsselpaar bestehend aus einem Public- und einem Private-Key. Sie können jedoch zur Integration in Ihre IT-Infrastruktur auch eine beliebige Zertifizierungsstelle bzw. -technologie verwenden, z. B. Active Directory oder OpenSSL. Für die einfache Verwaltung sowie den sicheren Zugriff auf Zertifikate ist die Einrichtung eines Global Discovery Server [▶ 119] sinnvoll.

Zur Einrichtung einer Vertrauensstellung zwischen einem OPC UA Client und dem TwinCAT OPC UA Server benötigen Sie den öffentlichen Schlüssel des Clientzertifikats. Der Server muss diesem entsprechend vertrauen. Der Server verwaltet die Vertrauenseinstellungen für Client-Zertifikate in einem Unterverzeichnis des Applikationsverzeichnisses [▶ 47].

Das folgende Schaubild verdeutlicht den Zusammenhang von Client- und Serverzertifikat beim Aufbau einer sicheren Kommunikationsverbindung:



Beim CreateSession Request übermittelt der Client seinen Public Key. Der Server hat daraufhin die Möglichkeit die Vertrauensstellung zu überprüfen. Vertraut der Server dem Client, so übermittelt er in seiner Response seinen eigenen Public Key. Der Client hat somit ebenfalls die Möglichkeit, die Vertrauensstellung mit dem Server zu überprüfen.

Ist die beiderseitige Vertrauensstellung gewährleistet, so wird die Kommunikationsverbindung initiiert. Für die Verschlüsselung eines Requests vom Client an den Server wird der Public Key des Servers verwendet. Die Response vom Server an den Client wird dann mit dem Public Key des Clients verschlüsselt. Beide Kommunikationsteilnehmer haben die Möglichkeit, die jeweils empfangene Nachricht mit ihrem Private-Key zu entschlüsseln.

Das Signieren von Nachrichten erfolgt jeweils umgekehrt: Die Signatur einer Nachricht erfolgt jeweils mit dem Private-Key des Absenders. Da der Empfänger den Public-Key des Absenders erkennt, kann damit die Signatur überprüft werden.

Vertrauensstellung per Dateisystem konfigurieren

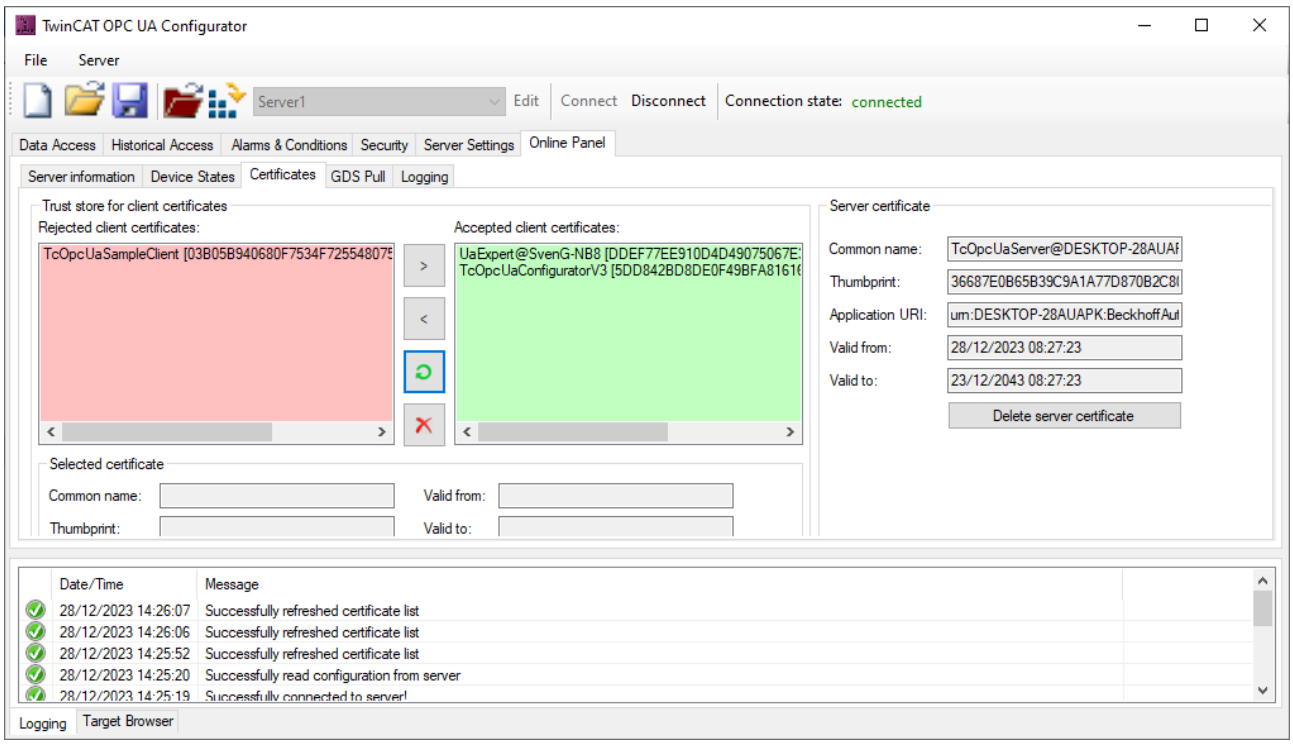
Durch das Verschieben von Clientzertifikaten zwischen den trusted-/rejected-Verzeichnissen können die Vertrauenseinstellungen entsprechend angepasst werden. Der öffentliche Schlüssel eines Clientzertifikats wird beim ersten Verbindungsversuch des Clients mit einem sicheren Endpunkt automatisch im Verzeichnis für nicht-vertrauenswürdige Zertifikate abgelegt. Durch das anschließende Verschieben des öffentlichen Schlüssels in das Verzeichnis für vertrauenswürdige Zertifikate wird dem Client beim nächsten Verbindungsversuch vertraut und er kann sich verbinden.

AutomaticallyTrustAllClientCertificates

Wenn diese Konfigurationsoption im Server aktiviert ist, vertraut der Server automatisch allen Clientzertifikaten. Diese werden in diesem Fall nicht in einem der oben genannten Verzeichnisse aufgelistet.

Vertrauensstellung per Konfigurator konfigurieren

Sie können die Vertrauenseinstellungen auch über den Konfigurator [▶ 40] durchführen. Der TwinCAT OPC UA Configurator beinhaltet entsprechend eine grafische Benutzeroberfläche zur Konfiguration der Vertrauenseinstellungen.



4.16.4 Authentifizierung

Eine OPC-UA-Client-Applikation kann sich über verschiedene IdentityToken am TwinCAT OPC UA Server authentifizieren. Hierbei werden die folgenden IdentityToken unterstützt:

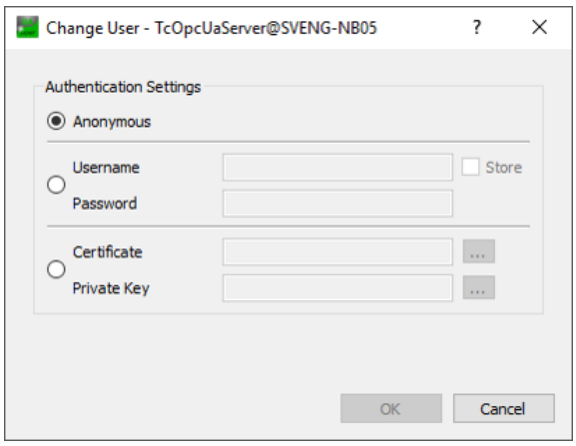
- Anonymous
- Benutzername/Passwort
- Benutzerzertifikat

Auslieferungszustand

Im Auslieferungszustand des Servers ist das IdentityToken „Anonymous“ aktiviert; der Server erfordert für die Inbetriebnahme jedoch eine einmalige Initialisierung. Anschließend wird dieses IdentityToken deaktiviert und Client-Applikationen müssen sich mit einem Benutzernamen am Server authentifizieren.

Anonymous

Diese Art der Authentifizierung ermöglicht es, beliebigen OPC UA Clients eine Verbindung zur Serverapplikation herzustellen. Die Angabe einer Benutzeridentität ist hierbei nicht erforderlich, wodurch sich auch keinerlei Möglichkeiten bieten, Zugriffsrechte auf dem Server zu definieren. Beckhoff empfiehlt, diese Authentifizierungsart nach der Inbetriebnahme des Servers zu deaktivieren. Dies kann über den TwinCAT-OPC-UA-Konfigurator erfolgen. Im Folgenden finden Sie einen Beispiel-Screenshot aus der OPC-UA-Client-Applikation "UA Expert":



Benutzername/Passwort

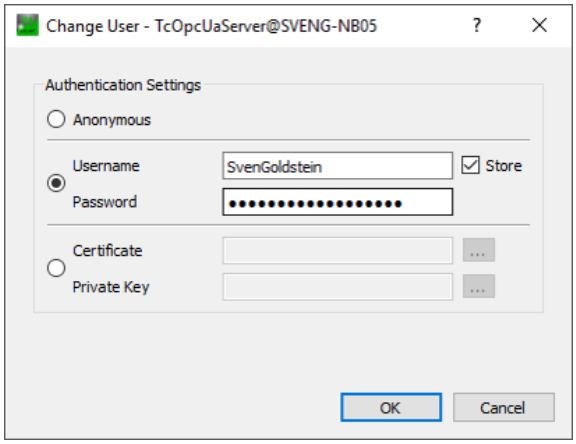
Diese Art der Authentifizierung verwendet eine Benutzername-/Passwort-Kombination zum Authentifizieren des Clients an der Serverapplikation. Auf dem Server lassen sich dann Zugriffsrechte für die jeweilige Benutzeridentität definieren. Die Benutzeridentität kann hierbei auf verschiedenen Ebenen definiert sein:

- Benutzeridentität ist im Server definiert
- Benutzeridentität kommt aus dem unterlagerten Betriebssystem (z. B. ein lokaler Windows Benutzer)
- Benutzeridentität kommt aus dem Active Directory (z. B., wenn der Industrie-PC Teil einer Windows Domäne ist)

Empfehlung bei Verwendung von User IdentityToken

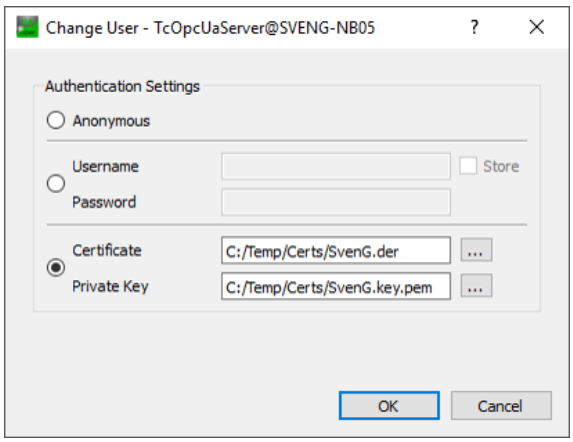
Wenn User IdentityToken zur Authentifzierung von Client-Applikationen verwendet werden sollen, empfiehlt Beckhoff die Verwendung von Betriebssystem-Benutzern.

Im Folgenden finden Sie einen Beispiel-Screenshot aus der OPC UA Client Applikation "UA Expert":



Benutzerzertifikat

Diese Art der Authentifizierung verwendet ein Zertifikat, um sich an der Serverapplikation zu authentifizieren. Die Handhabung der Benutzerzertifikate auf Serverseite ist identisch zur Verwendung von Zertifikaten auf Transportebene, d. h. der Server muss dem (Benutzer-) Zertifikat vertrauen, bevor sich der Client mit dem Zertifikat erfolgreich am Server authentifizieren kann. Ein separates Verzeichnis ("pkiuser") zur Verwaltung der Benutzerzertifikate steht hierfür im Server zur Verfügung. Im Folgenden finden Sie einen Beispiel-Screenshot aus der OPC UA Client Applikation "UA Expert":



Zusätzlich muss für die Verwendung dieser Authentifizierungsmethode über den TwinCAT OPC UA Configurator ein X.509-Benutzer angelegt werden. Der Name des Benutzers muss mit dem Common Name des Zertifikats übereinstimmen.

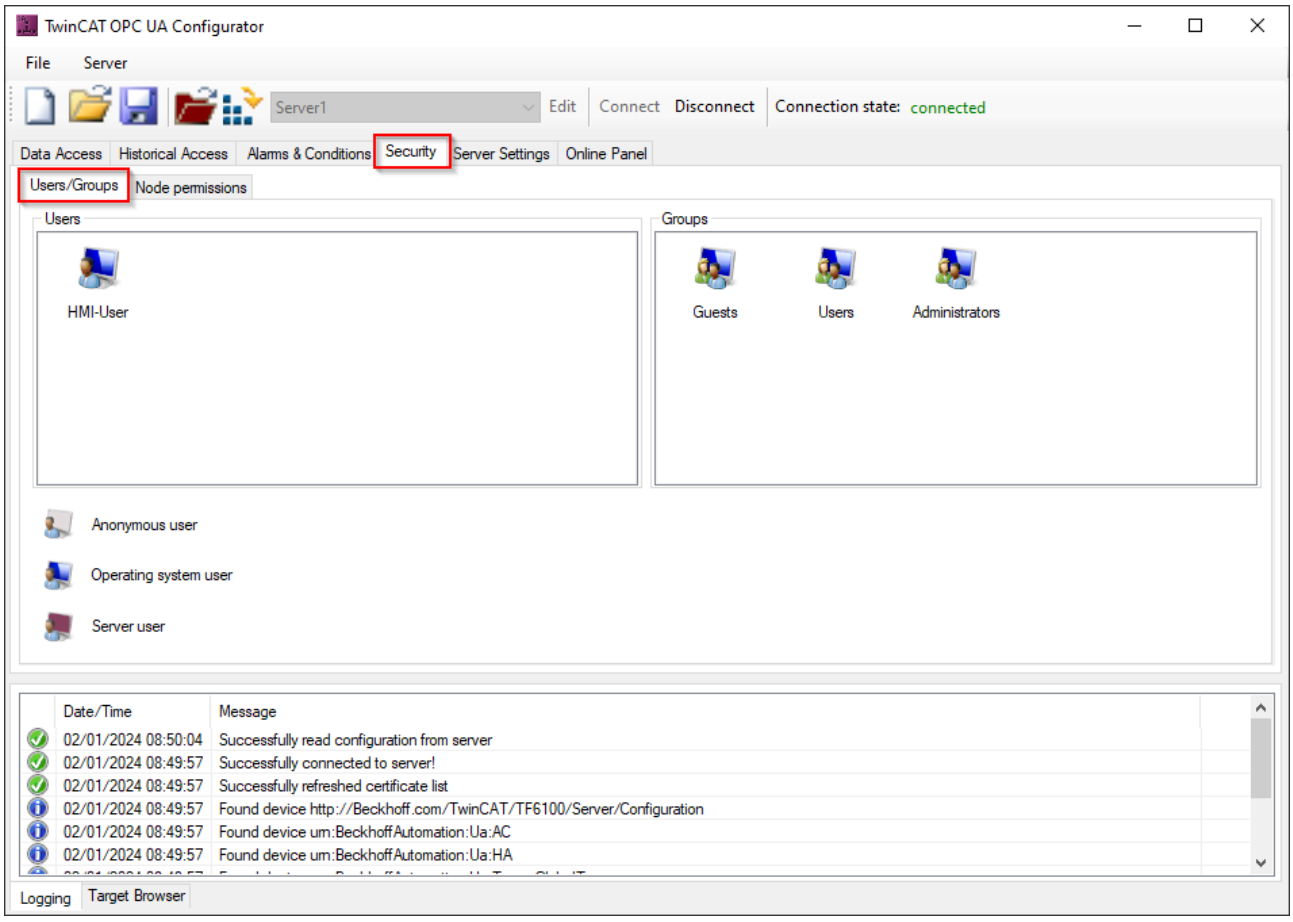
HINWEIS

Authentifizierung und Serverzertifikat

Der TwinCAT OPC UA Client benötigt bei der Verwendung des unverschlüsselten Endpunkts in Verbindung mit Authentifizierung dennoch den Public-Key vom OPC UA Server Zertifikat, um das Passwort bei der Übertragung zu verschlüsseln. Dazu muss diesem Zertifikat im TwinCAT OPC UA Client vertraut werden (siehe Zertifikatsaustausch [▶ 126]).

Konfiguration

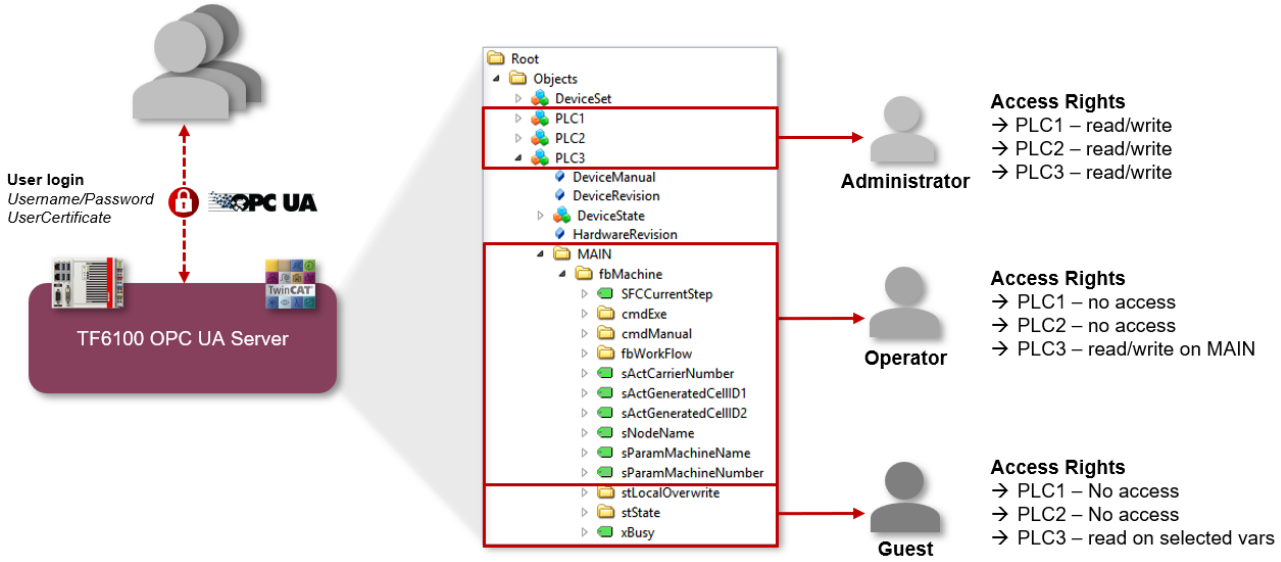
Die Konfiguration der IdentityToken erfolgt üblicherweise über den TwinCAT OPC UA Configurator. Hier steht Ihnen eine grafische Benutzeroberfläche zur Verfügung, um IdentityToken zu aktivieren, zum Beispiel im Standalone Konfigurator:



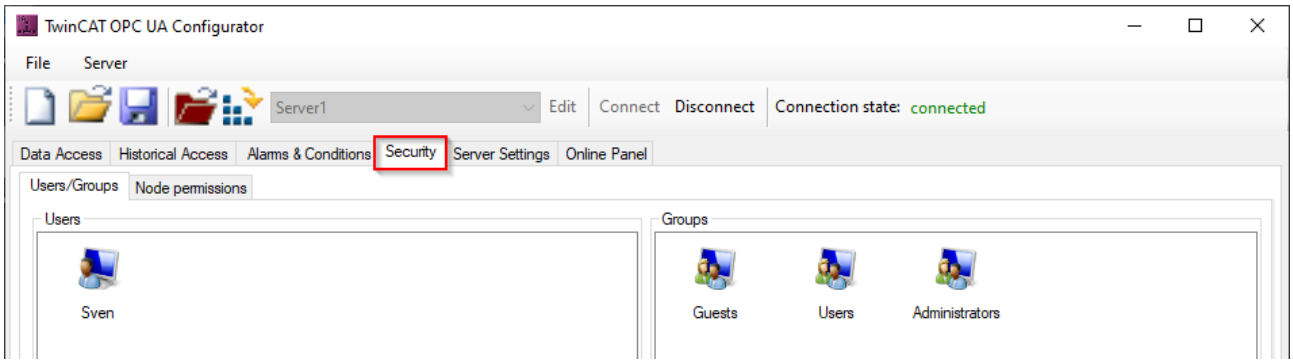
4.16.5 Zugriffsrechte

Der TwinCAT OPC UA Server ermöglicht die Konfiguration von Zugriffsrechten für bestimmte authentifizierte Benutzeridentitäten [▶ 128]. Diese Zugriffsrechte können sowohl für ganze Namespaces als auch für individuelle Nodes konfiguriert werden. Hierdurch kann sowohl der Zugriff auf ADS-Geräte (z. B. auf verschiedene SPS-Laufzeiten) als auch einzelne Symbole feingranular eingestellt werden.

Diese Sicherheitseinstellungen sind für alle Data-Access [▶ 49]-Geräte verfügbar, die im Server-Namespace dargestellt werden können.

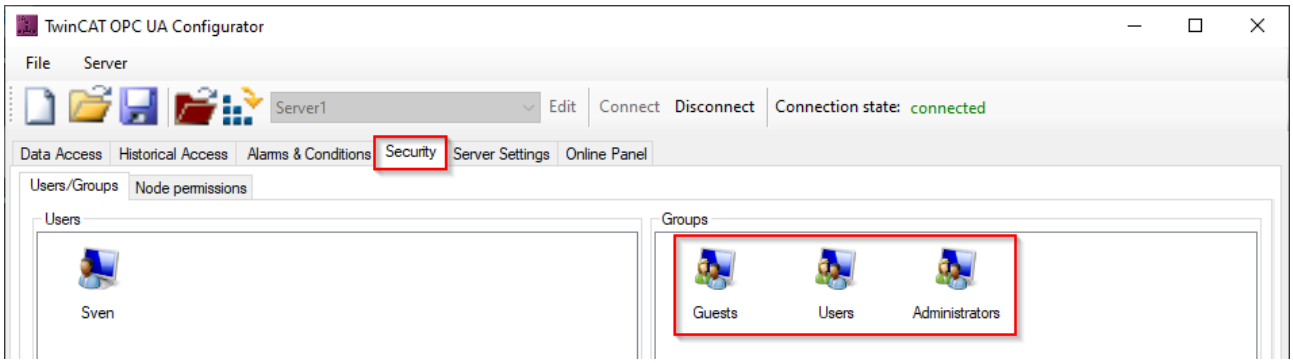


Die Konfiguration dieser Funktionalität erfolgt über den TwinCAT OPC UA Configurator. In der Standalone-Variante des Konfigurators findet man die entsprechende Konfigurationsoberfläche unterhalb der Registerkarte **Security**.



Konfiguration des Zugriffs auf Namespaces

Die Konfiguration des Benutzerzugriffs auf einzelne Namespaces erfolgt immer auf Basis der konfigurierten Benutzergruppen. In den Einstellungen der Benutzergruppe können Sie die entsprechenden Zugriffsrechte für einzelne Namespaces verwalten. Im Auslieferungszustand sind bereits einige Gruppen vorkonfiguriert an deren Konfigurationsparametern man sich orientieren kann.



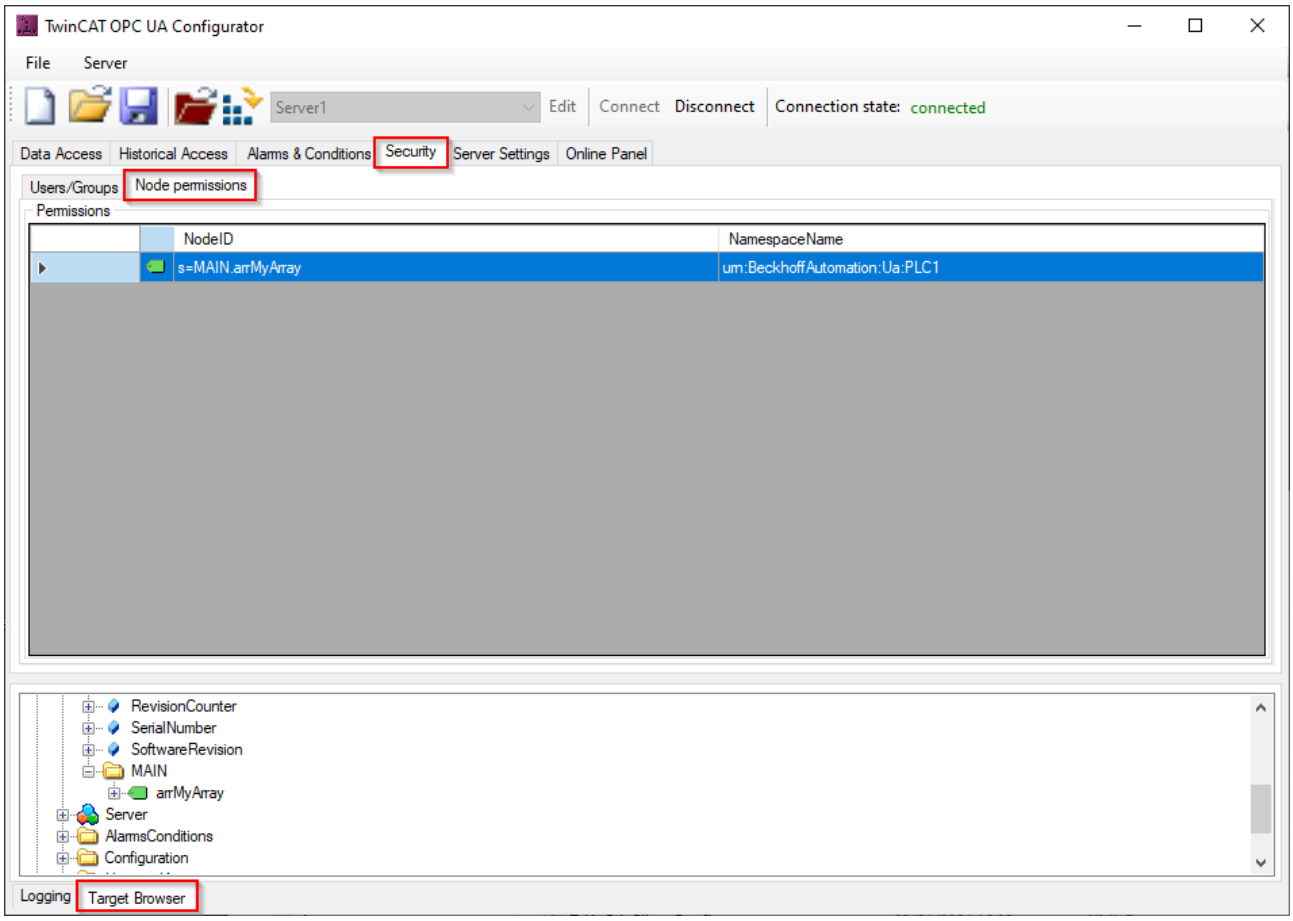
Die folgende Tabelle gibt einen Überblick über die im Auslieferungszustand definierten Berechtigungen der einzelnen Benutzergruppen.

Benutzergruppe	Beschreibung
Administrators	Vordefinierte Benutzergruppe für Serveradministratoren. Diese Benutzergruppe hat Vollzugriff auf alle Namespaces.
Guests	Vordefinierte Benutzergruppe für Gast-Benutzer. Diese Benutzergruppe hat nur eingeschränkten Zugriff auf den Server und hierbei nur auf den Default-Namespace „0“ mit den Berechtigungen ReadAttribute, ReadValue und Browse.
Users	Vordefinierte Benutzergruppe für normale Benutzer. Diese Benutzergruppe hat erweiterte Zugriffsrechte auf alle Namespaces, insbesondere Vollzugriff auf den Namespace des vorkonfigurierten Data-Access [▶ 49]-Geräts.

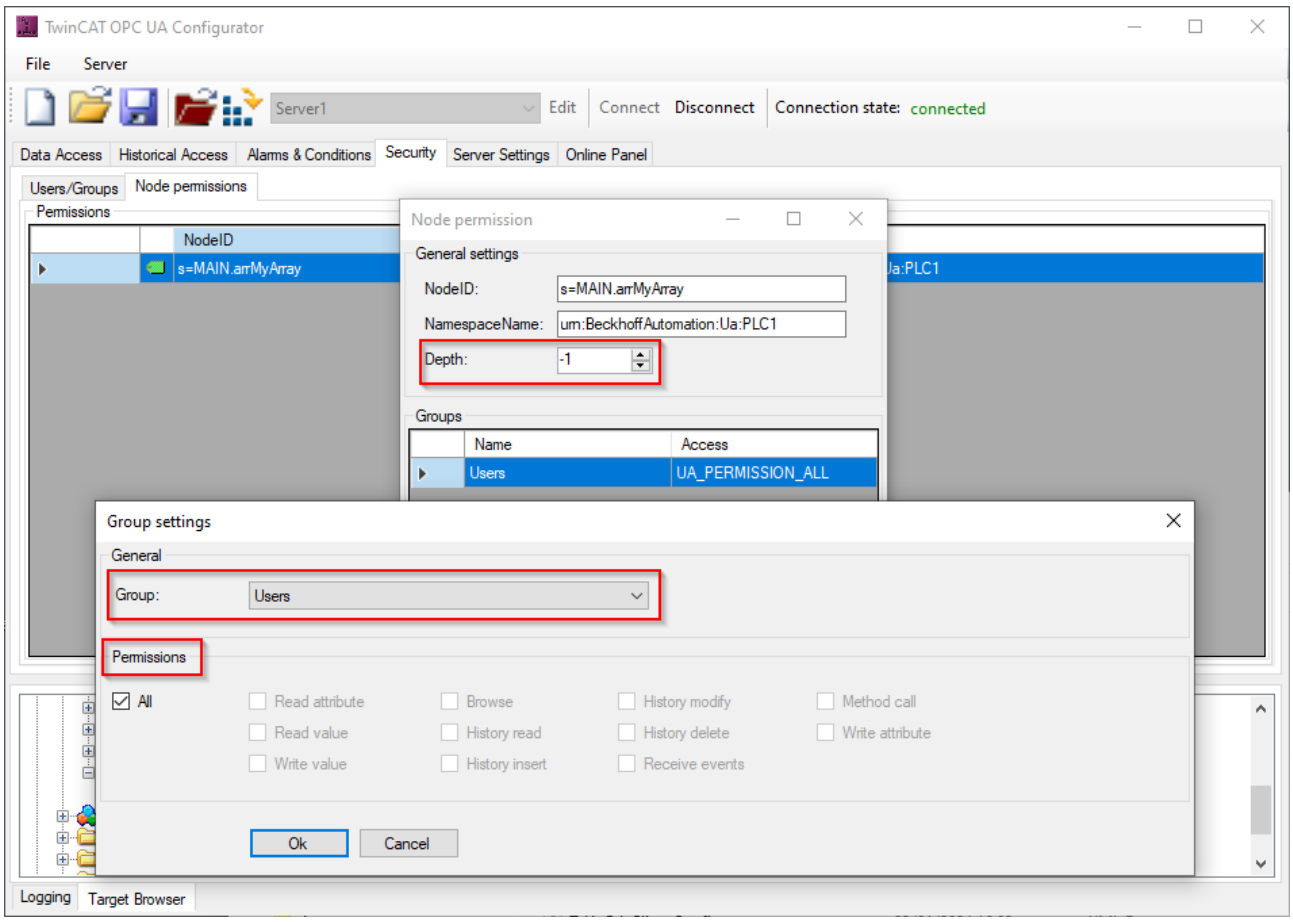
Durch das Hinzufügen von Benutzern zu den Benutzergruppen bekommen diese automatisch die entsprechenden Berechtigungen von der Gruppe vererbt.

Konfiguration des Zugriffs auf Node-Ebene

Über die Registerkarte **Node permissions** lassen sich erweiterte und sehr feingranulare Berechtigungen auf Node-Ebene konfigurieren. Die Berechtigungen lassen sich hierbei auch auf Unterelemente vererben. Über den Target Browser können Sie zu konfigurierende Nodes vom Server per Drag-and-Drop in die Konfiguration übernehmen.



Die einzelnen Berechtigungen an der Node lassen sich mit einer konfigurierten Benutzergruppe verknüpfen.

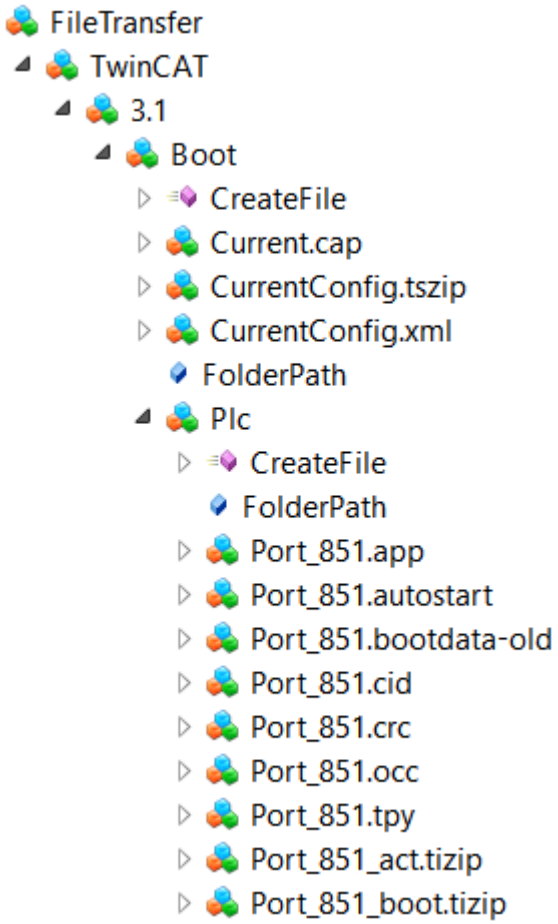


4.17 File Transfer

4.17.1 Überblick

Die OPC-UA-Spezifikation enthält einen speziellen Datentyp zur Übertragung von Dateien. Dieser spezielle ObjectType namens „FileType“ beschreibt das Informationsmodell für die Datenübertragung. Dateien können in OPC UA mit ByteStrings als einfache Variablen modelliert werden. FileType ist eine Datei mit Methoden zum Zugriff auf die Datei. In der OPC-UA-Spezifikation erhalten Sie weitere Informationen zu FileType sowie Aufbau und Handhabung der zugrunde liegenden Methoden und Eigenschaften zum Zugriff auf eine Datei im OPC-UA-Namensraum.

Beckhoff hat einen generischen Weg implementiert, um Dateien und Ordner von einer lokalen Festplatte in den OPC-UA-Namensraum zu laden. Jede Datei wird durch einen FileType repräsentiert und ermöglicht Lese- und Schreibvorgänge für diese Datei. Zusätzlich enthält jeder Ordner eine Methode CreateFile(), um neue Dateien auf der Festplatte zu erstellen und einen eigenen FolderPath, um den tatsächlichen Pfad zum Ordner auf dem OPC UA Server festzulegen.



FileTransfer im Device Manager OPC UA Server

Diese Funktion hat nur der OPC UA Server des Beckhoff Device Managers (IPC Diagnose). Der TwinCAT OPC UA Server stellt ebenfalls einige Teile dieser Dateiübertragung bereit. Die allgemeine Funktion, die eine Offenlegung aller Dateien und Ordner ermöglicht, steht aber nur im OPC UA Server zur Verfügung, der zum Gerätemanager gehört, der automatisch auf jedem Beckhoff Industrie-PC oder Embedded-PC verfügbar ist. In der Gerätemanagerdokumentation erhalten Sie weitere Informationen.

4.17.2 Konfiguration

FileType-Objekte werden in einem separaten Namensraum mit der Bezeichnung „FileTransfer" erstellt. Zur Konfiguration des Namensraums und zur Auswahl der über OPC UA verfügbaren Dateien und Ordner dient eine XML-Datei (*files.xml*), die in demselben Verzeichnis wie die ausführbare Datei des OPC UA Servers sein muss. Um die Konfiguration zu aktivieren, muss das System neu gestartet werden. Die XML-Datei enthält Informationen über den Ordnerpfad und eine Suchmaske, die definiert, welche Dateien im OPC-UA-Namensraum veröffentlicht werden:

```
<Files>
  <FolderObject DisplayName="TwinCAT">
    <FolderObject DisplayName="3.1">
      <FolderObject DisplayName="Boot" Path="c:/TwinCAT/3.1/Boot" Search="*.*" >
        <FolderObject DisplayName="Plc" Path="c:/TwinCAT/3.1/Boot/Plc" Search="*.*" ></FolderObject>
        <FolderObject DisplayName="Tmi" Path="c:/TwinCAT/3.1/Boot/Tmi" Search="*.*" ></FolderObject>
      </FolderObject>
    </FolderObject>
  </FolderObject>
</Files>
```

Lesen einer Datei mit einem OPC UA Client

Der allgemeine Umgang mit Dateien ist in Anhang C der OPC-UA-Spezifikation beschrieben.
Das Lesen einer Datei via OPC UA kann somit in folgende Schritte unterteilt werden:

- Aufruf der Open-Methode einer Datei. Diese Methode gibt ein Dateihandle zurück, das für den späteren Zugriff gespeichert werden muss. Der Modus legt fest, ob die Datei gelesen oder ob in sie geschrieben wird (siehe Dateimodi).
- Bestimmen der Größe der Datei mit der Eigenschaft „Size“. So kann die ganze Datei bei Aufruf der Read-Methode gelesen werden.
- Aufruf der Read-Methode. Dateihandle und Dateigröße als Eingaben einfügen. Zielordner wählen, in den der Dateiinhalt NACH dem Aufruf der Methode zu speichern ist.
- Aufruf der Close-Methode zur Freigabe des Dateihandles.

Dateimodi

Die folgende Tabelle zeigt alle verfügbaren Dateimodi:

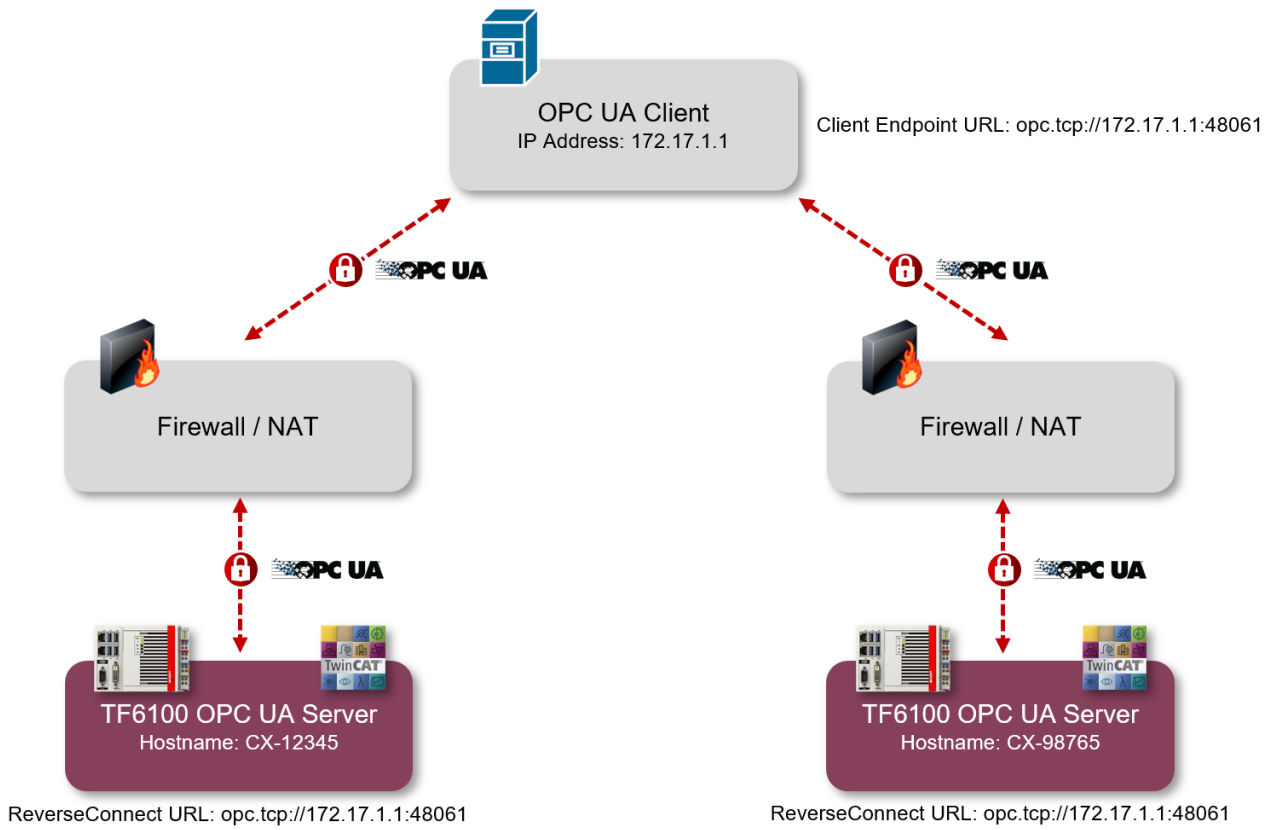
Feld	Bit	Beschreibung
Lesen	1	Die Datei wird zum Lesen geöffnet. Wenn dieses Bit nicht gesetzt ist, kann Read nicht ausgeführt werden.
Schreiben	4	Die Datei wird zum Schreiben geöffnet. Wenn dieses Bit nicht gesetzt ist, kann Write nicht ausgeführt werden.
EraseExisting	6	Der vorhandene Dateiinhalt wird gelöscht und es wird eine leere Datei zur Verfügung gestellt.
Append	10	Die Datei wird geöffnet und ans Ende positioniert, sonst auf den Anfang. Diese Position kann mit SetPosition geändert werden.

Generelles Verhalten

Die Anzahl an parallel geöffneten Dateien ist prinzipiell unlimitiert und unterliegt ggf. nur etwaigen Einschränkungen des zugrunde liegenden Betriebssystems. Dateien unterliegen jedoch einem 60-Sekunden Timeout. Nach diesem Timeout werden offene Dateien nicht sofort automatisch geschlossen. Anstelle dessen werden sie als „zu-schließend“ markiert. Wenn während dieser Zeit das entsprechende FileHandle für eine Read-/Write-Operation verwendet wird, wird der Timeout zurückgesetzt und das FileHandle bleibt gültig. Wenn während dieser Zeit eine Open-Operation auf derselben Datei durchgeführt wird, wird das alte FileHandle freigegeben. Wenn ein OPC UA Client seine Verbindung zum Server trennt und noch Dateien geöffnet hat, werden alle FileHandles, die zu dieser Sitzung gehören, automatisch geschlossen.

4.18 Reverse Connect

Der TwinCAT OPC UA Server unterstützt die ReverseConnect-Funktion von OPC UA, um eine rückwärtsgerichtete Kommunikationsverbindung vom Server zum Client aufzubauen. Um diese Funktion zu aktivieren, muss im Server eine Liste mit Client-Adressen hinterlegt werden. Anschliessend baut der Server für jeden Client in der Liste eine OPC-UA-TCP-Verbindung auf. Diese Art des Verbindungsaufbaus wird oftmals eingesetzt, wenn ein OPC UA Client Verbindungen mit Servern aufbauen soll, welche sich hinter einer Firewall oder NAT-Gerät befinden. Das folgende Schaubild verdeutlicht diesen Zusammenhang.



In diesem Beispiel gibt es einen OPC UA Client, welcher eine Verbindung zu zwei Servern aufbauen soll, die sich jeweils hinter einer Firewall befinden. Die Firewall blockiert hierbei sämtlichen eingehenden Kommunikationsverkehr und öffnet keinerlei Ports im internen Netzwerk. Der Client ist nun für ReverseConnect konfiguriert und öffnet einen eigenen Netzwerkport unter der Client Endpoint URL opc.tcp://172.17.1.1:48061. Jeder der unterlagerten Server ist ebenfalls für ReverseConnect eingerichtet worden und hat hierbei als ReverseConnect URL die Client Endpoint URL eingetragen. Der Server öffnet nun unter Verwendung dieser URL eine TCP-Verbindung zum Client und hält diese aufrecht. Durch diese TCP-Verbindung wird die eigentliche OPC-UA-Client-Kommunikation mit dem Server getunnelt. Aus Sicht der Firewall handelt es sich durch die initial aufgebaute TCP-Verbindung um eine „ausgehende“ Kommunikation. Hierbei muss nur der ausgehende Kommunikationsport (in diesem Beispiel 48061) in der Firewall freigegeben werden.

Kompatibilität von Clients

Bitte beachten Sie, dass der OPC UA Client ebenfalls diese Funktion unterstützen und über seine Client Endpoint URL erreichbar sein muss.

Konfiguration im Server

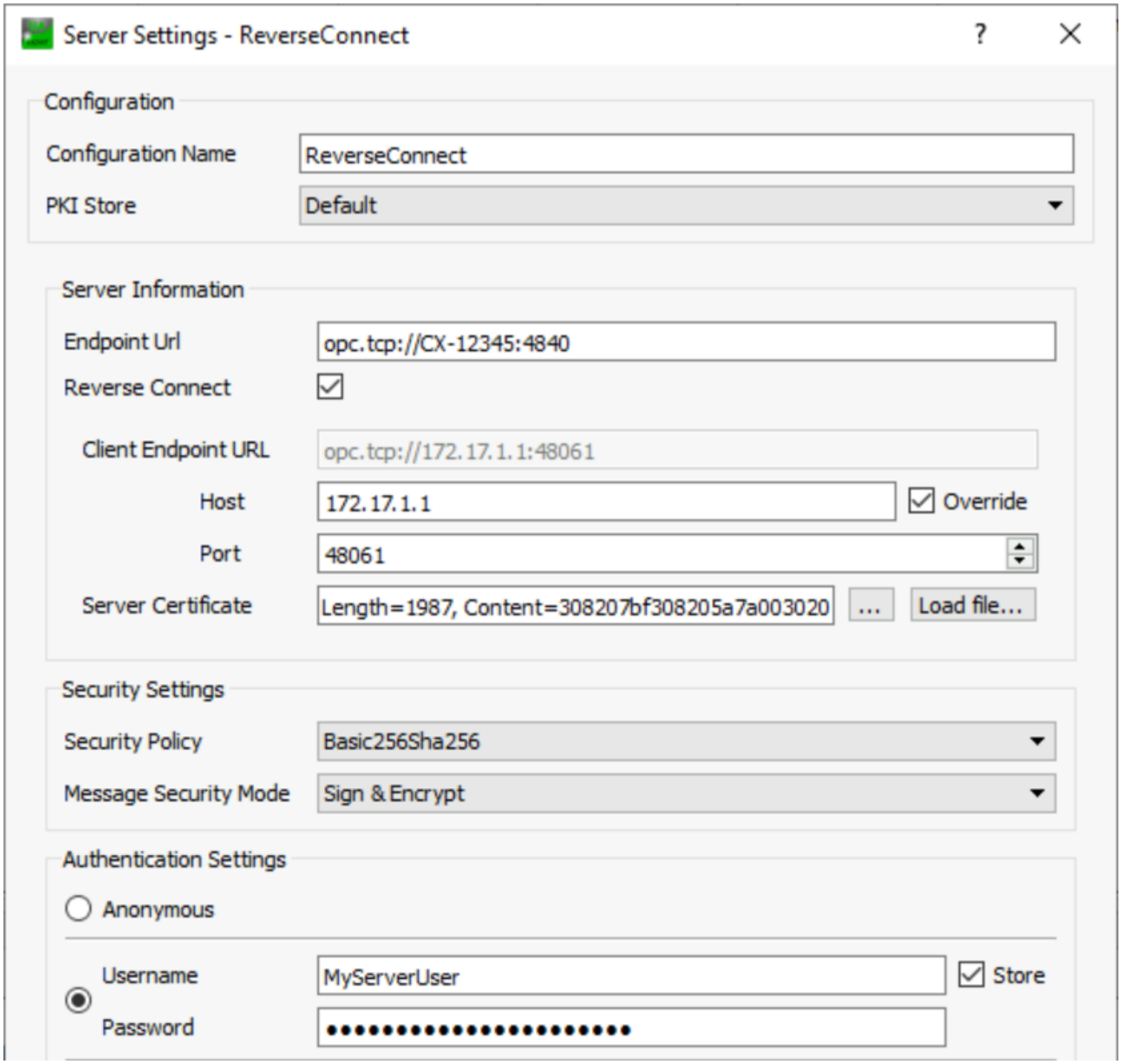
Eine Liste mit OPC UA Clients kann in der TcUaServerConfig.xml innerhalb des <UaEndpoint> konfiguriert werden. Hier wird die Client Endpoint URL eingetragen, unter der die jeweiligen Clients erreichbar sind.

```
<ReverseConnect>
  <Url>opc.tcp://172.17.1.1:48061</Url>
</ReverseConnect>
```

Konfiguration im OPC UA Client

Der folgende Screenshot zeigt exemplarisch die Konfiguration einer ReverseConnect-Verbindung in der OPC UA Client Software „UA Expert“ von Unified Automation. In den Verbindungseinstellungen wird hierzu der ReverseConnect aktiviert und die Client Endpoint URL eingetragen, unter der der Client für den Server erreichbar ist. In dem Feld EndpointURL wird die Server Endpoint URL eingetragen, welche der Client verwenden soll sobald vom Server eine ReverseConnect TCP-Verbindung aufgebaut wurde. Bezugnehmend auf unser obiges Schaubild wird hier zum Beispiel opc.tcp://CX-12345:4840 bzw. opc.tcp://CX-98765:4840 eingetragen. Die Einstellungen zu SecuritySettings, MessageSecurityMode sowie die Authentifizierungsparameter gelten dann ebenfalls für diese Verbindung.

Bitte beachten Sie, dass Sie ggf. noch das Serverzertifikat importieren müssen. Hierbei handelt es sich um den Public-Key des Servers, welcher im entsprechenden Applikationsverzeichnis [▶ 47] hinterlegt ist.



Kommunikationsverlauf

Das folgende Wireshark® Trace zeigt exemplarisch einen Verbindungsaufbau, basierend auf ReverseConnect. Der einzige Unterschied zu obigem Schaubild ist, dass der in diesem Mitschnitt verwendete Client für die Client Endpoint URL opc.tcp://172.30.3.86:48061 konfiguriert wurde. Der Server sitzt hinter einem NAT-Gerät, welches seinerseits die IP-Adresse 178.200.200.59 besitzt.

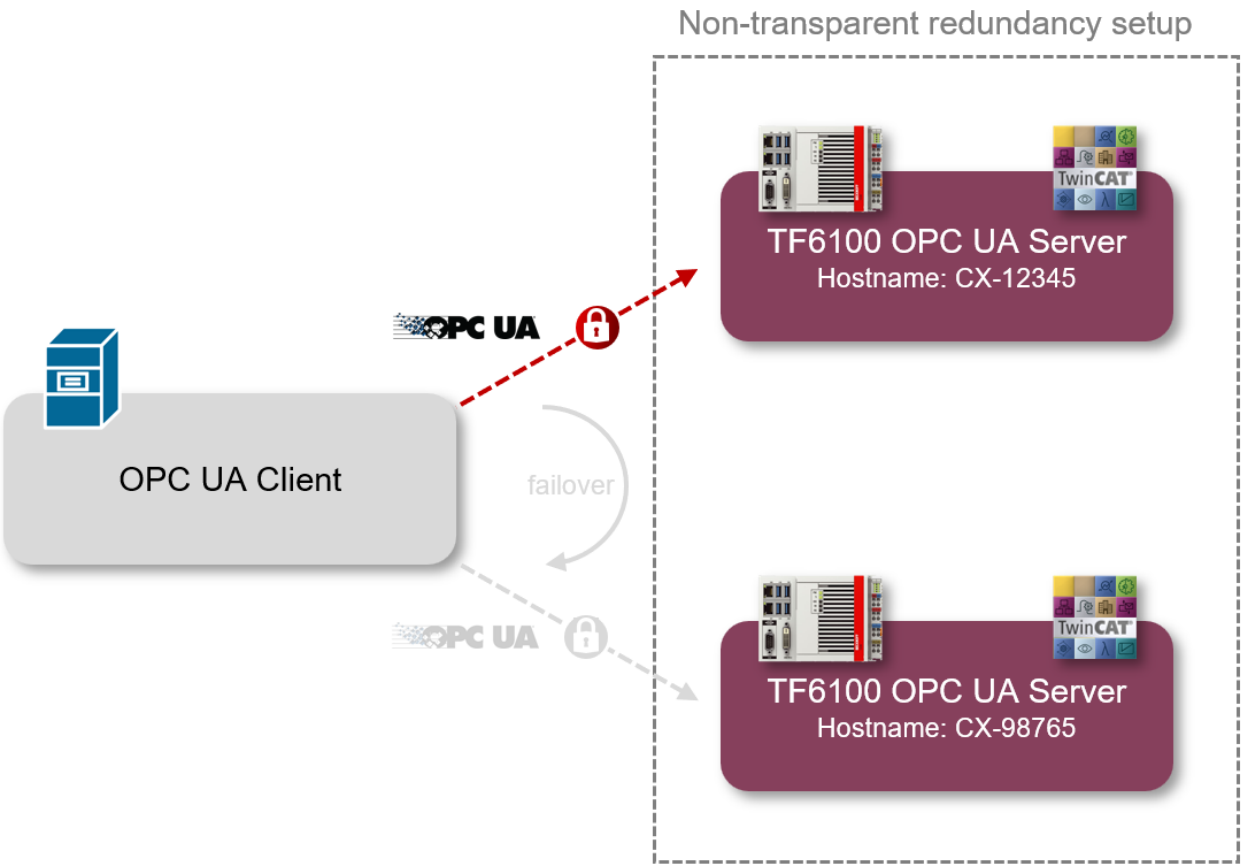
No.	Time	Source	Destination	Protocol	Length	Info
608	5.228616	178.200.200.59	172.30.3.86	OpcUa	154	Reverse Hello message
609	5.228927	172.30.3.86	178.200.200.59	OpcUa	116	Hello message
610	5.249617	178.200.200.59	172.30.3.86	OpcUa	82	Acknowledge message
612	5.344295	172.30.3.86	178.200.200.59	OpcUa	3039	OpenSecureChannel message: ServiceId 0
620	5.541799	178.200.200.59	172.30.3.86	OpcUa	246	OpenSecureChannel message: ServiceId 0
626	5.868707	172.30.3.86	178.200.200.59	OpcUa	2246	UA Secure Conversation Message: ServiceId 0
646	5.992669	178.200.200.59	172.30.3.86	OpcUa	1486	UA Secure Conversation Message: ServiceId 0

Beim Aufbau der TCP-Verbindung vom Server zum Client wird eine sogenannte „Reverse Hello“-Nachricht versendet. In dieser gibt der Server dem Client bekannt, unter welcher Server Endpoint URL er erreichbar ist. Dies ist dieselbe Server Endpoint URL, welche Sie im Client konfiguriert haben (siehe oben). Der Client verwendet diese Server Endpoint URL für den weiteren Verbindungsaufbau zum Server.

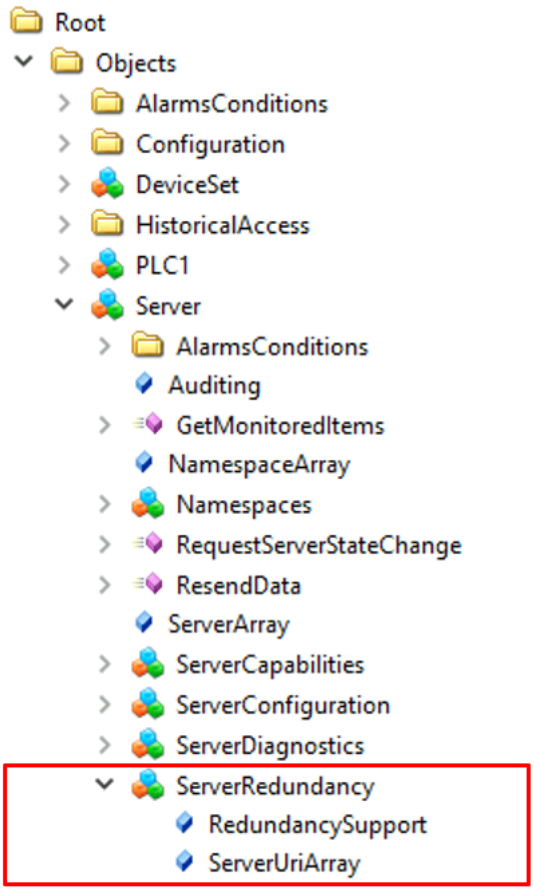
4.19 Redundanz

Die OPC UA Spezifikation definiert zwei Arten der Server-Redundanz: transparente und nicht-transparente Redundanz. Bei der transparenten Redundanz ist die Übertragung der Funktionen von einem Server zum anderen transparent für den Client, d.h. dieser weiß nicht, dass ein Failover-Vorgang stattgefunden hat und hat dementsprechend auch keinerlei Kontrolle über das Failover-Verhalten. Des Weiteren muss der Client keine Aktionen durchführen, um weiterhin Daten zu senden oder zu empfangen. Bei der nicht-transparenten Redundanz wird der Failover-Vorgang von einem Server zum anderen und die Aktionen zum weiteren Senden oder Empfangen von Daten vom Client durchgeführt. Der Client muss hierbei die Redundanzkonfiguration des Servers kennen und die erforderlichen Aktionen durchführen, um im Falle eines Failovers von der Server-Redundanz zu profitieren. Die nicht-transparente Redundanz definiert verschiedene Failover-Modi, welche unter Anderem angeben, wie viele Server aktiv an der Kommunikation beteiligt sind, oder als reine Passivsysteme für den Failover-Fall bereitstehen.

Der TwinCAT OPC UA Server unterstützt die nicht-transparente Redundanz mit den Betriebsmodi „cold“, „warm“ und „hot“. Die Unterschiede in diesen drei Betriebsmodi werden weiter unten erläutert.



Für die nicht-transparente Redundanz stellt OPC UA die Datenstrukturen zur Verfügung, die es dem Client ermöglichen, zu erkennen, welche Server in der redundanten Servergruppe verfügbar sind, sowie Serverinformationen, die dem Client mitteilen, welche Failover-Modi der Server unterstützt. Anhand dieser Informationen kann der Client feststellen, welche Maßnahmen er ergreifen muss, um Failover zu erreichen. Im Adressraum des Servers existiert hierfür das sogenannte ServerRedundancy Objekt, welches den vom Server unterstützten Redundanz-Modus und das Redundanz-Set, d.h. alle Server die Bestandteil der Redundanzkonfiguration sind, anzeigt.



#	Server	Node Id	Display Name	Value	Datatype
1	TcOpcUaServer...	NS0|Numeric|...	RedundancySupport	1 (Cold)	Int32
2	TcOpcUaServer...	NS0|Numeric|...	ServerUriArray	{'urn:EC2AMAZ-R1LR85K:BeckhoffAutomation:TcOpcUaServer:...	String

Failover-Modi

Die folgende Tabelle listet alle unterstützten Failover-Modi für die nicht-transparente Redundanz auf.

Modus	Beschreibung
Cold	Im Cold-Failover-Modus kann jeweils nur ein Server aktiv sein. Dies kann bedeuten, dass redundante Server nicht verfügbar sind (nicht eingeschaltet) oder zwar verfügbar sind, aber nicht laufen (der PC läuft, aber die Anwendung ist nicht gestartet).
Warm	Im Modus „Warm Failover“ können die Backup-Server aktiv sein, aber keine Verbindung zu den tatsächlichen Datenpunkten herstellen (typischerweise ein System, bei dem die zugrunde liegenden Geräte auf eine einzige Verbindung beschränkt sind). Die zugrundeliegenden Geräte, wie z. B. die SPS, verfügen möglicherweise über begrenzte Ressourcen, die nur eine einzige Serververbindung zulassen. Daher ist nur ein einziger Server in der Lage, Daten abzurufen.
Hot	Im Hot-Failover-Modus sind alle Server eingeschaltet und betriebsbereit. In Szenarien, in denen Server Daten von einem nachgeschalteten Gerät, z. B. einer SPS, abrufen, sind ein oder mehrere Server aktiv und parallel mit dem/den nachgeschalteten Gerät(en) verbunden. Diese Server haben nur minimale Kenntnis von den anderen Servern in ihrer Gruppe und arbeiten unabhängig. Wenn ein Server ausfällt oder ein schwerwiegendes Problem auftritt, sinkt sein ServiceLevel (siehe unten). Nach der Wiederherstellung kehrt der Server mit einem geeigneten ServiceLevel zum Redundant Server Set zurück, um anzuzeigen, dass er verfügbar ist.

ServiceLevel

Der ServiceLevel liefert einem Client Informationen über den Zustand eines Servers und dessen Fähigkeit, Daten zu liefern. Der ServiceLevel ist eine Node vom Datentyp Byte mit einem Wertebereich von 0 bis 255. Der TwinCAT OPC UA Server setzt den ServiceLevel aktuell statisch auf den Wert 255. Der Grund hierfür ist, dass der Server in der Lage ist, Daten von mehr als einer unterlagerten SPS-Steuerung abzurufen. Der ServiceLevel kann hierbei also ausschließlich den Zustand der Serveranwendung an sich, aber nicht der unterlagerten SPS-Steuerungen anzeigen. Für letzteres steht in den jeweiligen SPS-Adressräumen das DeviceState [▶ 91] Objekt zur Verfügung.

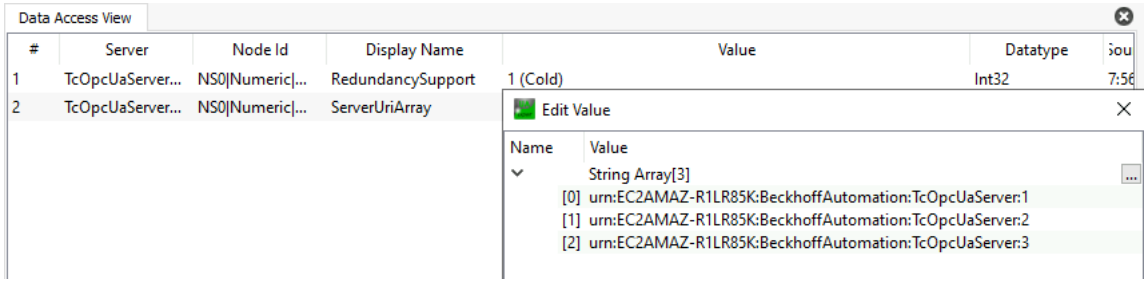
Konfiguration

Die Konfiguration der Server-Redundanz lässt sich mit Hilfe des TwinCAT OPC UA Configurator durchführen. Hierzu sind zwei Schritte notwendig:

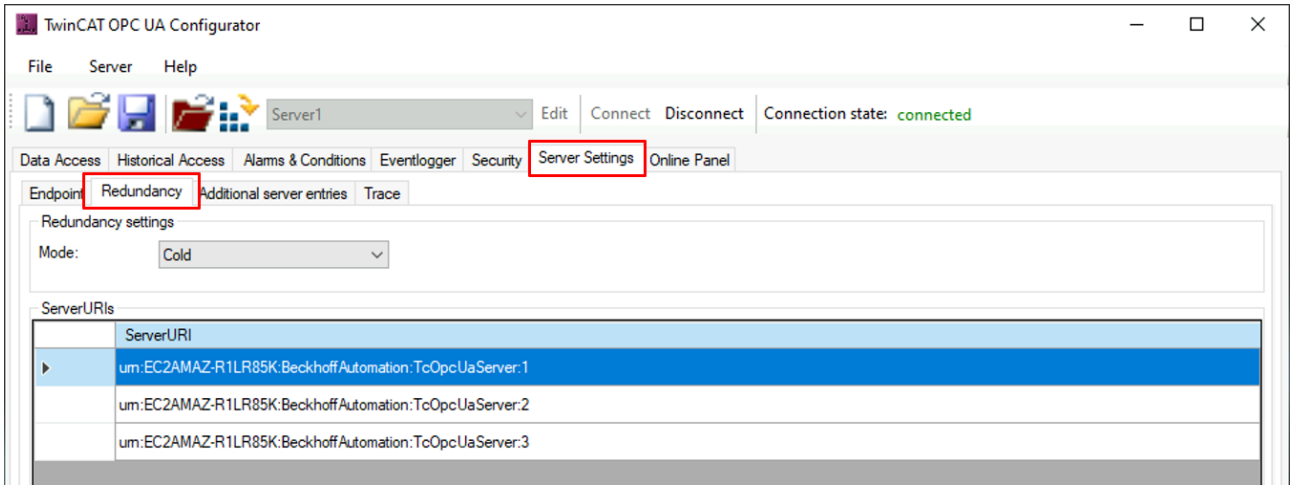
- Konfiguration der Redundanz
- Hinzufügen der Server aus dem Redundanz-Set als AdditionalServerEntry

Konfiguration der Redundanz

Bei der Konfiguration der Server-Redundanz wird definiert, welchen Failover-Modus der Server bereitstellt und welche anderen Server Bestandteil der Redundanz-Konfiguration sein sollen. Diese Server werden über deren ServerURI identifiziert. Der Failover-Modus, sowie eine Liste der ServerURIs werden dann vom Server in dessen ServerRedundancy-Objekt (s.o.) hinterlegt.



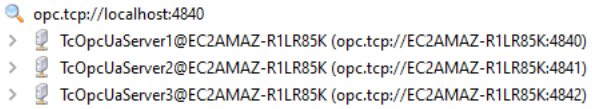
Die Konfiguration dieser Einstellungen kann über den TwinCAT OPC UA Configurator in der Registerkarte „Server Settings“ erfolgen, z.B.:



Da die ServerURI an sich noch keine Verbindungsinformationen (z.B. die Server- bzw. Discovery-URL) beinhalten, müssen diese Informationen dem Client an anderer Stelle zur Verfügung gestellt werden. Dies erfolgt über die sogenannte AdditionalServerEntry Konfiguration. Hierdurch kann ein Client durch einen FindServers-Aufruf alle erforderlichen Informationen vom Server abfragen und die Verbindungsinformationen für eine bestimmte ServerURI identifizieren. Dieser Konfigurationsschritt wird im Folgenden erläutert.

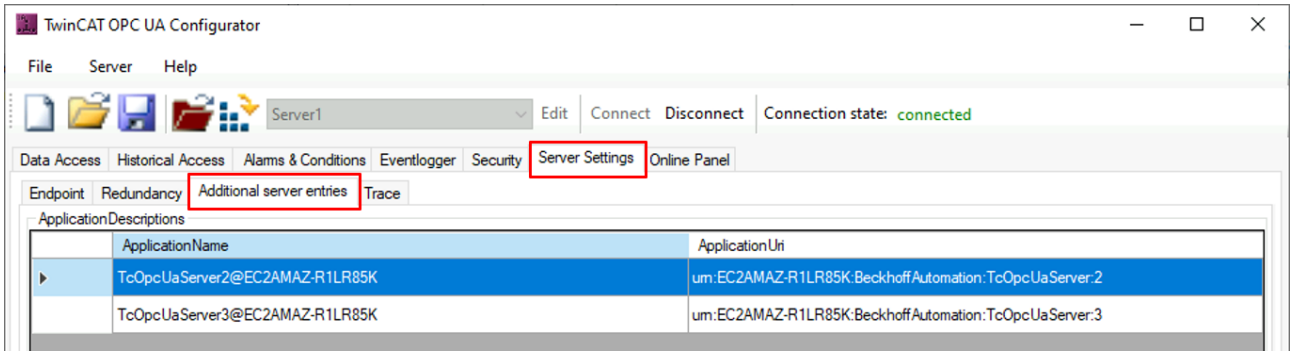
Hinzufügen der Server aus dem Redundanz-Set als AdditionalServerEntry

Bei der AdditionalServerEntry-Konfiguration werden Informationen von anderen TwinCAT OPC UA Servern zum FindServers Aufruf eines Servers hinzugefügt. Ein Client hat hierdurch die Möglichkeit, alle Server aus der Redundanz-Konfiguration aufzufinden und sich mit diesen zu verbinden. Der folgende Screenshot zeigt einen exemplarischen FindServers-Aufruf auf dem lokalen TwinCAT OPC UA Server. Der Aufruf returniert drei TwinCAT OPC UA Server, welche auch Bestandteil einer Redundanz-Konfiguration sind.

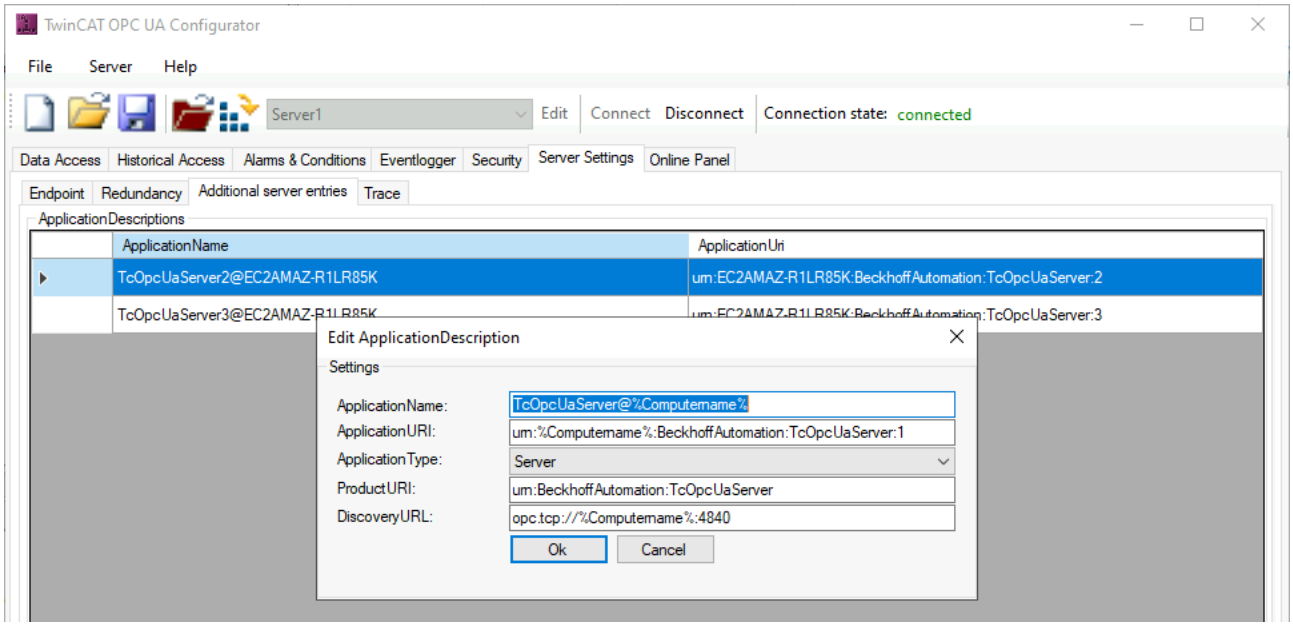


Zur Identifikation eines Servers verwendet der Client die ServerURI aus der Redundanz-Konfiguration.

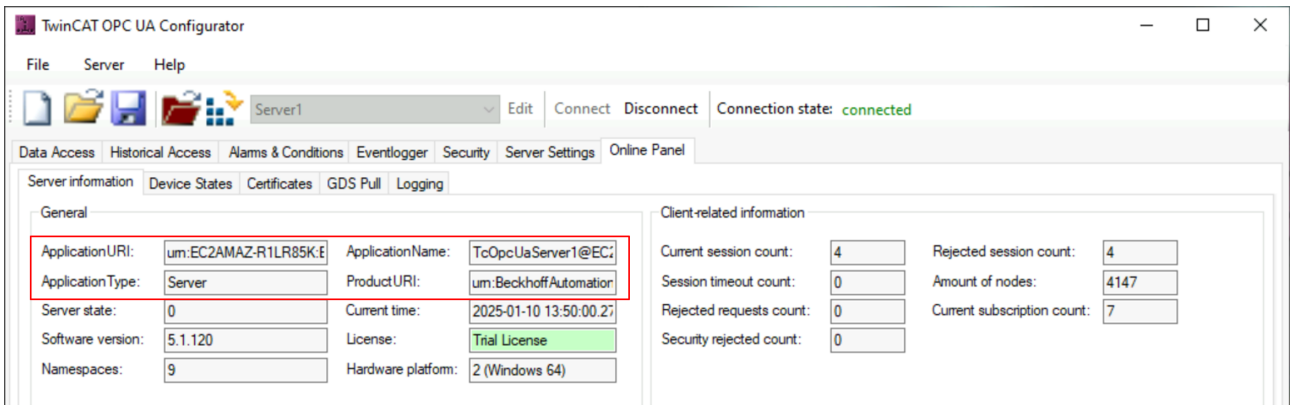
Die Konfiguration der AdditionalServerEntries kann über den TwinCAT OPC UA Configurator in der Registerkarte **Server Settings** erfolgen, z.B.:



Beim Hinzufügen eines neuen Eintrags, werden bestimmte Felder schon mit einer Vorlage ausgefüllt. Bitte passen Sie diese Einträge entsprechend Ihrer Konfiguration an.



Die konkreten Werte für ProductUri, ApplicationName, ApplicationUri und ApplicationType können Sie dem OnlinePanel des TwinCAT OPC UA Configurator entnehmen:



Die DiscoveryURL entspricht der ServerURL, welche Sie üblicherweise für den Verbindungsaufbau mit dem jeweiligen Server verwenden.

4.20 Logging

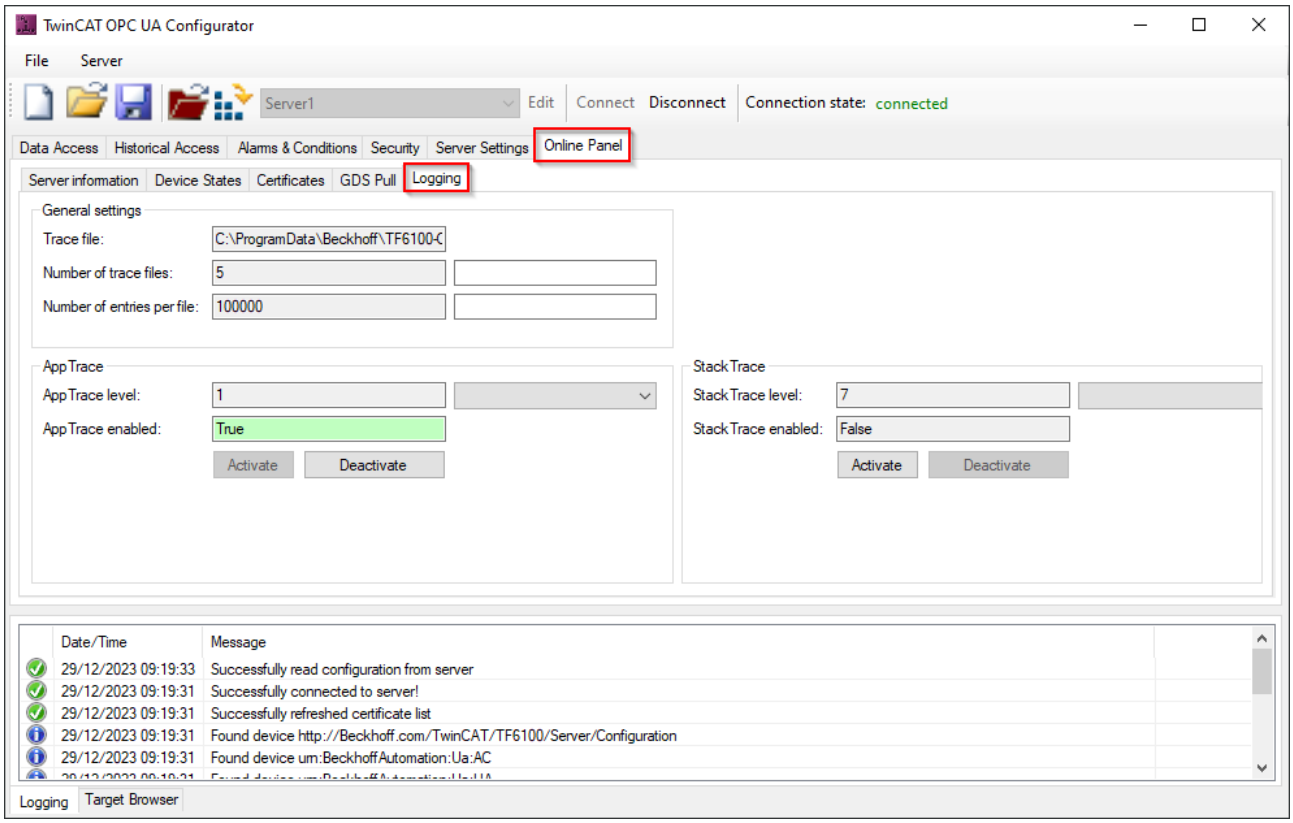
Sie können für eine erweiterte Diagnose eine Protokolldatei im Server aktivieren, in welcher dann auf Basis von unterschiedlichen Protokollleveln verschiedene Informationen mitgeschrieben werden.

Einfluss des Loggings auf das Betriebsverhalten

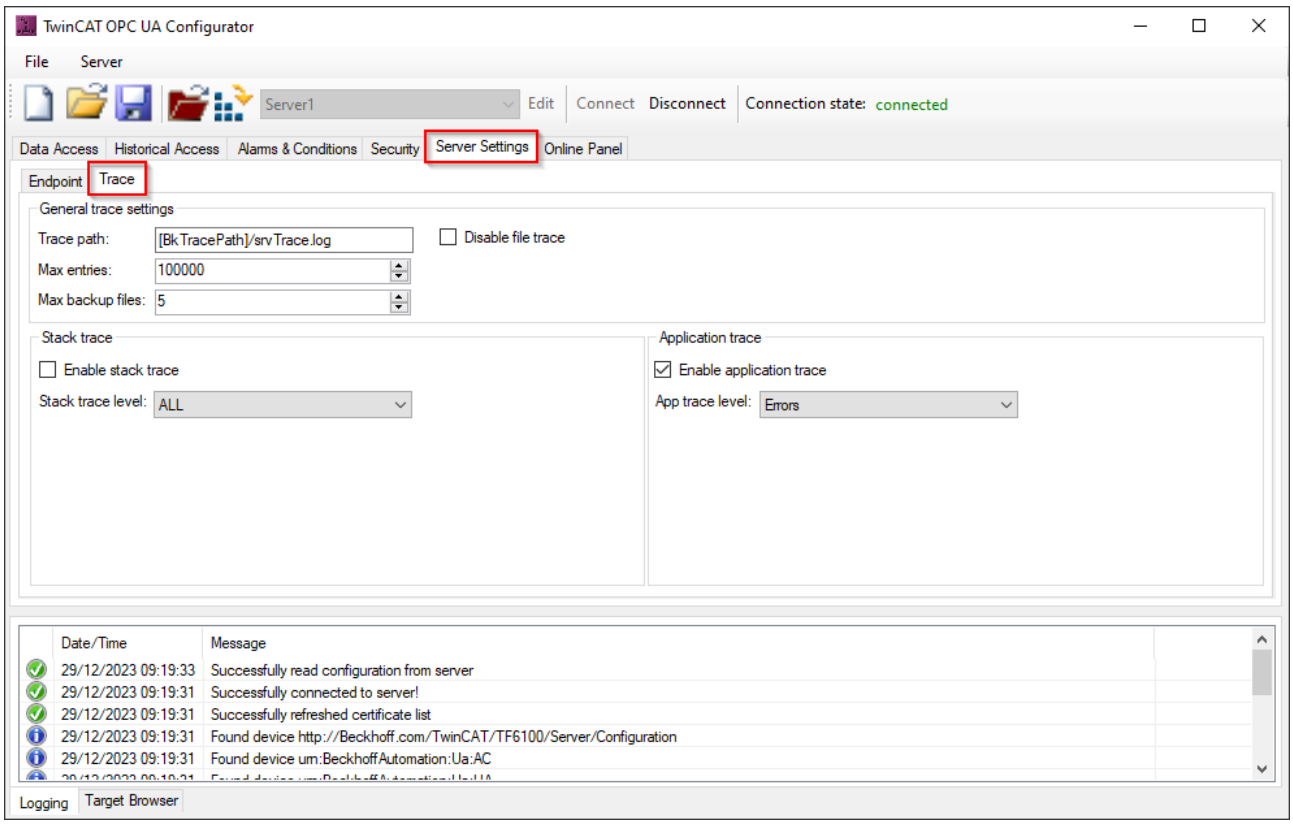
Bitte beachten Sie, dass die Aktivierung der Protokolldatei negative Einflüsse auf die Geschwindigkeit und das Betriebsverhalten des TwinCAT OPC UA Servers haben kann.

Der Standardpfad für die erstellten Protokolldateien ist abhängig vom verwendeten Betriebssystem und wird in dem Kapitel Applikationsverzeichnisse [▶ 47] näher beschrieben.

Das Aktivieren/Deaktivieren des Loggings erfolgt im Regelfall über den TwinCAT OPC UA Configurator. Hierbei können Sie das Logging sowohl online auf dem Server aktivieren/deaktivieren als auch offline über die entsprechende Konfigurationsdatei, wodurch das Logging erst nach einem Serverneustart aktiviert ist. Der folgende Screenshot zeigt die Konfigurationsoberfläche für das Online-Logging:



Der folgende Screenshot zeigt die Konfigurationsoberfläche für das Offline-Logging:



4.21 System Tray

Der TwinCAT OPC UA Server beinhaltet eine Applikation, welche als Icon im Windows System Tray aufrufbar ist. Diese Applikation ermöglicht das Antriggern eines Neustarts des Servers. Über das Kontexmenü ist die entsprechende Funktion aufrufbar.

5 SPS API

5.1 Tc2_OpcUa

5.1.1 Datentypen

5.1.1.1 ST_OpcUAServerInfo

ST_OpcUAServerInfo beinhaltet Sessioninformationen eines TwinCAT OPC UA Servers.

Syntax

```
TYPE ST_OpcUAServerInfo :
STRUCT
    nReserved : UDINT;
    nCummulatedSessionCount       : UDINT;
    nCurrentSessionCount          : UDINT;
    nRejectedSessionCount         : UDINT;
    nSecurityRejectedSessionCount : UDINT;
    nSessionTimeoutCount          : UDINT;
    nCurrentSubscriptionCount     : UDINT;
    nRejectedRequestCount         : UDINT;
    nSecurityRejectedRequestCount : UDINT;
END_STRUCT
END_TYPE
```

Parameter

Name	Typ	Beschreibung
nReserved	UDINT	Platzhalter.
nCummulatedSessionCount	UDINT	Gesamtanzahl der Client-Sessions seit Start des Servers.
nCurrentSessionCount	UDINT	Gesamtzahl der aktuellen Client-Sessions.
nRejectedSessionCount	UDINT	Gesamtzahl der vom Server abgelehnten Sessions.
nSecurityRejectedSessionCount	UDINT	Gesamtzahl der aus Security-Gründen vom Server abgelehnten Sessions (Beispiel: Falsche Kombination aus Benutzername und Passwort).
nSessionTimeoutCount	UDINT	Gesamtzahl der Sessions, die einen Timeout hatten.
nCurrentSubscriptionCount	UDINT	Gesamtzahl der aktuellen Subscriptions im Server.
nRejectedRequestCount	UDINT	Gesamtzahl der fehlgeschlagenen Requests.
nSecurityRejectedRequestCount	UDINT	Gesamtzahl der aus Security-Gründen fehlgeschlagenen Requests.

5.1.1.2 E_OpcUAServerOption

E_OpcUAServerOption legt fest welches Kommando an den TwinCAT OPC UA Server geschickt werden soll.

Syntax

```
TYPE E_OpcUAServerOption
(
    eOPCUAServerOption_None,
    eOPCUAServerOption_Restart,
    eOPCUAServerOption_Shutdown,
    eOPCUAServerOption_RefreshCfg,
    eOPCUAServerOption_ServerInfo
);
END_TYPE
```

Parameter

Name	Beschreibung
eOPCUAServerOption_None	Ausgangszustand der Aufzählung.
eOPCUAServerOption_Restart	Diese Option triggert einen Neustart des OPC UA-Interfaces des Servers.
eOPCUAServerOption_Shutdown	Diese Option triggert das Herunterfahren des OPC UA-Interfaces des Servers. Da die voranstehende Restart-Option über OPC UA funktioniert, ist diese nach Nutzen dieser Option bis zu einem kompletten Server-Neustart nicht mehr verfügbar.
eOPCUAServerOption_RefreshCfg	Diese Option hat aktuell keine Funktion.
eOPCUAServerOption_ServerInfo	Diese Option fragt die in ST_OpcUAServerInfo [▶ 144] enthaltenen Server-Informationen ab.

5.1.1.3 E_OpcUAServerStatus

E_OpcUAServerStatus repräsentiert den Laufzeitstatus eines TwinCAT OPC UA Servers.

Syntax

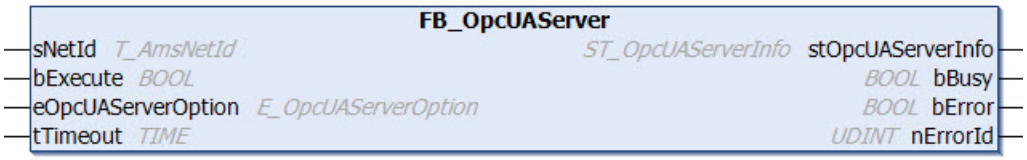
```
TYPE E_OpcUAServerStatus
(
    eOPCUAServerStatus_None,
    eOPCUAServerStatus_Alive,
    eOPCUAServerStatus_NotResponding
);
END_TYPE
```

Parameter

Name	Beschreibung
eOPCUAServerStatus_None	Ausgangszustand der Aufzählung.
eOPCUAServerStatus_Alive	Das ADS-Interface des TwinCAT OPC UA Servers ist erreichbar.
eOPCUAServerStatus_NotResponding	Das ADS-Interface des TwinCAT OPC UA Servers ist nicht erreichbar.

5.1.2 Funktionsbausteine

5.1.2.1 FB_OpcUAServer



Der Funktionsbaustein ermöglicht das Auslesen von Statusinformationen und Neustarten eines TwinCAT OPC UA Servers.

Syntax

Definition:

```
FUNCTION_BLOCK FB_OpcUAServer
VAR_INPUT
    sNetId             : T_AmsNetId;
    bExecute           : BOOL;
    eOpcUAServerOption : E_OpcUAServerOption;
```

```
    tTimeout            : TIME;
END_VAR
VAR_OUTPUT
    stOpcUAServerInfo : ST_OpcUAServerInfo;
    bBusy               : BOOL;
    bError              : BOOL;
    nErrorId            : UDINT;
END_VAR
```

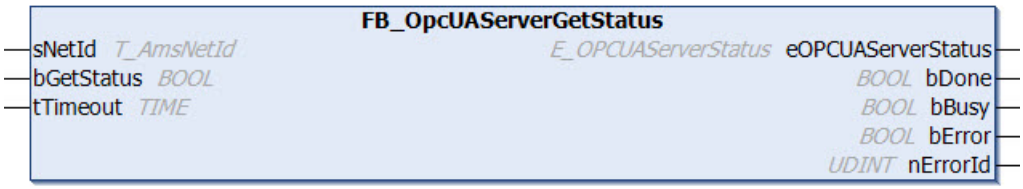
Eingänge

Name	Typ	Beschreibung
sNetId	T_AmsNetId	AmsNetId des Systems auf dem der TwinCAT OPC UA Server läuft.
bExecute	BOOL	Eine steigende Flanke startet die Abarbeitung des Funktionsbausteins.
eOpcUAServerOption	E_OpcUAServerOption [▶ 144]	Gibt die auszuführende Operation an.
tTimeout	TIME	ADS Timeout

Ausgänge

Name	Typ	Beschreibung
stOpcUAServerInfo	ST_OpcUAServerInfo [▶ 144]	Enthält Statusinformationen vom Server wenn beim Eingang eOpcUAServerOption "ServerInfo" ausgewählt wurde.
bBusy	BOOL	TRUE, solange die Abarbeitung des Funktionsbausteins nicht beendet ist.
bError	BOOL	Wird TRUE, sobald eine Fehlersituation eintritt.
nErrorId	UDINT	Enthält bei Auftreten eines Fehlers (bError) den Fehlercode.

5.1.2.2 FB_OpcUAServerGetStatus



Der Funktionsbaustein ermöglicht das Auslesen des aktuellen Status (Alive, NotResponding) eines TwinCAT OPC UA Servers. An dieser Stelle ist anzumerken, dass sich dieser Funktionsbaustein mit dem ADS-Interface des OPC UA Servers beschäftigt. Wenn der OPC UA-Server neugestartet oder heruntergefahren wird, bleibt das ADS-Interface des Servers erreichbar. Das ADS-Interface lässt sich nur durch Beenden des Server-Prozesses beenden.

Syntax

Definition:

```
FUNCTION_BLOCK FB_OpcUAServerGetStatus
VAR_INPUT
    sNetId              : T_AmsNetId;
    bGetStatus          : BOOL;
    tTimeout            : TIME;
END_VAR
VAR_OUTPUT
    eOPCUAServerStatus : E_OPCUAServerStatus;
    bDone               : BOOL;
    bBusy               : BOOL;
```

```
    bError              : BOOL;
    nErrorId            : UDINT;
END_VAR
```

Eingänge

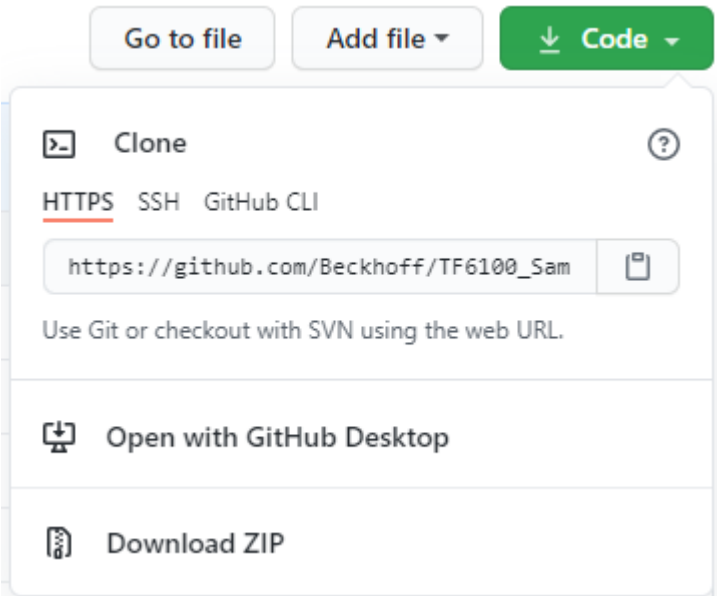
Name	Typ	Beschreibung
sNetId	T_AmsNetId	AmsNetId des Systems auf dem der TwinCAT OPC UA Server läuft.
bGetStatus	BOOL	Eine steigende Flanke startet die Abarbeitung des Funktionsbausteins.
tTimeout	TIME	ADS Timeout

Ausgänge

Name	Typ	Beschreibung
eOPCUAServerStatus	E_OpcUAServerStatus [▶ 145]	Enthält Statusinformationen des Servers.
bDone	BOOL	TRUE, wenn die Abarbeitung des Funktionsbausteins beendet ist.
bBusy	BOOL	TRUE, solange die Abarbeitung des Funktionsbausteins nicht beendet ist.
bError	BOOL	Wird TRUE, sobald eine Fehlersituation eintritt.
nErrorId	UDINT	Enthält bei Auftreten eines Fehlers (bError) den Fehlercode.

6 Beispiele

Beispielcode und -konfigurationen für dieses Produkt können über das entsprechende Repository auf GitHub bezogen werden: https://github.com/Beckhoff/TF6100_Samples. Sie haben dort die Möglichkeit, das Repository zu clonen oder ein ZIP-File mit dem Sample herunterzuladen.

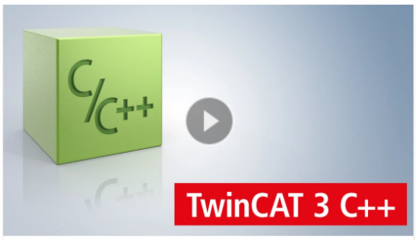


Es existieren folgende Samples:

Name	TwinCAT-Version	Beschreibung
TF6100_OpcUa_Client_Sample	TwinCAT 3	Dieses Sample beinhaltet Beispielcode für verschiedene Funktionen des TwinCAT OPC UA Clients (PLCOpen-Funktionsbausteine). Dazu gehören Browse, Connect, HistoryUpdate, MethodCall, Read und Write. Es ist zusätzlich das Server-Sample für den Zugriff enthalten.
TF6100_OpcUa_Server_Sample	TwinCAT 3	Dieses Sample beinhaltet eine SPS mit umfangreicher Bereitstellung von SPS-Daten für den TwinCAT OPC UA Server (OPC UA Data Access).
TS6100_OpcUa_Client_Sample	TwinCAT 2	Dieses Sample beinhaltet Beispielcode für verschiedene Funktionen des TwinCAT OPC UA Clients (PLCOpen-Funktionsbausteine). Dazu gehören MethodCall, Read und Write.
TS6100_OpcUa_Server_Sample	TwinCAT 2	Dieses Sample demonstriert die Verwendung von SPS Kommentaren zur Freigabe von Variablen über den TwinCAT OPC UA Server.

7 Tutorials

Video-Tutorials zu diesem Produkt finden Sie auf unserer Website unter https://www.beckhoff.com/tutorials. Die Video-Tutorials bieten einen schnellen Einstieg in das Produkt und die einzelnen Produktfacetten.



14.03.2024 | Multimedia
Tutorial: TwinCAT 3 C++ overview
Get an overview of the C++ integration into TwinCAT.
Learn more →

12.02.2024 | Multimedia
Tutorial: Chart synchronization in TwinCAT Scope
Learn how to dock and synchronize charts in TwinCAT Scope.
Learn more →

12.02.2024 | Multimedia
Tutorial: Getting started with TF6100 TwinCAT 3 OPC UA Server
Learn how to configure the TwinCAT OPC UA Server.
Learn more →

8 Anhang

8.1 Attribute und Kommentare

Die folgende Tabelle bietet einen Überblick über alle im Server konfigurierbaren Pragmas und Kommentare. Diese lassen sich in den verschiedenen Echtzeitumgebungen von TwinCAT zur Freischaltung von unterschiedlichen Funktionalitäten definieren. Eine detaillierte Beschreibung zur Verwendung von Attributen und Kommentaren erhalten Sie im Kapitel Freigabe von Symbolen [▶ 53].

Verwendung der Tags in der SPS

Bitte beachten Sie bei Verwendung der Pragmas in der SPS, dass sie den Key und Value in Hochkommata setzen. Ein Beispiel finden Sie im Kapitel Freigabe von Symbolen\SPS [▶ 55].

TwinCAT 3

Key	Value	Bedeutung
OPC.UA.DA	0	Sperrt ein Symbol explizit für OPC UA.
OPC.UA.DA	1	Gibt ein Symbol für OPC UA frei.
OPC.UA.DA	2	Gibt ein Symbol für OPC UA frei. Im Fall einer Struktur und dem StructuredDataType werden Membervariablen hierbei nicht als separate Nodes in den Adressraum des Servers geladen.
OPC.UA.DA.Access	1	Setzt eine Node Read-Only [▶ 84].
OPC.UA.DA.Access	2	Setzt eine Node Write-Only.
OPC.UA.DA.Access	3	Ermöglicht Lese-/Schreibzugriff auf eine Node (Default wenn nicht gesetzt).
OPC.UA.DA.Alias	<string>	Definiert einen anderen Namen (Alias [▶ 85]) für eine Node.
OPC.UA.DA.Description	<string>	Definiert den Inhalt des OPC UA Attributs „Description [▶ 83]“.
OPC.UA.DA.StructuredType	0	Deaktiviert den StructuredDataType für eine Struktur [▶ 75].
OPC.UA.DA.StructuredType	1	Aktiviert den StructuredDataType für eine Struktur [▶ 75].
OPC.UA.DA.Status	quality	StatusCode [▶ 79] eines Symbols im OPC UA Namensraum manuell festlegen.

TwinCAT 2

SPS Kommentar	Bedeutung
(*~ (OPC:0:not available) *)	Sperrt eine Variable für OPC UA, woraufhin diese im UA-Namensraum nicht mehr sichtbar ist.
(*~ (OPC:1:available) *)	Aktiviert eine Variable für OPC UA, woraufhin diese im UA-Namensraum sichtbar wird. Dieses Tag muss immer gesetzt sein, wenn eine Variable für UA verwendet werden soll.
(*~ (OPC_PROP[0005]:1:Schreibgeschützt) *)	Setzt den Schreibschutz für eine Variable. Muss zusammen mit (*~ (OPC:1: available) *) verwendet werden.
(*~ (OPC_UA_PROP[5100] : x: Alias name) *)	Bestimmt x als Knotennamen im UA-Namensraum, sogenanntes Alias Mapping.
(*~ (OPC_UA_PROP[5000]:x:Storage media) *)	Aktiviert eine Variable für „Historical Access“. Muss zusammen mit (*~ (OPC:1: available) *) verwendet werden. x definiert das Speichermedium für die Speicherung der Datenwerte: 1 = Speicher 2 = Datei 3 = SQL Compact Database 4 = SQL Server Database
(*~ (OPC_UA_PROP[5000][1]:x:SamplingRate) *)	Legt die Abtastrate fest, mit der die Variablenwerte zu speichern sind, in Abhängigkeit des Parameters „x“ in [ms]
(*~ (OPC_UA_PROP[5000][2]:x:Buffer) *)	Definiert die maximale Anzahl Werte, die im Datenspeicher bleiben, in Abhängigkeit des Parameters „x“.

8.2 32-bit und 64-bit Prozess

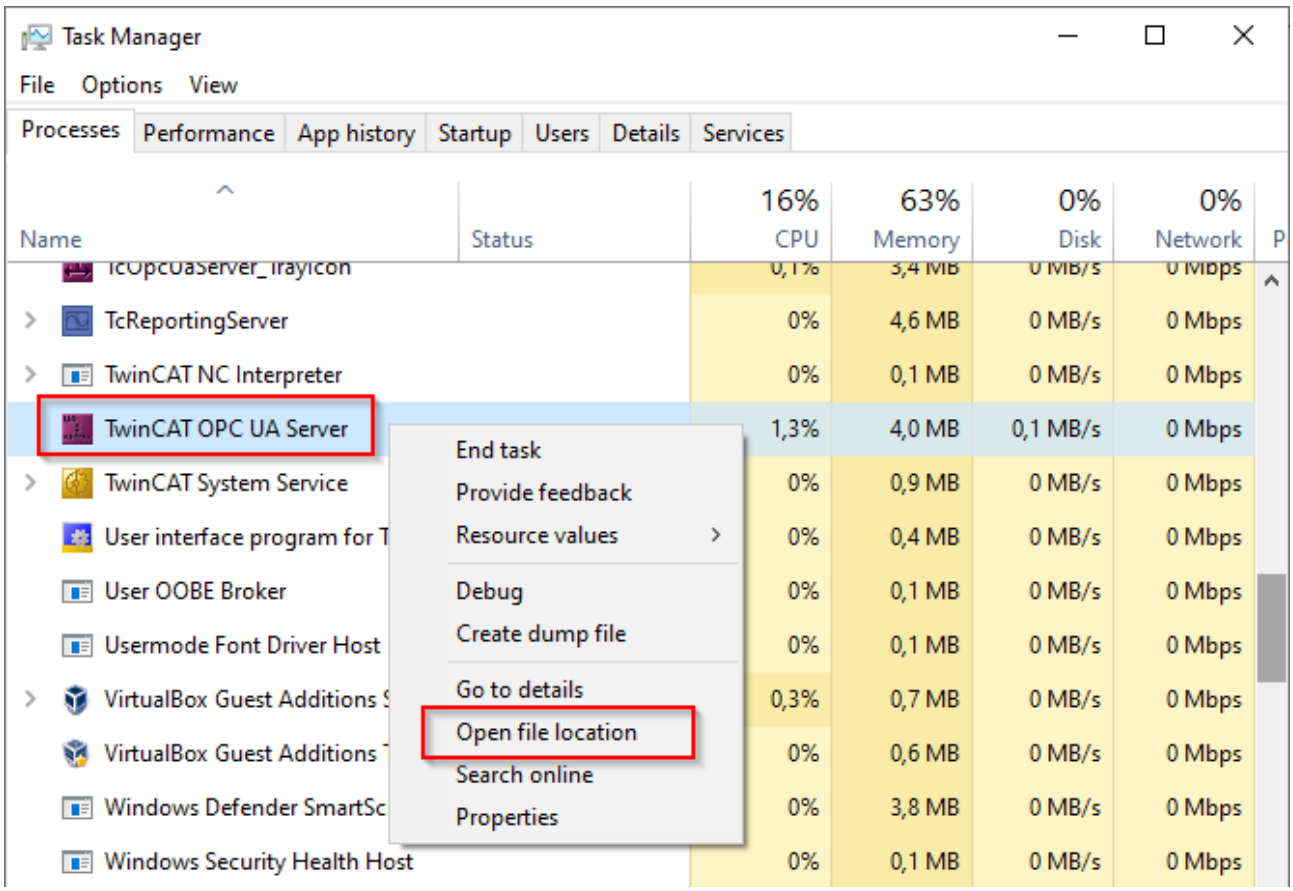
Der TwinCAT OPC UA Server ist als 32-bit und 64-bit Prozess erhältlich. Die entsprechenden Dateien werden automatisch über das Setup bzw. Package installiert.

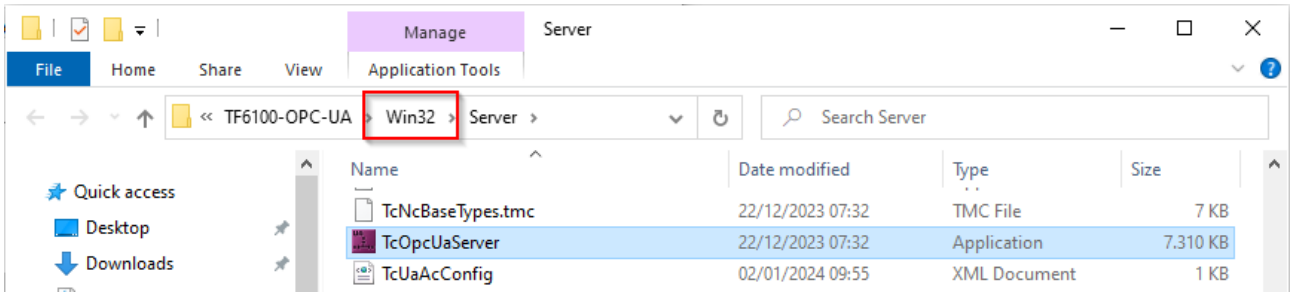
Welche Variante ist auf Ihrem System aktiv?

✓ Durch einen kleinen Trick können Sie im Windows Task Manager leicht erkennen, welche Variante gerade auf Ihrem System aktiv ist.

1. Öffnen Sie den Task Manager.
2. Navigieren Sie zum Prozess „TwinCAT OPC UA Server“.
3. Öffnen Sie über das Kontextmenü den Speicherort der Datei.

⇨ In der Adresszeile des File Explorers können Sie nun sehen, welche Variante der TcOpcUaServer.exe gerade auf Ihrem System registriert und somit aktiv ist:





Registrieren/Deregistrieren

Zum Registrieren bzw. Deregistrieren einer anderen Variante des TwinCAT OPC UA Servers, führen Sie bitte die folgenden Schritte durch.

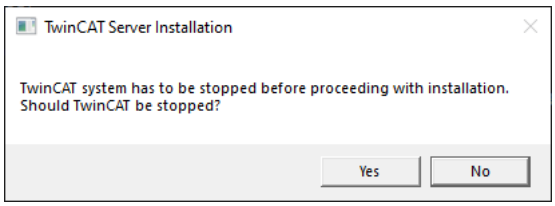
Beachten Sie vor dem Wechseln der Variante, dass Sie zunächst den laufenden Prozess beenden müssen.

Von 32-bit auf den 64-bit Prozess wechseln:

- Beenden Sie den laufenden Prozess der Datei TcOpcUaServer.exe im Task Manager.
- Öffnen Sie die Windows Eingabeaufforderung als Administrator und wechseln Sie zu dem Installationsverzeichnis [▶ 47] der 32-bit Variante.
- Geben Sie den folgenden Befehl ein, um den Prozess zu deregistrieren:

```
TcOpcUaServer.exe /UnRegTcServer2
```

Eine Dialogbox weist Sie darauf hin, dass der TwinCAT-System-Service für eine Deregistrierung neu gestartet werden muss. Bestätigen Sie diese Dialogbox mit **Yes**.



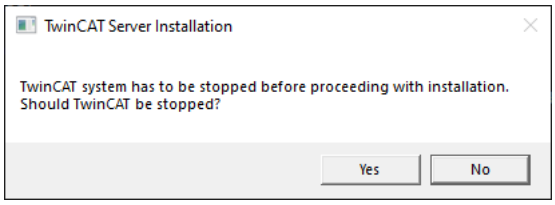
Als Folge dieses Befehls wird der TwinCAT OPC UA Server nun nicht mehr automatisch mit TwinCAT gestartet.

Im nächsten Schritt erfahren Sie, wie Sie die 64-bit Variante des Servers am TwinCAT-System-Service registrieren.

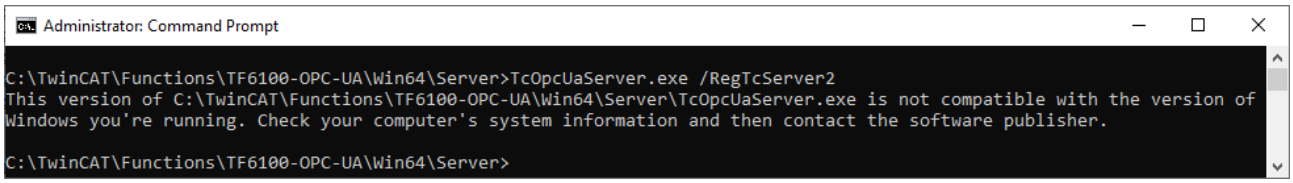
- Wechseln Sie zu dem Server-Installationsverzeichnis [▶ 47] der 64-bit Variante und geben Sie den folgenden Befehl ein, um den Prozess zu registrieren:

```
TcOpcUaServer.exe /RegTcServer2
```

Eine Dialogbox weist Sie abermals darauf hin, dass der TwinCAT-System-Service für eine Deregistrierung neu gestartet werden muss. Bestätigen Sie diese Dialogbox mit „Yes“.



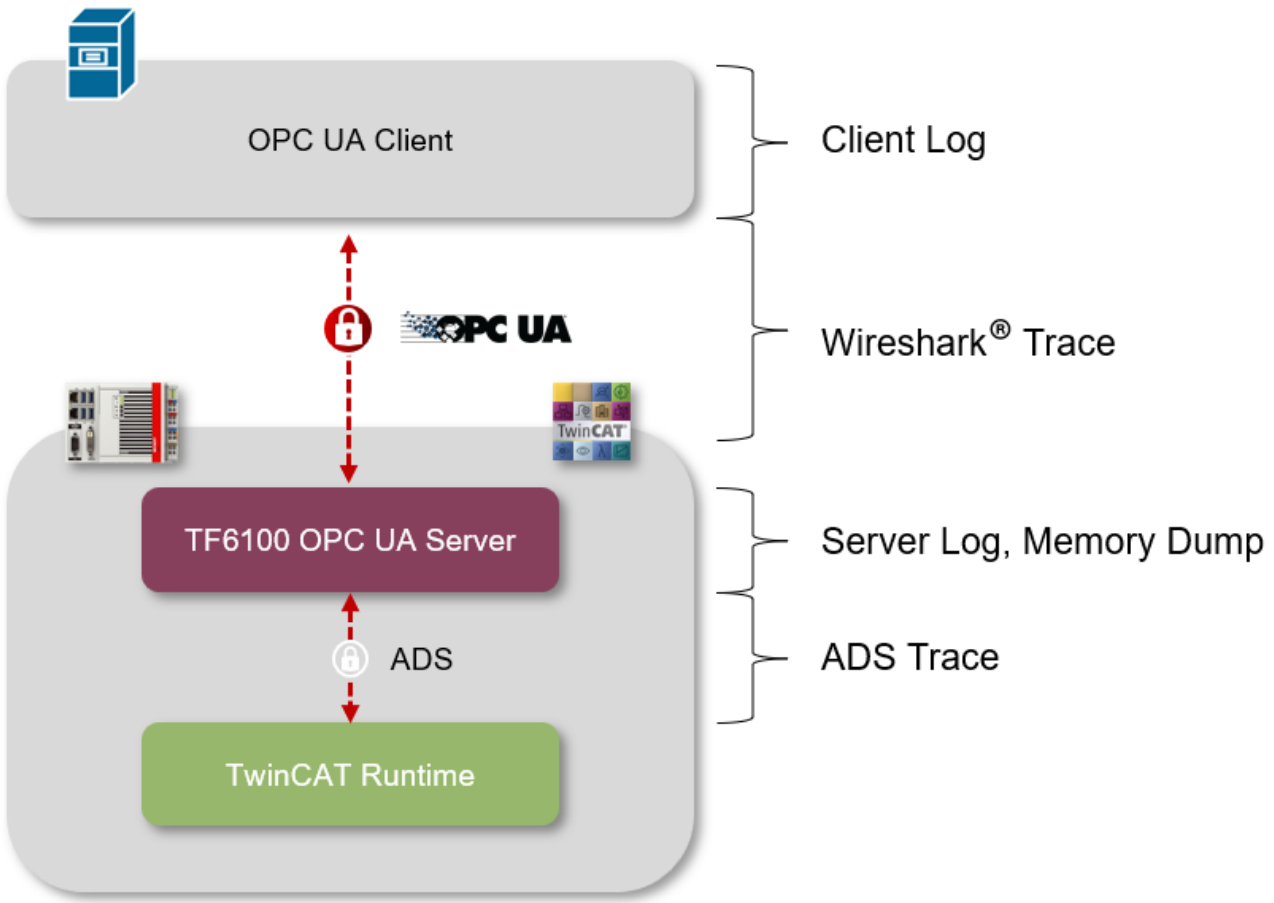
Als Folge dieses Befehls wird der TwinCAT OPC UA Server am TwinCAT-System-Service registriert und startet automatisch zusammen mit TwinCAT. Sollten Sie bei der Eingabe dieses Befehls die Fehlermeldung bekommen, stellen Sie sicher, dass Sie nicht versehentlich versuchen den 64-bit Prozess auf einem 32-bit System zu registrieren.



Umgekehrt kann der 32-bit Prozess natürlich auf einem 64-bit System registriert werden.

8.3 Fehlerdiagnose

Im Fall eines unerwünschten Betriebsverhaltens ist möglicherweise eine erweiterte Diagnose notwendig. Je nach Sachlage beinhaltet dies die folgenden Maßnahmen: ADS Trace, Wireshark® Trace, Protokolldatei, Memory Dump. Gegebenenfalls ist auch auf Clientseite eine Protokollfunktion vorhanden und sinnvoll. Die Sinnhaftigkeit zur Durchführung der jeweiligen Maßnahme hängt stark vom Erscheinungsbild des Verhaltens ab. Führen Sie sich hierzu noch einmal die Softwarearchitektur des TwinCAT OPC UA Servers vor Augen und gleichen Sie diese mit dem Betriebsverhalten ab. Das folgende Schaubild verdeutlicht, an welchen Stellen die einzelnen Maßnahmen (genauer beschrieben in der Tabelle) hilfreich sein könnten.



Maßnahme	Anwendungsfall	Link
Client Log	Hilfreich zum Aufzeichnen von erweiterten Protokollfunktionen auf Clientseite.	Bitte wenden Sie sich an den Hersteller des Clients, um weitere Informationen zu dessen Logging-Funktionen zu erhalten.
Wireshark Trace	Sinnvoll zur Untersuchung der Kommunikation zwischen Client und Server.	https://www.wireshark.org/
Server Log	Hilfreich zum Aufzeichnen von erweiterten Protokollfunktionen auf Serverseite.	Siehe Kapitel Logging [▶ 142] in dieser Dokumentation.
Memory Dump	Die Erstellung eines Memory Dumps kann bei einer unerwarteten Terminierung des TwinCAT OPC UA Servers sinnvoll sein. Die hierfür notwendigen Tools sind abhängig vom Betriebssystem.	Microsoft Debug Diagnostics
ADS Trace	Sinnvoll zur Untersuchung der Kommunikation zwischen Server und SPS.	Beckhoff Download Finder (TF6010 TC3 ADS Monitor)

Support

Die Supportabteilung von Beckhoff [▶ 161] unterstützt Sie gern bei der Durchführung der geeigneten Maßnahmen.

Die folgende Tabelle gibt einen Überblick über möglicherweise unerwartet auftretendes Betriebsverhalten und Maßnahmen zu dessen Lösung.

Verhalten	Maßnahme(n)
Ein OPC UA Client sieht den PLC Namespace nicht.	TF6100 Setup Version 3.x und kleiner: Dieser Status ist ein Hinweis auf eine fehlende Lizenz. Überprüfen Sie, ob Sie eine gültige TF6100 Lizenz aktiviert haben.
Ein OPC UA Client bekommt bei Auslesen von Nodes den StatusCode 0x810e0000.	TF6100 Setup Version 4.x: Dieser Status ist ein Hinweis auf eine fehlende Lizenz. Überprüfen Sie, ob Sie eine gültige TF6100 Lizenz aktiviert haben.
Die über Kommentare/Attribute freigegebenen Variablen werden nicht im OPC UA Server angezeigt.	Überprüfen Sie, ob die Symboldatei ordnungsgemäß auf die Steuerung übertragen wurde (z. B. Auswahlkästchen im SPS-Projekt), im Bootverzeichnis vorhanden ist und der Pfad zur Symboldatei in der Konfigurationsdatei des Servers auf die richtige Symboldatei verweist. Sie können auch über die DeviceState Node im jeweiligen Namespace eventuell aufgetretene ErrorMessages überprüfen. Hier wird z. B. auch eingetragen, wenn die Symboldatei nicht gefunden wurde. Überprüfen Sie auch die ordnungsgemäße Schreibweise der Kommentare/Attribute.
Die vom OPC UA Client geforderte Samplingrate/PublishingInterval werden vom Server nicht eingehalten.	OPC UA Client/Server ist kein Echtzeitprotokoll, d. h. dass es keine Garantie gibt, dass der Server eine vom Client geforderte Samplingrate oder PublishingInterval auch immer zu 100 % einhält. Die zur Verfügung stehenden Samplingraten und PublishingIntervalle können in der Konfigurationsdatei des Servers eingesehen und bei Bedarf modifiziert werden (<AvailableSamplingRates> und <MinPublishingInterval>).
Ein OPC UA Client kann sich nicht mit dem Server verbinden, obwohl der Server im Windows Task Manager angezeigt wird. Es erscheint die Fehlermeldung „Host unreachable“ (o.Ä.).	Überprüfen Sie, ob gegebenenfalls Firewall-Einstellungen eine Kommunikation mit dem Server verhindern. Der Serverport muss für eine eingehende TCP-Kommunikation geöffnet sein, damit sich ein Client verbinden kann.
Ein OPC UA Client sieht zwar die Endpunkte des Servers, eine Verbindung mit diesen schlägt jedoch mit der Fehlermeldung „Host unreachable“ fehl.	Überprüfen Sie, ob die Namensauflösung in Ihrem Netzwerk ordnungsgemäß funktioniert und der Server unter seinem Hostnamen erreichbar ist. Auch wenn sich der OPC UA Client augenscheinlich mit der IP-Adresse des Servers verbindet (z. B. opc.tcp://192.168.0.1:4840), um auf die Endpunkte des Servers zuzugreifen, so returniert der Server in seinen Endpunkten dennoch immer den eigenen Hostnamen. Wenn sich nun der Client direkt mit einem der Endpunkte verbindet, verwendet er wieder den Hostnamen des Servers. Im Fall einer nicht funktionierenden Namensauflösung schlägt dann die Verbindung fehl.
Ein OPC UA Client sieht zwar die Endpunkte des Servers, eine Verbindung mit einem sicheren Endpunkt schlägt jedoch fehl. Es erscheint die Fehlermeldung „BadSecurityChecksFailed“	Überprüfen Sie, ob der Server dem Client-Zertifikat vertraut. Die notwendigen Konfigurationsschritte können Sie im Kapitel Zertifikatsaustausch nachlesen. Hier ist zusätzlich darauf zu achten, dass ein Signaturhashalgorithmus der SHA 2-Gruppe (SHA256, SHA364, SHA512) zur Signierung des Client-Zertifikats verwendet wird. Werden veraltete Algorithmen wie SHA1 verwendet, lässt der TwinCAT OPC UA Server keine Verbindung zu.
Bei Verwendung eines SQL Servers zur Speicherung von Historical Access Informationen, werden die Werte nicht zur SQL-Datenbank hinzugefügt.	Überprüfen Sie die Zugangsdaten zum SQL Server und dass der SQL Server auch im Netzwerk erreichbar ist. Stellen Sie auch sicher, dass Sie ein „Big Windows“-Betriebssystem auf dem TwinCAT OPC UA Server verwenden, da SQL Server nicht unter Windows CE für Historical Access verwendet werden können (SQL Compact hingegen schon).

Verhalten	Maßnahme(n)
Beim Auslesen von Nodes erhält ein OPC UA Client die Fehlermeldung „BadDeviceFailure“.	Dies ist ein Hinweis darauf, dass das zugehörige ADS-Gerät nicht erreichbar ist, z. B. wenn kein SPS-Programm gestartet ist. Überprüfen Sie die Konnektivität mit dem ADS-Gerät und stellen Sie sicher, dass die entsprechende Laufzeit aktiv ist.
Beim Auslesen von Nodes erhält ein OPC UA Client die Fehlermeldung „BadLicenseExpired“.	Dies ist ein Hinweis darauf, dass keine TF6100 Lizenz auf dem System aktiv bzw. dass diese abgelaufen ist. Bitte stellen Sie sicher, dass Sie eine TF6100 Lizenz auf dem Gerät aktiviert haben, auf dem der Server installiert wurde.
Arrays werden nicht voll aufgelöst im Namespace dargestellt.	Standardmäßig werden Arrays von einfachen Datentypen nicht „aufgeklappt“ im Namensraum dargestellt. Einzelne Array-Indizes sind dennoch über die sogenannte IndexRange Funktion von OPC UA adressierbar. Ein OPC UA Client sollte diese Funktion entsprechend unterstützen. Ist dies nicht der Fall, so kann über einen Optionsschalter in der Konfigurationsdatei des Servers bewirkt werden, dass ein Array „aufgeklappt“ dargestellt und somit jedes einzelne Arrayelement als separate Node adressierbar ist. Dieser Optionsschalter ist im Kapitel Arrays [▶ 60] beschrieben.
Im Errorlog taucht wiederholt die folgende Meldung auf: "CBkUaPlcSamplingEngine_Poll : !!CRITICAL!! Sampling exceeded n times in a row, leaving no time left to rest."	Diese Meldung ist ein Hinweis darauf, dass der Server die vom Client zu einer Subscription hinzugefügten Variablen nicht in der angeforderten Samplingrate verarbeiten kann. Der Grund hierfür ist typischerweise eine zu hohe Datenmenge, z.B. beim Sampling von großen Datenstrukturen (StructuredTypes [▶ 75]). Bitte beachten Sie, dass diese Meldung nicht zwangsläufig ein Problem darstellen muss. Bei vielen OPC UA Client Applikationen werden die Einstellungen zur Samplingrate generalisiert und für alle Variablen getroffen, auch wenn diese gegebenenfalls langsamer ge-sampled werden könnten. In jedem Fall sollte ein erster Schritt zur Optimierung jedoch immer eine Überprüfung und ggf. Reduktion der Samplingrate beinhalten. Auch sollte überprüft werden, ob im Falle einer Verwendung von großen StructuredTypes diese in der Datenmenge reduziert werden können. Auch das Upgrade auf eine schnellere CPU ist hilfreich. Allerdings kann aus der Erfahrung heraus gesagt werden, dass sich schon durch eine Reduzierung der Samplingrate und Verringerung der Datenmenge erste gute Erfolge erzielen lassen.
Der Aufruf einer Job-Methode [▶ 107] schlägt fehl mit dem Statuscode „BadResourceUnavailable“	Wenn eine Job-Methode mit diesem Statuscode returniert, dann ist dies ein Indiz dafür, dass die Signatur des Funktionsbausteins, welcher die Job-Methode repräsentiert, nicht korrekt gesetzt wurde. Überprüfen Sie als erstes das Vorhandensein der Start() Methode. Weiterer Hinweis: in einem ADS-Trace würden Sie in diesem Fall sehen, dass beim zugehörigen Aufruf (AdsReadWriteRequest) der Methode der Result 0x710 („Symbol not found“) zurückgegeben wird.

8.4 ADS Return Codes

Gruppierung der Fehlercodes:
Globale Fehlercodes: 0x0000 [▶ 157]... (0x9811_0000 ...)
Router Fehlercodes: 0x0500 [▶ 157]... (0x9811_0500 ...)
Allgemeine ADS Fehler: 0x0700 [▶ 158]... (0x9811_0700 ...)
RTime Fehlercodes: 0x1000 [▶ 160]... (0x9811_1000 ...)

Globale Fehlercodes

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

Router Fehlercodes

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

Allgemeine ADS Fehlercodes

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

Hex	Dec	HRESULT	Name	Beschreibung
0x732	1842	0x98110732	ADSERR_DEVICE_LICENSEDEMODENIED	Demo-Lizenz untersagt.
0x733	1843	0x98110733	ADSERR_DEVICE_INVALIDFNCID	Funktions-ID ungültig.
0x734	1844	0x98110734	ADSERR_DEVICE_OUTOFRANGE	Außerhalb des gültigen Bereiches.
0x735	1845	0x98110735	ADSERR_DEVICE_INVALIDALIGNMENT	Ungültiges Alignment.
0x736	1846	0x98110736	ADSERR_DEVICE_LICENSEPLATFORM	Ungültiger Plattform Level.
0x737	1847	0x98110737	ADSERR_DEVICE_FORWARD_PL	Kontext – Weiterleitung zum Passiv-Level.
0x738	1848	0x98110738	ADSERR_DEVICE_FORWARD_DL	Kontext – Weiterleitung zum Dispatch-Level.
0x739	1849	0x98110739	ADSERR_DEVICE_FORWARD_RT	Kontext – Weiterleitung zur Echtzeit.
0x740	1856	0x98110740	ADSERR_CLIENT_ERROR	Clientfehler.
0x741	1857	0x98110741	ADSERR_CLIENT_INVALIDPARM	Dienst enthält einen ungültigen Parameter.
0x742	1858	0x98110742	ADSERR_CLIENT_LISTEMPTY	Polling-Liste ist leer.
0x743	1859	0x98110743	ADSERR_CLIENT_VARUSED	Var-Verbindung bereits im Einsatz.
0x744	1860	0x98110744	ADSERR_CLIENT_DUPLINVOKEID	Die aufgerufene ID ist bereits in Benutzung.
0x745	1861	0x98110745	ADSERR_CLIENT_SYNCTIMEOUT	Timeout ist aufgetreten – Die Gegenstelle antwortet nicht im vorgegebenen ADS Timeout. Die Routeneinstellung der Gegenstelle kann falsch konfiguriert sein.
0x746	1862	0x98110746	ADSERR_CLIENT_W32ERROR	Fehler im Win32 Subsystem.
0x747	1863	0x98110747	ADSERR_CLIENT_TIMEOUTINVALID	Ungültiger Client Timeout-Wert.
0x748	1864	0x98110748	ADSERR_CLIENT_PORTNOTOPEN	Port nicht geöffnet.
0x749	1865	0x98110749	ADSERR_CLIENT_NOAMSADDR	Keine AMS Adresse.
0x750	1872	0x98110750	ADSERR_CLIENT_SYNCINTERNAL	Interner Fehler in Ads-Sync.
0x751	1873	0x98110751	ADSERR_CLIENT_ADDHASH	Überlauf der Hash-Tabelle.
0x752	1874	0x98110752	ADSERR_CLIENT_REMOVEHASH	Schlüssel in der Tabelle nicht gefunden.
0x753	1875	0x98110753	ADSERR_CLIENT_NOMORESYM	Keine Symbole im Cache.
0x754	1876	0x98110754	ADSERR_CLIENT_SYNCRESINVALID	Ungültige Antwort erhalten.
0x755	1877	0x98110755	ADSERR_CLIENT_SYNCPORTLOCKED	Sync Port ist verriegelt.
0x756	1878	0x98110756	ADSERR_CLIENT_REQUESTCANCELLED	Die Anfrage wurde abgebrochen.

RTime Fehlercodes

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

Spezifische positive HRESULT Return Codes:

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

TCP Winsock-Fehlercodes

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

8.5 Support und Service

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

Downloadfinder

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

Die Downloads sind in verschiedenen Formaten erhältlich.

Beckhoff Niederlassungen und Vertretungen

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

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

Dort finden Sie auch weitere Dokumentationen zu Beckhoff Komponenten.

Beckhoff Support

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

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

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

Beckhoff Service

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

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

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

Beckhoff Unternehmenszentrale

Beckhoff Automation GmbH & Co. KG

Hülshorstweg 20
33415 Verl
Deutschland

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

Trademark statements

Beckhoff®, TwinCAT®, TwinCAT/BSD®, TC/BSD®, EtherCAT®, EtherCAT G®, EtherCAT G10®, EtherCAT P®, Safety over EtherCAT®, TwinSAFE®, XFC®, XTS® and XPlanar® are registered trademarks of and licensed by Beckhoff Automation GmbH.

Third-party trademark statements

Arm, Arm9 and Cortex are trademarks or registered trademarks of Arm Limited (or its subsidiaries or affiliates) in the US and/or elsewhere.

DSP System Toolbox, Embedded Coder, MATLAB, MATLAB Coder, MATLAB Compiler, MathWorks, Predictive Maintenance Toolbox, Simscape, Simscape™ Multibody™, Simulink, Simulink Coder, Stateflow and ThingSpeak are registered trademarks of The MathWorks, Inc.

Intel, the Intel logo, Intel Core, Xeon, Intel Atom, Celeron and Pentium are trademarks of Intel Corporation or its subsidiaries.

The registered trademark Linux® is used pursuant to a sublicense from the Linux Foundation, the exclusive licensee of Linus Torvalds, owner of the mark on a worldwide basis.

Microsoft, Microsoft Azure, Microsoft Edge, PowerShell, Visual Studio, Windows and Xbox are trademarks of the Microsoft group of companies.

Wireshark is a registered trademark of Sysdig, Inc.

Mehr Informationen:
www.beckhoff.com/ts6100

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