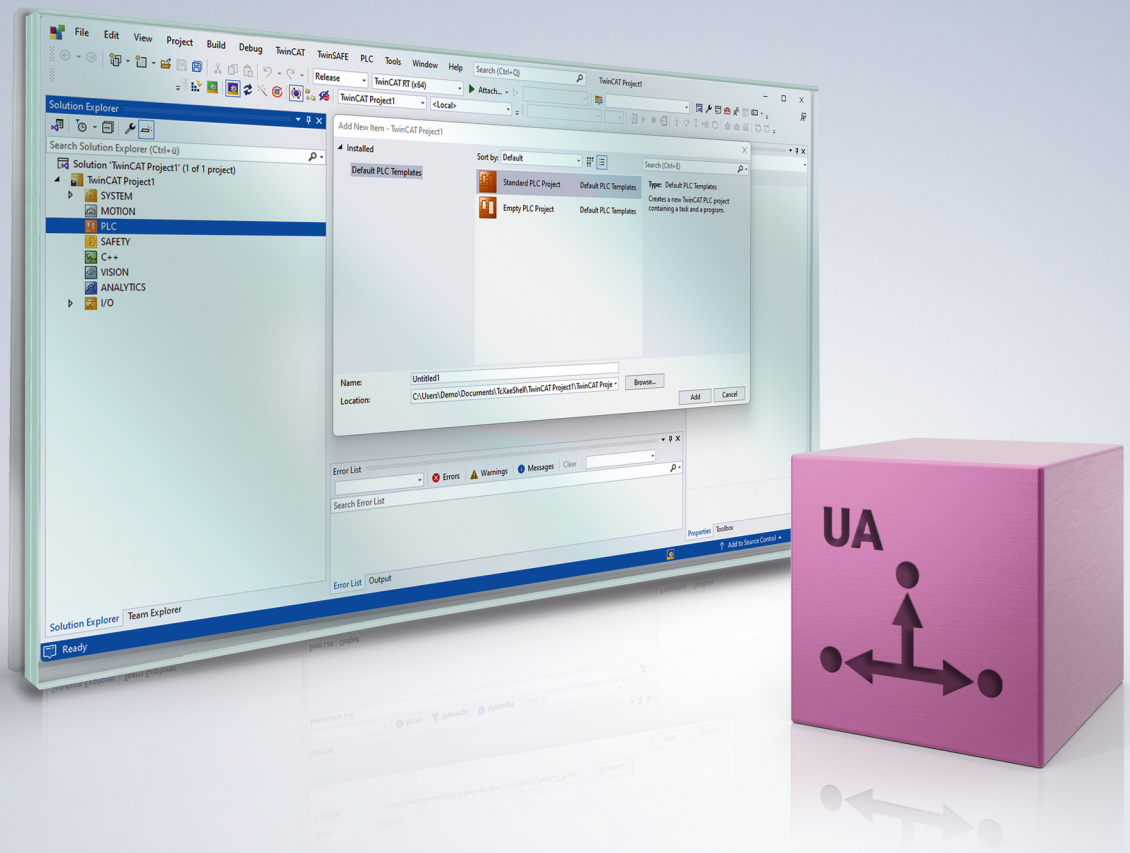
BECKHOFF New Automation Technology

手册 | ZH

TF6100

TwinCAT 3 | OPC UA Server



2026-07-07 | 版本： 1.5.0

目录

1 前言

1.1 文档说明

本说明仅适用于熟悉国家标准且经过培训的控制和自动化工程专家。
在安装和调试组件时，必须遵循文档和以下说明及解释。
操作人员应具备相关资质，并始终使用最新的生效文档。

相关负责人员必须确保所述产品的应用或使用符合所有安全要求，包括所有相关法律、法规、准则和标准。

免责声明

本文档经过精心准备。然而，所述产品正在不断开发中。
我们保留随时修改和更改本文档的权利，恕不另行通知。
不得依据本文档中的数据、图表和说明对已供货产品的修改提出赔偿。

商标

Beckhoff®、ATRO®、EtherCAT®、EtherCAT G®、EtherCAT G10®、EtherCAT P®、MX-System®、Safety over EtherCAT®、TC/BSD®、 TwinCAT®、TwinCAT/BSD®、TwinSAFE®、XFC®、XPlanar® 和 XTS®是 Beckhoff Automation GmbH
的注册商标并由其授权使用。本出版物中所使用的其它名称可能是商标名称，任何第三方出于其自身目的使用它们可能会侵犯商标所有者的权利。

EtherCAT®是注册商标和专利技术，由 Beckhoff Automation GmbH 授权使用。

版权所有

第三方商标

本文档可能使用了第三方商标。有关商标信息，可以访问：https://www.beckhoff.com/trademarks。

1.2 安全信息

安全规范

为了确保您的使用安全，请务必仔细阅读
并遵守本文档中每个产品的安全使用说明。

责任免除

所有组件在供货时都配有适合应用的特定硬件和软件配置。严禁未按文档所述修改硬件或软件配置，否则，德国倍福自动化有限公司对由此产生的后果不承担责任。

人员资格

本说明仅供熟悉适用国家标准的控制、自动化和驱动工程专家使用。

警示性词语

文档中使用的警示信号词分类如下。为避免人身伤害和财产损失，请阅读并遵守安全和警告注意事项。

人身伤害警告

⚠ 危险
存在死亡或重伤的高度风险。

⚠ 警告
存在死亡或重伤的中度风险。

⚠ 谨慎
存在可能导致中度或轻度伤害的低度风险。

财产或环境损害警告

注意
可能会损坏环境、设备或数据。

操作产品的信息

这些信息包括：
有关产品的操作、帮助或进一步信息的建议。

1.3 信息安全说明

Beckhoff Automation GmbH & Co.KG (简称 Beckhoff) 的产品，只要可以在线访问，都配备了安全功能，支持工厂、系统、机器和网络的安全运行。尽管配备了安全功能，但为了保护相应的工厂、系统、机器和网络免受网络威胁，必须建立、实施和不断更新整个操作安全概念。Beckhoff 所销售的产品只是整个安全概念的一部分。客户有责任防止第三方未经授权访问其设备、系统、机器和网络。它们只有在采取了适当的保护措施的情况下，方可与公司网络或互联网连接。

此外，还应遵守 Beckhoff 关于采取适当保护措施的建议。关于信息安全和工业安全的更多信息，请访问本公司网站 https://www.beckhoff.com/secguide。

Beckhoff 的产品和解决方案持续进行改进。这也适用于安全功能。鉴于持续进行改进，Beckhoff 明确建议始终保持产品的最新状态，并在产品更新可用后马上进行安装。使用过时的或不支持的产品版本可能会增加网络威胁的风险。

如需了解 Beckhoff 产品信息安全的信息，请订阅 https://www.beckhoff.com/secinfo 上的 RSS 源。

1.4 文档发行状态

版本	修改
1.5.0	手动添加 TOFU 用户 [▶ 34] 错误诊断 [▶ 153]:记录新错误 **新增：** 安装 [▶ 12]信息：TwinCAT 3 Usermode Runtime、Beckhoff RT Linux® 和 TwinCAT/BSD 容器 [▶ 18] 命名空间 [▶ 48] ServiceLevel [▶ 133] Tc3_OpcUa [▶ 140]
1.4.x	**新增：**

版本	修改
	字符串 [▶ 70]
1.3.x	安装/系统要求 [▶ 11]：更改系统要求服务器
1.2.x	**新增：** 安装 [▶ 12]：TwinCAT 2 安装程序 授权 [▶ 20]：TwinCAT 2 下的授权 冗余 [▶ 130]
1.1.0	**新增：** 快速入门 [▶ 26] (TwinCAT 2) 支持的功能 [▶ 37] 教程 [▶ 150]

2 概述

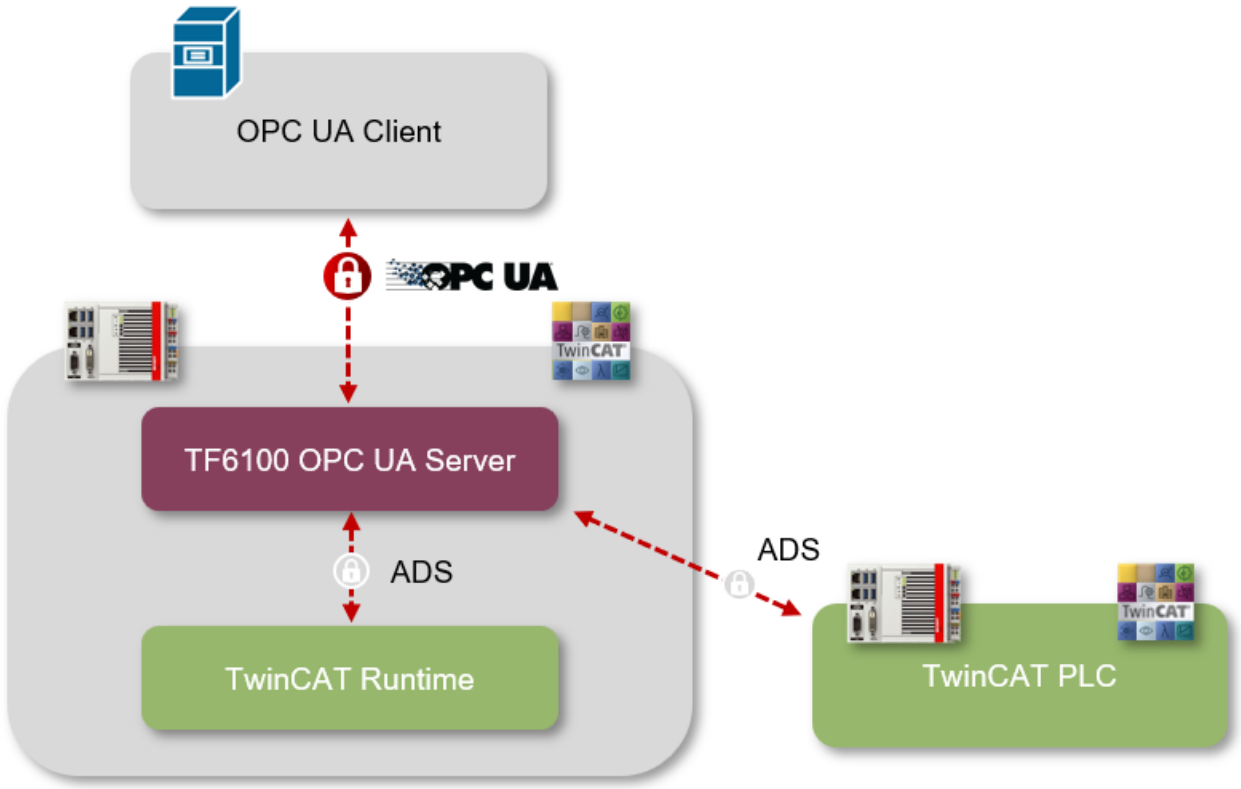
OPC 统一架构 (OPC UA) 是熟知的 OPC 标准的下一代产品。此为全球标准化通信协议，无论制造商和平台如何，都可以通过它交换机器数据。OPC UA 已经在协议中直接集成了通用安全标准。与 OPC 常规标准相比，OPC UA 的另一大优势是独立于 COM/DCOM 系统。

有关 OPC UA 的详细信息，请访问 OPC 基金会网页。

TwinCAT 3 功能 TF6100 OPC UA 由多个软件组件组成，可根据 OPC UA 与 TwinCAT 进行数据交换。
下表概述了各个产品组件。

软件组件	描述
TwinCAT OPC UA 服务器	提供 OPC UA 服务器接口，以便 UA 客户端访问 TwinCAT 运行时。
TwinCAT OPC UA 客户端	根据 PLCopen 标准化功能块和易于配置的 I/O 设备，提供 OPC UA 客户端功能，实现与其他 OPC UA 服务器的通信。
TwinCAT OPC UA 配置器	用于配置 TwinCAT OPC UA 服务器的图形用户界面。
TwinCAT OPC UA 示例客户端	OPC UA 客户端的图形示例实现，用于与 TwinCAT OPC UA 服务器进行首次连接测试。
TwinCAT OPC UA 网关	可提供 OPC COM DA 服务器接口和 OPC UA 服务器聚合功能的封装技术。

本文档介绍了 TwinCAT OPC UA Server，它是一个 OPC UA 服务器应用程序，可用于访问 TwinCAT 实时环境中的符号。可以直接在控制器上运行该服务器，也可以在网关 PC 上运行该服务器。
下图以简化形式展示了这两种关系，更详细的说明请参见“安装型号 [▶ 16]”一章。



3 安装

3.1 系统要求

以下系统要求适用于本产品的安装和操作。

服务器

技术数据	描述
操作系统	Windows 10 Windows 11 Windows CE 6/7（仅限 5.1.x 及之前的安装版本） Windows Server 2022 TwinCAT/BSD Beckhoff RT Linux®
目标平台	PC 架构（x86、x64、Arm®）
.NET Framework	---
TwinCAT 版本	TwinCAT 2、TwinCAT 3
TwinCAT 安装级别	TwinCAT 2 CP、PLC、NC PTP TwinCAT 3 XAE、XAR、ADS
所需 TwinCAT 授权	TS6100 TwinCAT OPC UA（适用于 TwinCAT 2） TF6100 TC3 OPC UA（适用于 TwinCAT 3）

示例客户端

技术数据	描述
操作系统	Windows 10 Windows 11 Windows Server 2022
目标平台	PC 架构（x86、x64、Arm®）
.NET Framework	4.7.2
TwinCAT 版本	TwinCAT 2、TwinCAT 3
TwinCAT 安装级别	TwinCAT 2 CP、PLC、NC PTP TwinCAT 3 XAE、XAR、ADS
所需 TwinCAT 授权	---

与 TwinCAT 2/TwinCAT 3 兼容

本文档适用于 TwinCAT 2 扩展组件（TS6100）和 TwinCAT 3 功能组件（TF6100）。
如果个别子功能存在差异，信息框将予以说明，并在必要时提供补充说明。

防火墙端口

若要与 TwinCAT OPC UA Server 进行通信，必须在该设备的防火墙中打开以下网络端口：

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

例如，如果在倍福工业 PC 上安装了 TwinCAT OPC UA Server，必须在操作系统防火墙中将该端口打开，作为传入通信端口。

硬件要求

TwinCAT OPC UA Server 对底层工业 PC 硬件的要求在很大程度上取决于相应的应用场景。一般来说，有 2 个关键指标会影响服务器的利用率：
主存和 CPU 负载。

服务器的主存负载取决于通过数据访问 [▶ 48]设备导入服务器地址空间的符号数量以及 OPC UA 客户端创建的并发连接、订阅、历史访问调用等的数量。但是，服务器的基本负载基于的是符号数量，可以将其设置为每个符号约 1,024 个字节。以 1,000,000 个符号为例，这意味着主存的使用量约为 1,000,000 个符号。

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

注意工业 PC 的主存负载

请确保您使用的工业 PC 有足够的主存。

为了进行初始测试，您可以在工程 PC 上启动项目，并在 Windows 任务管理器中读取 TwinCAT OPC UA 服务器的主存负载。注意操作系统的限值，如 32 位进程的 2 GB 主存限值。例如，这会影响到 TwinCAT OPC UA 服务器的 32 位变体。如果进程需要超过 2 GB 的主存，则必须使用 64 位版本的服务器。

另一方面，服务器的 CPU 负载完全取决于运行时时的条件，特别是 OPC UA 客户端向服务器请求的订阅和监控项目数量及其参数配置。请参见“优化 [▶ 42]”一章了解更多信息。

3.2 安装

根据所使用的 TwinCAT 版本和操作系统，该 TwinCAT 3 功能可以采用不同的安装方式，下面将详细介绍。

注意

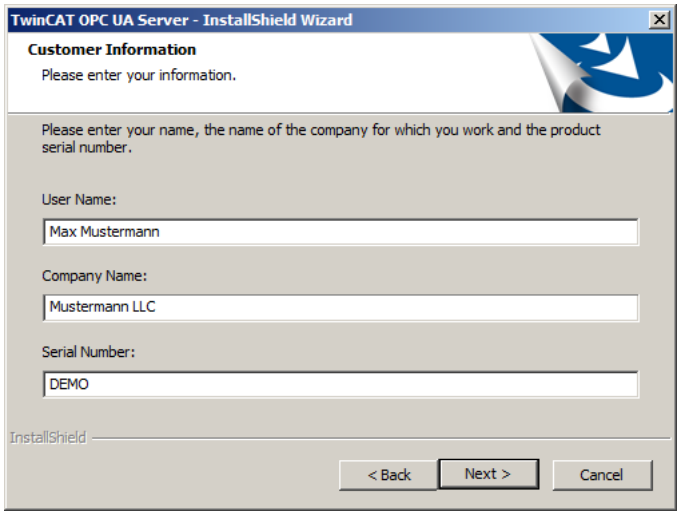
更新安装

更新安装总是会卸载之前安装的版本。请确保事先已备份配置文件。

TwinCAT 2 设置

如果您使用的是 TwinCAT 2，则可以通过安装包安装该插件，安装包可从倍福网站 https://www.beckhoff.com/download 下载。

安装可以在工程设计或运行时端进行，具体取决于您需要补充的系统。请注意，在 TwinCAT 2 下，安装过程中该插件会直接获得授权。另一个对话框会提示您输入授权密钥。如果您想安装 30 天试用版的插件，请输入 DEMO 作为授权密钥。



Windows CE 上的 TwinCAT 2 有单独的设置下载，名称为 TS6100-0030 OPC UA。该设置会安装 CAB 文件，然后您可以将其传输到 Windows CE 设备并进行安装。

TwinCAT 3 Package Manager

如果在 Microsoft Windows 操作系统上使用 TwinCAT 3.1 Build 4026（及更高版本），则可以通过 TwinCAT Package Manager 安装此功能，请参见 安装文档。

通常情况下，可通过相应的工作量安装此功能；然而，您也可以单独安装工作量中包含的软件包。本文档简要介绍了通过工作量进行安装的过程。

注意

未做好准备的 TwinCAT 重启会导致数据丢失

安装该功能可能会导致 TwinCAT 重启。
确保系统上没有运行重要的 TwinCAT 应用程序，或者先有序关闭这些应用程序。

命令行程序 TcPkg

您可以使用 TcPkg **C**ommand **L**ine **I**nterface（命令行界面，CLI）来显示系统上的可用工作量，并安装特定的工作量。

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

TwinCAT Package Manager UI

您可以使用 **U**ser **I**nterface（用户界面，UI）显示所有可用的工作量，并根据需要进行安装。为此，请按照界面中的相应说明操作。

TwinCAT 3 Usermode Runtime

本产品支持安装在 TwinCAT 3 Usermode Runtime 上。在此类环境中安装时，会自动部署相应的 UM 软件包。

UM 软件包安装了一个配置文件，用于说明 TwinCAT Usermode 系统服务在启动阶段会启动哪些应用程序。默认情况下，该配置文件位于 TwinCAT Usermode Runtime 目录中：

C:\ProgramData\Beckhoff\TwinCAT\3.1\Runtimes\UmRT_Default\3.1\Target\StartManConfig

每个产品都会提供各自的配置文件，因此在该目录中发现多个配置文件是正常现象。每个配置文件都会为其相关的 TwinCAT 3 功能组件定义与启动相关的信息，如：

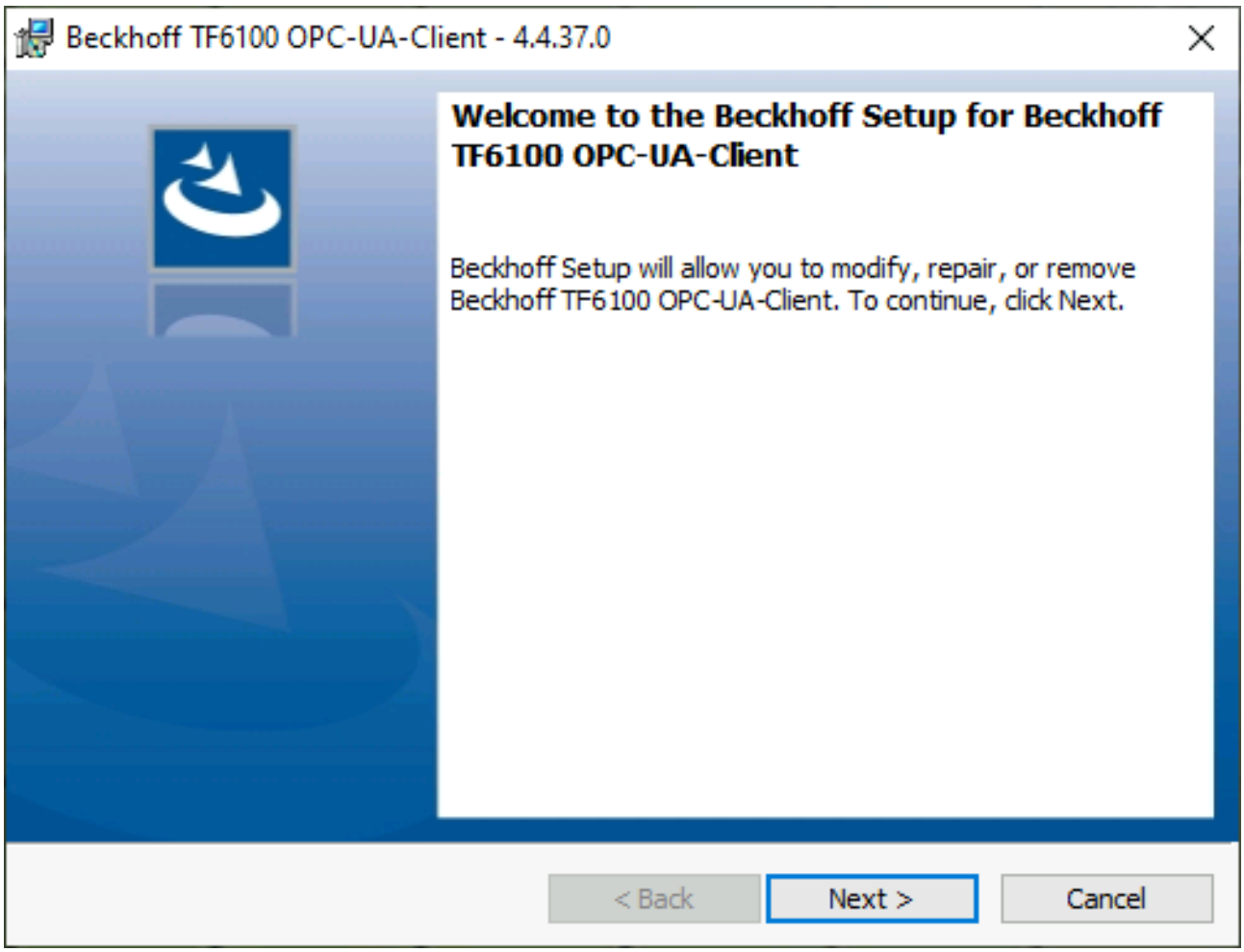
- 要启动的可执行文件的安装路径
- 启动 TwinCAT 3 功能组件所需的参数

TwinCAT Usermode 系统服务会将这些条目用于确定在运行时初始化过程中启动哪些应用程序以及何时执行这些应用程序。

TwinCAT 3 设置

如果您在 Microsoft Windows 操作系统上使用的是 TwinCAT 3.1 Build 4024，您可以通过一个安装包来安装该功能，该安装包可以从倍福网站 https://www.beckhoff.com/download 上下载。

根据需要使用该功能的系统，可以在工程设计或运行时端进行安装。下方截图显示的是使用 TF6100 TwinCAT OPC UA 客户端设置的设置界面示例。



要完成安装过程，请按照设置“对话框”中的说明进行操作。

注意

未做好准备的 TwinCAT 重启会导致数据丢失

安装该功能可能会导致 TwinCAT 重启。
确保系统上没有运行重要的 TwinCAT 应用程序，或者先有序关闭这些应用程序。

TwinCAT/BSD

如果使用 TwinCAT/BSD 作为操作系统，则可以通过 TwinCAT/BSD 软件包服务器安装此功能。有关 TwinCAT/BSD 软件包服务器及其配置的更多信息，请参阅 TwinCAT/BSD 文档。

要搜索软件包，可以在 TwinCAT/BSD 控制台中使用以下调用：

```
pkg search TF6100
```

要安装一个功能，可以在 TwinCAT/BSD 控制台中使用以下调用：

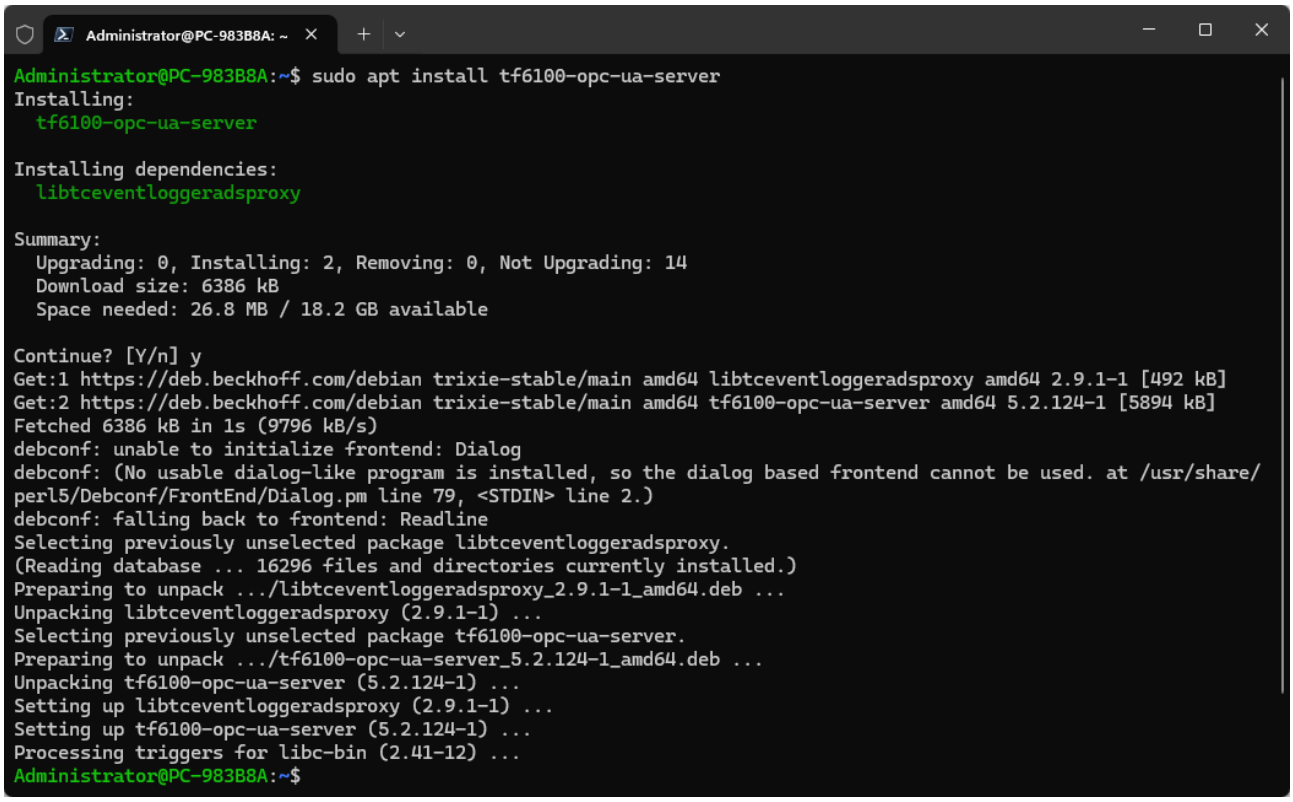
```
doas pkg install TF6100-OPC-UA-Server-<version>
```

Beckhoff RT Linux®

1. 直接在设备上使用 CLI。或者，您也可以通过 SSH 连接。
2. 以管理员身份登录。
3. 更新 Package Manager 中的软件包。
   ```
   sudo apt update
   ```
4. 搜索现有的 TF6100 软件包。
   ```
   sudo apt search tf6100
   ```

5. 通过 Package Manager 安装 TF6100 OPC UA 服务器。
   `sudo apt install tf6100-opc-ua-server`

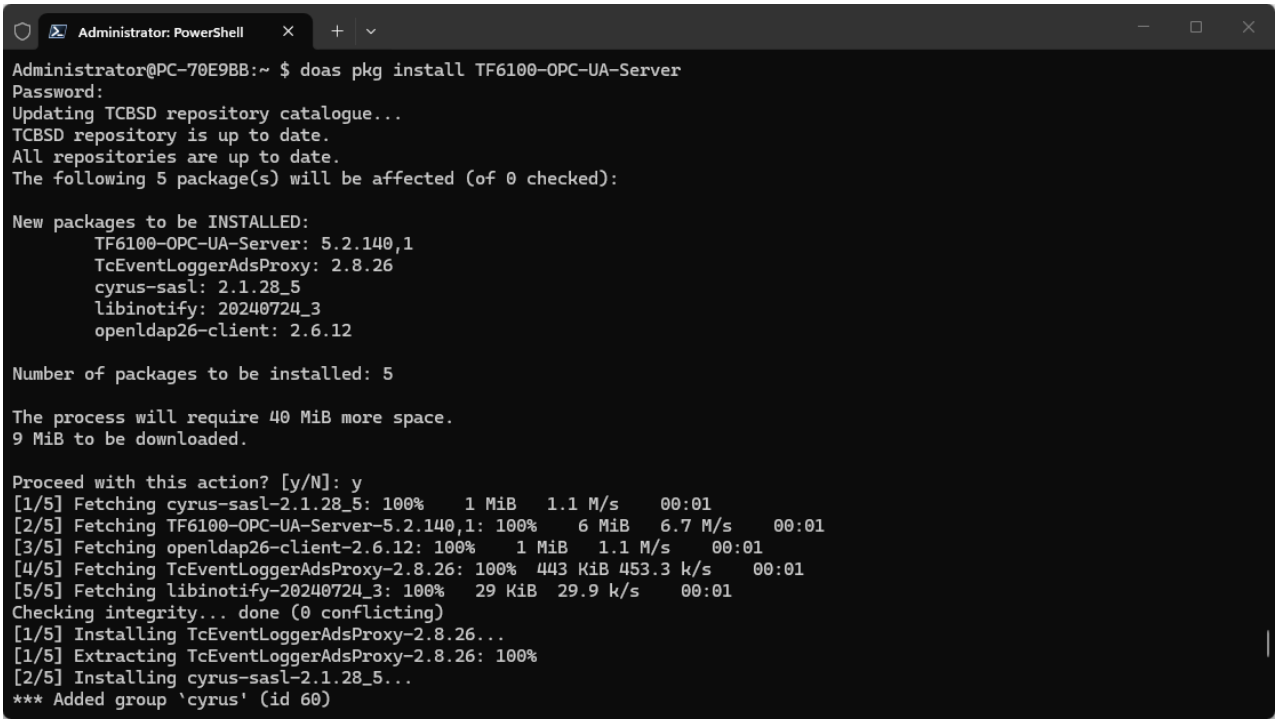
⇨ TF6100 OPC UA 服务器已安装。



TwinCAT/BSD

1. 直接在设备上使用 CLI。或者，您也可以通过 SSH 连接。
2. 以管理员身份登录。
3. 更新 Package Manager 中的软件包。
   `doas pkg update`
4. 搜索现有的 TF6100 软件包。
   `doas pkg search TF6100`
5. 通过 Package Manager 安装 TF610x OPC UA Client PubSub。
   `doas pkg install TF6100-OPC-UA-Server`

⇨ TF6100 OPC UA 服务器已安装。



Windows CE

如果使用 Microsoft Windows CE 作为操作系统，则可以通过相应的 CAB 文件来安装该功能，这些文件可以从倍福网站 https://www.beckhoff.com/download 单独下载。

下载完成后，可通过文件传输功能将 CAB 文件传输到 Windows CE 设备并执行。然后，CAB 文件会在相应的系统上安装和注册该功能。

请务必使用适合您系统的 CAB 文件。具体来说，这意味着：

- CE-ARMV4I：基于 Arm® 的设备，例如 CX8190、CX9020
- CE-X86：基于 x86 的设备，例如 CX51xx、CX52xx、CX20xx

CAB 文件可通过 CF/SD 卡或 Windows CE 中集成的 FTP 服务器传输到设备。

重启设备

安装此功能后，需要重启设备才能使用该功能。

3.3 更新安装

在更新安装过程中，会将旧版应用程序文件替换为新版应用程序文件。现有配置文件的备份将以 *.xml.bak 的形式创建并保存在应用程序目录 [▶ 46]中。此进程同样适用于证书目录。

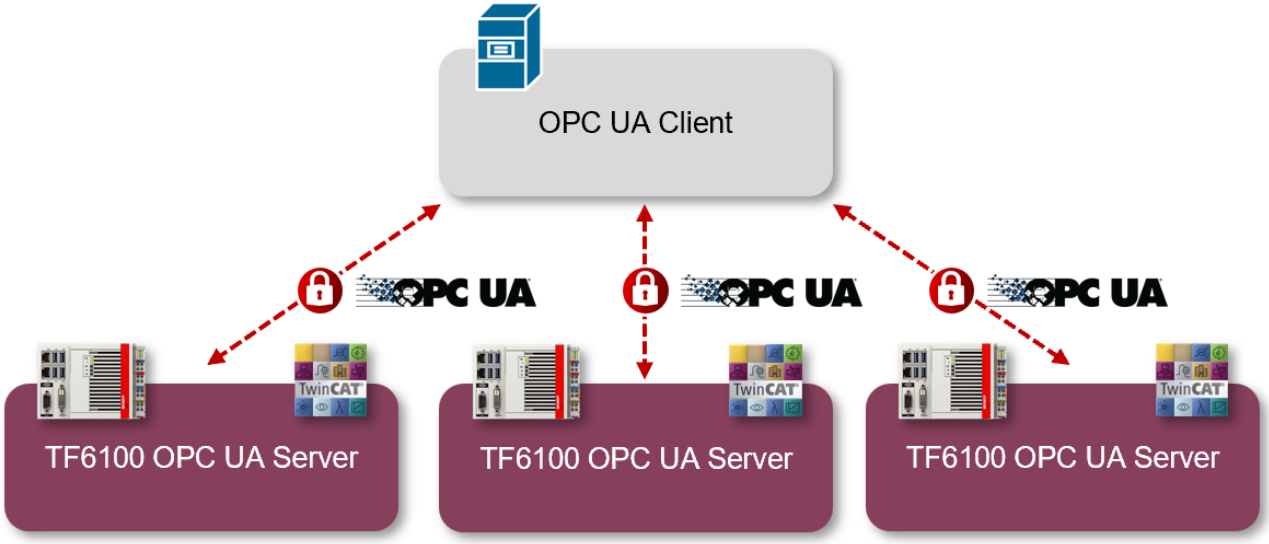
3.4 安装型号

下面将介绍两种安装方式，根据不同的应用和基础架构安装 TwinCAT OPC UA 服务器。

1. 将服务器直接集成在控制系统中
2. 在网关 PC 上运行服务器

将服务器直接集成在控制系统中

该方案描述了 TwinCAT OPC UA 服务器的正常运行方式。该软件的最大优势之一是可以集成到最小的嵌入式平台（如 CX8000 系列）中。有了这种集成，一般的操作都非常简单方便。OPC UA 客户端，例如 HMI 或 MES/ERP 系统，可以与网络中相应的 TwinCAT OPC UA Server 建立连接，并读取和写入 TwinCAT 运行时提供的符号信息。

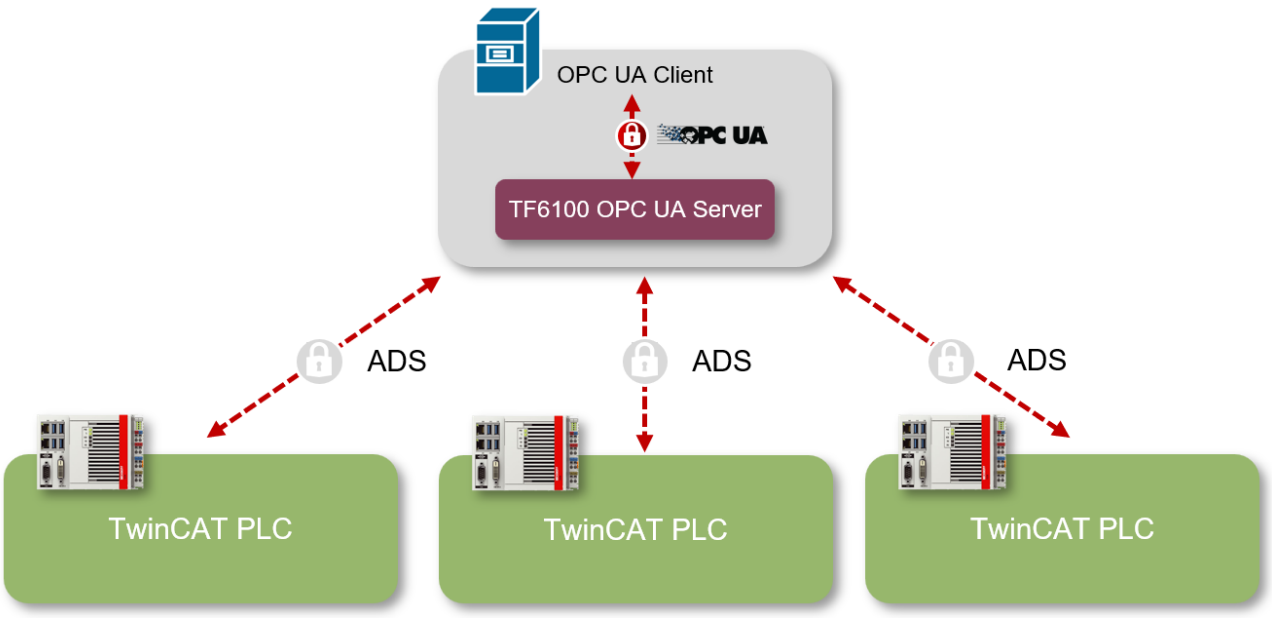


这种方案有以下优点：

- 由于可以使用订阅（"OnDataChange communication"）等 OPC UA 机构，因此优化了网络负载。
- 分散式内存使用 每个 TwinCAT OPC UA 服务器设备完全负责自己的存储空间需求，因为在服务器的地址空间中只需要提供其 "自己的 "PLC 符号。
- 直到控制系统的安全通信。OPC UA 具有直接集成在协议中的安全机构。控制器防火墙中只启用 OPC UA 服务器端口，然后用于安全通信。就反向互联互通 [▶ 127]功能而言，甚至无需打开（传入）防火墙端口。

在网关 PC 上运行服务器

该方案描述了在网关 PC 上使用 TwinCAT OPC UA 服务器的情况。这种用例通常用于需要为现有控制系统提供 OPC UA 接口的棕地方案。在这种情况下，TwinCAT OPC UA 服务器安装在网关 PC（通常称为 "边缘 PC"）上，并与一个或多个下属 TwinCAT PLC 控制器互联互通。然后，服务器与 PLC 之间通过 TwinCAT ADS 进行通信。



从经济角度来看，该方案非常有吸引力，因为只需购买 1 个 TwinCAT OPC UA Server 软件授权即可。不过，与将服务器集成到控制器中相比，这种方案也有一些技术上的缺点：

- 根据设备和符号的数量，网络负载可能会非常高。TwinCAT OPC UA Server 使用循环 ADS 采样从 TwinCAT 运行时快速高效地查询符号值，还必须能够同时为数千个符号提供服务。符号越多（互联互通的 PLC 控制器越多），网络中的循环通信就越多。
- 对中央服务器的内存需求非常高，因为 TwinCAT OPC UA Server 必须从每个 TwinCAT PLC 中导入符号信息并将其存储在其地址空间中。
- 服务器与 PLC 之间基于 TwinCAT ADS 进行通信，而 TwinCAT ADS 仅在较新版本的 TwinCAT 中才能实现传输通道的集成加密。旧版本系统作为棕地应用的一部分，尚不支持此功能。
- 每次更改 PLC 程序时，都必须在 TwinCAT PLC 与中央服务器之间交换符号文件。如果直接在 PLC 控制器中运行服务器，则无需执行此步骤。

3.5 容器

TwinCAT Runtime for Linux® 是一款功能强大的全新操作系统，专为倍福控制器打造，兼具 TwinCAT 的实时性与先进 Linux® 系统的灵活性。其核心创新之一是对 Docker® 等容器技术的原生支持，首次实现了 TwinCAT 功能组件以容器应用程序的形式进行安装和运行。

Docker® 容器的使用为 TwinCAT 功能组件的提供、版本化和维护开辟了新的可能性。用户可以对功能组件进行封装、重复部署，并独立于系统的其余部分进行更新。同时，用户无需放弃倍福控制平台的确定性实时特性，即可使用成熟的 Linux® 和 DevOps 工作流程。

本文将介绍如何在 Linux® 系统下，在 Docker® 容器中安装和运行此 TwinCAT 功能组件。其旨在以通俗易懂的方式介绍基本概念、前提条件和步骤，从而为实际使用基于容器的 TwinCAT 功能组件奠定了基础。

以下示例设置展示了在 Linux® 系统下以 Docker® 容器形式运行 TwinCAT 功能组件的典型系统架构。TwinCAT 功能组件以独立容器的形式运行，并通过指定的接口集成到 TwinCAT 系统中。

该控制器基于 TwinCAT Runtime for Linux® 运行，并提供了 TwinCAT 的实时环境和容器运行时环境（Docker®）。TwinCAT 功能组件在 Docker® 容器内运行，并使用系统提供的资源以及相应的 TwinCAT 接口与控制器和其他应用程序进行通信。在本例中，该功能组件与 TwinCAT PLC 运行时在同一容器中一起运行，但这些组件也可以独立运行。

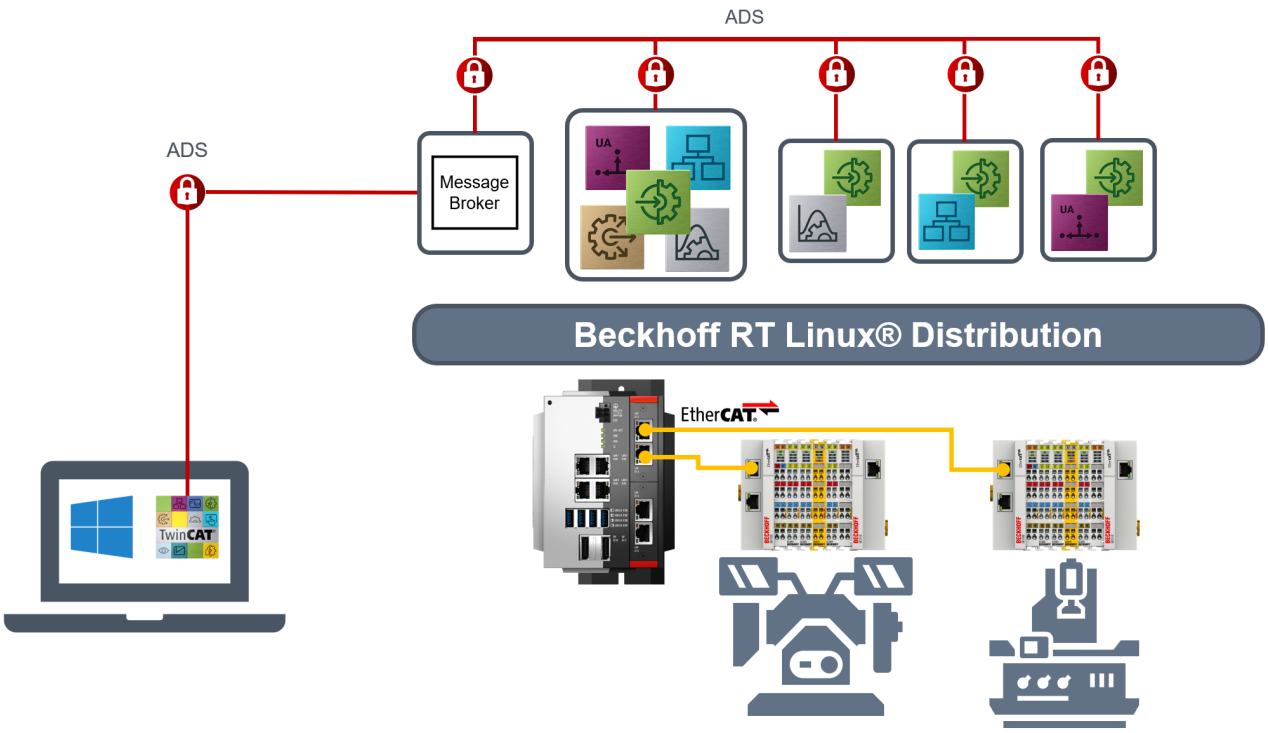
可以使用 ADS-over-MQTT 进行多个容器之间的 TwinCAT 端通信。通过将 MQTT 作为中间传输层，各个 TwinCAT 组件之间实现了解耦，从而大大简化了 ADS 路由的设置和管理工作。

相关容器通过共享的 MQTT 消息代理进行通信，而不是通过特定的静态点对点 ADS 路由进行通信。因此，更改 IP 地址、容器名称或部署结构时，通常不需要对 ADS 路由作任何调整，这一点对于容器化和动态环境的优势尤为明显。

另外，ADS-over-MQTT 所需的消息代理也可以作为 Docker® 容器在倍福控制器或独立系统上运行，从而无缝集成到基于容器的系统结构中。

注意

下图为概念性概览，并不代表完整的系统配置。



设置概述

以下步骤概述了将 TwinCAT 功能组件作为 Docker® 容器进行安装和调试的基本程序。有关详细说明和完整配置示例，可访问链接中的 GitHub 存储库（参见下文）。

查看要求

确保控制器配备了 TwinCAT Runtime for Linux®，并且安装和激活了 Docker®，作为容器运行时使用。

创建容器图像

通过 Docker® 文件创建一个新容器图像，图像中包含该 TwinCAT 功能组件和所有附属组件。

配置容器

使用 Docker® Compose File 配置容器运行时所需的参数。此类参数的示例如下：

- 网络和通信接口的设置
- 访问权限和资源
- TwinCAT 专用设置（例如 ADS 路由）
- 用于存储用户专用配置文件的卷映射

启动容器

在倍福控制器上启动 Docker® 容器，并检查 TwinCAT 功能组件是否正常启动。

集成到 TwinCAT 系统中

将 TwinCAT 功能组件集成到现有的 TwinCAT 项目中，并对通信和功能进行测试。

监控和维护

监控容器状态，评估日志，并在必要时执行更新或重启。

下方的 GitHub 存储库提供了一个完整的可执行示例，其中包含了详细的文档、配置文件和源代码：

https://www.github.com/Beckhoff/TF6100_Samples

3.6 授权

该 TwinCAT 产品需要获得授权才能完全运行。此外，还可以激活试用授权，以便对产品进行评估。

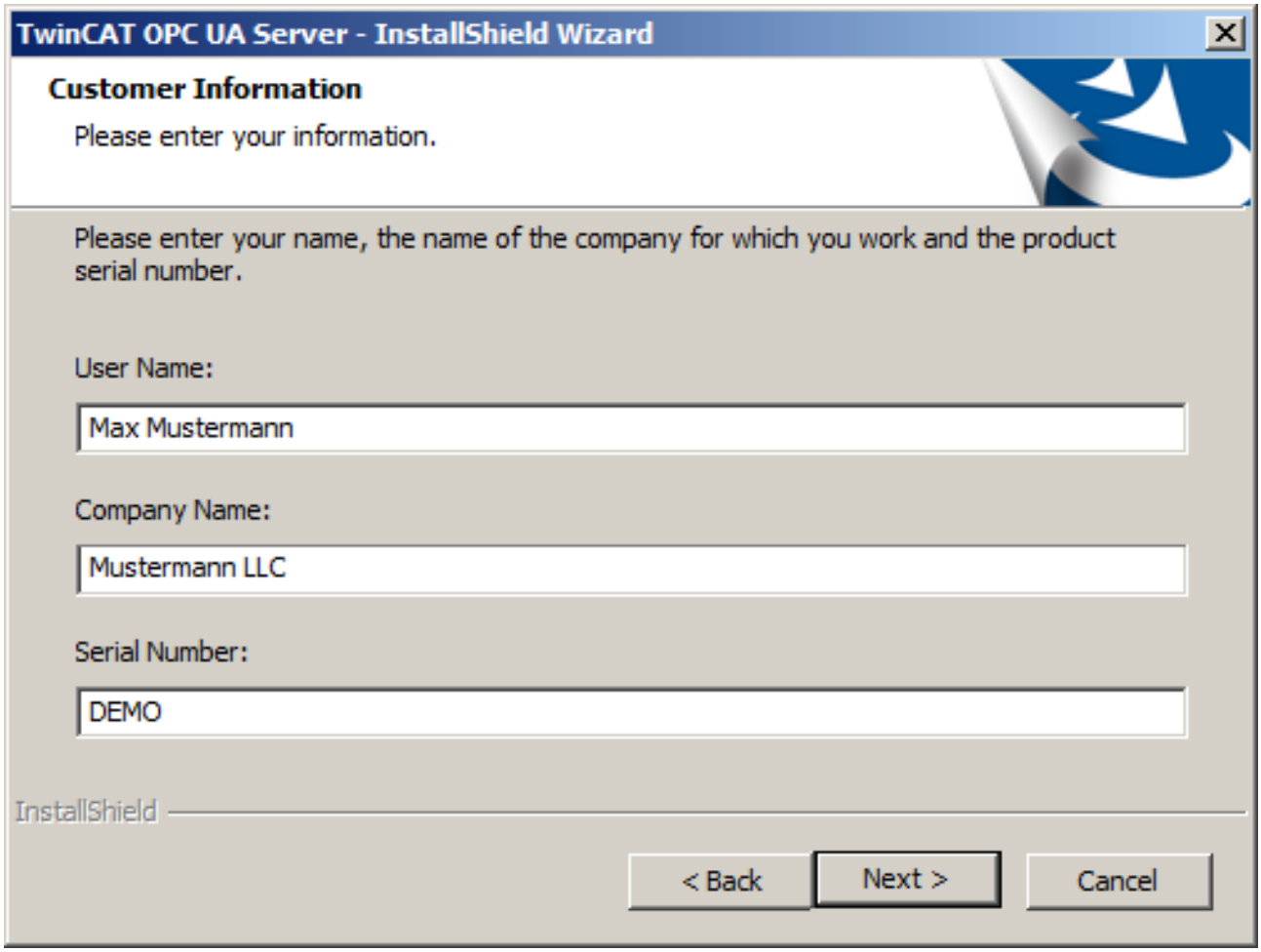
有关授权的更多信息，请参见以下章节：

- TwinCAT 2 下的授权
- TwinCAT 3 下 7 天试用版的授权
- TwinCAT 3 下完整版的授权

有关 TwinCAT 3 授权的更多信息，请参见倍福信息系统中的“授权”文档（TwinCAT 3 > Licensing（授权））。

TwinCAT 2 下的授权

在安装过程中，该产品已获得 TwinCAT 2 授权；您可以在相应的文本字段中输入授权密钥。要评估该产品，您可以将其激活 30 天。为此，请输入“Demo”作为授权密钥。



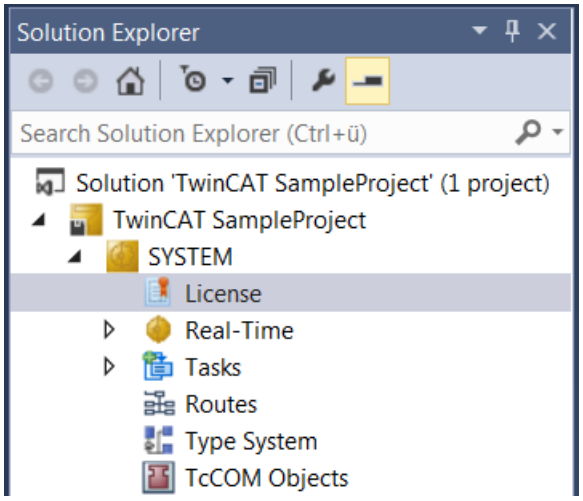
授权一个 TwinCAT 3 功能的 7 天测试版本

对于TwinCAT 3 许可证加密狗无法启用 7 天测试版本。

1. 启动 TwinCAT 3 开发环境（XAE）。
2. 打开一个现有的 TwinCAT 3 项目或创建一个新项目。
3. 如果想为一个远程设备激活授权，请设置所需的目标系统。为此，从工具栏的**选择目标系统**下拉列表中选择目标系统。

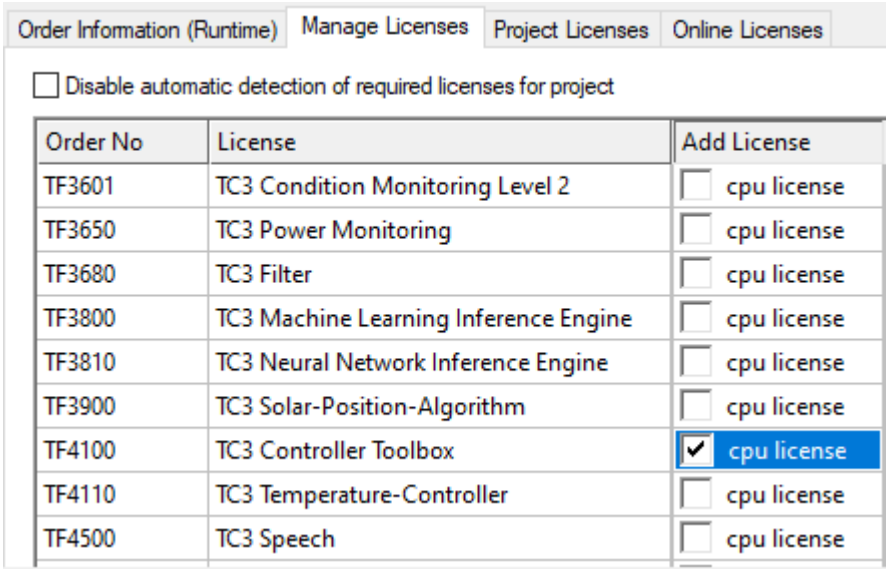
⇨ 授权设置总是指选定的目标系统。在目标系统上激活项目时，自动将相应的 TwinCAT 3 许可证复制到该系统中。

4. 在**解决方案资源管理器**中，双击**系统**子目录的**许可证**。



⇨ TwinCAT 3 许可证管理器打开。

5. 打开**管理许可证**选项卡。在**添加许可证**栏中，选中希望添加到项目的许可证的复选框（例如“TF4100 TC3 控制器工具箱”）。

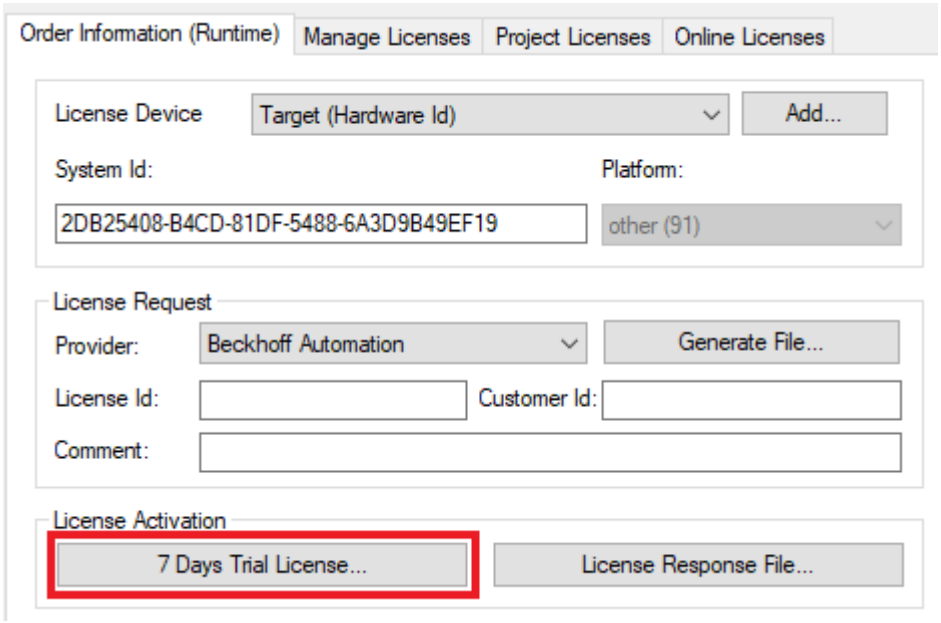


Order No	License	Add License
TF3601	TC3 Condition Monitoring Level 2	cpu license
TF3650	TC3 Power Monitoring	cpu license
TF3680	TC3 Filter	cpu license
TF3800	TC3 Machine Learning Inference Engine	cpu license
TF3810	TC3 Neural Network Inference Engine	cpu license
TF3900	TC3 Solar-Position-Algorithm	cpu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

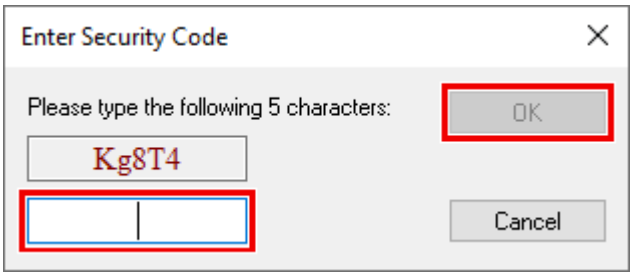
6. 打开**订单信息（运行时间）**选项卡。

⇨ 在许可证的表格概览中，以前选择的许可证显示为“缺失”状态。

7. 点击**7 天试用许可证...**，以激活 7 天试用许可证。



⇨ 一个对话框打开，提示输入对话框中显示的安全代码。



8. 准确输入显示的代码并确认输入。
9. 确认随后的对话框，表明激活成功。
   ⇨ 在许可证的表格概览中，许可证状态现在显示许可证的到期日。
10. 重新启动 TwinCAT 系统。

⇨ 7 天试用版启用。

授权使用 TwinCAT 3 功能的完整版本

关于许可证完整版本的程序描述，可参见倍福信息系统的文档“TwinCAT 3 授权”。

4 技术简介

4.1 快速入门

下一章将介绍 TwinCAT OPC UA Server 的快速入门方法。您可以按照这些说明对交付状态x下的 TwinCAT OPC UA Server 进行初始化，然后创建一个 TwinCAT PLC 项目，最后通过 OPC UA 设置一个编译指示，从而启用 PLC 变量。然后即可在服务器的地址空间中使用该变量。
下文将按执行顺序详细介绍这些行动步骤：

- 服务器初始化
- 创建 TwinCAT PLC 项目
- 激活符号文件的下载
- 添加产品授权
- 启用变量
- 连接 OPC UA 客户端

服务器初始化

安装完成后，需要根据所谓的 TOFU 原则（**T**rust-**O**n-**F**irst-**U**se（首次使用即信任））对服务器进行一次初始化 [▶ 29]。在此过程中，您需要配置一个用户账户，然后才能与服务器建立连接。

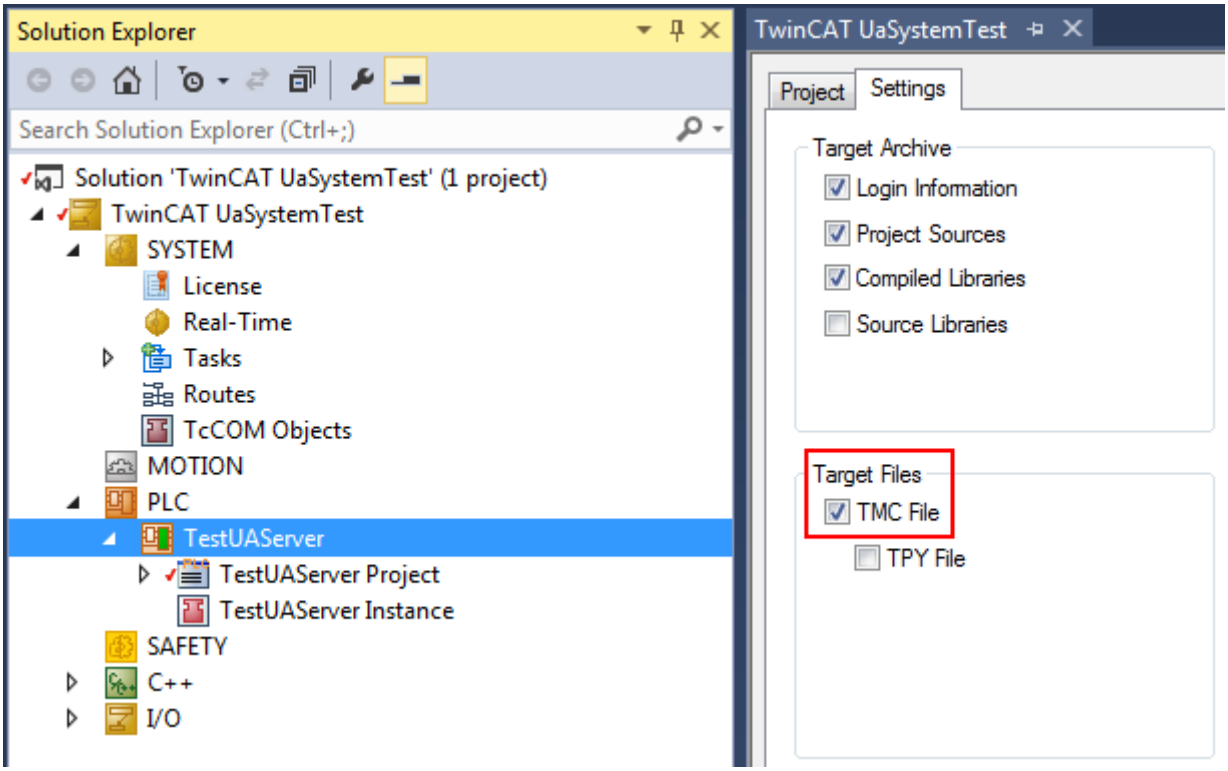
由于该主题十分重要且与安全息息相关，我们将在单独的文档章节中对初始化 [▶ 29]进行详细说明。本节的后续步骤假设您已执行一次该进程，并且已使用用户账户对服务器进行初始化。

服务器初始化所用的工具
您可以使用 TwinCAT OPC UA Configurator 对服务器进行初始化。此操作可能需要另行安装一个软件包。

创建 TwinCAT PLC 项目（下载符号文件）

1. 打开 TwinCAT XAE Shell（或 Visual Studio）。
2. 在“**File**（文件）”菜单中选择命令“**New > Project**（新建 > 项目）”。
3. 在项目中添加一个空的 PLC 项目。
4. 在 PLC 项目属性中激活自动下载 TMC 文件，如下方的屏幕截图所示。

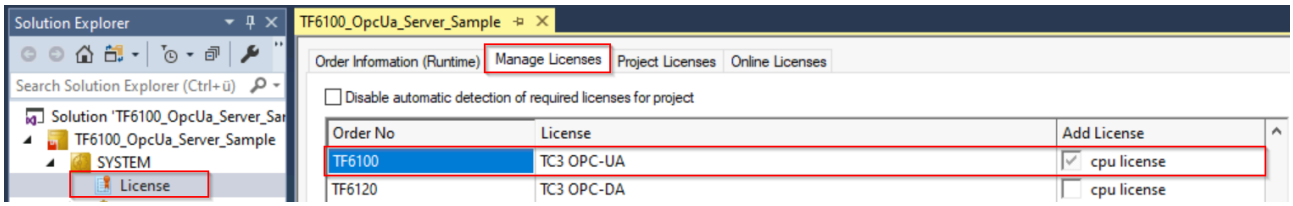
⇨ 新 TwinCAT 项目创建完成。



添加产品授权

✓ 在 TwinCAT XAE 授权对话框中查看 TF6100 授权是否可用。

1. 如果不可用，您可以使用本快速入门教程中的 7 天试用授权。



启用变量

2. 在 MAIN 程序中添加一个数据类型为 INT 的新变量。
3. 在此变量上设置 pragma 可通过 OPC UA 启用变量。

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

4. 在执行 MAIN 程序时，每次循环该变量都会递增 1。

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

5. 在您的系统上激活 TwinCAT 项目。

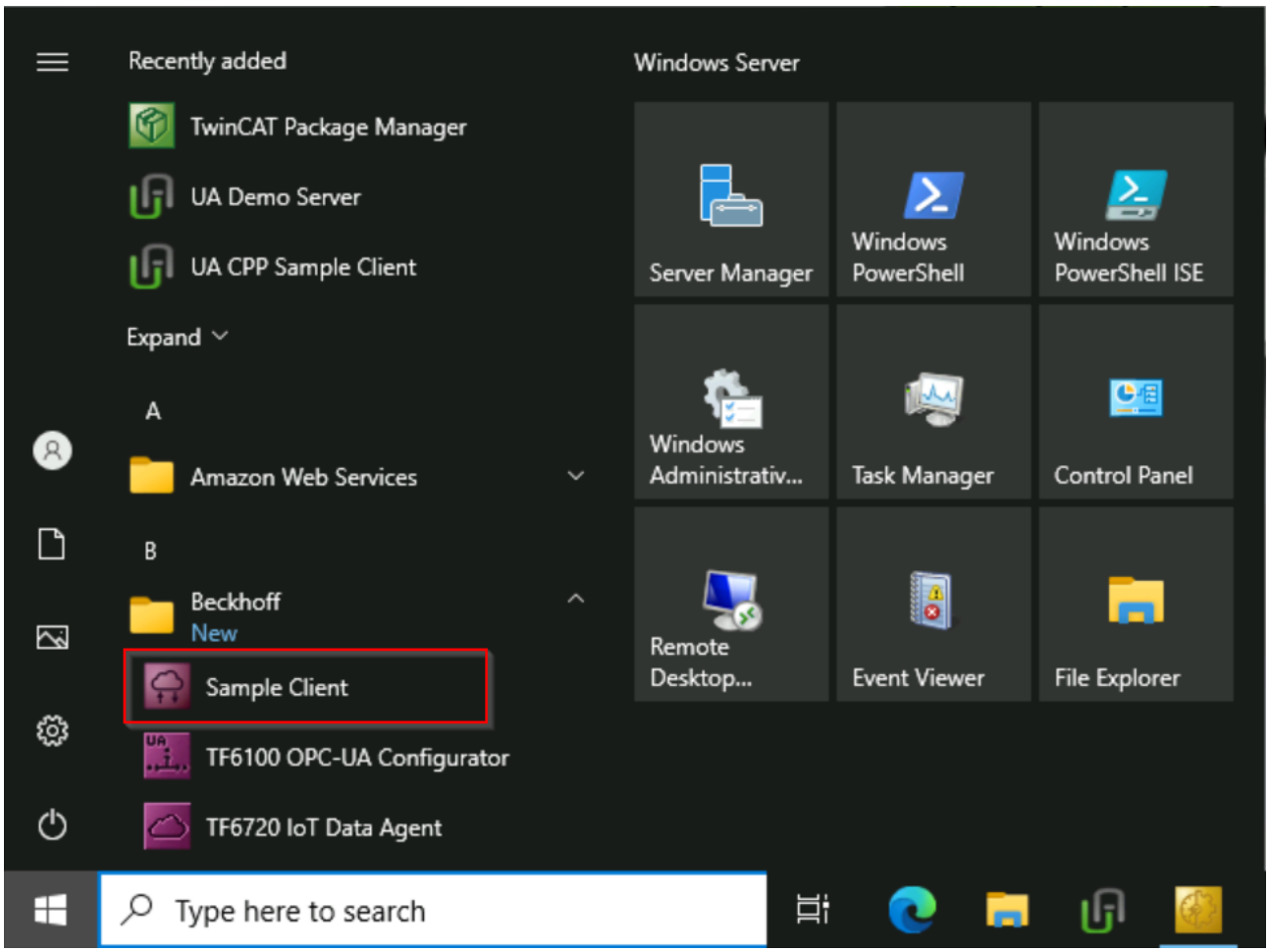
连接 OPC UA 客户端

OPC UA 客户端使用所谓的 ServerURL 来连接服务器。ServerURL 包含安装该服务器的设备的 IP 地址或主机名。在本教程中，我们假设该客户端和服务器在同一系统上运行。因此，该客户端会连接到以下 ServerURL：

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

作为客户端，我们使用了 TF6100 产品包中所包含的 TwinCAT OPC UA Sample Client。

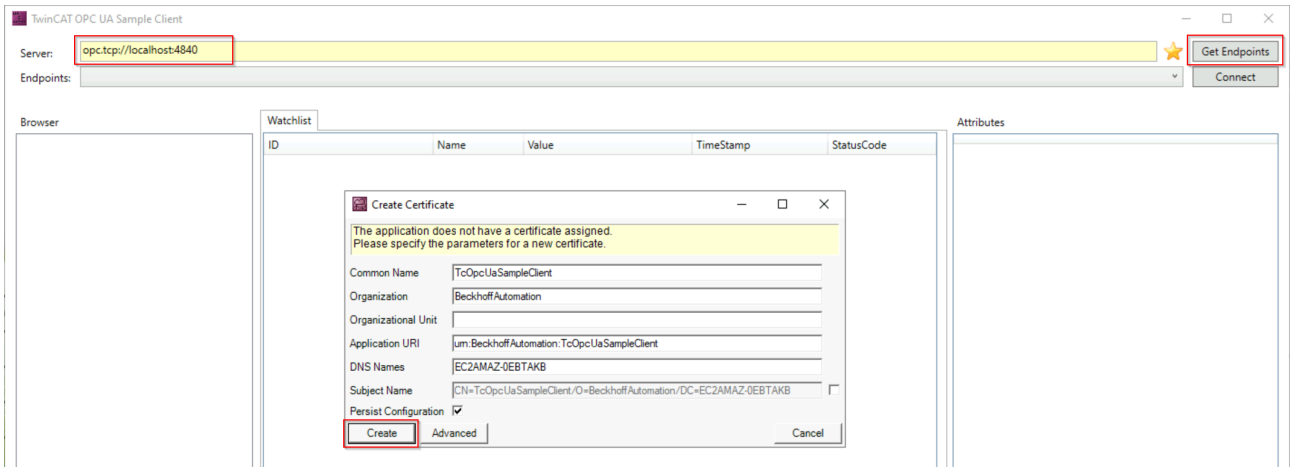
安装后，可通过 Windows 开始菜单调用。



在 TwinCAT OPC UA 示例客户端中，已经默认输入了本地主机的服务器 URL。

1. 单击 "**获取端点**"按钮。

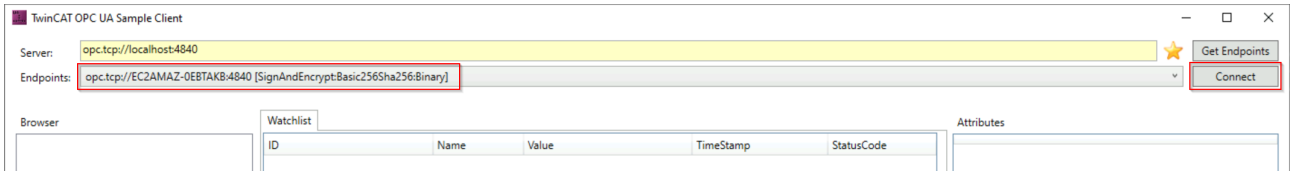
   ⇨ 如果第一次使用示例客户端，会出现一个对话框，接受生成应用程序证书的设置。



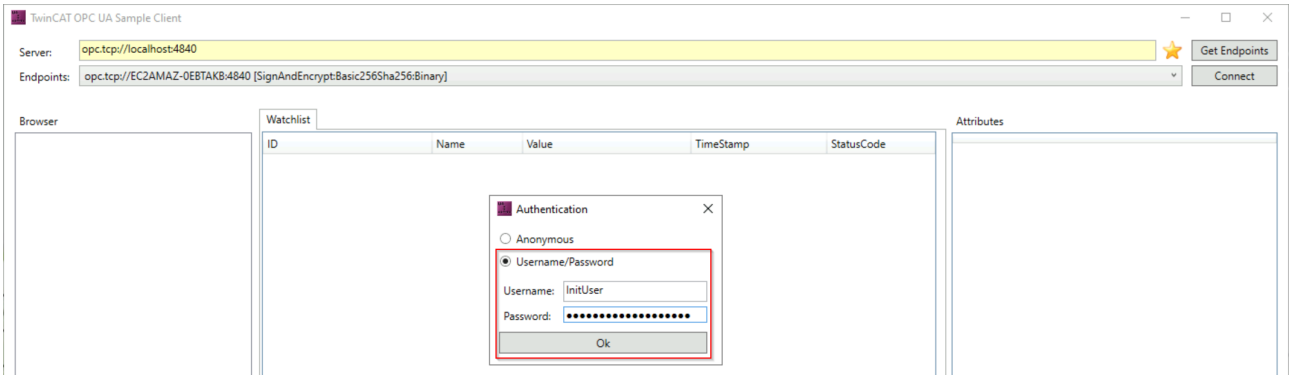
2. 点击“**Create**（创建）”按钮确认该对话框。

   ⇨ 现在已从服务器读取并显示所有连接端点。

3. 选择端点“SignAndEncrypt:Basic256Sha256:Binary”，然后点击“**Connect**（连接）”按钮。



4. 输入您在本文档文章第一步中为初始化服务器而配置的用户账户数据。

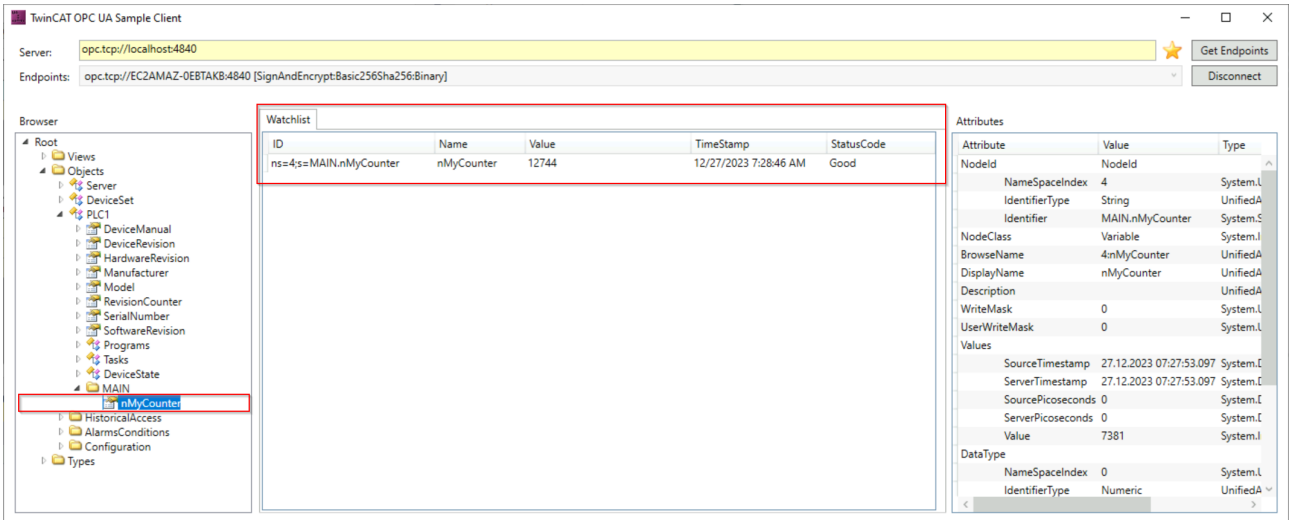


5. 点击“**OK**（确定）”。

⇨ 您现在已与服务器连接。
服务器的地址空间以树形结构显示在应用程序的左侧，您可以通过 PLC 程序的符号进行浏览。在本例中，我们为 OPC UA 启用了一个 PLC 变量。这些信息可在以下路径找到：

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

双击该变量即可将其添加至“观察列表”中。这意味着为该变量创建了一个订阅，一旦变量值发生变化，就会将该变量值从服务器传输至客户端。



4.2 快速入门（TwinCAT 2）

下一章将为您提供如何在 TwinCAT 2 下运行 TwinCAT OPC UA Server 的快速入门指南。您可以按照这些说明对交付状态下的 TwinCAT OPC UA Server 进行初始化，然后创建一个 TwinCAT PLC 项目，最后通过 OPC UA 设置一个注释，从而启用某个 PLC 变量。然后即可在服务器的地址空间中使用该变量。

下文将按执行顺序详细介绍这些行动步骤：

- 服务器初始化
- 创建 TwinCAT PLC 项目
- 激活符号文件的下载
- 启用变量
- 连接 OPC UA 客户端

服务器初始化

安装完成后，需要根据所谓的 TOFU 原则（**T**rust-**O**n-**F**irst-**U**se（首次使用即信任））对服务器进行一次初始化 [▶ 29]。在此过程中，您需要配置一个用户账户，然后才能与服务器建立连接。

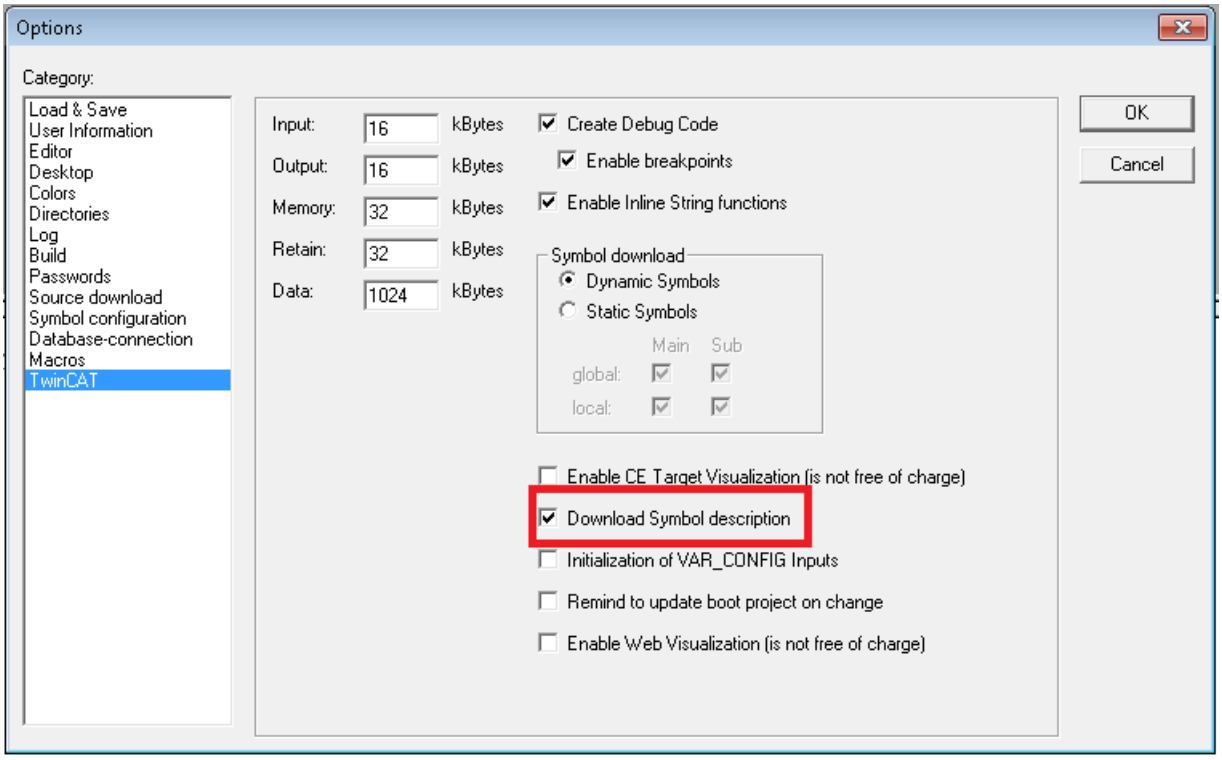
由于该主题十分重要且与安全息息相关，我们将在单独的文档章节中对初始化 [▶ 29]进行详细说明。本节的后续步骤假设您已执行一次该进程，并且已使用用户账户对服务器进行初始化。

服务器初始化所用的工具

您可以使用 TwinCAT OPC UA Configurator 对服务器进行初始化。此操作可能需要另行安装一个软件包。

创建 TwinCAT PLC 项目并激活符号文件的下载

在 TwinCAT PLC Control 中创建一个新项目或打开您现有的 PLC 项目。打开项目属性，激活“Download Symbol description（下载符号说明）”选项，以便在激活 PLC 程序时，将其符号文件（*.tpy）传输至目标设备的启动目录中。



启用变量

1. 在 MAIN 程序中添加一个数据类型为 INT 的新变量。
2. 在该变量上设置注释，以便通过 OPC UA 启用变量。可以通过“Description（描述）”来定义文本，在 OPC UA 地址空间的相应节点上也会以“描述”属性显示该文本。

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

3. 您可以添加注释，将该变量标记为只读。随后，服务器会拒绝通过 OPC UA 进行写入操作，状态代码为 BadNotWriteable。

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

4. 您可以使用以下注释为变量定义别名，即在 OPC UA 命名空间中为该变量指定一个不同的名称。您为 x 指定的值将与新的变量名相对应。

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

5. 在执行 MAIN 程序时，每次循环该变量都会递增 1。

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

6. 在您的系统上激活 TwinCAT 项目。

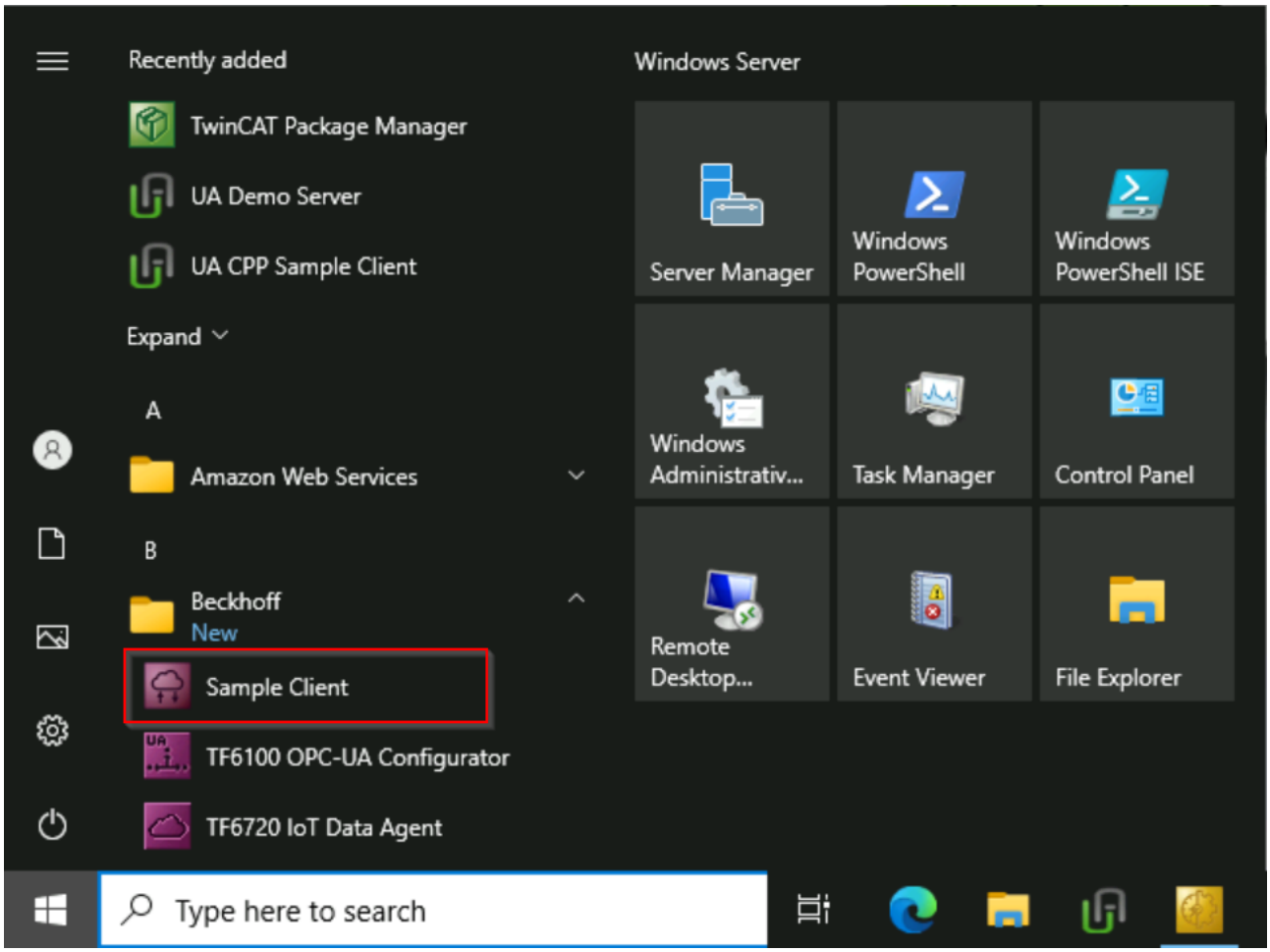
连接 OPC UA 客户端

OPC UA 客户端使用所谓的 ServerURL 来连接服务器。ServerURL 包含安装该服务器的设备的 IP 地址或主机名。在本教程中，我们假设该客户端和服务器在同一系统上运行。因此，该客户端会连接到以下 ServerURL：

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

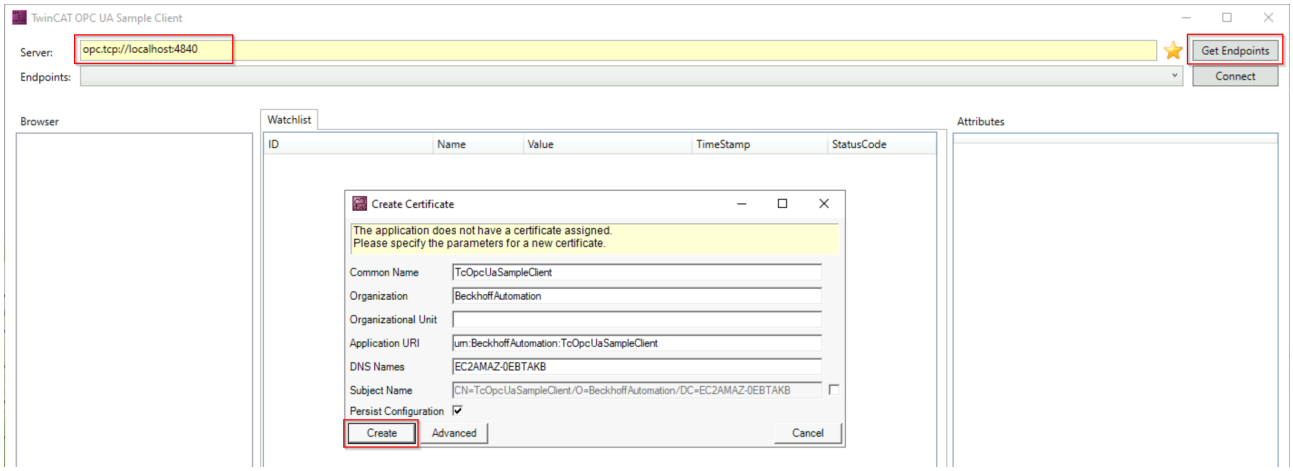
作为客户端，我们使用了 TF6100 产品包中所包含的 TwinCAT OPC UA Sample Client。

安装后，可通过 Windows 开始菜单调用。

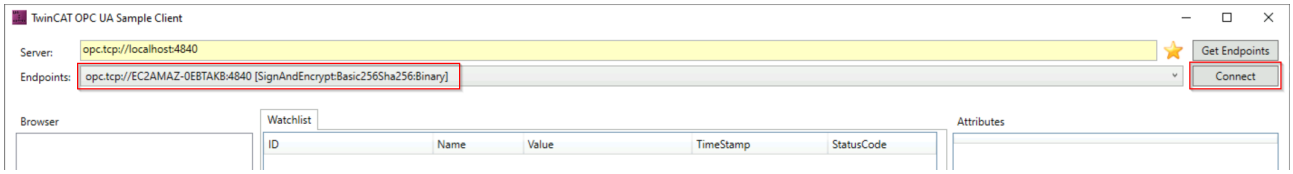


在 TwinCAT OPC UA 示例客户端中，已经默认输入了本地主机的服务器 URL。

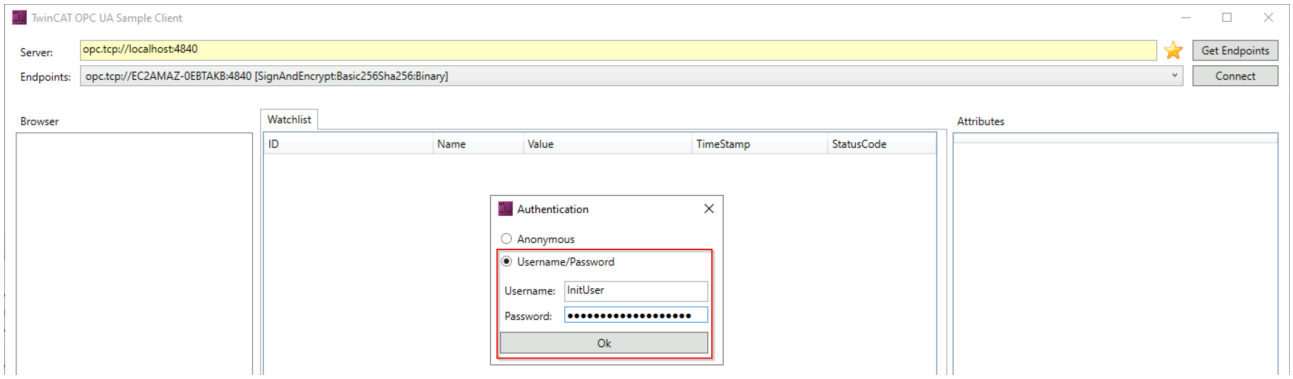
1. 单击 "**获取端点**"按钮。
   ⇨ 如果第一次使用示例客户端，会出现一个对话框，接受生成应用程序证书的设置。



2. 点击“**Create**（创建）”按钮确认该对话框。
   ⇨ 现在已从服务器读取并显示所有连接端点。
3. 选择端点“SignAndEncrypt:Basic256Sha256:Binary”，然后点击“**Connect**（连接）”按钮。



4. 输入您在本文档文章第一步中为初始化服务器而配置的用户账户数据。

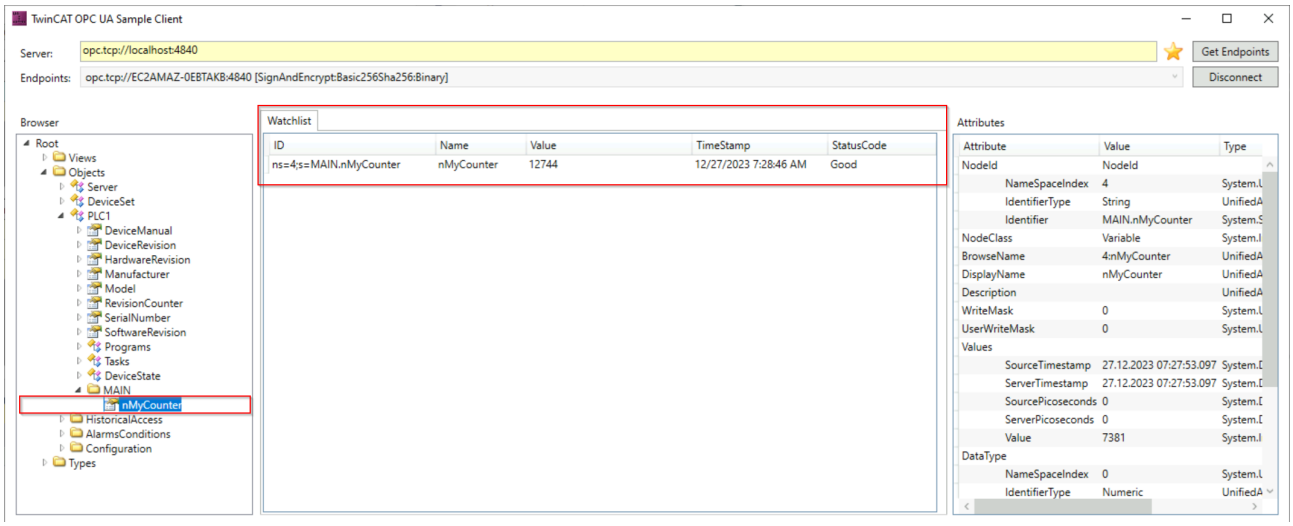


5. 点击“**OK**（确定）”。

⇨ 您现在已与服务器连接。
服务器的地址空间以树形结构显示在应用程序的左侧，您可以通过 PLC 程序的符号进行浏览。在本例中，我们为 OPC UA 启用了一个 PLC 变量。这些信息可在以下路径找到：

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

双击该变量即可将其添加至“观察列表”中。这意味着为该变量创建了一个订阅，一旦变量值发生变化，就会将该变量值从服务器传输至客户端。

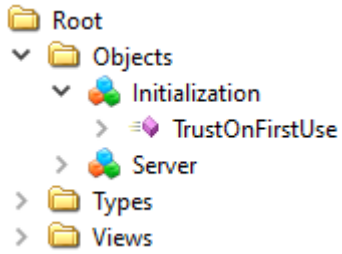


4.3 初始化

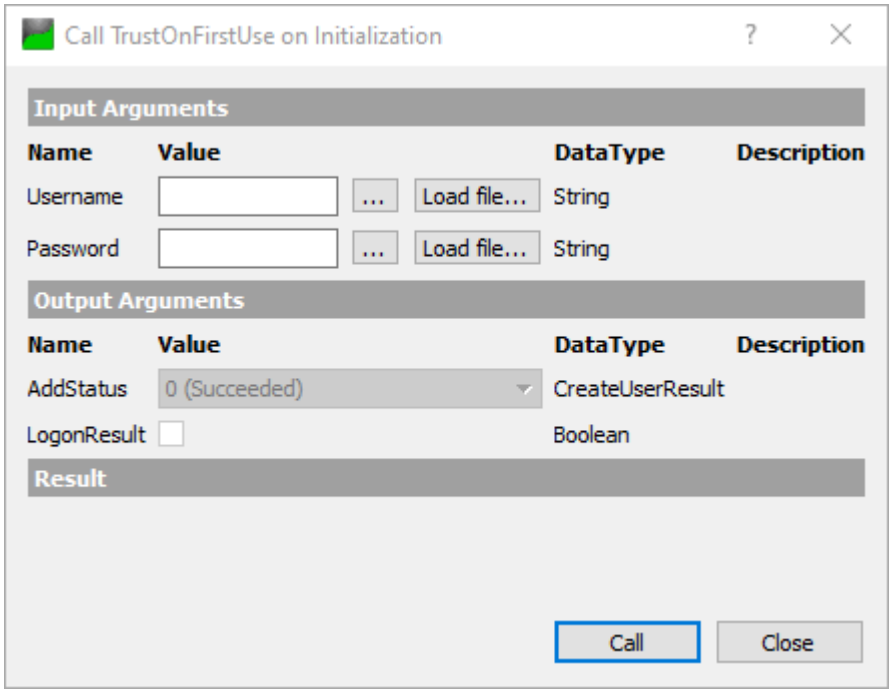
从安装版本 4.4.0 开始，TwinCAT OPC UA Server 需要完成一个基于 TOFU 原则（**T**rust **O**n **F**irst **U**se（首次使用即信任））的初始化阶段。这意味着用户必须主动对该服务器进行初始化，之后才能使用其各种功能（数据访问、历史访问等）。

默认情况下，服务器允许客户端建立未经验证的互联互通（"匿名"）。而一次性 TOFU 初始化现在需要配置一个操作系统用户，OPC UA 客户端随后必须使用该用户账户才能成功登录该服务器。

为此，服务器仅在未初始化状态下提供一个特殊的初始化命名空间。该命名空间包含一个对象“Initialization”和一个方法“TrustOnFirstUse”。



该方法定义了以下输入/输出参数：



参数	描述
[in] Username	待创建的操作系统用户的用户名。如果用户已存在，服务器会尝试使用指定的密码进行登录测试，若登录成功，则会将现有用户转移至其安全配置中。
[in] Password	操作系统用户的密码。 **密码不存储在服务器配置中，只能在操作系统的用户数据库中获取。请注意，密码类型可能取决于操作系统的安全设置（关键词“复杂密码”）。**
[out] AddStatus	指示操作系统用户是否创建成功，或用户是否已经存在。
[out] LogonResult	指示服务器是否能用指定的用户名/密码组合登录操作系统。如果用户已存在，可通过此参数有效检查是否输入了错误的密码。
[out] OPC UA Statuscode	调用方法时采用的常规 OPC UA 状态代码。如果在 OPC UA 层成功调用了该方法，状态代码将返回“GOOD”，否则将返回“BAD”。

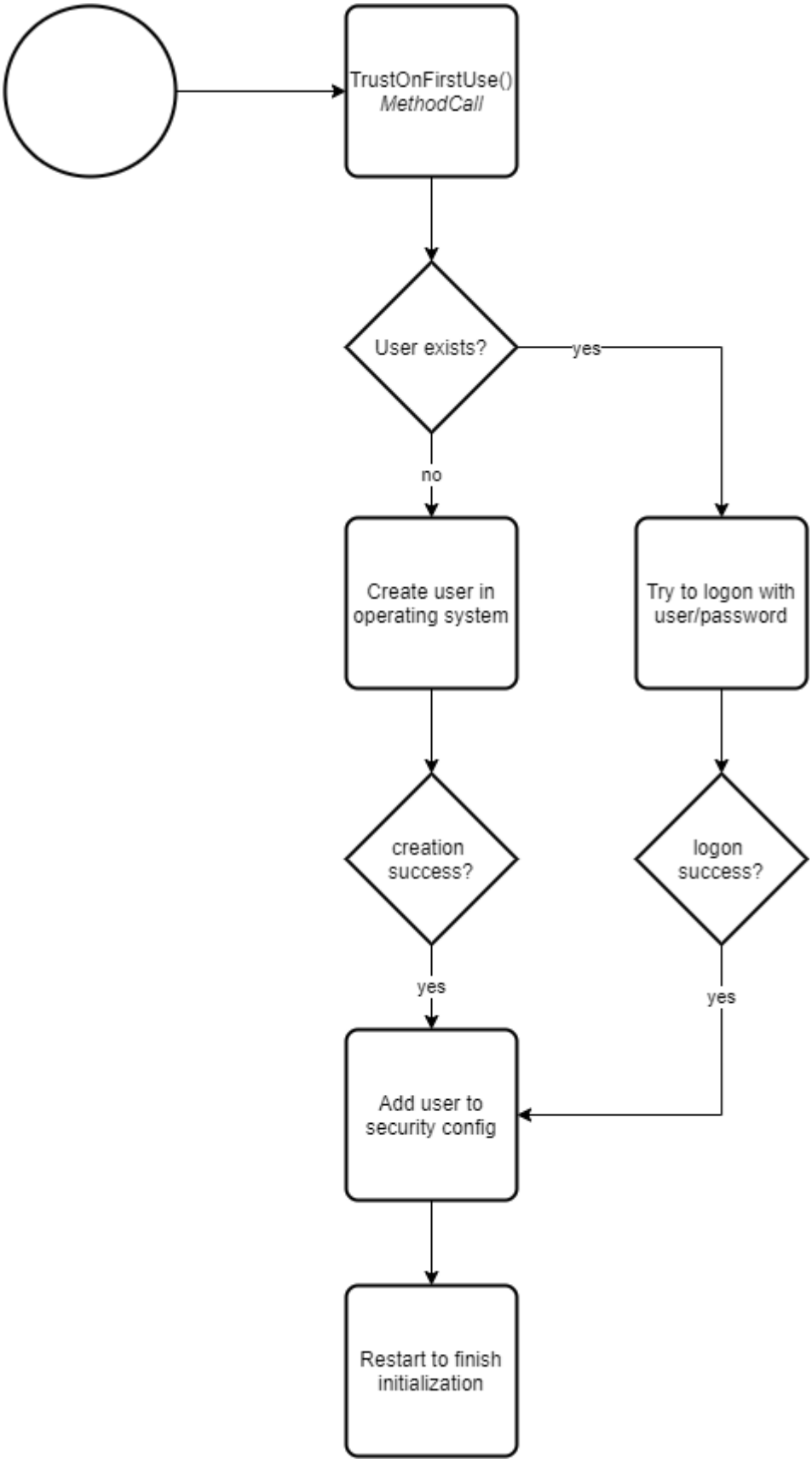
服务器通过调用此方法来完成初始化。该方法尝试在服务器的下级操作系统中创建用户指定的用户。如果创建成功，系统会将该用户自动添加至服务器的安全配置（TcUaSecurityConfig.xml）中，并将其定义为服务器管理员。方法调用结束后，服务器会自动重启，然后 OPC UA 客户端即可用该用户账户登录服务器。

如果操作系统中已存在指定的用户，将会通过输出参数（AddStatus）提示这一情况。在这种情况下，服务器会尝试使用指定的密码来登录操作系统。如果登录成功，系统会将该用户输入服务器的安全配置，服务器将自动重启，从而成功完成初始化。如果操作系统登录失败（例如，因为输入的密码不正确），将会通过输出参数（LogonResult）提示这一情况，初始化进程将终止。这样可以防止在初始化服务器时不小心使用了错误的用户名/密码组合，从而导致“将自己锁在系统之外”情况的发生。

用户密码过期

当 OPC UA 服务器创建操作系统用户时，并**不会**为该用户明确启用“密码永不过期”的设置。而是采用操作系统的设置，即以密码策略中定义的最长密码有效期为准。如果将最长密码有效期设为 0，表示密码永不过期；否则，密码将在操作系统规定的天数后过期。

下图以高度简化的形式再次阐释了该进程：

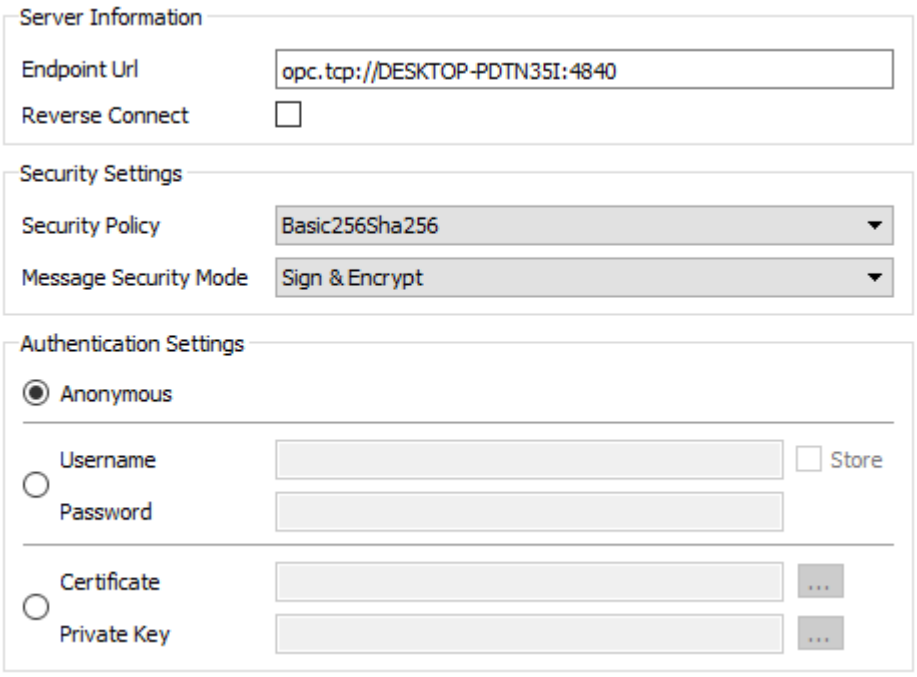


服务器重启后，OPC UA 客户端在建立连接时必须使用初始化时使用的操作系统用户进行身份验证。

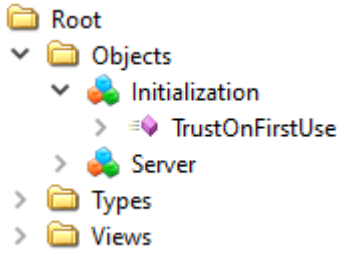
下方的屏幕截图以 OPC UA 客户端“UA Expert”为例展示了整个过程。在本例中，我们假设用户在操作系统中尚不存在，因此由服务器进行创建。

第 1 步：OPC UA 客户端首次连接服务器

服务器已安装完毕，UA Expert 首次与服务器连接。该连接仍可使用匿名访问。

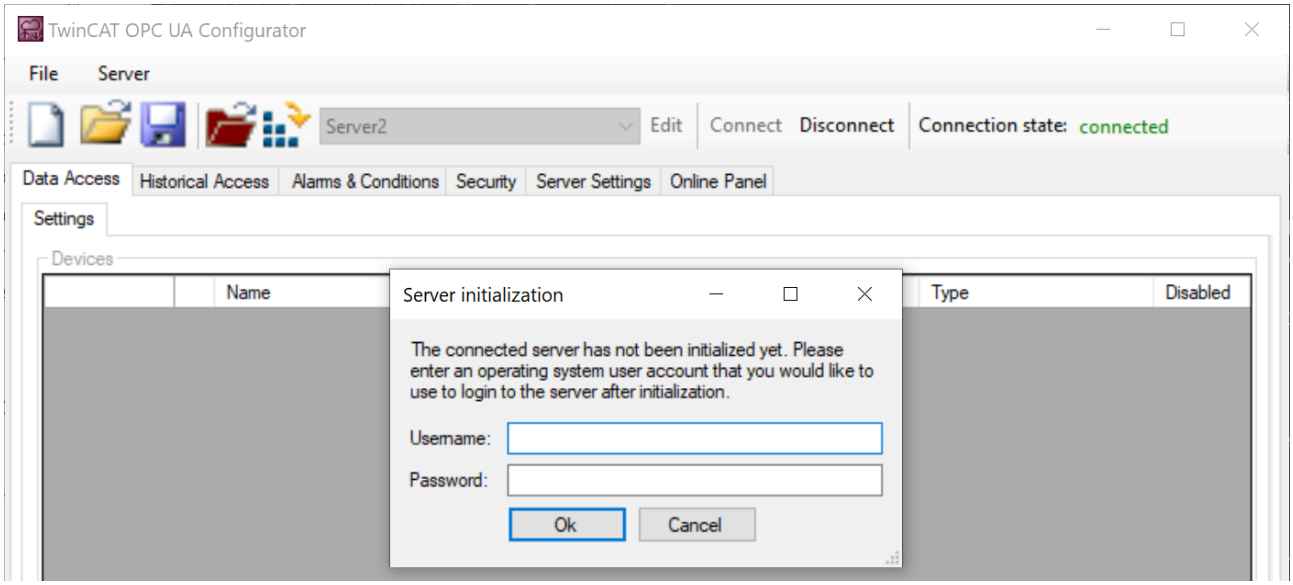


建立连接后，即可在服务器的地址空间中查找初始化对象和 TrustOnFirstUse 方法。



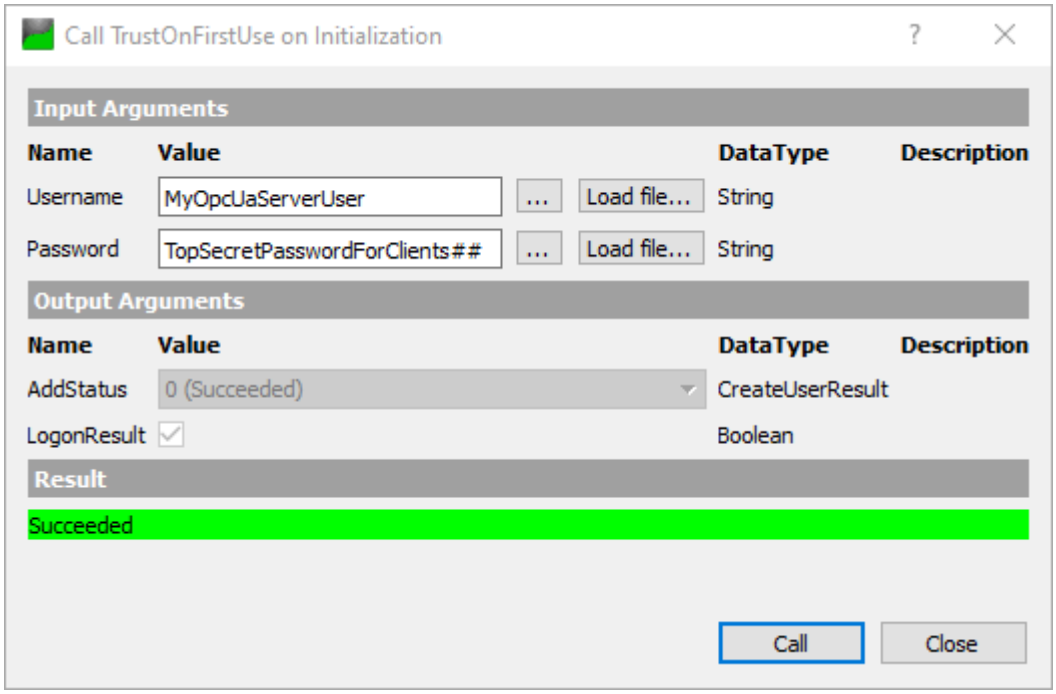
第 2 步：OPC UA 客户端启动 TrustOnFirstUse

可通过任何 OPC UA 客户端（如 UA Expert）调用 TrustOnFirstUse 方法。不过，倍福自有的配置工具也支持使用该初始化接口。建立连接时，TwinCAT OPC UA Configurator（独立版本或与 Visual Studio 集成的版本）会自动检测到未初始化的服务器，并支持通过相应的配置接口进行初始化：



以下步骤以 UA Expert 软件为例，展示了如何手动完成同一进程：

在 UA Expert 中，系统会调用 TrustOnFirstUse 方法来创建用户并为该用户配置服务器。本例中使用的用户名为“MyOpcUaServerUser”。密码必须符合操作系统的复杂度要求，否则初始化将失败。下方的屏幕截图展示了该方法调用成功的情况。



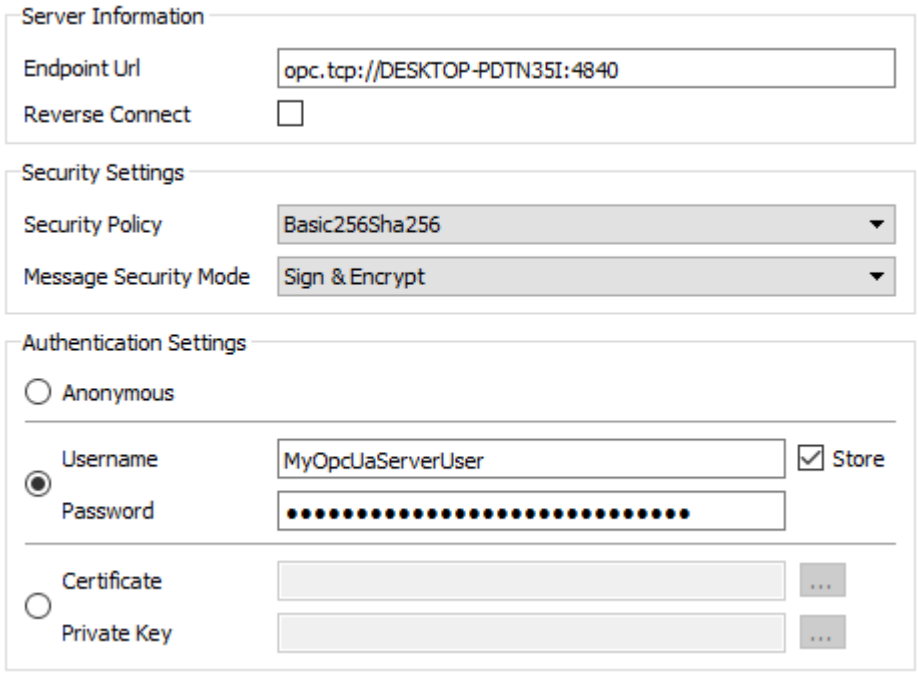
参数“AddStatus”表示已在操作系统的用户数据库中成功创建用户。参数“LogonResult”表示已成功通过指定用户信息对服务器进行初始测试身份验证。

方法调用成功后，服务器会自动重启。

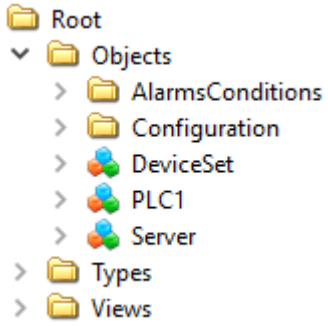
第 3 步：OPC UA 客户端登录初始化完成的服务器

用户名/密码禁用匿名访问

请注意，在方法调用完成后，UA Expert 无法自动重新连接服务器，因为匿名访问已被禁用，此后必须使用指定的用户名进行登录。



建立连接后，即可在服务器的地址空间中重新找到常规命名空间和对象，并开始进行应用程序配置。



TOFU 用户的权限

通过 TOFU 机制配置的用户拥有对服务器的完全访问权限，但这可能并不可取。因此，Beckhoff 建议在下一步中为纯数据访问创建一个明确的用户，请参见 。

在用户数据库中手动添加 TOFU 用户

在某些系统配置下，TwinCAT OPC UA Server 可能没有足够的授权，无法在初始化过程中自动将指定用户添加至用户数据库中。

例如，在 Microsoft Windows 下的 TwinCAT UMRT 独立系统上使用服务器版本 5.4 时，就会出现这种情况。在这种环境下，TwinCAT OPC UA 服务器以受限的系统权限执行。这意味着它不具备对操作系统用户数据库的写入权限，从而无法自动创建用户账户。

程序

为确保所配置的用户可以正常使用，必须在操作系统中手动创建该用户，然后再对服务器进行初始化。

例如，可以通过 Windows 计算机管理进行操作：

- 打开计算机管理器
- 导航至“**Local users and groups**（本地用户和群组）”
- 创建用户。该用户无需拥有任何其他授权，也无需加入用户组。

只有在用户账户创建成功后，才能对 TwinCAT OPC UA Server 进行初始化。在初始化过程中，您将收到一条提示“Access denied (not root)（拒绝访问（非 root 用户））”的错误消息，但使用创建的用户将可以成功进行登录（LogonResult），进而完成初始化。

4.4 建议的步骤

初次调试完成后，倍福建议您注意以下几点，以便进一步配置服务器并确保运行环境稳定、安全。

数据访问

数据访问是 OPC UA 的一个功能，用于在服务器地址空间中显示符号和相应的符号访问权限。在 TwinCAT OPC UA Server 中，配置数据访问设备是一个基本环节，也是进一步使用各项功能的前提。因此，我们建议您接下来阅读关于数字访问 [▶ 48]的章节，其中还介绍了如何与运行时连接 [▶ 49]。

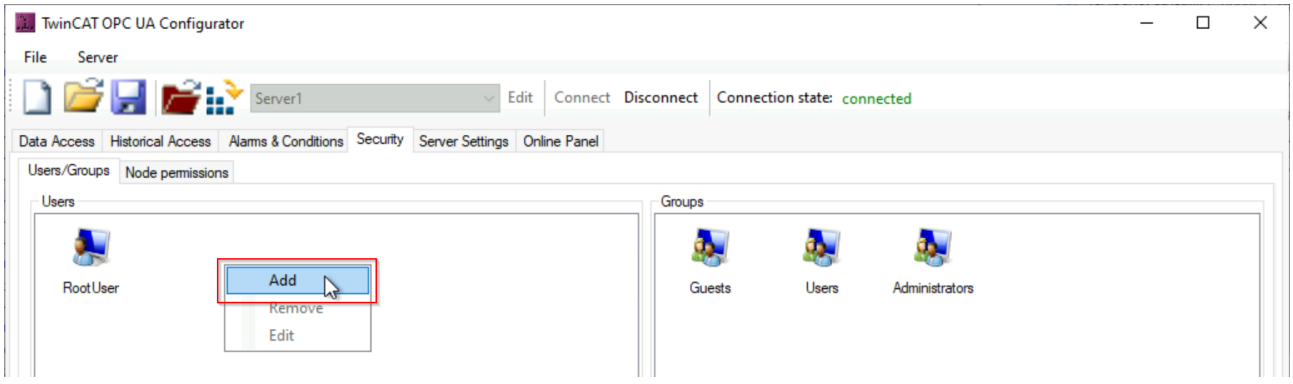
只使用安全的身份代币

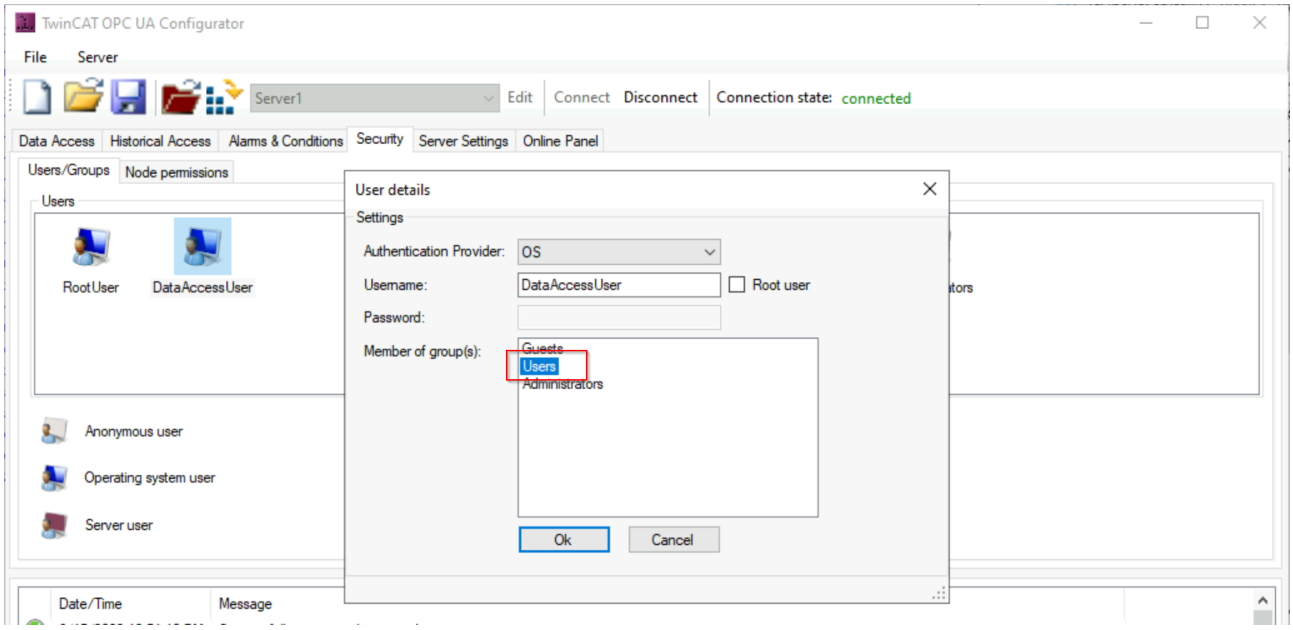
对服务器进行一次性初始化时会禁用 IdentityToken“Anonymous（匿名）”。**出于安全考虑，您应将其禁用。**只有经过身份验证的客户端应用程序才能访问服务器，如初始化时默认配置的用户名/密码身份验证。

创建纯数据访问用户

上文所述的服务器初始化会配置一个用于访问服务器的用户，随后会禁用服务器匿名访问功能。已配置的用户拥有对服务器命名空间中所有对象的完全访问权限。在大多数应用场景中，我们并不希望出现这种情况，应将管理员用户与应用程序用户分开。

因此，倍福建议配置一个额外的专用用户，赋予其访问数据访问设备上任意变量的必要权限，但不允许其访问配置命名空间。可以通过配置器进行此设置，具体方法为：添加一个新用户并将其分配至“Users（用户）”群组中。





这样，新配置的用户即拥有访问 TwinCAT 变量、读取类型系统的所有必要权限，但不具备更改服务器配置的权限。请注意，如果使用身份验证提供程序“OS”，也必须在操作系统中创建用户，即用户必须存在于操作系统中。

注意

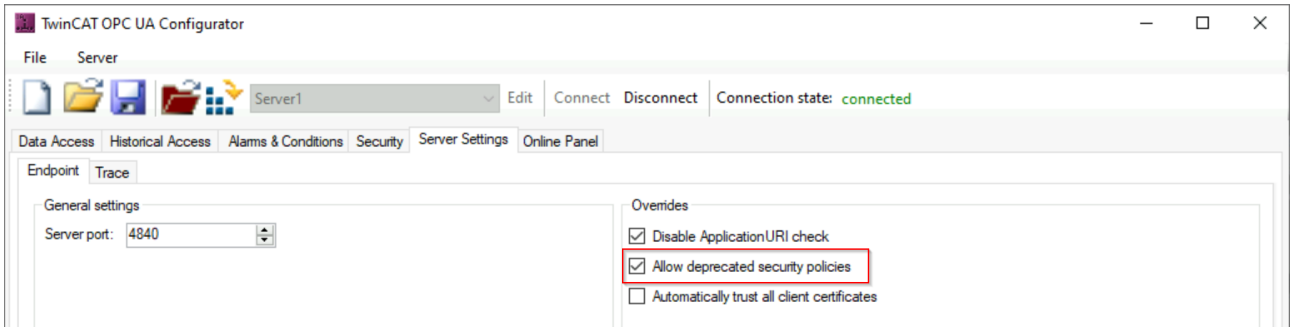
将不安全端点保持禁用状态

✓ TwinCAT OPC UA Server 默认不提供列为不安全的端点。可以通过配置开关在服务器中使用此类端点，但倍福不建议这样做！

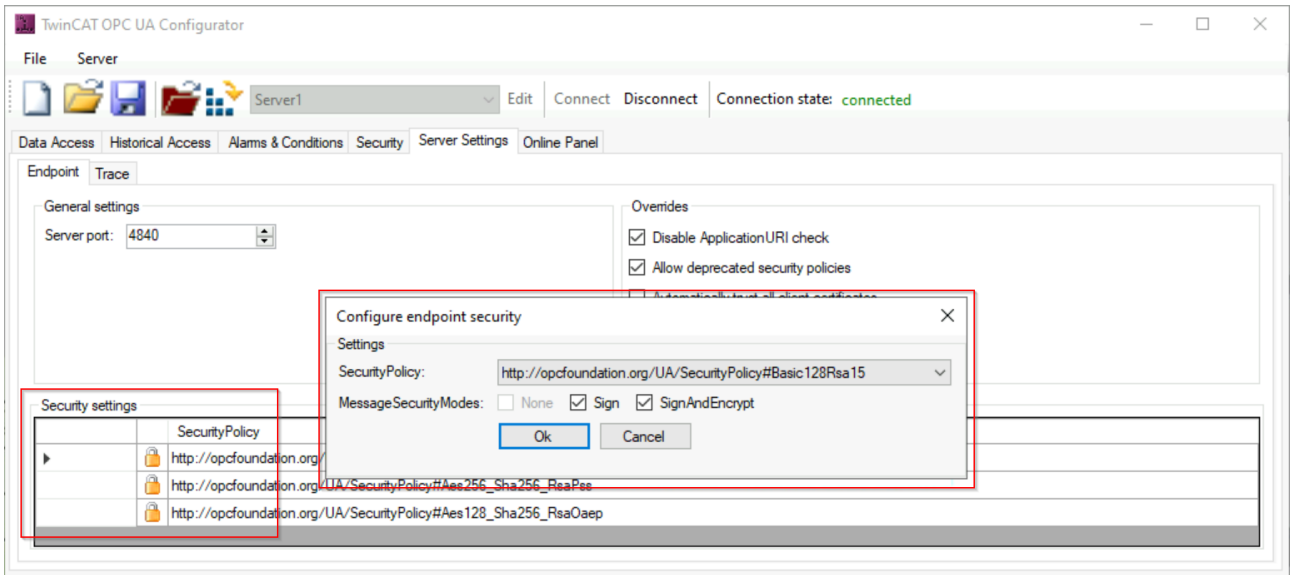
a) 只能使用目前公认安全的端点。

b) 遵守并遵循下一节中与安全相关的其他建议。

下方的屏幕截图展示了如何在 TwinCAT OPC UA Configurator 中启用旧版服务器端点。



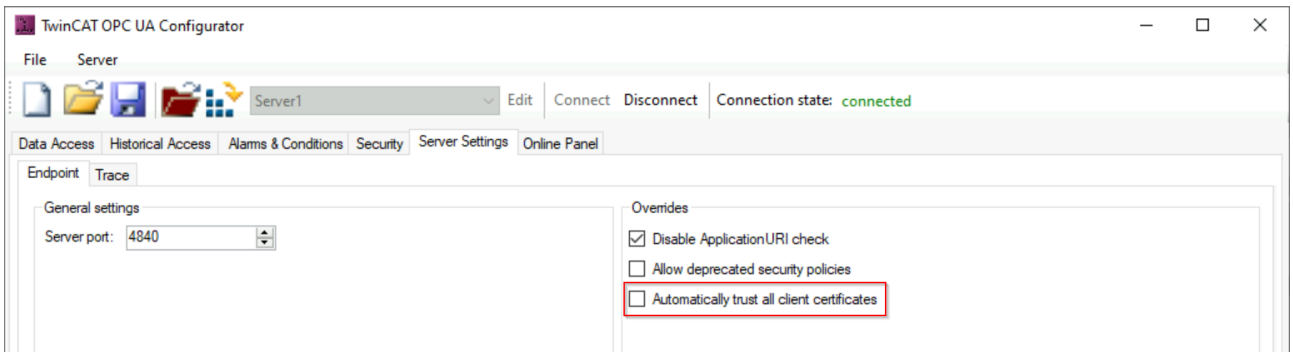
然后，您可以将不安全的端点重新添加到服务器配置中，例如，通过该配置器中“Security Settings（安全设置）”区域的上下文菜单：



服务器在交付时已经禁用 None/None 端点。出于安全考虑，倍福建议您也禁用该端点，仅允许通过安全端点访问服务器。如果需要，可以使用上述方法将 None/None 端点重新添加到服务器配置中。

禁用“AutomaticallyTrustAllClientCertificates”

默认情况下，服务器是为方便调试而配置的，因此它会自动信任所有客户端证书，无需在服务器端手动交换证书。出于安全考虑，倍福建议禁用此设置。可通过 TwinCAT OPC UA Configurator 进行此设置，如下方的屏幕截图所示：



禁用此设置后，必须在客户端和服务器之间建立信任关系，即这两个应用程序相互信任对方的证书。

4.5 支持的功能

OPC UA 具备众多功能。本产品在初始阶段可能无法支持所有功能。随着时间的推移，我们将通过产品更新逐步增加更多功能。下表显示了当前产品版本目前支持的所有功能。

与 OPC UA Pub/Sub 规范兼容

本产品的当前版本与第 1.05.03 版 OPC UA Pub/Sub 规范兼容。

认证配置文件

本产品的当前版本已通过“标准 UA 服务器”配置文件的认证。本认证中涉及的符合性单元由 OPC 基金会记录在文件中。

规范

OPC UA 规范分为若干部分。第 1 至第 5 部分规定了 OPC UA 的核心功能。下表概述了该规范的各个部分，并说明了本产品是否提供相应的支持。如有相关情况，您可以在下文中查看各部分所支持功能的详细信息。请注意，并非所有部分均受到支持，因为该规范的某些部分超出了产品定义的范围。

部分	支持
OPC10000-1 概述和概念	是
OPC10000-2 安全模型	是
OPC10000-3 地址空间模型	是
OPC10000-4 服务	是
OPC10000-5 信息模型	是
OPC10000-6 映射	是
OPC10000-7 配置文件	是
OPC10000-8 数据访问	是
OPC10000-9 报警与条件	是
OPC10000-10 程序	否
OPC10000-11 历史访问	是
OPC10000-12 查找和全局服务	是
OPC10000-13 骨料	否
OPC10000-14 Pub/Sub	否 *1
OPC10000-15 安全	否
OPC10000-16 状态机	否
OPC10000-17 别名	否
OPC10000-18 基于角色的安全性	否 *2
OPC10000-19 字典引用	否

部分	支持
OPC10000-20 文件传输	是
OPC10000-21 设备上线	否
OPC10000-22 基本网络型号	---
OPC10000-23 常见引用类型	---
OPC10000-24 调度程序	否

*1：OPC UA Pub/Sub 由产品 TF6105 TC3 OPC UA Pub/Sub 进行映射

*2：本产品包含其特有的用户权限执行机制，该机制在第 18 部分发布之前即已存在，并采用基于用户/群组的授权系统 [▶ 123]。

OPC10000-4 服务

下表概述了 OPC UA 规范第 4 部分所述的本产品支持的服务（服务集、服务性能等）。

功能	支持
Attributes Read/Write	是
Attributes HistoryRead	是（值属性）
Attributes HistoryUpdate	是（值属性）
订阅	是
Durable Subscriptions	否
Monitored items	是
Method Calls	是
Auditing	否
Redundancy	否

OPC10000-8 数据访问

下表概述了 OPC UA 规范第 8 部分所述的本产品支持的数据访问功能。

功能	支持
数据项类型	是
模拟项目类型	是
离散项目类型	是 *1
数组项目类型	是
欧盟设备准入合规信息	是 *1
复数类型	是 *1
双复数类型	是 *1
轴信息	是 *1
轴刻度枚举	是 *1
XVType	是 *1
对象类型	是（功能块）
结构化类型	是
ServerTimestamp	是
SourceTimestamp	是
Quality	是

*1：通过节点集导入和 TE6100 OPC UA 节点集编辑器支持。

OPC10000-9 报警与条件

下表概述了 OPC UA 规范第 9 部分所述本产品支持的警报和条件功能。

功能	支持
限制警报类型	是
OffNormalAlarmType	是
多语言警报/事件文本	是
TwinCAT 3 事件记录器	是 *1
自定义事件类型	是 *2
Custom AlarmTypes	是 *2

*1：通过专有的 Alarm/EventTypes 支持 TwinCAT 3 EventLogger

*2：通过节点集导入和 TE6100 OPC UA Nodeset Editor 提供支持。

OPC10000-11 历史访问

下表概述了 OPC UA 规范第 11 部分所述的本产品支持的历史访问功能。

功能	支持
HistoricalDataNodes [▶ 89]	是
历史事件节点	否
历史审计事件	否
HistoryUpdate [▶ 92]	是
外部历史资料来源	否

OPC10000-20 文件传输

本产品支持在 OPC UA 端交换用于调试产品的配置文件。我们在倍福设备管理器软件的 OPC UA 服务器中实现了通用文件传输 [▶ 126]功能。

安全

下表概述了本产品支持的安全功能。此处显示的某些功能分布在规范的多个部分中。

功能	支持
SecureChannel	是
SecureEndpoints [▶ 118]	是
Application Instance Certificate	是
Certificate Trust Relationship	是
IdentityToken Anonymous	是
IdentityToken Username/Password	是
IdentityToken user certificates	是
User rights at Namespace level	是
User rights at Node level	是

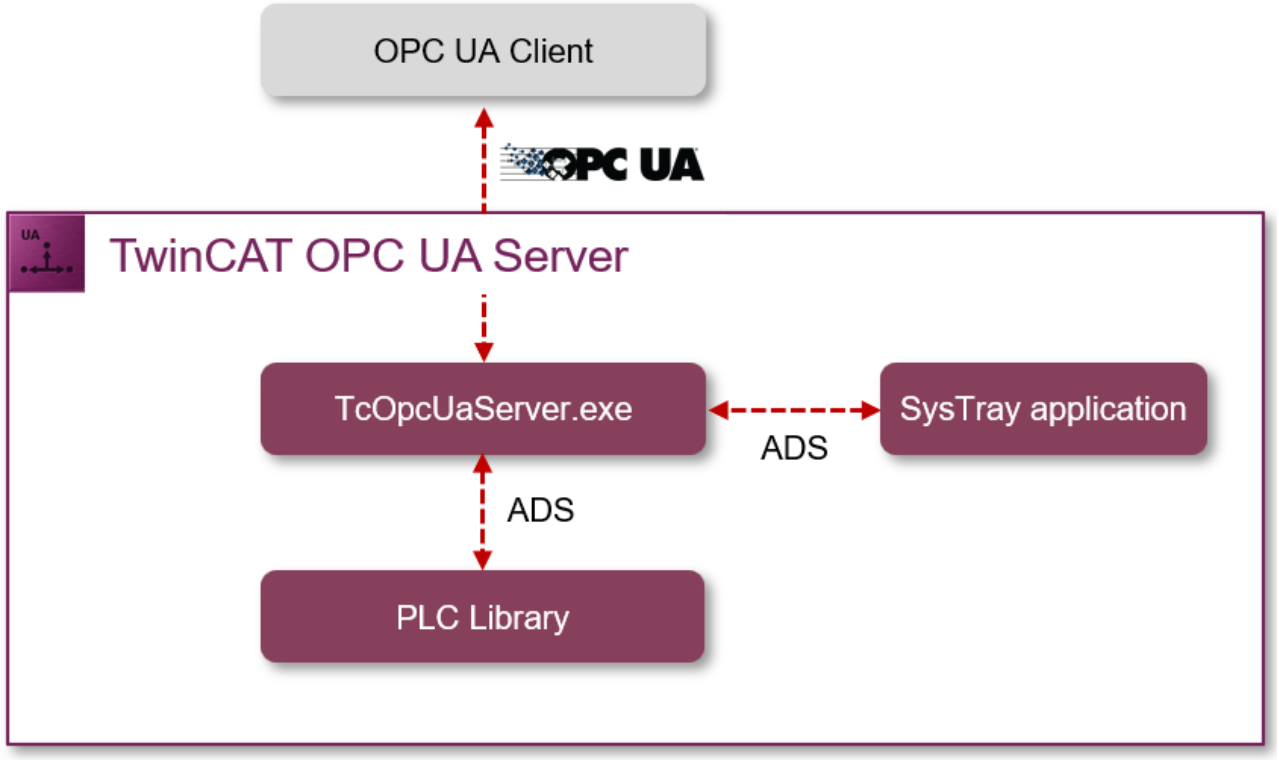
4.6 软件架构

您不需要了解本产品的内部软件架构就可以使用它，但在某些情况下可能会对它感兴趣。因此，我们将在下文向您简要介绍这些情况。

TwinCAT OPC UA Server 主要由以下组件组成：

- 操作系统中的进程
- Windows System Tray [▶ 135] 中的应用程序
- PLC 库 Tc2_OpcUa [▶ 137]

下图详细介绍了各个组件之间的相互作用：



操作系统中的进程

操作系统中的进程（TcOpcUaServer.exe）负责处理 OPC UA 协议功能（即与 OPC UA 客户端通信）、提供 OPC UA 地址空间以及与下级设备 [▶ 48]通信。此外，该进程还提供了一个 ADS 服务器接口，以便 PLC 库 Tc2_OpcUa 和 Windows System Tray 应用程序能够与服务器应用程序进行交互。

Windows 系统托盘应用程序

该应用程序可触发 TwinCAT OPC UA Server 重启。可通过 Windows System Tray 中的图标调用相应的功能。

PLC 库

PLC 库 Tc2_OpcUa 可与 TwinCAT OPC UA Server 进行交互，用于检索状态信息和触发重启。

4.7 配置器

作为 TwinCAT OPC UA Configurator 的一部分，有 2 个图形用户界面可用于简单配置 TwinCAT OPC UA Server：一个是集成到 Visual Studio（或 XAE Shell）中的界面，另一个是一款独立工具。

TwinCAT OPC UA 配置器文档

尽管本文档的多个章节均讨论了 TwinCAT OPC UA Configurator，但该组件也有专属的产品文档，您可以在倍福信息系统中查阅这些文档。

通过 OPC UA 进行配置

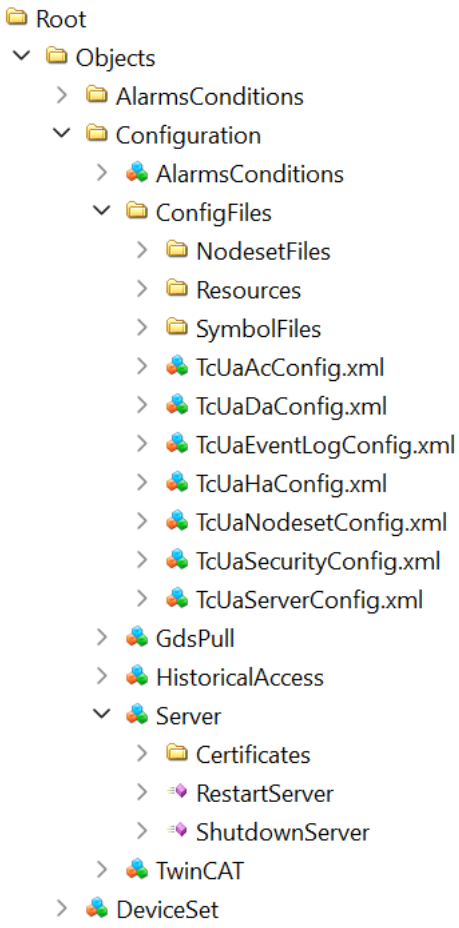
TwinCAT OPC UA Server 包含一个所谓的配置命名空间，可通过 OPC UA 对服务器进行本地和远程配置。TwinCAT OPC UA Configurator 利用该接口访问服务器的各个配置文件。

配置命名空间中涵盖了以下功能：

- 管理服务器配置文件
- 管理证书

- 重新启动服务器

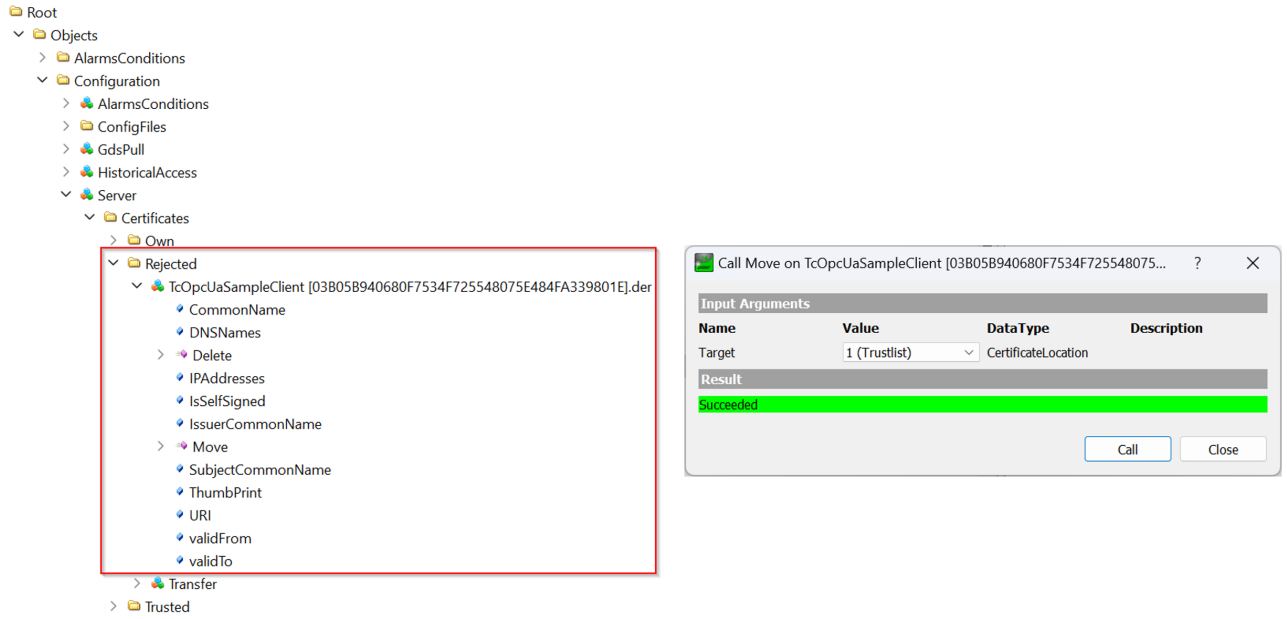
初始化 [▶ 29]服务器后，在此配置的用户可访问配置命名空间，并可用于进一步配置服务器。下面的截图从 OPC UA 客户端（本例中为 Unified Automation 的 UA Expert）的角度显示了配置命名空间：



所有服务器配置文件均以 FileType 型对象的形式提供。该对象类型及其访问由 OPC UA 标准化。在配置命名空间中以 CertificateType 形式创建证书文件。

证书管理

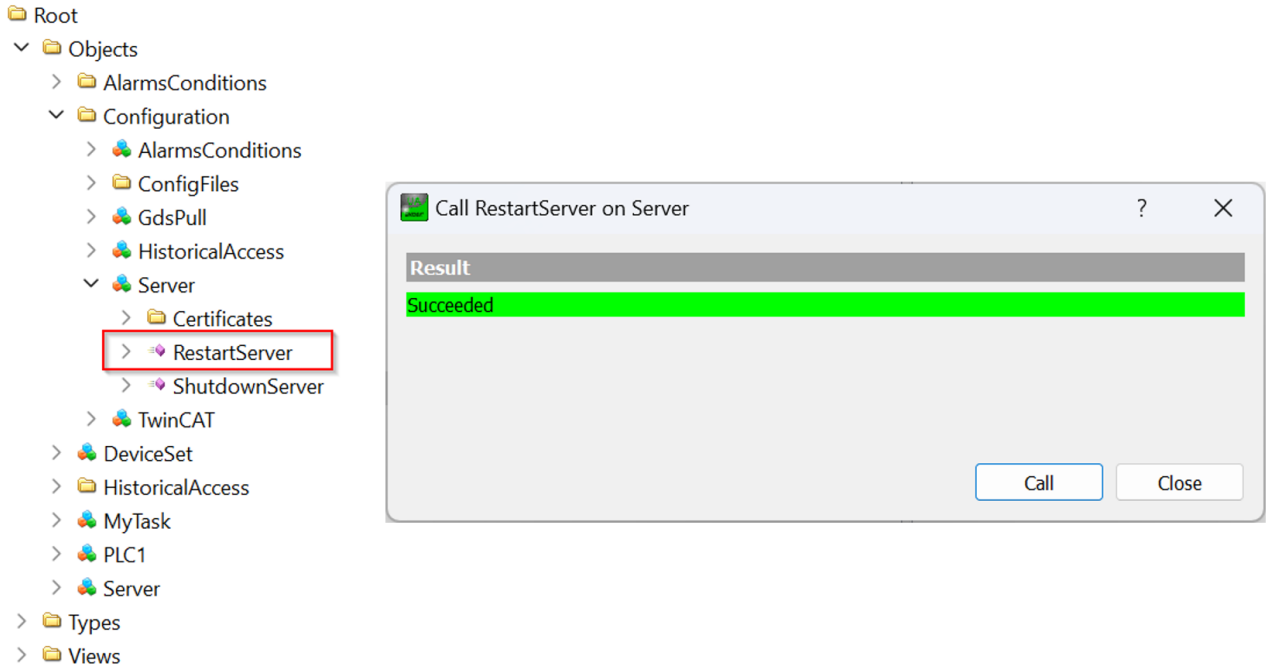
客户端证书分为“被拒绝”证书和“受信任”证书，这两种证书在命名空间中分别以不同的文件夹表示。在证书对象上调用`Move()`方法，可在受信任列表之间移动证书。此外，各种属性还提供了有关证书本身的更多信息，更加便于进行证书识别。



TwinCAT OPC UA Configurator 为您提供了一个用户界面，为您信任/拒绝证书提供了更大的便利。更多信息请参见“证书交换 [▶ 119]”一章。

重新启动服务器

配置命名空间包含一种方法，通过该方法，您无需重启 TwinCAT 即可重启 TwinCAT OPC UA Server。



4.8 优化

有多种方法可以优化 OPC UA 客户端与服务器或 PLC 之间的通信连接。此外，还可以通过各种参数来优化 TwinCAT OPC UA Server 的运行时性能，特别是 CPU 和内存负载。本章将对各种优化方案进行介绍，特别涵盖以下几个主题：

- SamplingInterval vs. 出版间隔
- 结构化类型
- StructuredTypes 及其成员变量

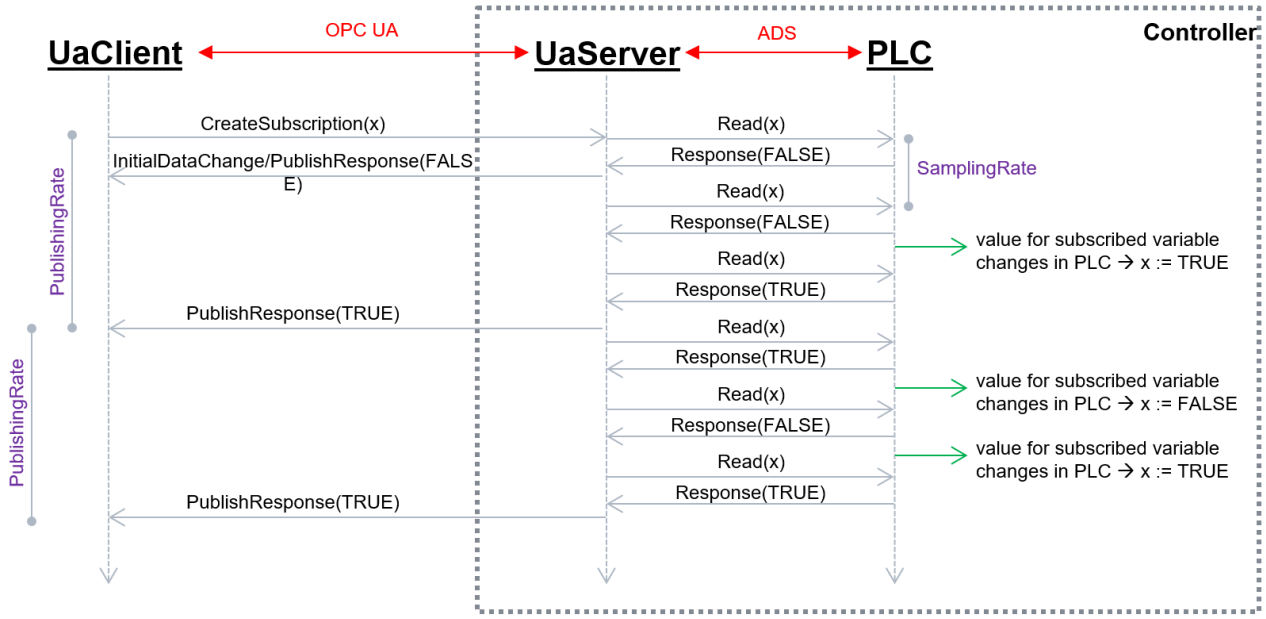
此处显示的屏幕截图和性能值是在实验室条件下于不同硬件设备上运行的示例。因此，无法将其 1:1 传输至客户项目中，仅用于说明某些事实情况。

SamplingInterval vs. 出版间隔

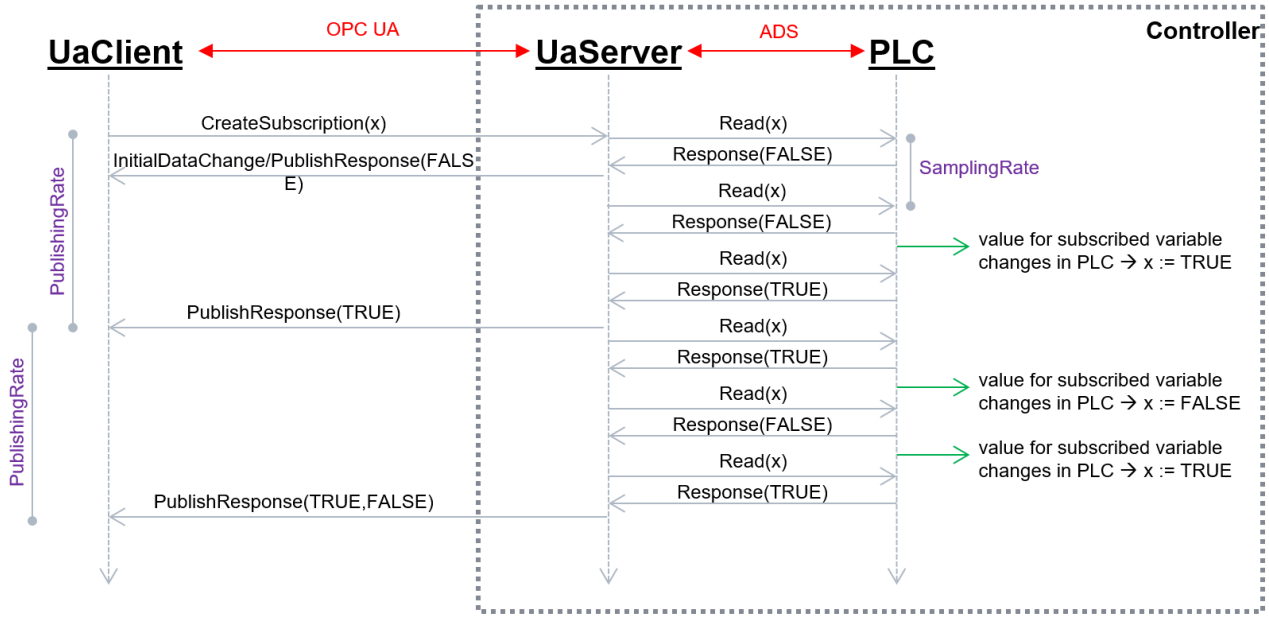
在创建订阅时，OPC UA 客户端会使用订阅的各种参数以及订阅中包含的所谓的 MonitoredItems 来接收有关变量更改的通知。下表对其中的 2 个参数进行了说明，我们将在后续章节中对其进行更详细的说明。

参数	描述
出版间隔	PublishingInterval 指定了服务器向 OPC UA 客户端发送值变化通知的速率。OPC UA 规范第 4 部分对 PublishingInterval 进行了详细说明。
采样间隔	SamplingInterval 指定了 OPC UA 服务器对其底层数据源的值变化进行采样的速率（若为 TwinCAT OPC UA Server，则通过 ADS 连接进行采样）。OPC UA 规范第 4 部分对 SamplingInterval 进行了详细说明。

下图再次阐释了这两个参数之间的关系。此时，我们假设 PLC 控制器上已安装 TwinCAT OPC UA Server，并且 OPC UA 客户端从外部系统访问服务器。



如图所示，例如，如果 PLC 中的值变化发生得“太快”，采样率不够高，或者变量值返回到原始值（如上图所示），可能会出现 OPC UA 客户端未发现 PLC 中某些值变化的情况。通过另一个参数，即所谓的 QueueSize（队列大小），可以记录发布间隔（PublishingIntervals）之间的若干数值变化，并将其传输到 OPC UA 客户端。在上例中，选择的 QueueSize 为 1，即 "最后已知值 "始终传输给客户端。而在下图中，选择的 QueueSize 为 2，即向客户端传输最后 2 个已知值的变化。

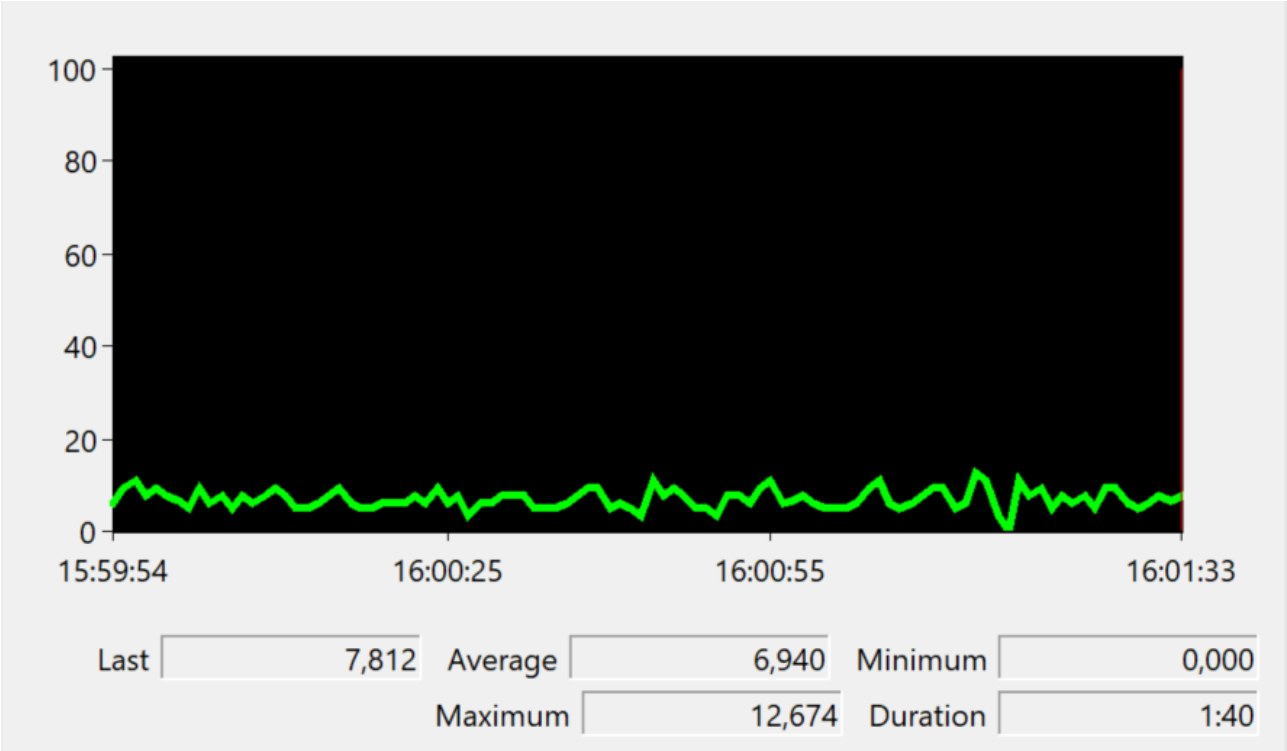


从第一个 PublishResponse 可以看出，只有 1 个值变化被传输至客户端，因为 PLC 中也只发生了 1 个值变化。从第二个 PublishResponse 可以看出，PLC 中发生了两次数值变化，这两次变化也都传送到了客户端。

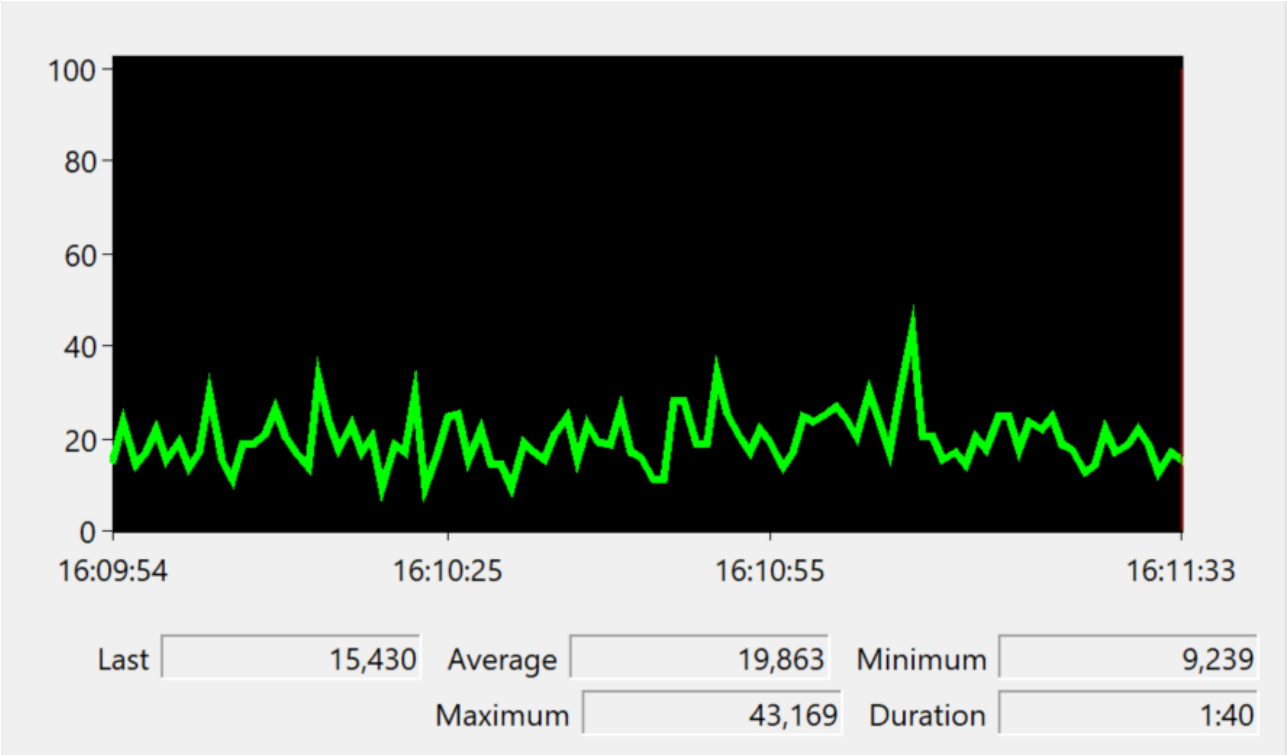
上述参数均为通常由客户端控制并请求服务器提供的设置。正是在这些参数上，存在着巨大的优化空间，因为这两个参数对 OPC UA 客户端和服务器所需的 CPU 时间有着显著影响，影响程度取决于需要处理或请求的相应的信息量。

应根据具体的应用场景合理设置这两个参数。例如，如果 OPC UA 客户端是可视化设备，那么，设置过快的 PublishingIntervals 和 SamplingRates 意义有限，因为人眼处理信息的速度无论如何都不会超过 200 ms 左右。此外，还应根据具体情况合理选择是否使用 QueueSize。如果 OPC UA 客户端并不处理队列中的任何值，那么，将 QueueSize 设为 1 既足够又合理，因为需要传输的信息相应地减少，从而进一步优化了系统。

在下面的示例中，一个 OPC UA 客户端在 TwinCAT OPC UA 服务器上创建了一个包含 10,000 个任意变量的订阅。其中，PublishingInterval 设置为 500 ms，SamplingInterval 设置为 250 ms。TwinCAT OPC UA Server 的 CPU 负载平均值约为 6.9%，请参见下方的 Windows Performance Monitor 屏幕截图。



然后将 PublishingInterval 设置为 200 ms，将 SamplingInterval 设置为 100 ms。这样，TwinCAT OPC UA Server 的 CPU 负载平均值增加了约 19%。



结构化类型

默认情况下，TwinCAT OPC UA Server 在其地址空间中以 FolderType 的形式提供结构 [▶ 74]。然后，成员变量就会以单独节点的形式显示在文件夹下方，并可进行访问。该服务器也支持结构使用 StructuredTypes，在这种模式下，客户端会处理结构的类型信息，并且可以通过确保数据一致性的方式对该结构进行读取/写入操作。在与 PLC 通信时（通过 TwinCAT ADS 协议），这两种模式的表现存在本质区别。

当 OPC UA 客户端访问各个成员变量时，可能会根据变量的数量将 ADS 通信拆分为多个 ADS 读取/写入命令。这反过来又会导致 PLC 在不同的 PLC 循环中处理 ADS 命令。因此无法保证数据的一致性。

另一方面，当 OPC UA 客户端访问 StructuredType 时，可确保 PLC 在单个 ADS 读取/写入命令中处理 StructuredType，从而确保数据的一致性。

在对大型 PLC 数据结构使用 StructuredTypes 时，应考虑以下几点：

- 根据结构的大小，ADS 读取/写入命令或相应的响应可能会变得非常大，从而需要相应占用 TwinCAT ADS 路由器中的大量内存。**因此，请注意路由器的内存。**
- 与 TwinCAT 实时环境进行通信时，TwinCAT OPC UA Server 必须对 StructuredTypes 进行编码或解码，此操作需要额外的 CPU 处理时间。
- 读取/写入命令可能需要很长时间，具体取决于结构的大小，同时也在很大程度上取决于服务器进行编码/解码所需的时间。

由于这些原因，在出厂状态下，我们对服务器的 StructuredType 的最大占用空间进行了限制，但可以根据需要使用配置文件 TcUaDaConfig.xml 中的参数 <MaxStructureSize> 对其进行调整。该参数以字节为单位指定结构体的最大大小。如果结构超过 <MaxStructureSize>，系统会将其作为 FolderType 进行导入，其中，每个结构元素均可作为独立的节点使用。有关更多信息，请参见“结构 [▶ 74]”一章。

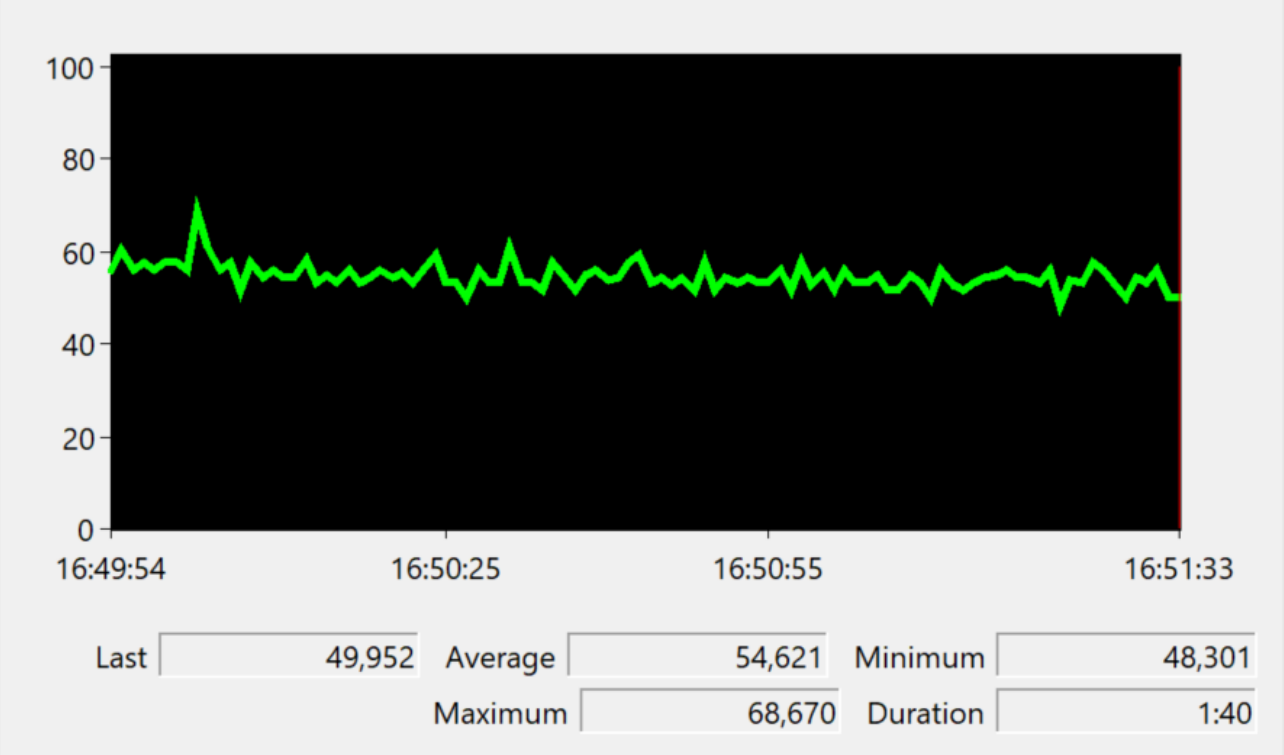
特别是在为某个订阅设置 SamplingInterval 和 PublishingInterval 参数时，可以通过 StructuredType 进行一些优化，这样可能会对服务器的运行时性能产生重大影响。

在下方的示例中，一个 OPC UA 客户端对 StructuredType 创建了一个订阅。下级 PLC 数据结构如下：

```
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
```

作为成员变量的 ST_Complex_1 结构大小约为 91 字节。这是一个总大小约为 1 MB 的数据结构。其中，PublishingInterval 设置为 500 ms，SamplingInterval 设置为 250 ms。创建订阅后，TwinCAT OPC UA Server 的 CPU 负载平均值约为 54.6%，请参见下方的 Windows Performance Monitor 屏幕截图。



根据所选的 SamplingInterval，下级 ADS 通信每 250 ms 发送一次 ADS 读取命令。在相应的响应中，您可以清楚地看到，数据结构的大小约为 1 MB，即每 250 ms 通过 TwinCAT ADS 路由器传输一个 1 MB 的数据包（并且服务器必须进行相应的处理）。服务器的编码器/解码过程需要大量的 CPU 能力，这就是 CPU 负荷高的原因。

```
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 及其成员变量

默认情况下，StructuredType 的成员变量在服务器地址空间中以独立节点的形式来表示和使用。这需要占用额外的主存，因为 TwinCAT OPC UA Server 会为每个节点分配主存。只与结构类型（即结构的 "根元素"）一起工作的 OPC UA 客户端不需要这些附加节点。可以使用特殊的编译指示显式隐藏这些节点，从而降低服务器的内存负载。有关编译指示的详细说明，请参见“结构 [▶ 74]”一章。

4.9 应用程序目录

该应用程序使用各种目录来存储相关信息，例如配置或证书文件。

安装目录

在所有操作系统上，应用程序的基本安装目录均始终与 TwinCAT 安装目录相关。

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

然后，应用程序即安装在该目录下的以下目录中。此目录会因平台（x86/x64）而异。从 TwinCAT Build 4026 开始，将默认在系统上注册并启动 64 位版本的服务器。

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

PKI 基础设施的基本目录

在所有操作系统上，用于建立安全连接的证书文件均存储在以下目录中：

```
%ProgramData%\Beckhoff\TF6100-OPC-UA\TcOpcUaServer\PKI
```

受信任证书目录

声明称，该目录中的客户端证书为“受信任”证书。该路径在所有操作系统上均相同。

```
%ProgramData%\Beckhoff\TF6100-OPC-UA\TcOpcUaServer\PKI\CA\trusted
```

被拒绝证书目录

声明称，该目录中的客户端证书为“被拒绝”证书。该路径在所有操作系统上均相同。

```
%ProgramData%\Beckhoff\TF6100-OPC-UA\TcOpcUaServer\PKI\CA\rejected
```

服务器证书目录

服务器证书目录的定义如下，其中，公钥目录（“certs”）与私钥目录（“private”）之间存在区别。

```
%ProgramData%\Beckhoff\TF6100-OPC-UA\TcOpcUaServer\PKI\CA\own\certs
```

日志文件

日志文件存储在以下目录中。不同操作系统的文件存储目录也有所不同。

```
%ProgramData%\Beckhoff\TF6100-OPC-UA\TcOpcUaServer (Windows)
/var/log/TF6100-OPC-UA-Server (TwinCAT/BSD)
```

配置文件

TwinCAT OPC UA Server 使用各种配置文件来配置各项功能，具体定义如下表所示。

文件	描述
TcUaAcConfig.xml	服务器报警与条件 [▶ 96]（A&C）功能的配置文件。为 A&C 配置的节点将保存在此文件中。
TcUaDaConfig.xml	服务器数据访问 [▶ 48]（DA）功能的配置文件。在此文件中配置与下级 ADS 设备的连接通信。
TcUaGdsClientConfig.xml	用于配置 全局查找服务器 [▶ 113] 功能的配置文件。在此文件中配置与 GDS 的连接。
TcUaHaConfig.xml	用于配置服务器历史访问 [▶ 89]（HA）功能的配置文件。该文件可存储 HA 所用的数据存储器和为此目的配置的节点。
TcUaNodesetConfig.xml	用于导入服务器节点集 [▶ 68]的配置文件。在此文件中定义要导入的节点集及其与 ADS 设备的链接。
TcUaSecurityConfig.xml	用户、群组和访问控制列表（ACL）的配置文件，是访问权限 [▶ 123]和身份验证 [▶ 121]机制定义的一部分。
TcUaServerConfig.xml	服务器各种参数的配置文件，例如端点 [▶ 118]、日志记录 [▶ 134]、反向连接 [▶ 127]的配置。

默认情况下，所有配置文件均存储在 ProgramData 目录中。通常不通过直接编辑 XML 文件来进行访问，而是通过 TwinCAT OPC UA Configurator 进行访问。

```
%ProgramData%\Beckhoff\TF6100-OPC-UA\TcOpcUaServer
```

4.10 命名空间

在 OPC UA 中，我们将命名空间用于以符合逻辑的方式分配节点，并避免不同信息模型或制造商之间的命名冲突。唯一标识是通过 NamespaceUri 而非数字形式的 NamespaceIndex 来实现的。

NamespaceIndex 仅与命名空间在服务器 NamespaceArray 中的当前位置相对应。由于该分配方式可能会发生变化，因此客户端不应静态接受或永久保存 NamespaceIndex。相反，应使用相关的 NamespaceUri 从每个连接的 NamespaceArray 中再次确定所需的索引。

下表显示了 TwinCAT OPC UA Server 中 NamespaceArray 的一个示例。

Name	Value
⌄	String Array[9]
[0]	http://opcfoundation.org/UA/
[1]	urn:MichaelKn-NB04:BeckhoffAutomation:TcOpcUaServer:1
[2]	http://opcfoundation.org/UA/DI/
[3]	http://PLCopen.org/OpcUa/IEC61131-3/
[4]	urn:BeckhoffAutomation:Ua:PLC1
[5]	urn:BeckhoffAutomation:Ua:Types:GlobalTypes
[6]	urn:BeckhoffAutomation:Ua:HA
[7]	urn:BeckhoffAutomation:Ua:AC
[8]	http://Beckhoff.com/TwinCAT/TF6100/Server/Configuration

在 TwinCAT OPC UA Server 中，用户无法更改命名空间的顺序或自由定义 NamespaceIndex。

4.11 数据访问

4.11.1 概述

本章将介绍在 TwinCAT OPC UA Server 命名空间中配置用于**数据访问**（DA）的变量所需的步骤。

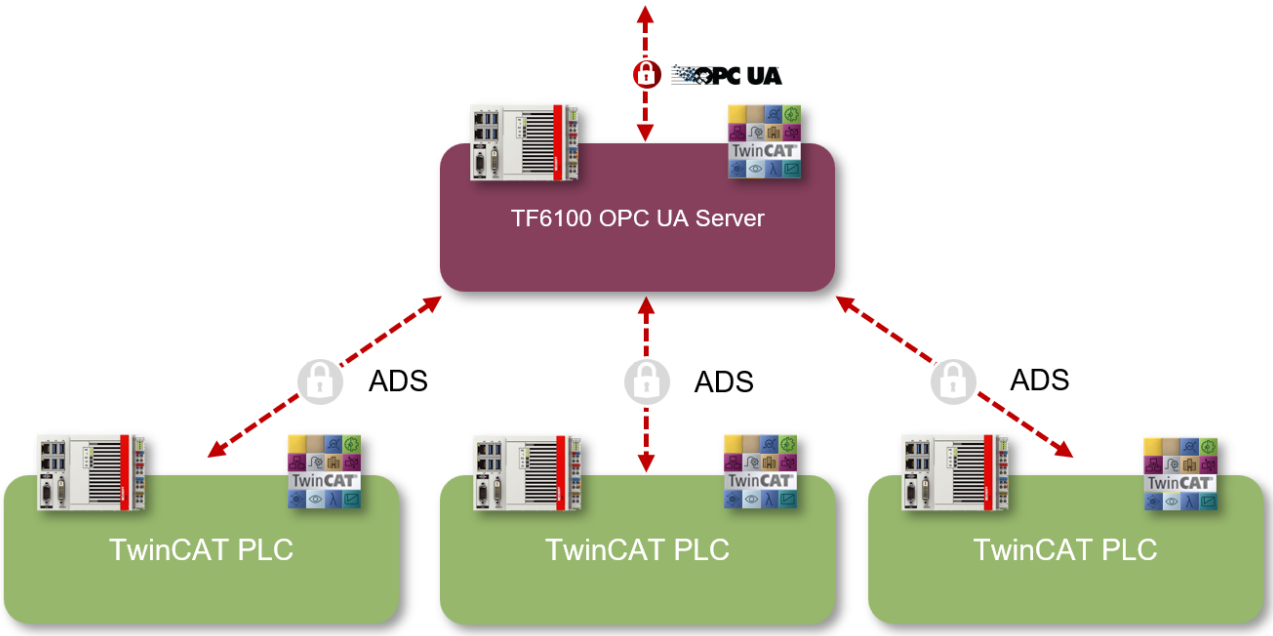
数据访问是一项 OPC UA 功能，用于描述变量值的表示方式和使用方法。例如，它是关于 OPC UA 客户端如何访问变量值的服务功能。

TwinCAT OPC UA Server 可以提供所有 TwinCAT 实时环境中的变量。例如，这些实时环境包括 TwinCAT PLC、TwinCAT 3 C++、TwinCAT 3 MATLAB®/Simulink® 或 TwinCAT I/O 任务。该服务器可以访问多个实时环境，并在其地址空间中提供这些环境的符号信息。

为实现更多功能奠定基础

为实现数据访问功能而需执行的“与运行时连接 [▶ 49]”和“启用符号 [▶ 52]”这两个操作为实现历史访问及报警与条件等其他功能奠定了基础。
请确保您熟悉此处所需的设置。

不同的实时环境不一定要位于同一个系统上，而是可以分布在不同的控制器上。在这种情况下，必须为每个系统设置一个 ADS 路由。下图阐释了这种关系。



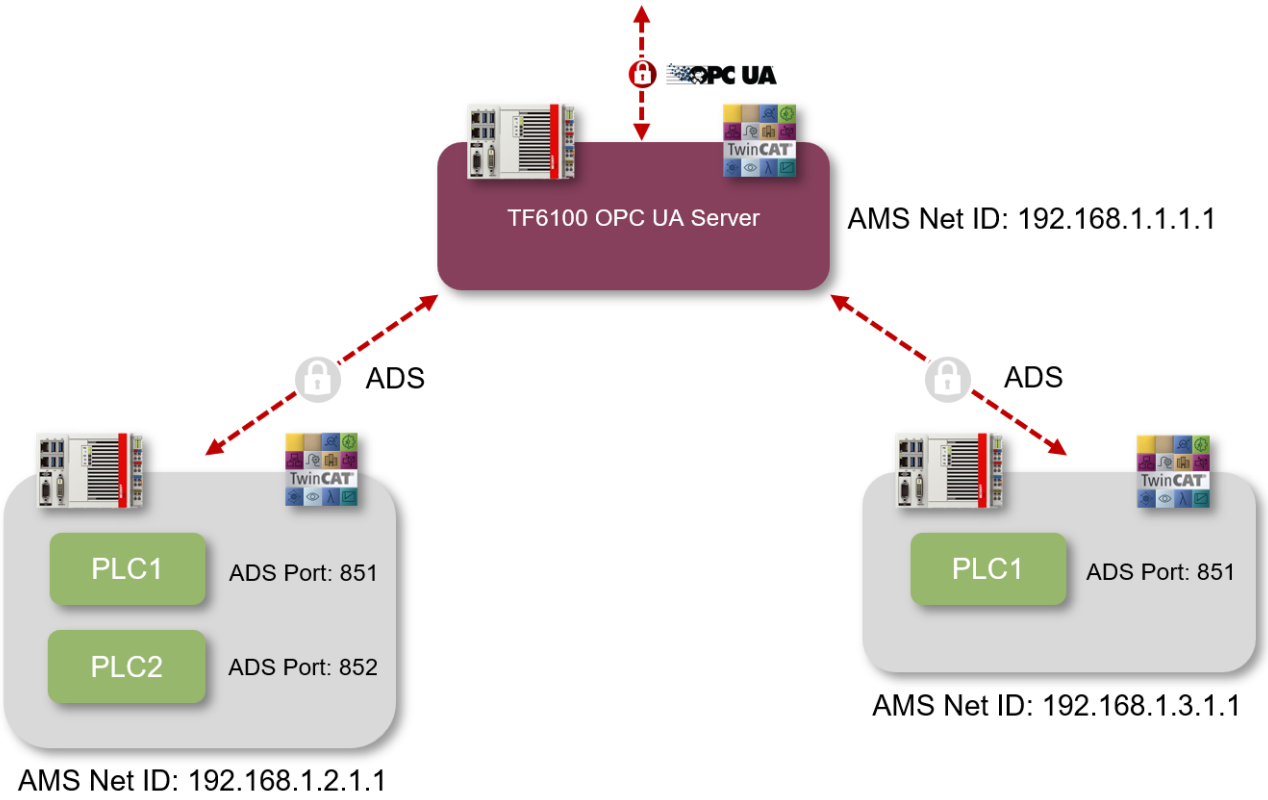
在 TwinCAT OPC UA Configurator 中，您可以在“**Data Access**（数据访问）”选项卡中进行数据访问配置。所有连接的实时环境均作为独立的“设备”显示在其中。“与运行时连接 [▶ 49]”一章介绍了如何添加数据访问设备，以及为此需要提供哪些参数。然后，您可以从“启用符号 [▶ 52]”开始操作。

还请参阅有关此

概述 [▶ 89]

4.11.2 与运行时连接

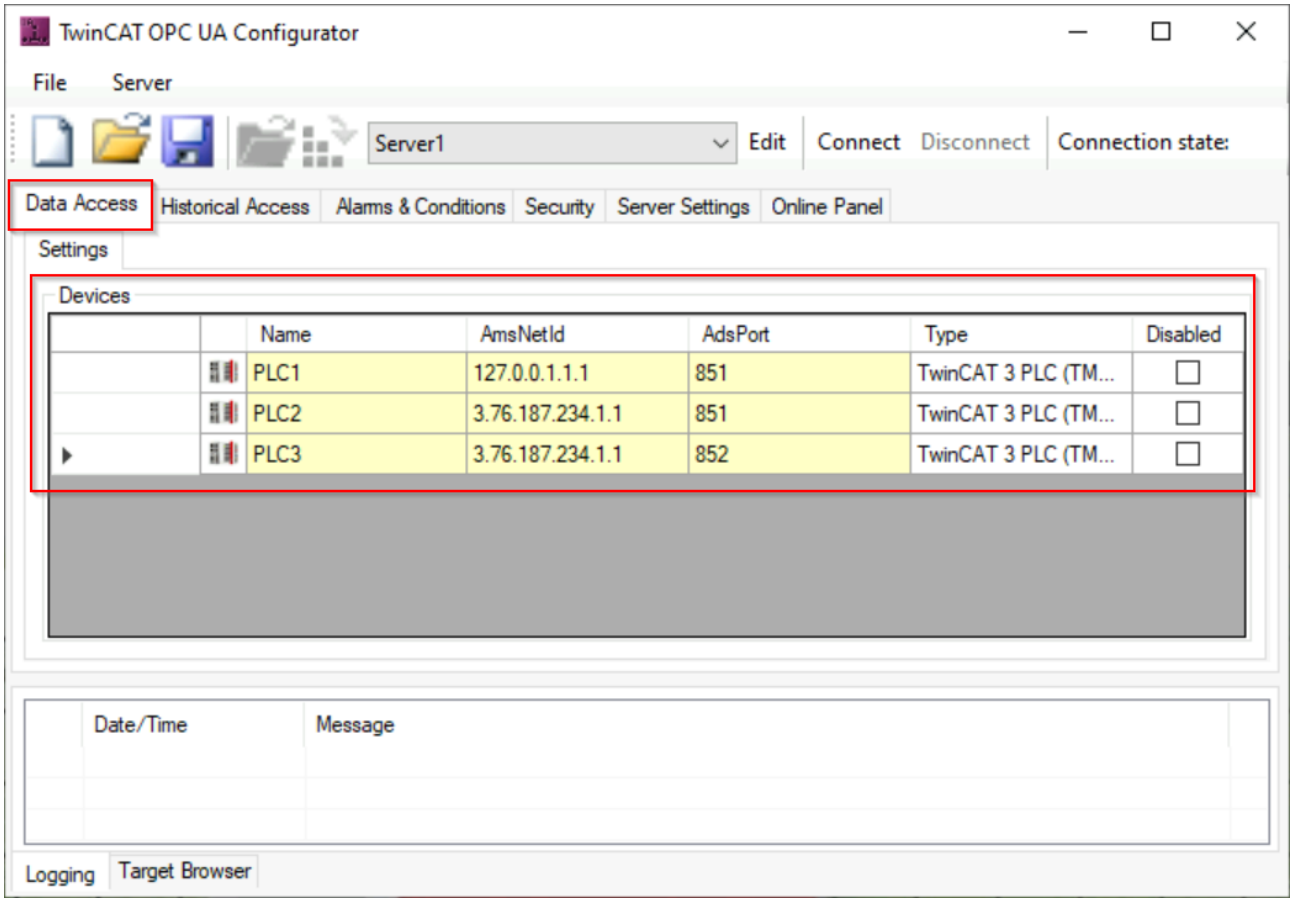
TwinCAT OPC UA Server 可以提供 1 个或多个实时环境中的符号。这些符号也可以位于不同的物理控制系统上。在这种情况下，必须建立一个与相应目标系统的 ADS 路由。所谓的“AMS Net ID（AMS 网络 ID）”是控制系统在网络中的唯一标识，建立与该系统的 ADS 路由时需要使用该 ID。另一方面，ADS 端口则可以识别该系统上的特定应用程序，例如 TwinCAT PLC。
下图阐释了这种关系。



本例中有 3 台控制设备。TwinCAT OPC UA Server 安装在第一台设备上，该设备标识的 AMS Net ID 为 192.168.1.1.1.1。在标识的 AMS Net ID 为 192.168.1.2.1.1 的第二台设备上，启动了 2 个 PLC 运行时系统，这两个系统分别由各自的 ADS 端口进行寻址。第三台设备（192.168.1.3.1.1）上只运行 1 个 PLC 运行时系统。

配置

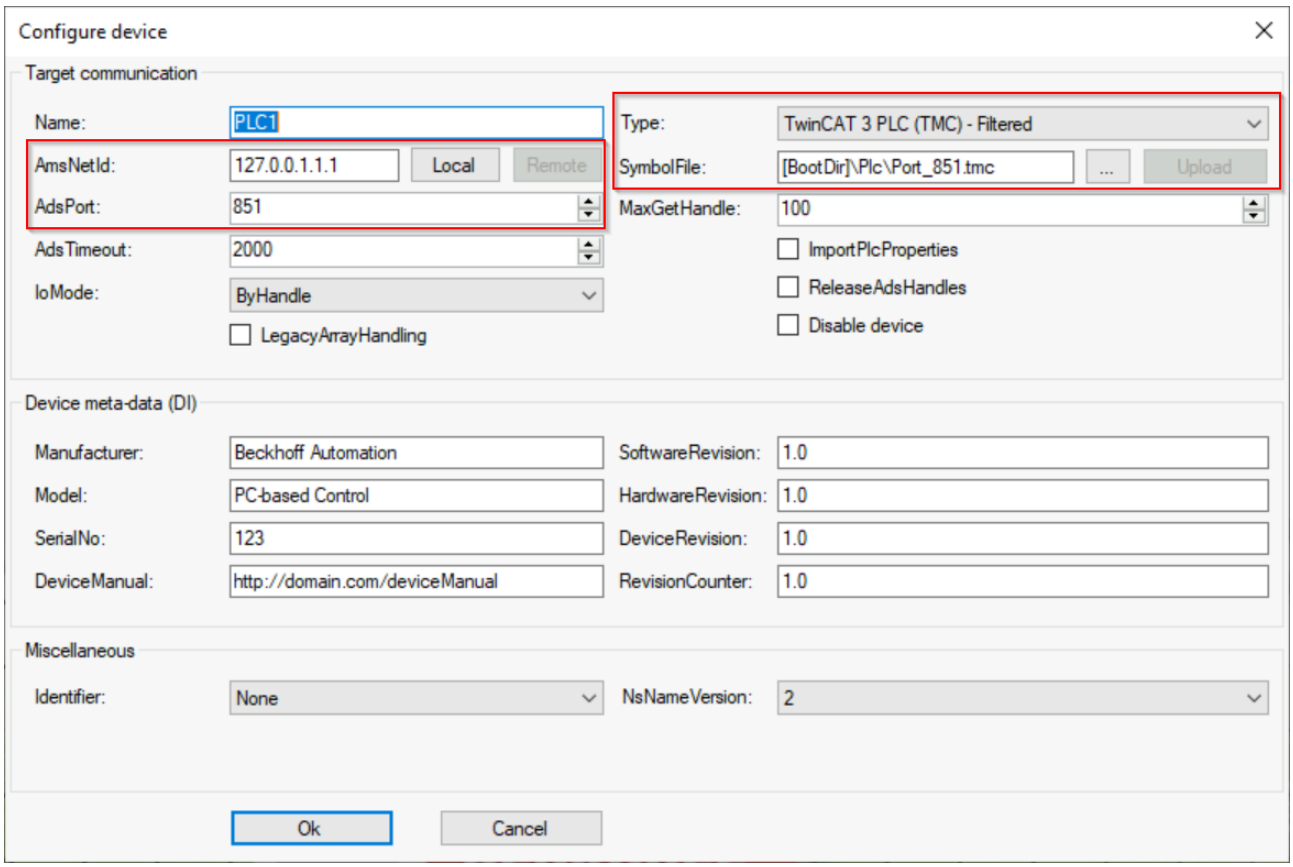
可以使用 TwinCAT OPC UA Configurator 对数据访问设备进行配置。“**Data Access**（数据访问）”选项卡为您提供了所有已配置设备的概览，并且您可以在其中添加或删除设备。



例如，在该屏幕截图中，您可以看到有 3 台数据访问设备的配置。这些设备的配置如下：

- PLC1：本地 PLC 运行时系统（127.0.0.1.1.1），端口 851
- PLC2：远程 PLC 运行时系统（3.76.187.234.1.1），端口 851
- PLC3：远程 PLC 运行时系统（3.76.187.234.1.1），端口 852

与设备连接时需要提供的所有参数及其符号现在均可在设备属性中进行配置。



您可以根据所使用的实时环境，在此处选择一个符号文件、选择要使用的 ADS 路由并输入相应应用程序的 ADS 端口等。有关各种符号文件的更多信息，请参见“启用符号 [▶ 52]”一章。

4.11.3 启用符号

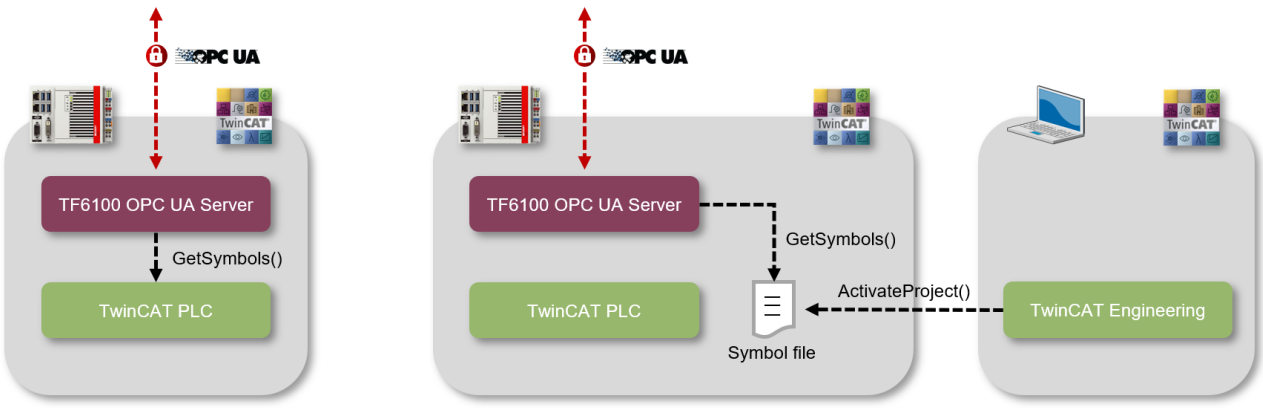
您在快速入门 [▶ 23]教程中已经了解，在 TwinCAT 3 PLC 中，符号是通过所谓的编译指示（也称为“属性”）来启用的。在其他实时环境中，启用机制可能会有所不同。下一章将对此进行概述。

“符号”一词

本文档中使用“符号”一词来代替变量、结构、功能块实例、方法等。

符号

TwinCAT OPC UA Server 通过实时变量（如地址和数据类型）接收其信息，即所谓的“符号”。根据所使用的实时环境（如 TwinCAT PLC 或 TwinCAT 3 C++）的不同，有各种所谓的“符号文件”可用于读取符号（也称为“离线符号”）。此外，还可在 Runtime 环境启动时通过 TwinCAT ADS 读取符号（也称为 "在线符号"）。



Online Symbolic Offline Symbolic

使用在线符号的优势在于无需交换符号文件。特别是在网关 PC 上运行 TwinCAT OPC UA Server 的安装方案 [▶ 16]中，无需再手动复制所有已连接控制器的符号文件。在线符号的缺点是，只有在 PLC 环境运行时才能使用符号。

若要访问符号的地址信息，必须将该服务器导入符号文件中。或者，服务器也可以直接使用在线符号。这样，只有在实时环境可用时，符号才会显示在服务器的地址空间中。

可以使用 TwinCAT OPC UA Configurator 来配置服务器及其应读取的符号文件。作为数据访问设备的一部分，可以在服务器中配置符号文件的相应路径。

根据所用实时环境的不同，为获取现有符号的地址信息而必须从服务器导入的符号文件以及符号启用机制也会有所不同。TwinCAT 3 PLC 使用所谓的编译指示来启用 OPC UA 符号，而 TwinCAT 2 仍然通过在符号上使用特殊格式的注释来实现此目的。TwinCAT 3 C++ 则使用所谓的 TMC 代码生成器来启用 OPC UA 符号。

可以在 TwinCAT 开发环境界面配置符号文件，以便在激活项目时自动将其复制到目标设备的启动目录中。TwinCAT OPC UA Server 通常配置为从启动目录读取符号文件。

下表概述了各种符号文件与其实时环境和启用机制的关系。

实时的环境	符号文件	启用机制……	在启动目录中的路径
TwinCAT 2 PLC	TPY	注释	CurrentPlc_1.tpy
TwinCAT 3 PLC [▶ 53]	TMC	编译指示	Plc\Port_%AdsPort%.tmc
TwinCAT 3 C++ [▶ 55]	TMI	TMC 代码生成器	Tmi\%ObjectId%.tmi
TwinCAT 3 MATLAB®/ Simulink® [▶ 59]	TMI	TMC 代码生成器	Tmi\%ObjectId%.tmi
TwinCAT 3 I/O Task [▶ 65]	XML	注释	CurrentConfig.xml
在线符号 [▶ 67]	---	编译指示 TMC 代码生成器	---

有关详细信息，请参见各实时环境的子章节。

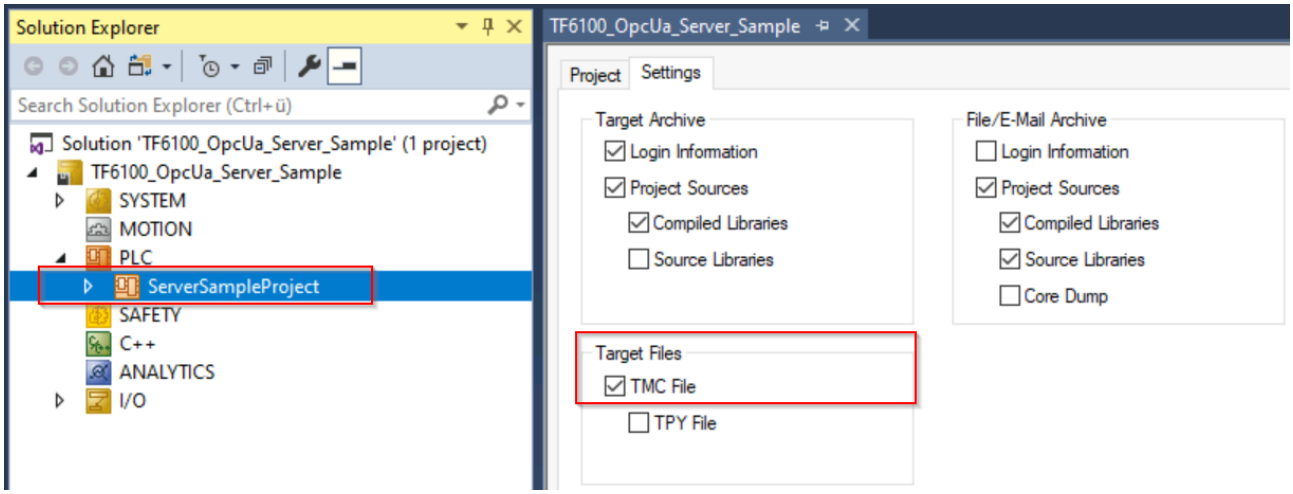
4.11.3.1 PLC

本节将介绍如何配置 TwinCAT OPC UA Server 的命名空间，以便获取 PLC 项目符号的访问权限。为此，必须执行以下几个步骤：

- 激活符号文件
- 启用符号
- 服务器的配置

激活符号文件

若要通过 OPC UA 提供 PLC 项目的符号，必须首先在 PLC 项目属性中激活符号文件（TMC）的下载。



这意味着在激活项目时，会将符号文件自动传输到相应的目标设备，然后即可在该目标设备的 TwinCAT 启动目录中获取该符号文件。例如，符号文件的名称基于 PLC 运行时系统的 ADS 端口来进行命名（见下文）：

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

快速入门

如果 TwinCAT OPC UA Server 也安装在同一设备上，它始终会自动读取交付状态下第一个 PLC 运行时系统的符号文件，即无需在服务器上进行进一步设置。因此，在激活项目时，会将符号文件传输至目标设备，随后在重启 TwinCAT 时，TwinCAT OPC UA Server 也会随之重启。现在即可在启动过程中找到启动目录中的符号文件并读取该文件。服务器通过设置的编译指示来识别其地址空间中应提供的符号。

启用符号

在 TwinCAT 3 PLC 中，我们通过所谓的编译指示来启用 OPC UA 的符号。例如，在快速入门 [▶ 23]教程中也使用了这样的编译指示来启用 OPC UA 的变量。该编译指示的插入位置位于其适用的符号之前。可以使用以下编译指示启用 OPC UA 的变量、数组或结构。TwinCAT OPC UA Server 在读取符号时会识别该编译指示，并将相应的符号导入其命名空间。然后将相应的类型信息 [▶ 69]传输和映射到 OPC UA。

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

对于数据访问的个别子功能，可能需要更多的编译指示，例如结构 [▶ 74]、属性 [▶ 77]、AnalogItemType [▶ 81]、StatusCode [▶ 77]、设置描述 [▶ 81]、别名或节点的只读 [▶ 83]标志。然后在单独一行中插入附加的编译指示，例如：

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

（设置该变量的只读标志）

系统会将用于启用某个符号的编译指示自动沿用至所有子符号。若要从某一点开始阻止这种沿用，例如从结构的某个成员开始，可以使用下方的编译指示：

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

对于功能块和结构而言，编译指示的定义定位起到了决定性作用。如果在某个实例上定义编译指示，则仅会为 OPC UA 启用该实例（以及所有子元素）。反之，如果在定义功能块或结构时定义了编译指示，则该功能块或结构的所有实例都将得到启用。在下方的示例中，我们将通过 OPC UA 提供功能块 FB_BLOCK1 的 2 个实例 fbTest1 和 fbTest2。启用整个实例后，其所有符号也可通过 OPC UA 调用。PLC 程序如下所示：

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

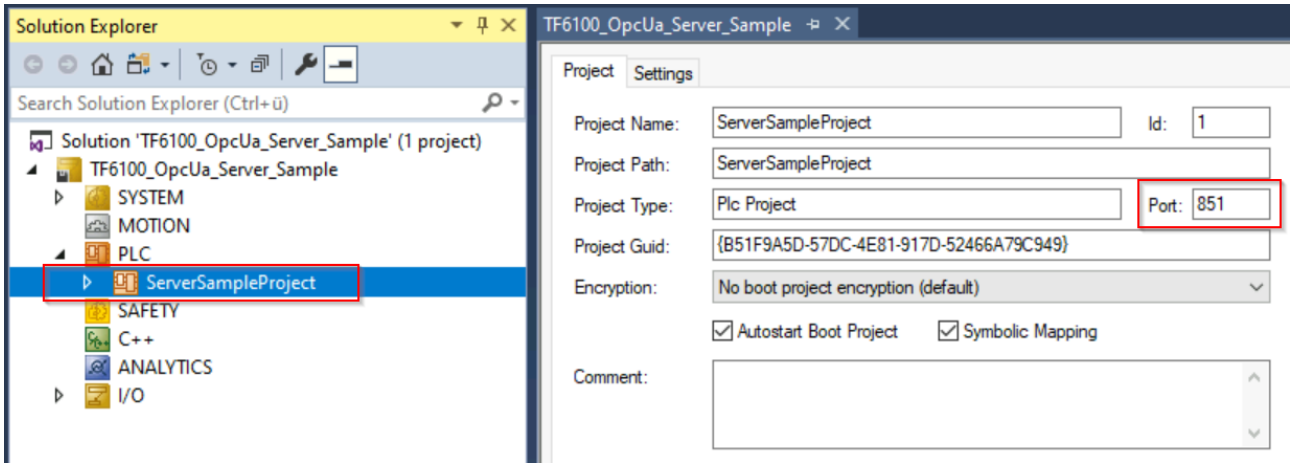
```
FUNCTION_BLOCK FB_BLOCK1
VAR_INPUT
  {attribute 'OPC.UA.DA' := '1'}
  ni1 : INT;
  ni2 : INT;
END_VAR
VAR_OUTPUT
  {attribute 'OPC.UA.DA' := '1'}
  no1 : INT;
  no2 : INT;
END_VAR
VAR
  {attribute 'OPC.UA.DA' := '1'}
  nx1 : INT;
  nx2 : INT;
END_VAR
```

实例 fbTest1 现在收到了编译指示，这意味着其中所包含的所有符号（即 fbTest.ni1、fbTest.ni2 等）也自动启用，供 OPC UA 使用。另一方面，实例 fbTest2 没有收到编译指示，但它包含的 3 个变量 ni1、no1 和 nx1 均标记了该编译指示。因此，在所有实例中均可通过 OPC UA 获取这些变量。

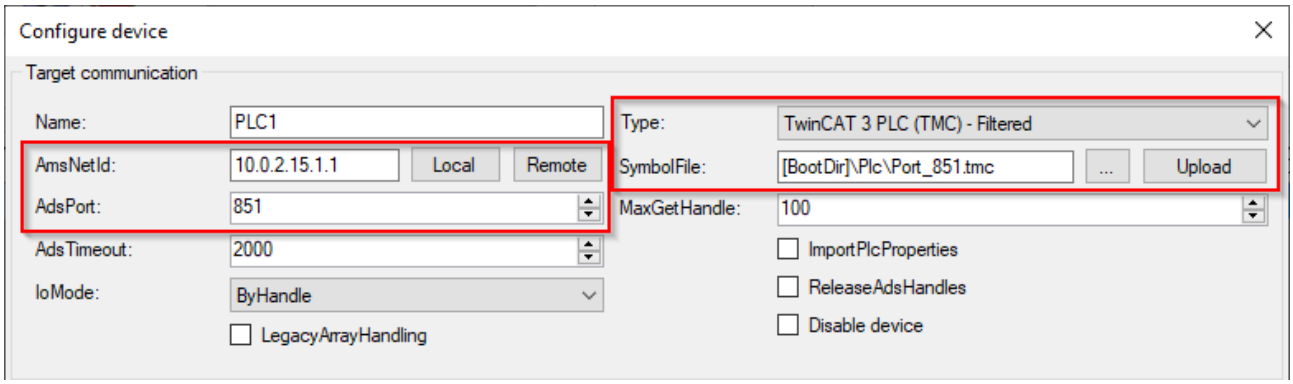
服务器的配置

现在，您可以使用 TwinCAT OPC UA Configurator 来配置 TwinCAT OPC UA Server，以便其读取生成的符号文件，并将 PLC 运行时系统作为其地址空间中的数据访问设备供用户使用。

为此，请在 TwinCAT OPC UA 配置器的数据访问选项卡中添加一个新设备。选择设备的 AMS Net ID 并输入 PLC Runtime 的 ADS 端口。您可以在 PLC 项目属性中找到 ADS 端口：



现在，从“Type（类型）”字段的选择列表中选择“TwinCAT 3 PLC (TMC) - Filtered（TwinCAT 3 PLC（TMC） - 已过滤）”条目。在“SymbolFile”字段中，选择在激活 TwinCAT PLC 项目后创建的符号文件（TMC），该文件现在应位于启动目录中。或者，您也可以使用在线符号学 [▶ 67]。



所有其他设置均可保留默认值。

4.11.3.2 C++

本节介绍如何配置 TwinCAT OPC UA 服务器的命名空间，以便获得对 TwinCAT 3 C++ 模块符号的访问权限。为此，必须执行以下几个步骤：

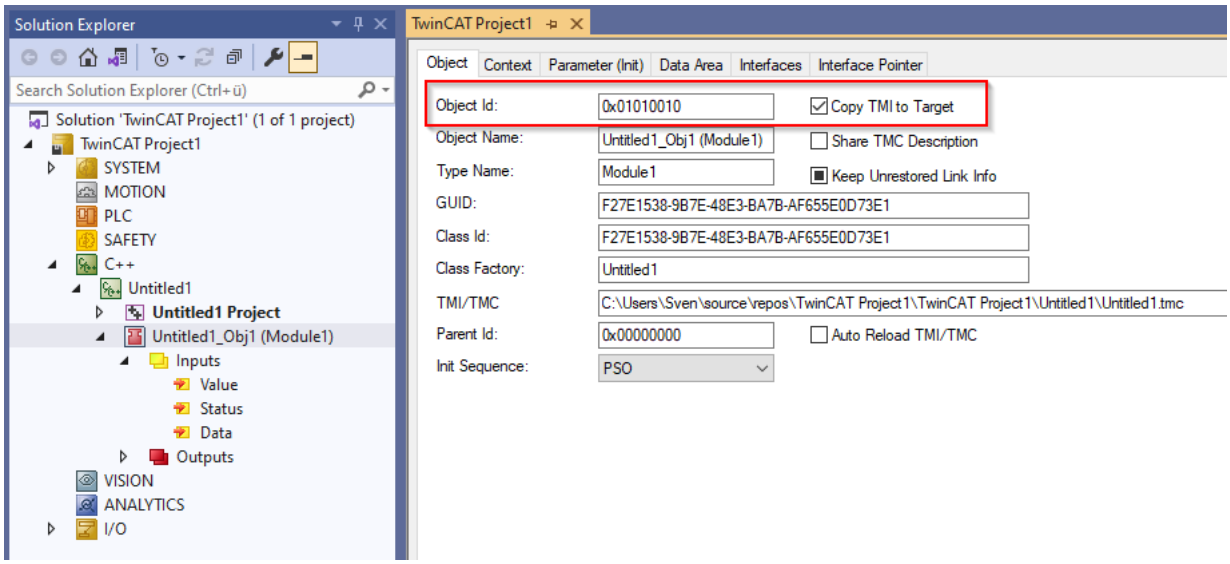
- 激活符号文件
- 启用符号
- 服务器的配置

这些步骤在下文会有详细说明。

激活符号文件

若要配置 C++ 模块某一实例中包含的某些符号，以便通过 OPC UA 访问这些符号，必须首先激活该模块实例符号文件的下载。这将在激活配置时将符号文件复制到目标设备的启动目录。

请通过在模块实例的属性中设置以下选项来激活符号文件的下载：

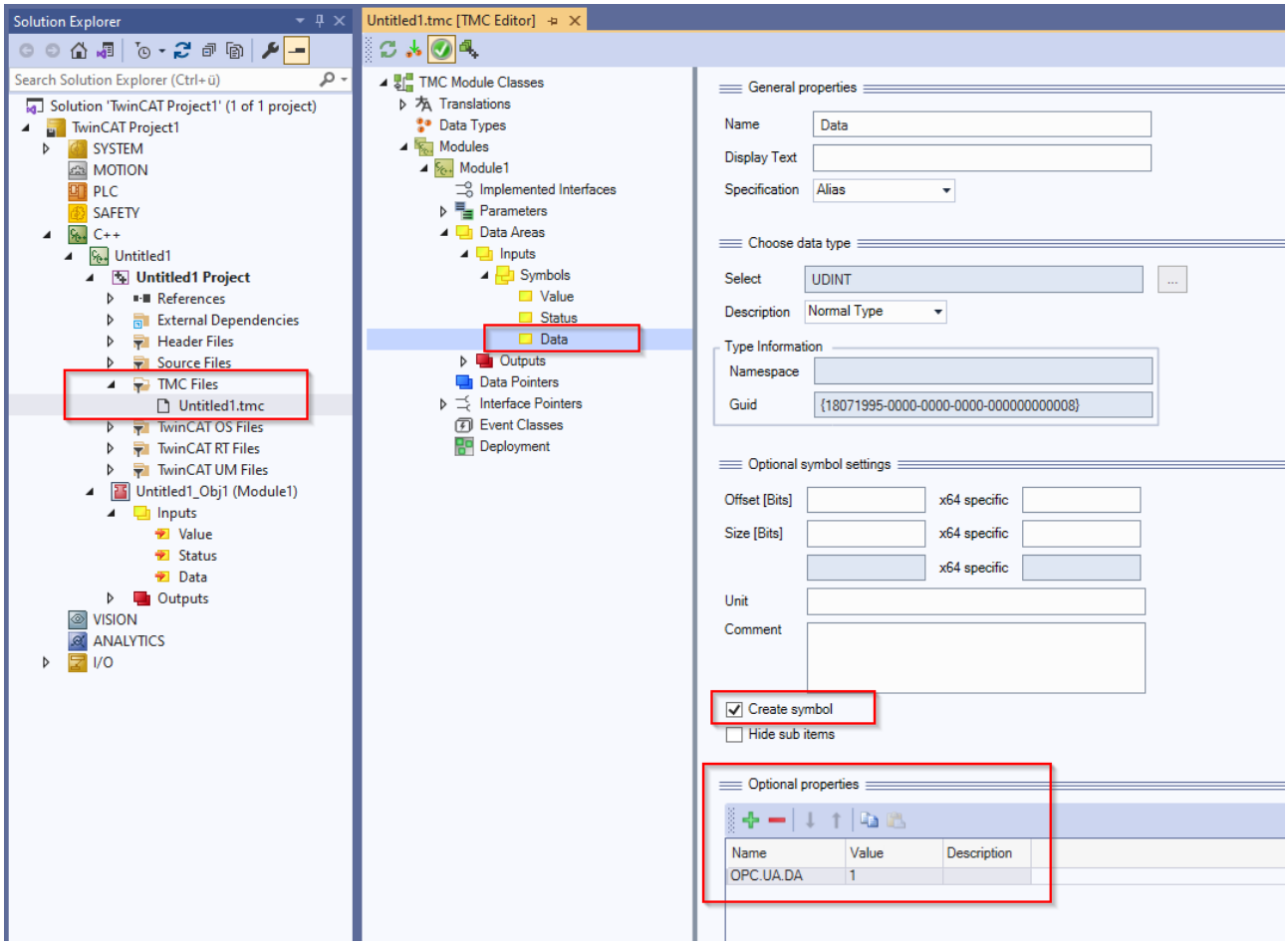


生成的符号文件（TMI）始终以模块实例的 ObjectId 命名，现在，系统会在激活配置时将其复制到目标设备的启动目录中，例如：

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

启用符号

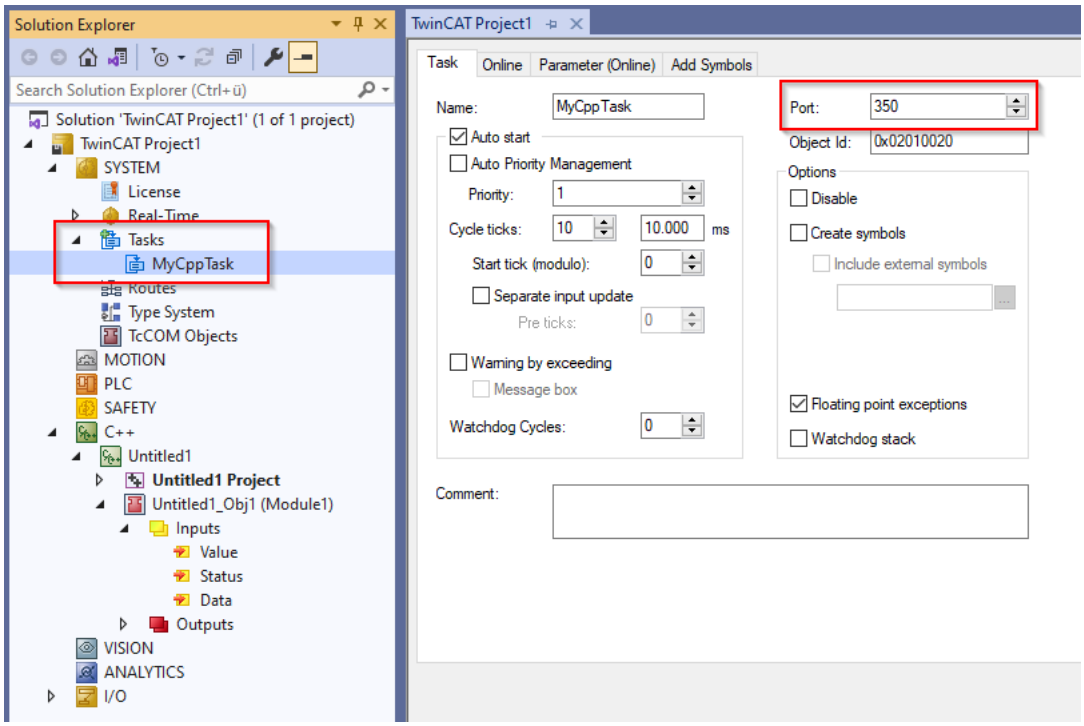
您可以使用 TMC 代码生成器来定义通过 OPC UA 启用的符号。为此需要进行 2 项设置。首先为要在 TMC 代码生成器中启用的每个符号激活“Create Symbol（创建符号）”复选框。然后添加一个键值为“OPC.UA.DA”、值为“1”的可选属性。



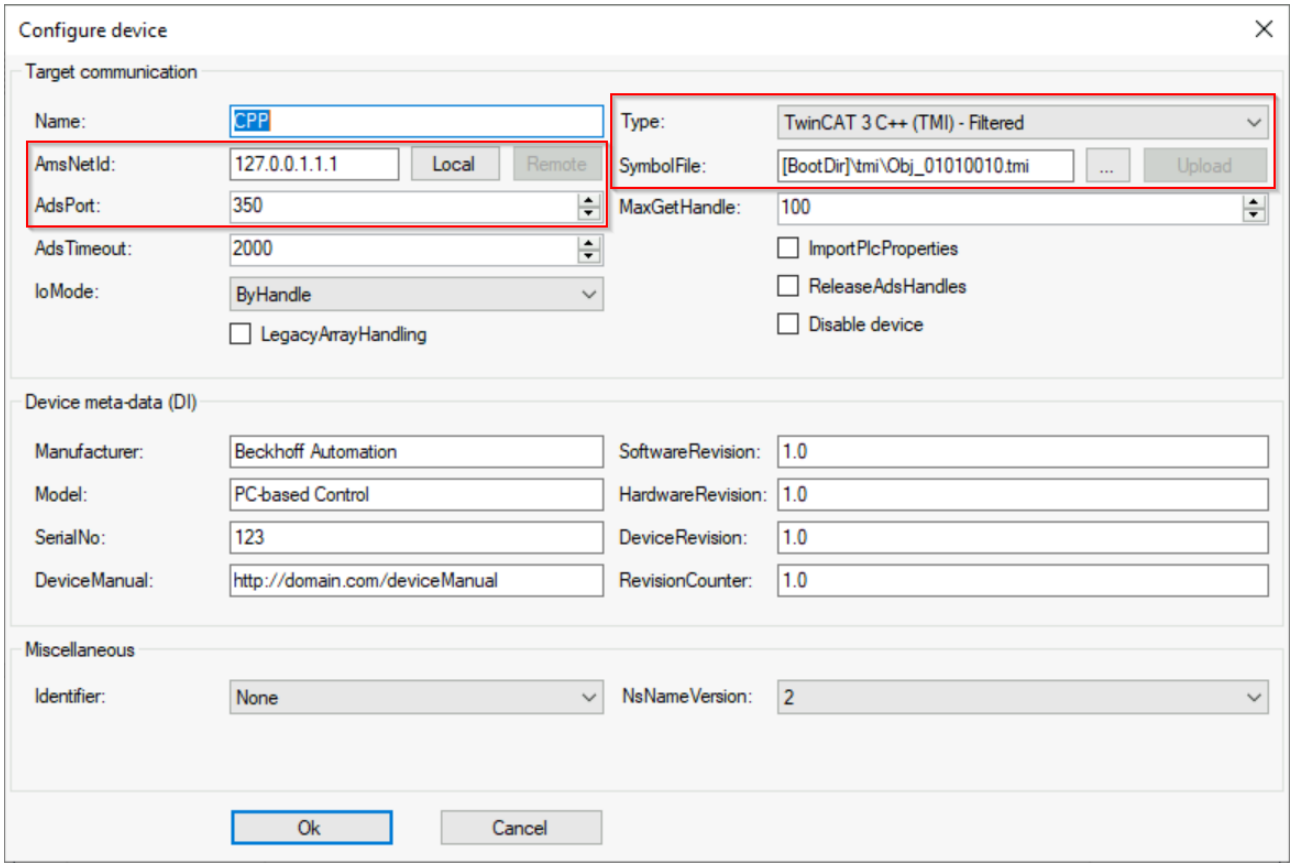
服务器的配置

现在，您可以使用 TwinCAT OPC UA Configurator 来配置 TwinCAT OPC UA Server，以便其读取生成的符号文件，并将 C++ 模块实例作为其地址空间中的数据访问设备供用户使用。

为此，请在 TwinCAT OPC UA 配置器的数据访问选项卡中添加一个新设备。选择设备的 AMS Net ID 并输入 TwinCAT 3 C++ 模块实例的 ADS 端口。您可以在与 C++ 模块实例连接的任务属性中找到该 ADS 端口：



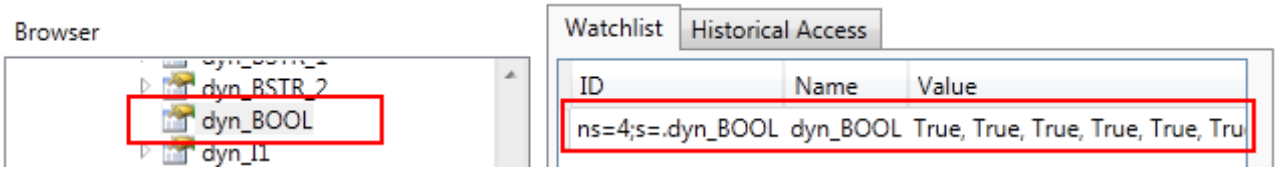
现在，从“Type（类型）”字段的选择列表中选择“TwinCAT 3 C++(TMI) - Filtered（TwinCAT 3 C++（TMI）- 已过滤）”条目。在“SymbolFile”字段中，选择在激活 TwinCAT 3 C++ 项目后创建的符号文件（TMI），该文件现在应位于启动目录中。或者，您也可以使用在线符号学 [▶ 67]。



所有其他设置均可保留默认值。

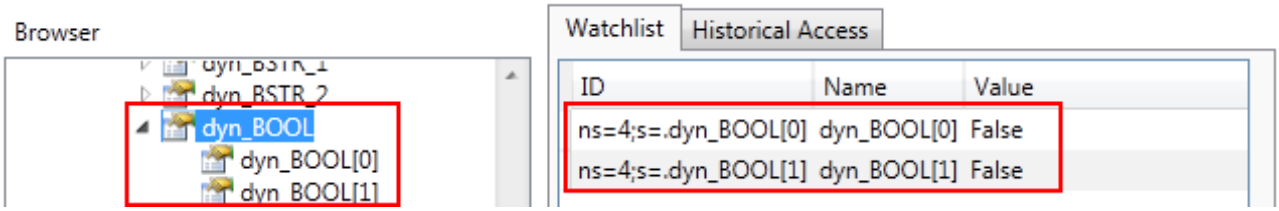
4.11.3.2.1 数组

默认情况下，数组被视为 UA 名称空间中的单个节点。这意味着，例如，如果您在 PLC 中定义了一个数组 dyn_BOOL[10]（也为 OPC UA 启用了该数组），该数组随后将会出现在 UA 命名空间中，如下所示：



这种方法的优点在于大大降低了 UA 命名空间的复杂度和内存占用，因为不需要将数组的每个位置都作为命名空间中的独立节点。但是，现代 UA 客户端仍然可以通过所谓的“RangeOffset”来访问各个数组位置。

不过，为了支持不提供此功能的旧版 UA 客户端，您也可以在 UA 命名空间中将数组的位置作为独立节点使用。详见下方的插图：



通过在相应的命名空间配置中激活 UA 配置器中的“**Legacy Array Handling（传统数组处理）**”选项，即可进行此设置。

根据 PLC 项目范围的不同，UA 命名空间可能会变得更加复杂，进而导致 UA 服务器的内存占用增加。

对上述设置的更改只有在重启 UA 服务器后才会生效。

4.11.3.3 MATLAB®/Simulink®

本节将介绍如何配置 TwinCAT OPC UA Server 的命名空间，以便获得对 TwinCAT 3 MATLAB®/Simulink® 模块符号的访问权限。为此，必须执行以下几个步骤：

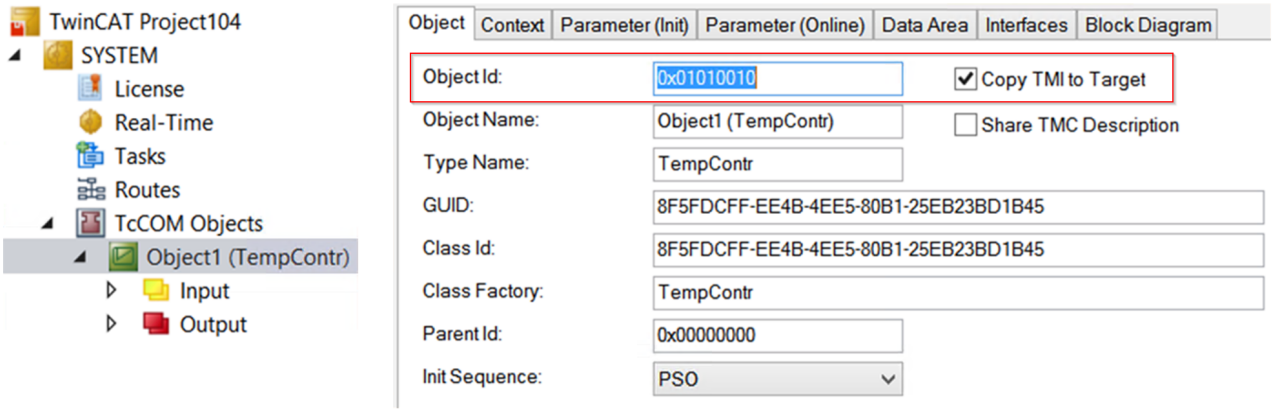
- 激活符号文件
- 启用符号
- 服务器的配置

这些步骤在下文会有详细说明。

激活符号文件

若要配置 C++ 模块某一实例中包含的某些符号，以便通过 OPC UA 访问这些符号，必须首先激活该模块实例符号文件的下载。这将在激活配置时将符号文件复制到目标设备的启动目录。

请通过在模块实例的属性中设置以下选项来激活符号文件的下载：



生成的符号文件（TMI）始终以模块实例的 ObjectId 命名，现在，系统会在激活配置时将其复制到目标设备的启动目录中，例如：

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

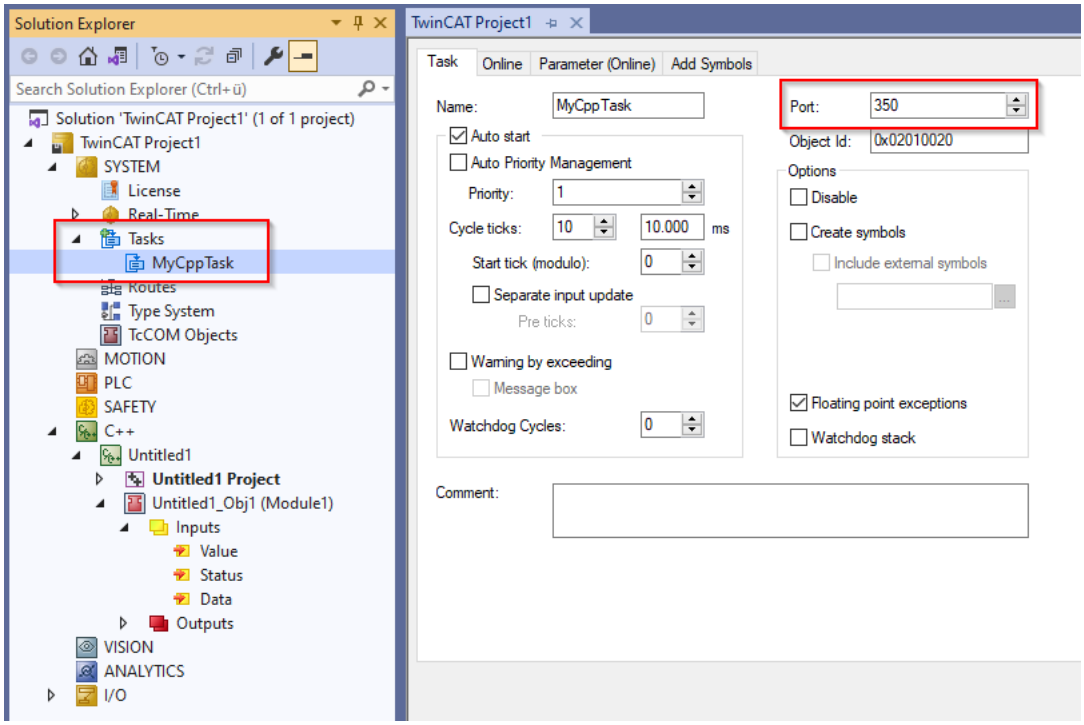
启用符号

TE1400 Target for Simulink® 产品提供多种设置选项，用于选择要通过 OPC UA 启用的任意变量。有关这些设置的说明，详见相应的 TE1400 产品文档。

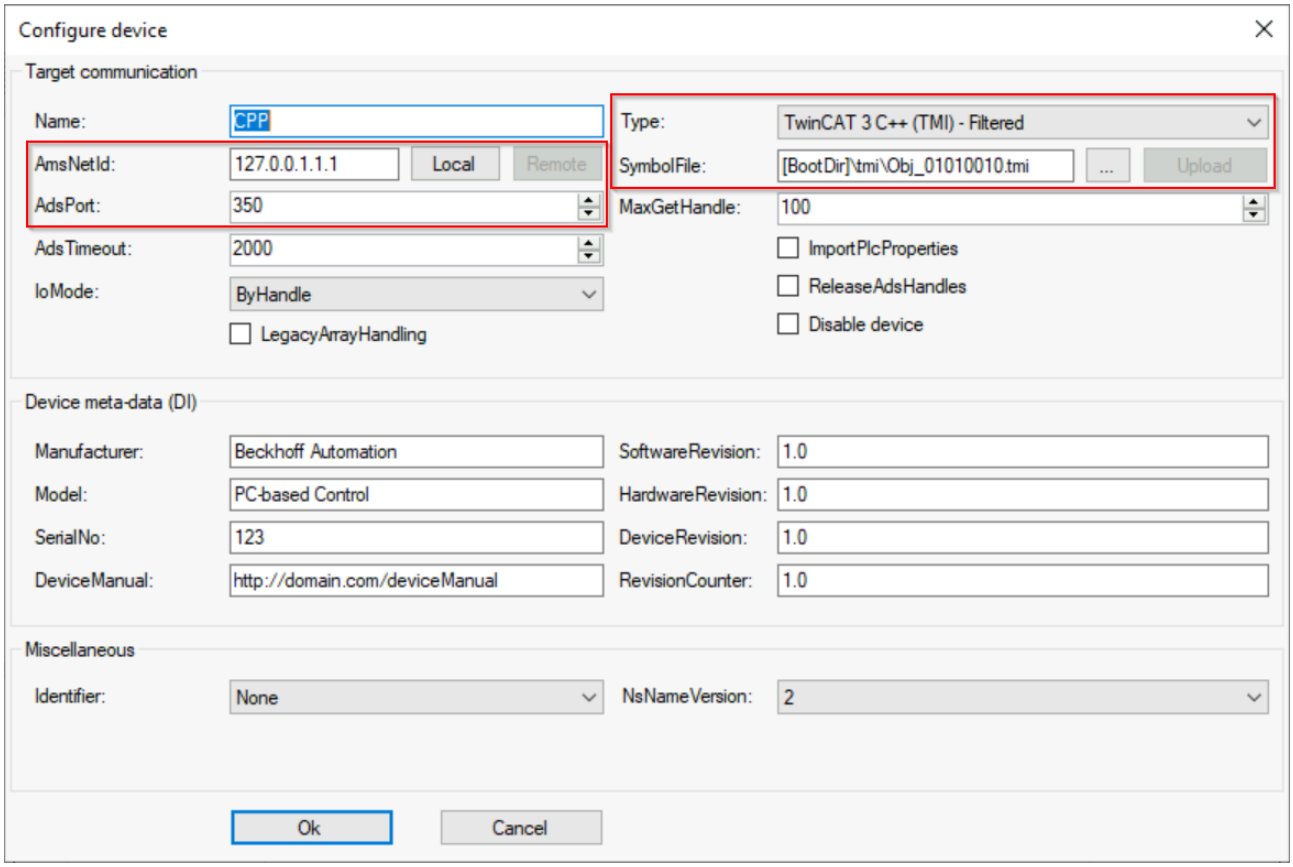
服务器的配置

现在，您可以使用 TwinCAT OPC UA Configurator 来配置 TwinCAT OPC UA Server，以便其读取生成的符号文件，并将 MATLAB®/Simulink® 模块实例作为其地址空间中的数据访问设备供用户使用。

为此，请在 TwinCAT OPC UA 配置器的数据访问选项卡中添加一个新设备。选择设备的 AMS Net ID 并输入 TwinCAT 3 MATLAB®/Simulink® 模块实例的 ADS 端口。您可以在与该模块实例连接的任务属性中找到该 ADS 端口：



现在，从“Type（类型）”字段的选择列表中选择“TwinCAT 3 C++(TMI) - Filtered（TwinCAT 3 C++（TMI） - 已过滤）”条目。在“SymbolFile”字段中，选择在激活 TwinCAT 3 C++ 项目后创建的符号文件（TMI），该文件现在应位于启动目录中。或者，您也可以使用在线符号学 [▶ 67]。



所有其他设置均可保留默认值。

4.11.3.4 EtherCAT 主站

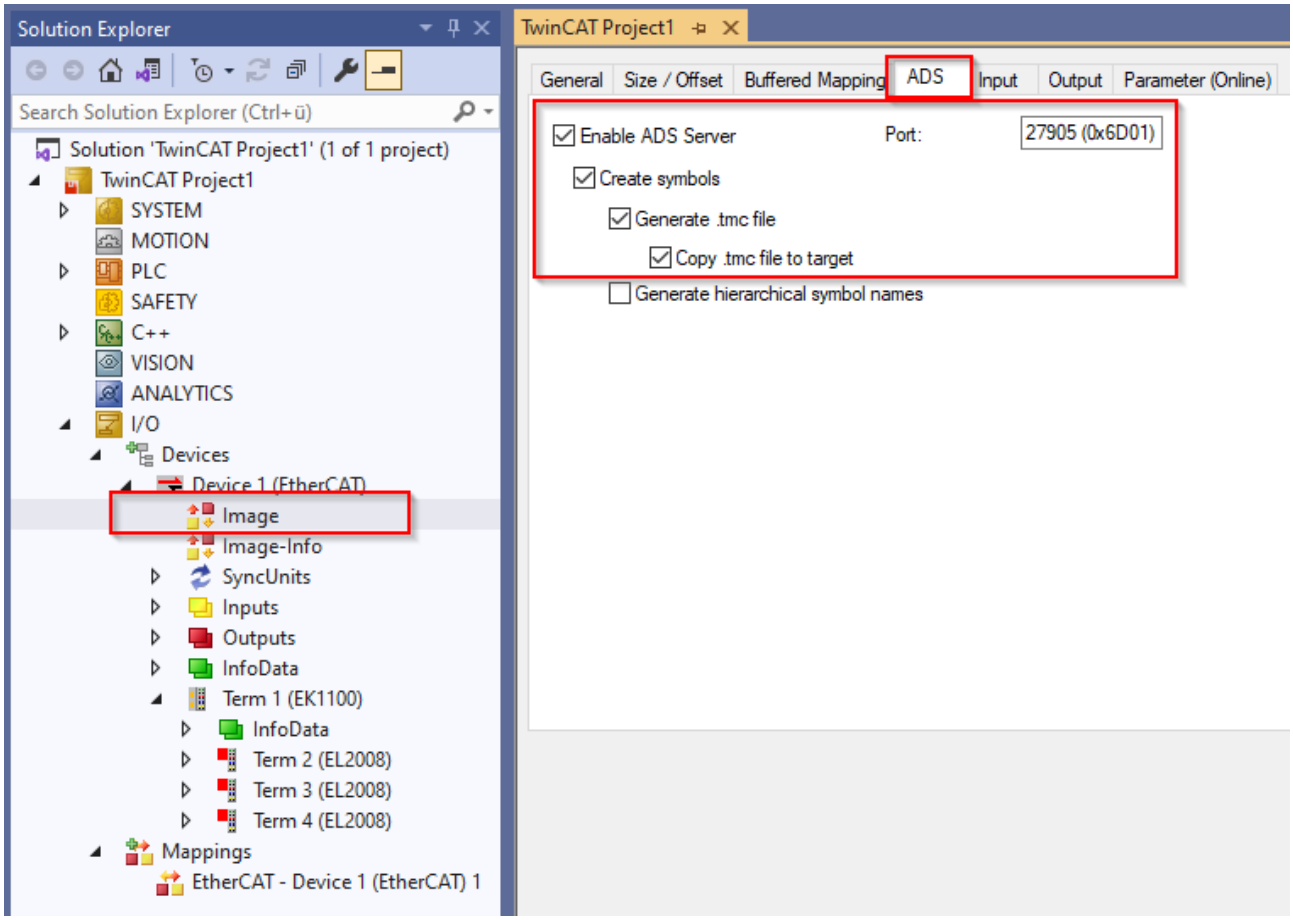
本节将介绍如何配置 TwinCAT OPC UA 服务器的命名空间，以获取对 EtherCAT 主站符号的访问权限。为此，必须执行以下几个步骤：

- 激活符号文件
- 服务器的配置
- ［可选］更改命名空间结构

这些步骤在下文会有详细说明。

激活符号文件

若要通过 OPC UA 提供 EtherCAT 主站的符号，必须首先激活 EtherCAT 主站图像中的 ADS 服务器以及符号的生成。这样就可以通过 ADS 访问符号。您还可以在同一对话框中找到 ADS 端口，在稍后的配置过程中将需要使用该端口。



通过激活参数“Generate .tmc file（生成 .tmc 文件）”和“Copy .tmc file to target（将 .tmc 文件复制到目标设备）”，将生成一个符号文件，并且将在激活配置时将其自动下载至目标设备。该文件随后将位于 TwinCAT 启动目录中，并且文件名中包含 ADS 端口，以便于识别，示例如下：

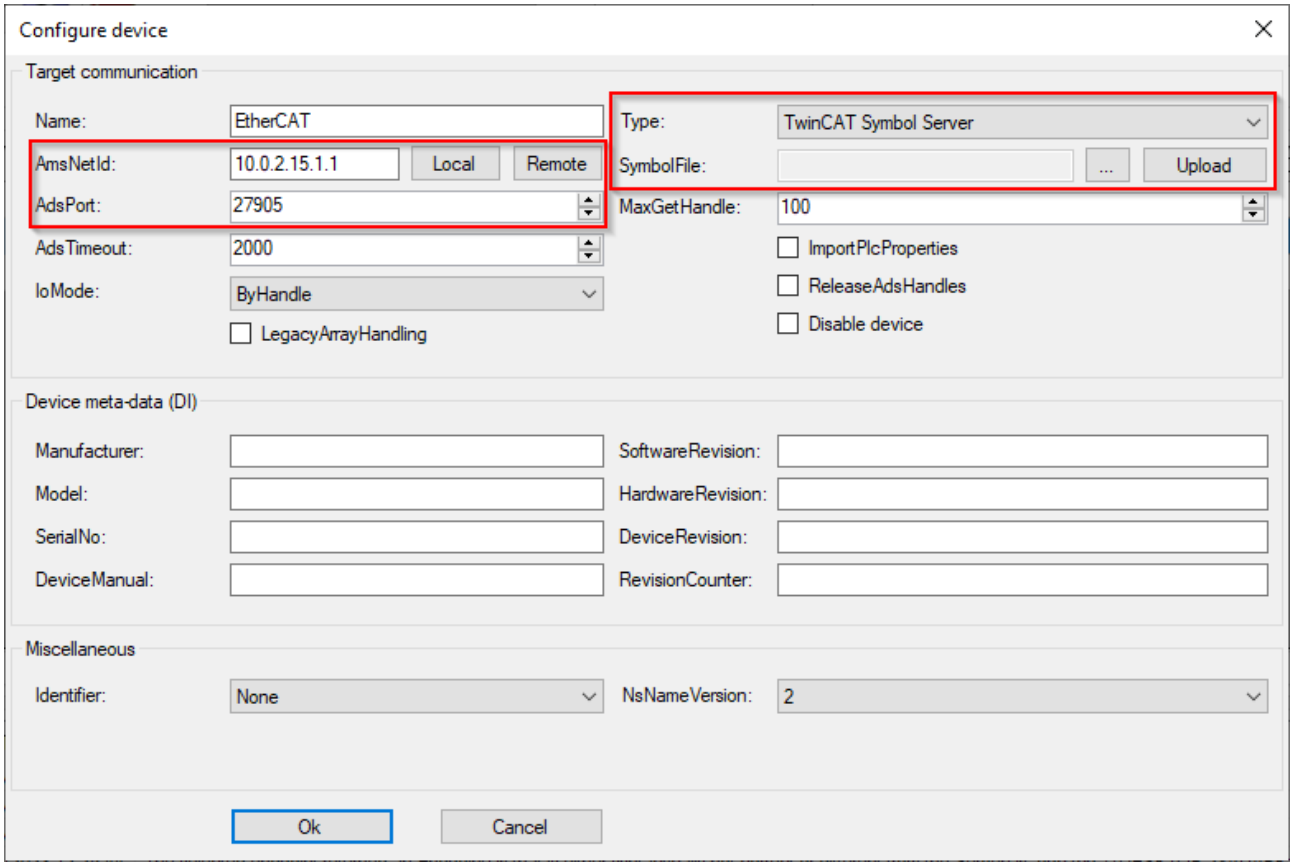
```
%TwinCATInstallDir%\3.1\Boot\Io\Port_27905.tmc
```

服务器的配置

现在，您可以使用 TwinCAT OPC UA Configurator 来配置 TwinCAT OPC UA Server，以便其读取生成的符号文件，并将 EtherCAT 主站作为其地址空间中的数据访问设备供用户使用。

在 TwinCAT OPC UA Configurator 的“Data Access（数据访问）”选项卡中添加一台新设备。选择设备的 AMS Net ID 并输入 EtherCAT 主设备图像的 ADS 端口（见上文）。

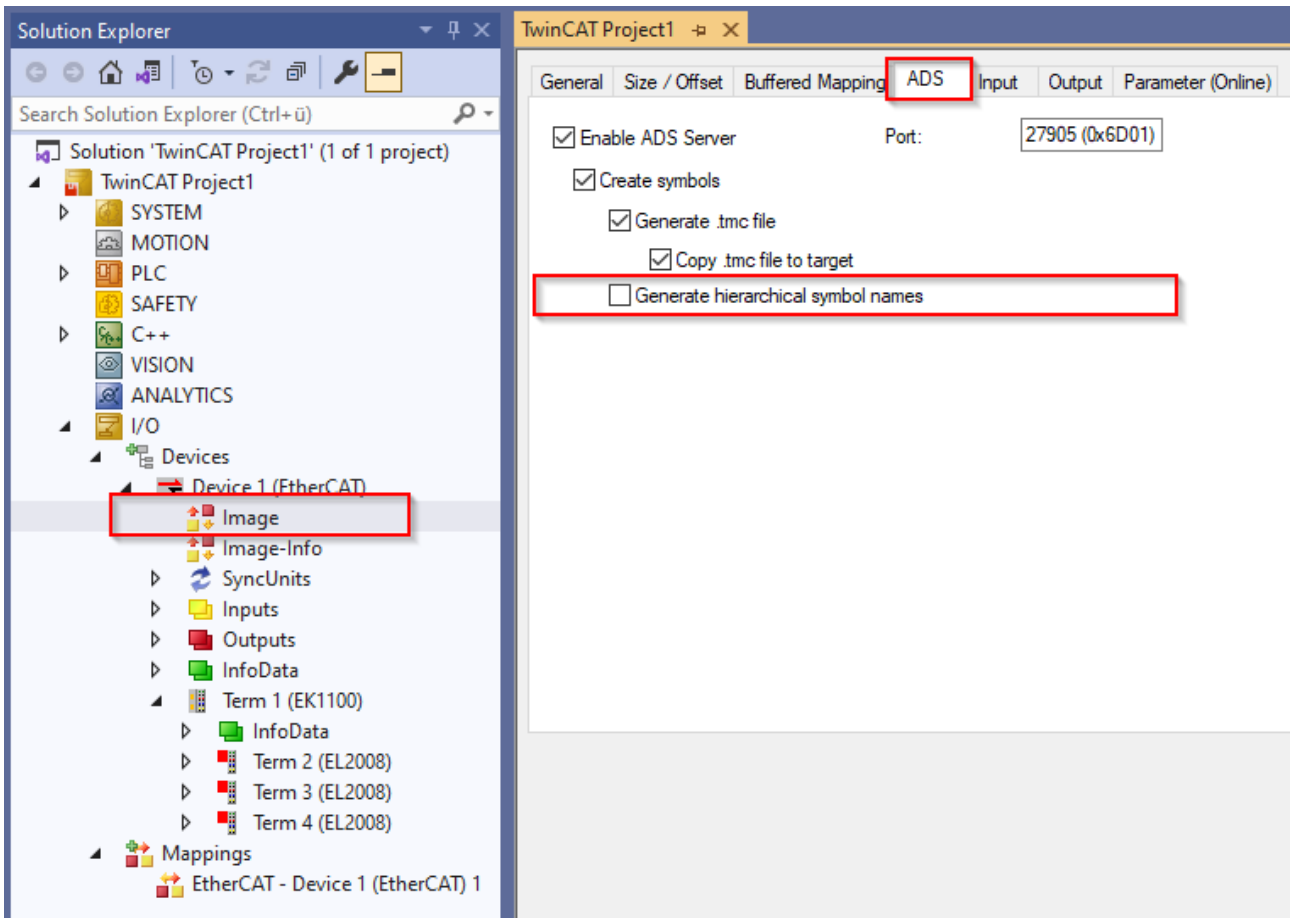
现在，从“Type（类型）”字段的选择列表中选择“EtherCAT Master (TMC) - Filtered（EtherCAT 主站（TMC） - 已过滤）”条目。在“SymbolFile”字段中，选择在激活 TwinCAT 项目后创建的符号文件（TMC），该文件现在应位于启动目录中。或者，您也可以使用在线符号学 [▶ 67]。



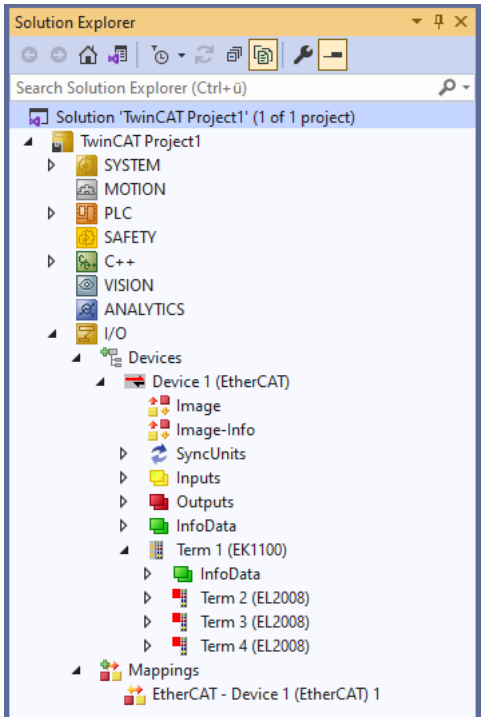
所有其他设置均可保留默认值。

更改命名空间结构

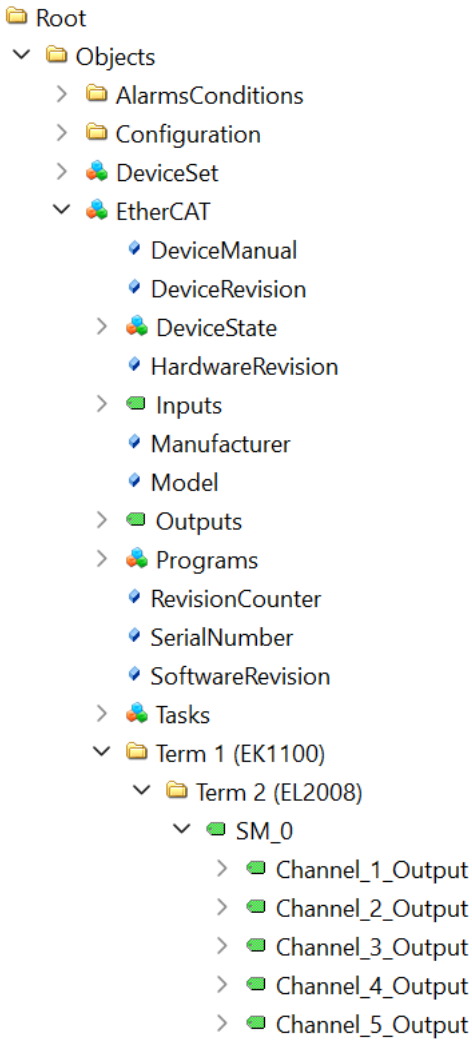
通过参数“Generate hierarchical symbol names（生成层级符号名称）”可以更改符号空间的结构，这也会直接影响 TwinCAT OPC UA Server 的命名空间。



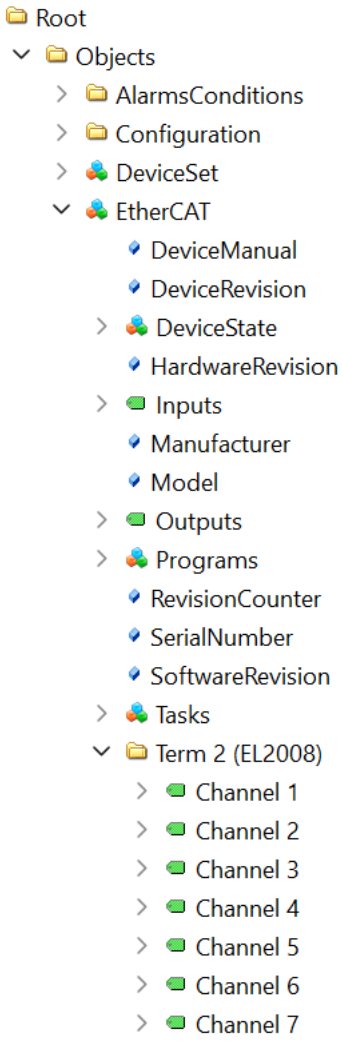
下面以 TwinCAT XAE 中的 EtherCAT 主站配置为例：



如果激活了该选项，命名空间将反映 TwinCAT XAE 中的结构：



如果禁用了该选项，命名空间的结构会“更扁平”：



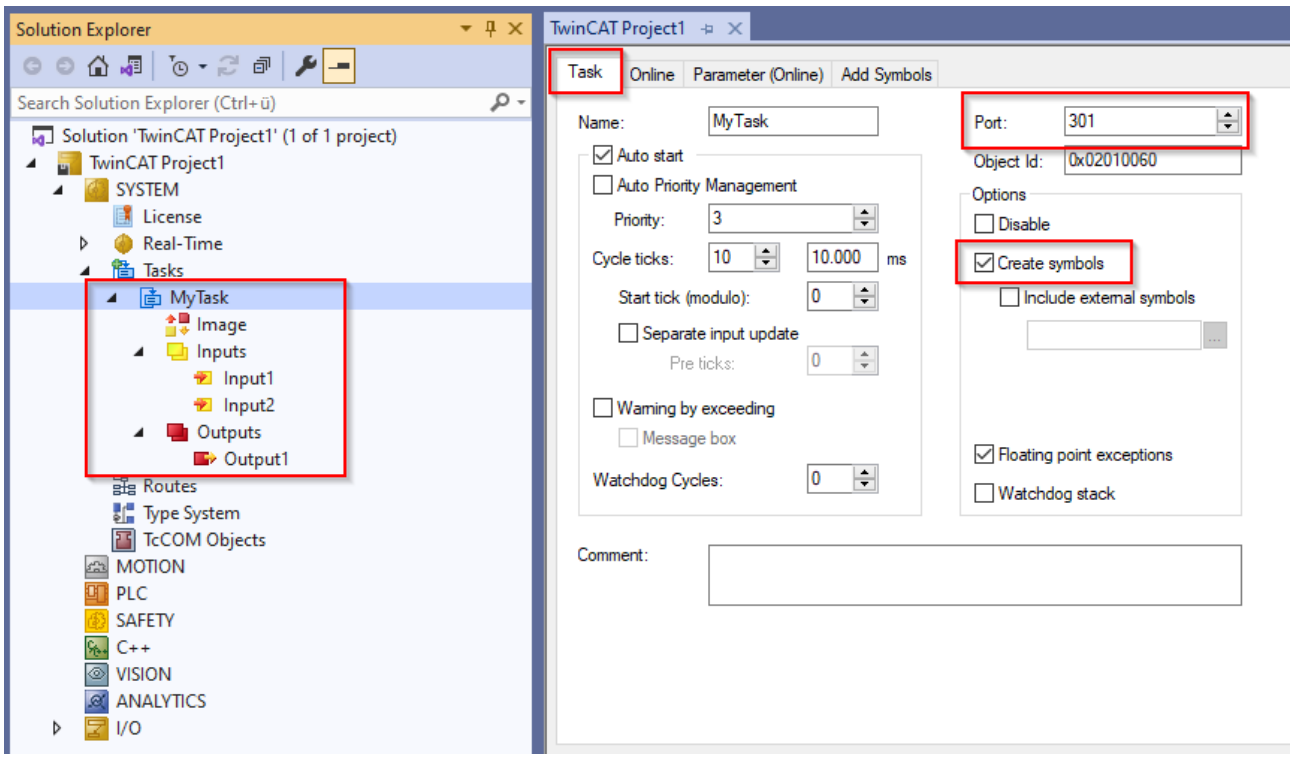
4.11.3.5 I/O 任务

本节将介绍如何配置 TwinCAT OPC UA Server 的命名空间，以便获取包含过程映像的任务符号的访问权限。为此，必须执行以下几个步骤：

- 激活符号
- 服务器的配置

激活符号

您可以通过 OPC UA 提供包含过程映像的任务的变量。为此，您必须在任务设置中启用“Create symbols（创建符号）”选项。这样即可生成符号，并将这些符号作为在线符号 [▶ 67]使用。您还可以在同一对话框中找到 ADS 端口，在稍后的配置过程中将需要使用该端口。

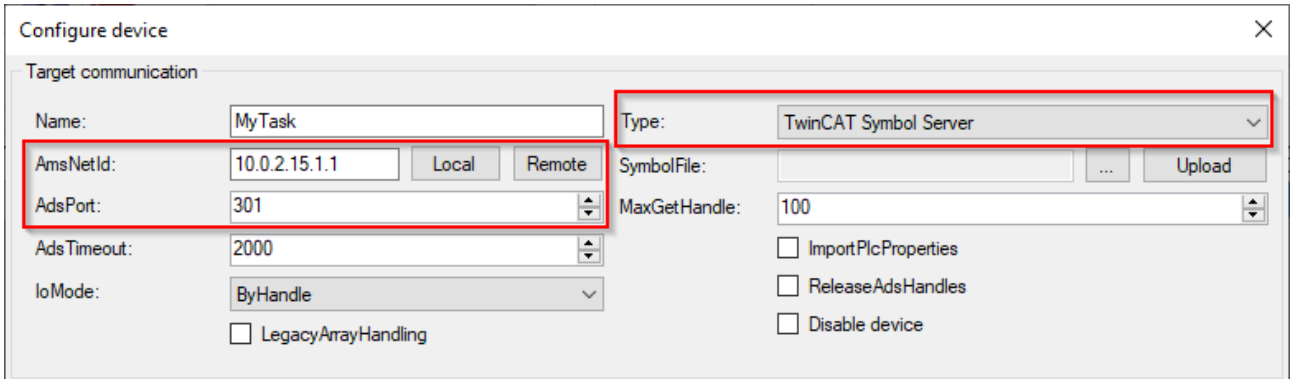


服务器的配置

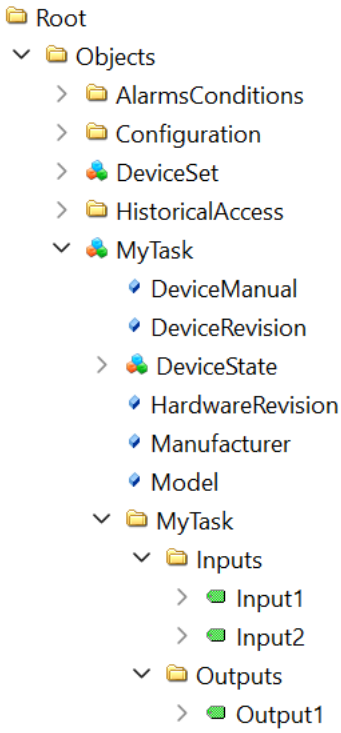
现在，您可以使用 TwinCAT OPC UA Configurator 来配置 TwinCAT OPC UA Server，以便其访问任务的在线符号，并在其地址空间中提供这些符号。

在 TwinCAT OPC UA Configurator 的“Data Access（数据访问）”选项卡中添加一台新设备。选择您设备的 AMS Net ID 并输入任务的 ADS 端口（参见上文）。

在“Type（类型）”字段的选择列表中选择”TwinCAT Symbol Server”条目。



激活 TwinCAT 项目后，符号即准备就绪，TwinCAT OPC UA Server 即可读取这些符号并相应地建立地址空间。



4.11.3.6 在线符号

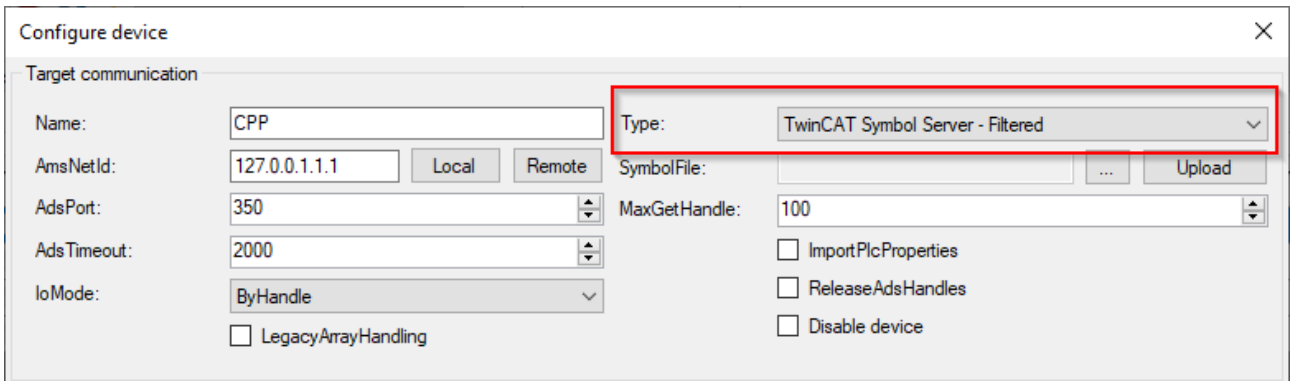
TwinCAT OPC UA Server 可通过所谓的“在线符号”读取符号信息。这是一个由 TwinCAT 中的一些 ADS 服务器提供的标准化 ADS 接口。我们通常将该接口称为“符号上传”。通过该接口，ADS 客户端可以从 ADS 服务器读取所有可用符号，并在必要时对其进行进一步处理。除了符号的地址信息外，还提供了其他信息，如数据类型说明和可选属性。

可选属性可用于显式启用符号或为此目的进行进一步的设置 — 处理方式与符号文件的使用和运行方式相同，例如通过 PLC 中的编译指示或 TwinCAT 3 C++ 的 TMC 代码生成器进行处理。这意味着，您可以通过导入符号文件和在线符号来使用此前设置的所有启用功能。请注意，并非所有的实时环境都允许显式启用符号。在这种情况下，系统始终提供所有符号。下表概述了提供在线符号接口的一些 ADS 服务器，以及是否提供显式符号启用功能。

类型	ADS 端口	符号启用
TwinCAT 2 PLC	801、811、821……	是
TwinCAT 3 PLC	851、852、853……	是
TwinCAT 3 C++ / MATLAB®/Simulink®	350、351、352……	是
TwinCAT 3 EtherCAT Master	27905、27906……	否（启用所有符号）
TwinCAT 3 I/O Task with Image	301、302、303……	否（启用所有符号）

在 TwinCAT OPC UA Configurator 中配置数据访问设备时，在线符号机制有 2 种形式可供使用：

- TwinCAT Symbol Server：导入所有符号
- TwinCAT Symbol Server - 已过滤：只导入启用的符号



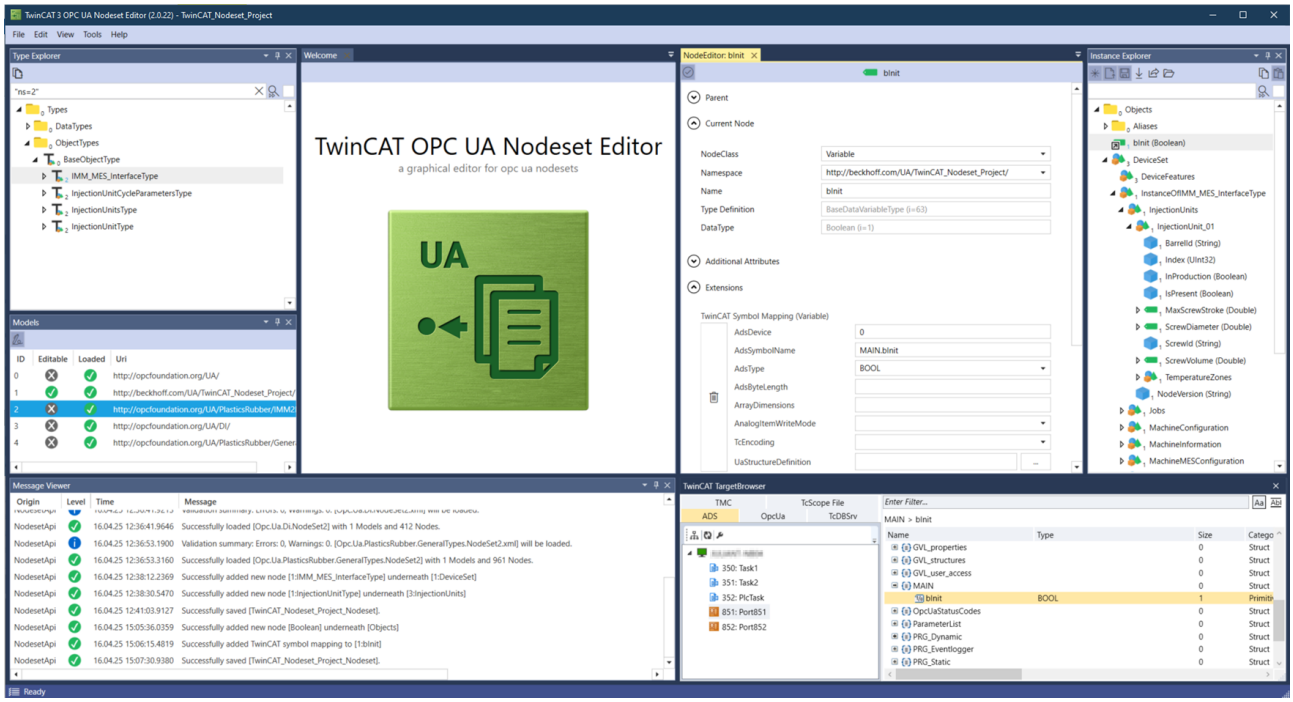
TwinCAT OPC UA Server 的命名空间结构依托于 ADS 符号的命名空间。由于可在此处获取数据，因此也可通过 OPC UA 提供这些数据。

4.11.4 节点集

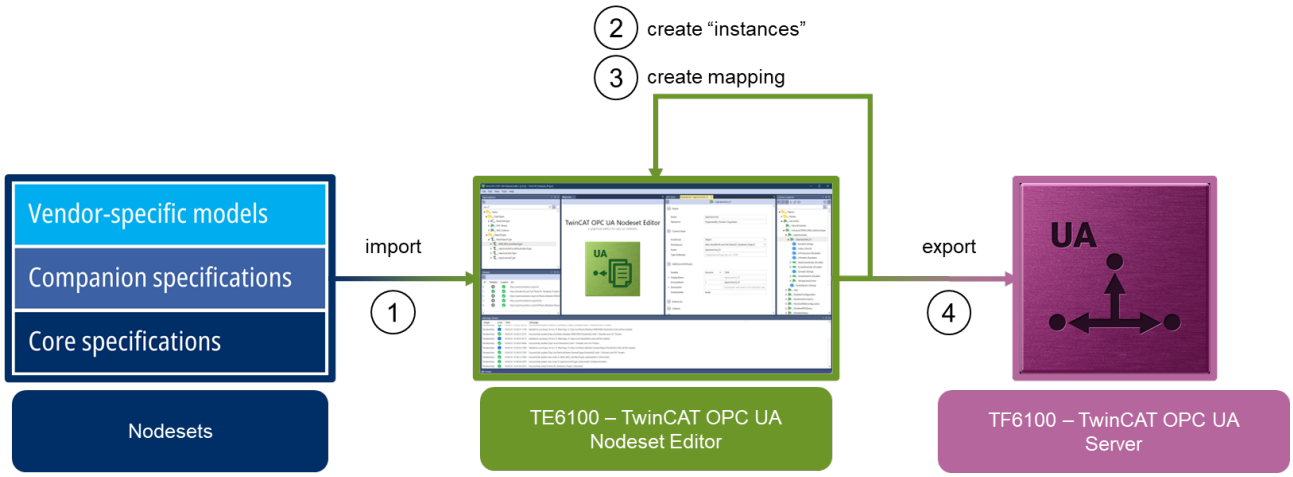
TwinCAT OPC UA Server 支持导入 OPC UA 节点集文件。节点集可以定义数据类型和实例，例如配套规范中包含的数据类型和实例，其在节点集内以 XML 结构提供（由 OPC 基金会进行标准化）。

您可以使用 TE6100 OPC UA Nodeset Editor 软件来创建、修改节点集文件，并与 TwinCAT 实时环境中的符号相连接。然后可将节点集导出至 TwinCAT OPC UA Server 并在该服务器中进行使用。

有关此主题的更多信息，请参见 TE6100 OPC UA Nodeset Editor 产品文档



下方的屏幕截图展示了使用这两种 TwinCAT 功能组件的工作流程示例。



4.11.5 数据类型

根据 PLCopen 映射规范将 IEC 61131-3 数据类型映射到 OPC UA 数据类型。下表显示了指定的映射。

PLC	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 [▶ 69]	Int64
LTIME [▶ 69]	Int64
STRING [▶ 70]	String
WSTRING [▶ 70]	String

4.11.5.1 TIME/LTIME

TIME/LTIME

对于 TIME 和 LTIME 数据类型，需要注意 PLC 与 OPC UA 数据类型之间映射的特殊性，请参见下一节。

TIME 和 LTIME

在 IEC 61131-3 的 PLCopen 映射中，TIME 和 LTIME 均映射为 Int64 数据类型。而 IEC61131-3 数据类型 UDINT 和 ULINT 的数据区则与 UInt32 和 UInt64 相对应。

数据类型	下限值	上限值	映射时间
TIME (UDINT)	0	4294967295	49d17h2m47s295ms
LTIME (ULINT)	0	18446744073709551615	213503d23h34m33s709ms551us615ns
Int64	-9223372036854775808	9223372036854775807	-

服务器在处理这两种数据类型时的性能将在下方的列表中予以说明：

TIME

- 在表中可以看到可能的取值范围。
- 对于所有其他值（<0 和 >4294967295），当 OPC UA 客户端执行写入操作时，服务器将返回状态代码 **BadOutOfRange**。

LTIME

- 表中也可以看到可能的取值范围。
- 对于所有 <0 的值，当 OPC UA 客户端执行写入操作时，服务器将返回状态代码 **BadOutOfRange**。
- >9223372036854775807 的值无法与 Int64 的取值范围进行映射，因此无法通过 OPC UA 写入。但是，可以在 PLC 中写入这些值。在这种情况下，服务器中受影响的 OPC UA 变量的值将变为**空值**。此外，系统还会将变量的 StatusCode 设置为 **BadOutOfRange**。

4.11.5.2 字符串

根据 PLCopen 映射规范，PLC 数据类型 STRING 和 WSTRING 均映射为 OPC UA 数据类型 String。区别在于内存不同：

- STRING：字符长度为 1 个字节。
- WSTRING：字符长度为 2 个字节。

下文将对这两种数据类型进行说明。有关详细信息，请参阅 TwinCAT 3 PLC 文档。

STRING

在 TwinCAT 中，我们可以将 STRING 数据类型解读为 Latin-1 或 UTF-8。
通过 {attribute 'TcEncoding'} 属性控制解读方式：

- 在不带属性的情况下使用 Latin-1。
- 在自带属性的情况下使用 UTF-8。

示例：

```
// Standard: Latin-1
sMyStringText    : STRING := 'This is a STRING';

// UTF-8 from STRING
{attribute 'TcEncoding' := 'UTF-8'}
sMyUTF8Text1     : STRING  := sLiteral_TO_UTF8('The dinner costs 30 €.');

// UFT-8 from WSTRING
{attribute 'TcEncoding' := 'UTF-8'}
sMyUTF8Text2     : STRING  := wsLiteral_TO_UTF8("The dinner costs 30 €.");

// UFT-8 simple STRING
{attribute 'TcEncoding' := 'UTF-8'}
sMyUTF8Text3     : STRING  := 'Hello World.';
```

▾ Value	
SourceTimestamp	1/21/2026 1:25:24.973 AM
SourcePicoseconds	0
ServerTimestamp	1/21/2026 1:25:24.973 AM
ServerPicoseconds	0
StatusCode	Good (0x00000000)
Value	The dinner costs 30 €.
▾ DataType	STRING
NamespaceIndex	3
IdentifierType	Numeric
Identifier	3013

WSTRING

WSTRING 数据类型采用的是 UCS2 编码（每个字符 2 个字节）。初始化时务必使用双引号。

示例：

```
wsMyWSTRING : WSTRING := "This is a WSTRING.";
```

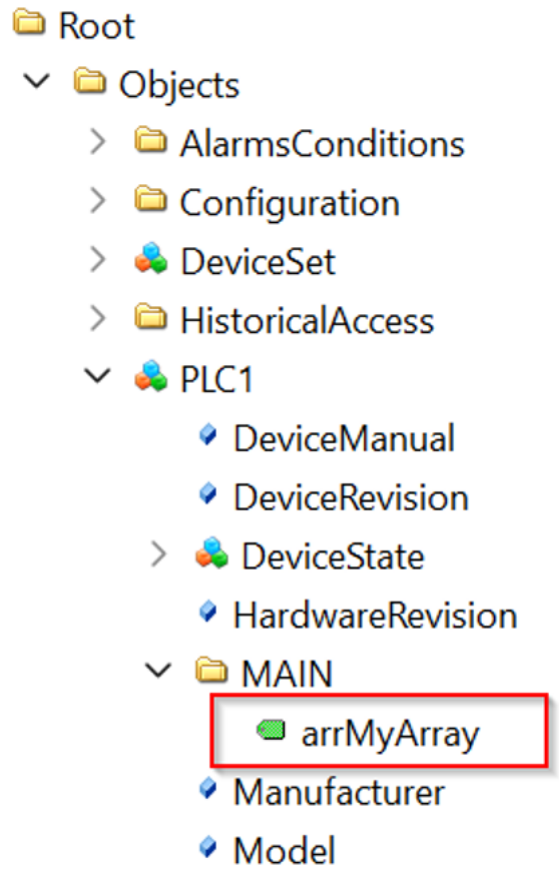
Value	
SourceTimestamp	1/21/2026 1:29:20.945 AM
SourcePicoseconds	0
ServerTimestamp	1/21/2026 1:29:20.945 AM
ServerPicoseconds	0
StatusCode	Good (0x00000000)
Value	This is a WSTRING.
DataType	String
NamespaceIndex	0
IdentifierType	Numeric
Identifier	12 [String]

4.11.6 数组

默认情况下，数组在服务器地址空间中以独立节点的形式进行映射。OPC UA 客户端仍可选择访问数组中的各个元素。

因此，PLC 中的以下数组将以单个节点的形式显示，并通过节点属性来定义数组限值。

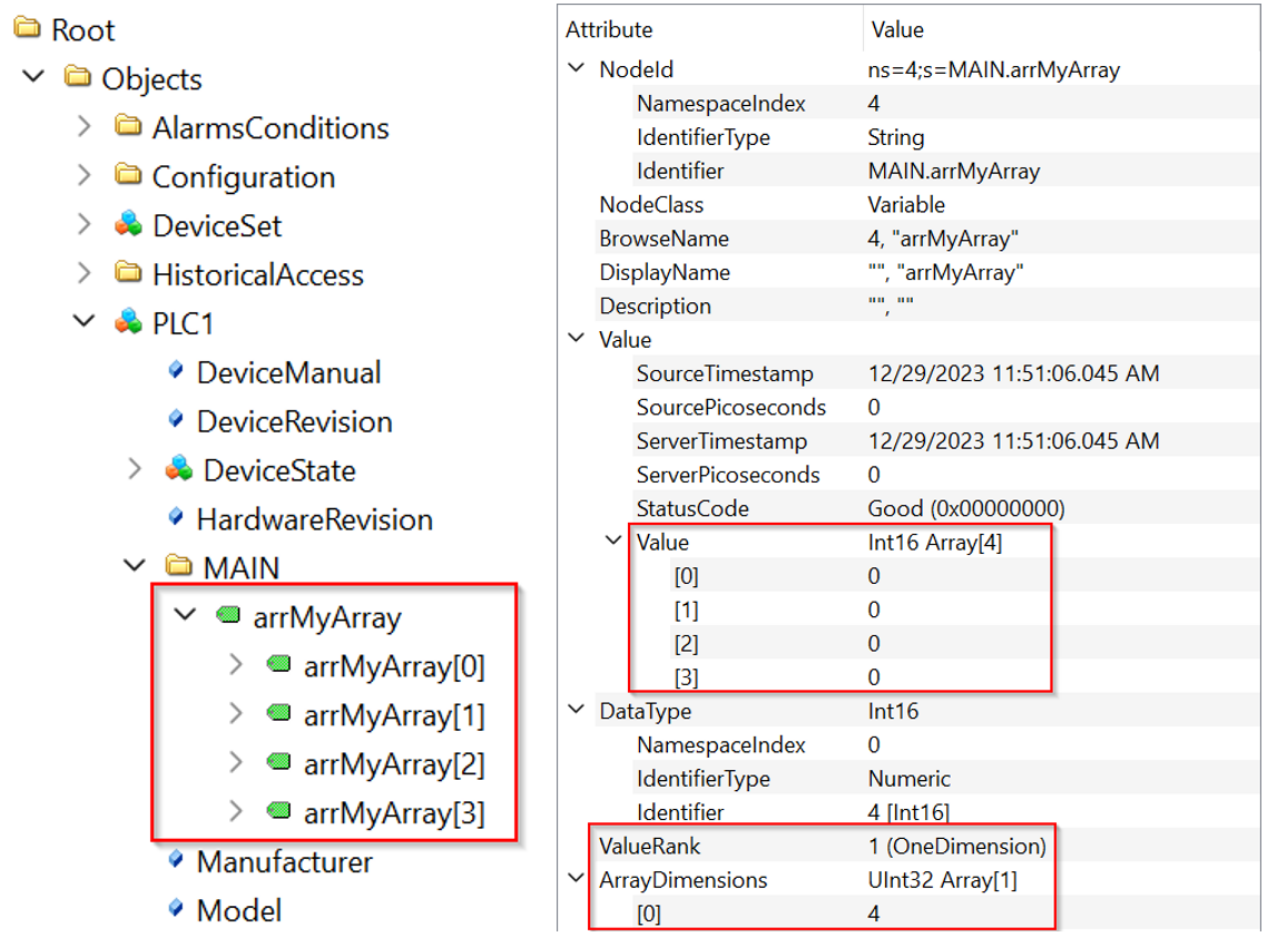
```
{attribute 'OPC.UA.DA' := '1'}
arrMyArray : ARRAY[0..3] OF INT;
```



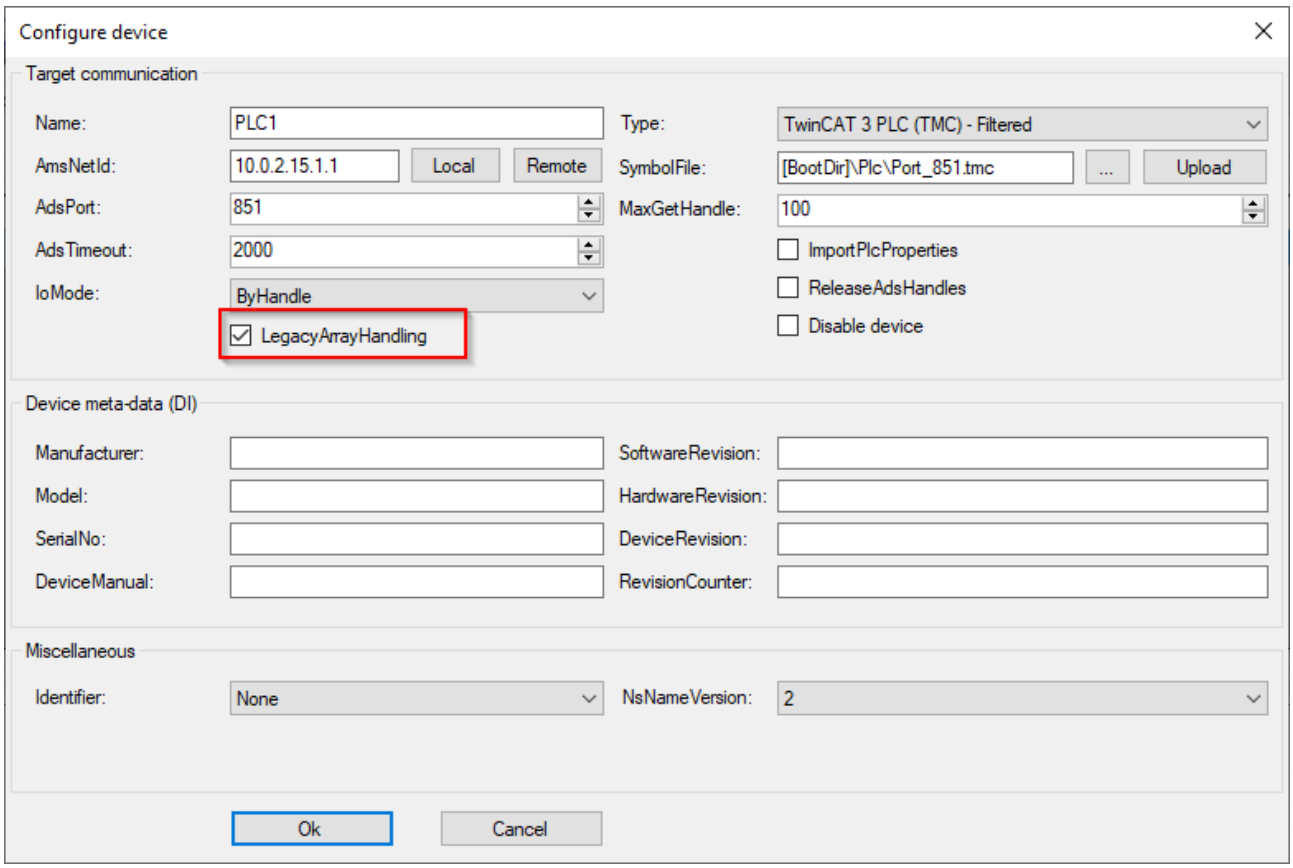
Attribute	Value
NodeId	ns=4;s=MAIN.arrMyArray
NamespaceIndex	4
IdentifierType	String
Identifier	MAIN.arrMyArray
NodeClass	Variable
BrowseName	4, "arrMyArray"
DisplayName	"", "arrMyArray"
Description	"", ""
Value	
SourceTimestamp	12/29/2023 11:51:06.045 AM
SourcePicoseconds	0
ServerTimestamp	12/29/2023 11:51:06.045 AM
ServerPicoseconds	0
StatusCode	Good (0x00000000)
Value	Int16 Array[4]
[0]	0
[1]	0
[2]	0
[3]	0
DataType	Int16
NamespaceIndex	0
IdentifierType	Numeric
Identifier	4 [Int16]
ValueRank	1 (OneDimension)
ArrayDimensions	UInt32 Array[1]
[0]	4

其优势在于大大降低了服务器地址空间的复杂度和内存占用，因为不需要将数组的每个位置都作为命名空间中的独立节点。

旧版的 OPC UA 客户端可能还不支持此功能。因此，通过一个特殊的配置开关，可以将数组中的所有独立元素作为命名空间中的独立节点来提供。如下图所示：



除了数组的根元素外，所有独立元素也可以作为独立节点使用。通过在 TwinCAT OPC UA Configurator 中激活数据访问设备上的“LegacyArrayHandling”选项，即可使用该配置开关。



存储需求增加

激活此配置开关可大大增加服务器的地址空间。根据项目范围的不同，特别是在使用嵌套的复杂数组时，此操作会增加 TwinCAT OPC UA Server 的内存负载。
请确保硬件设备有足够的主存。

4.11.7 枚举

OPC UA 中的枚举始终采用 Int32 数据类型。但是，IEC 61131-3 标准允许定义数据类型大于 Int32 的枚举。为确保正确处理这些枚举，TwinCAT OPC UA Server 提供了配置选项 <ImportBigEnumsNumeric>，可在其数据访问配置文件中启用该选项。

该选项的默认设置为 FALSE。这意味着，如果枚举值超出了 Int32 的范围，系统将触发状态代码异常 BadOutOfRange。

如果将该选项设置为 TRUE，系统会将数据类型大于 Int32 的枚举变量视为该特定数据类型的常规变量。

假设我们在 PLC 代码中定义了以下枚举：

```
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
```

如果激活了上述配置选项，系统会将 E_Enum_VeryBig 的实例作为服务器命名空间中数据类型为 Int64 的常规变量来处理，而 E_Enum_Normal 和 E_Enum_NotSoBig 的实例则作为 OPC UA 枚举（数据类型为 Int32）来处理：

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

4.11.8 结构

要求

该功能仅适用于基于 TwinCAT 3 的数据访问 [▶ 48]设备和 TMC 符号文件 [▶ 52]的导入。

TwinCAT OPC UA Server 支持对 TwinCAT 3 PLC 中的结构使用所谓的 StructuredTypes。StructuredTypes 允许以确保数据一致性的方式读取或写入结构，并向客户端提供该结构的类型描述。

您可以使用特殊的编译指示将结构定义为 StructuredType 并使用该结构。如果未设置此编译指示，系统不会将结构作为 StructuredType 加载到服务器的地址空间，而是作为 FolderType 显示。结构的成员变量将以可访问的独立节点形式显示。

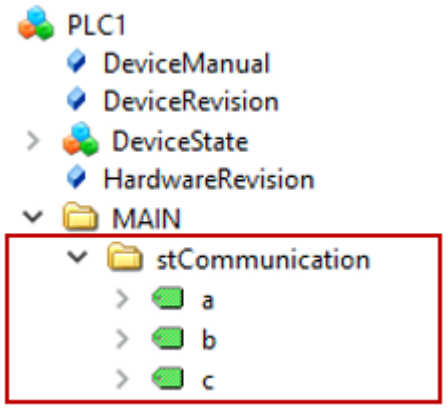
以下是 TwinCAT 3 PLC 中结构的一个示例：

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

该结构现已在 MAIN 程序中完成实例化，并且已按照“启用（PLC）符号 [▶ 53]”一章中的说明通过编译指示为 OPC UA 启用了该结构。

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

现在，服务器中默认提供的结构实例如下：

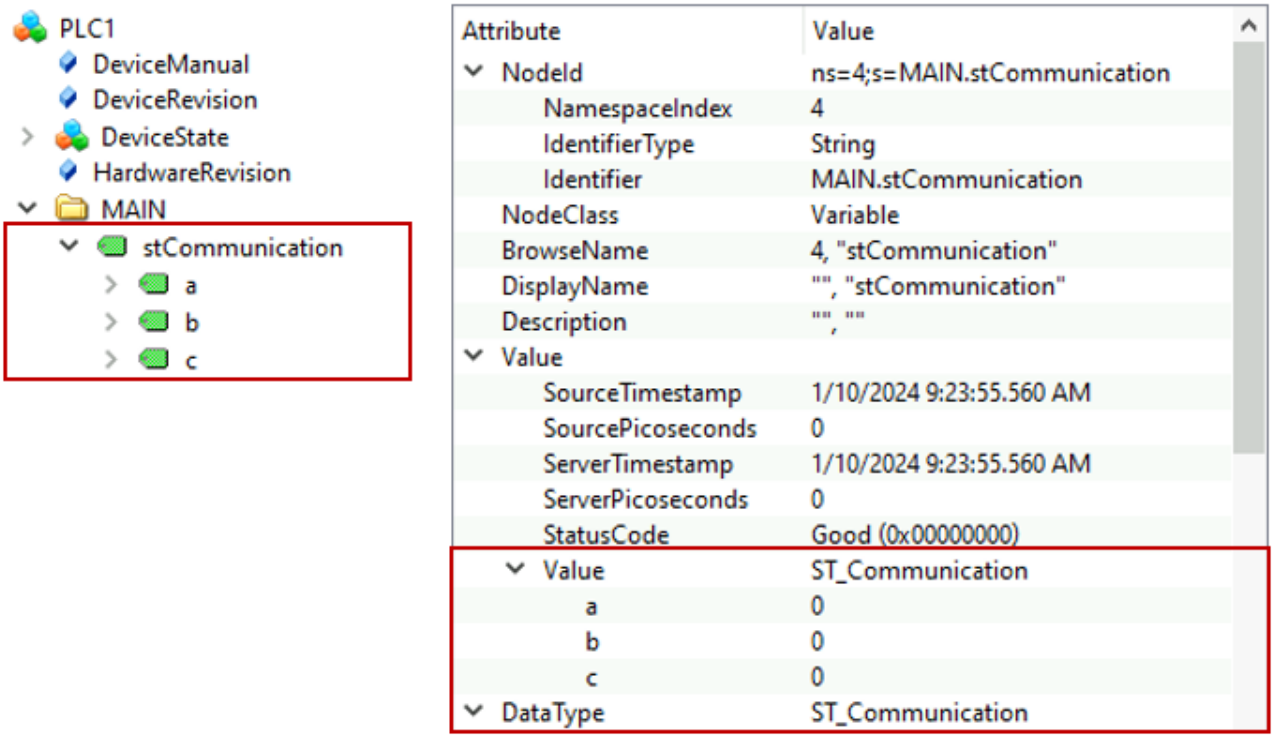


现在可以访问结构的成员变量，但无法访问结构的根元素。因此，可能无法对整个结构进行数据一致性访问。

现在，我们使用其他编译指示，将该结构实例定义为 StructuredType。

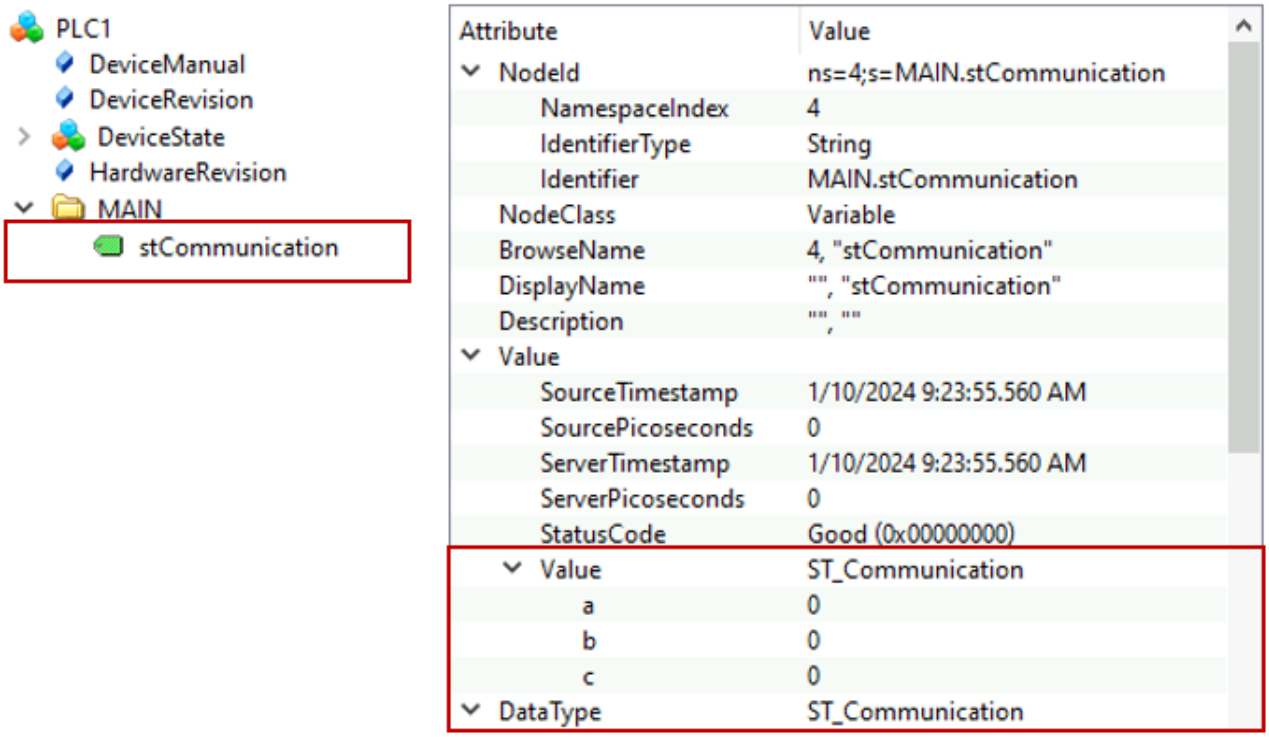
```
PROGRAM MAIN
VAR
  {attribute 'OPC.UA.DA' := '1'}
  {attribute 'OPC.UA.DA.StructuredType' := '1'}
  stCommunication : ST_Communication;
END_VAR
```

然后在 TwinCAT OPC UA Server 中提供实例，如下图所示：



在读取根元素时，我们通过 OPC UA 以确保数据一致性的方式提供结构信息，然后客户端即可对这些结构信息进行处理。此外，成员变量也以独立节点的形式显示，可供用户访问。您可以通过特殊属性停用该功能。然后，只能通过 StructuredType 访问成员变量，如以下示例所示：

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



这主要是对服务器内存的优化，“优化 [▶ 42]”一章对此也进行了介绍，因为必须为地址空间中的每个节点分配主存。

您不必在结构的每个实例中都放置编译指示，也可以在结构定义时放置一次编译指示。在这种情况下，我们会为 OPC UA 将该结构的所有实例作为 StructuredType 自动启用。

```
{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
```

在这种情况下，您也可以使用以下属性，显式停用某些实例的 StructuredType 定义：

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

功能块的 StructuredType

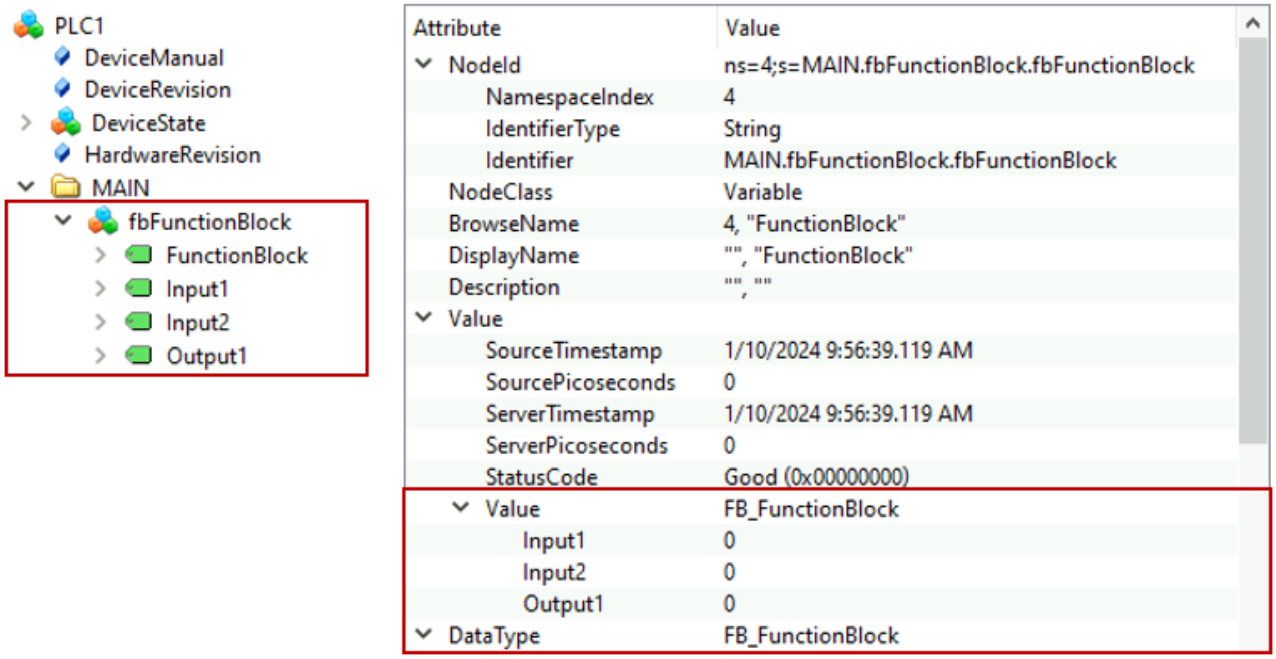
您还可以在 PLC 功能块中使用 StructuredTypes。在这种情况下，某个功能块实例会收到一个名为“FunctionBlock”的子节点，该子节点以 StructuredType 的形式表示整个功能块。
请参见下方的示例：

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

在 MAIN 程序中创建了一个功能块实例，为 OPC UA 启用了这两个实例并通过编译指示将其定义为 StructuredType。

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

然后，TwinCAT OPC UA 将提供以下功能块实例：



上述适用于结构的编译指示定义位置也适用于功能块。在这里，您还可以显式隐藏功能块的成员变量。

限制条件

使用 StructuredTypes 时需遵循以下限制条件。

指针和引用

如果在结构中使用了指针和引用类型，则无法将其转换为 StructuredType。然后，OPC UA 服务器会将这些结构呈现为包含相应成员变量的常规 FolderType。

结构的最大占用空间

默认情况下，结构的最大占用空间限值为 16 kB。如果结构的占用空间超过此限值，结构实例将以 FolderType 的形式显示。如有必要，您可以提高该限值。有关背景的详细说明，请参见“优化 [▶ 42]”一章。每个 STRUCT 会不断与基本 ADS 设备交换数据，即每次对 StructuredType 执行读取/写入命令时都会发送一条大型 ADS 消息。为防止 ADS 路由器被大量大型消息淹没，系统对最大占用空间进行了限制。

4.11.9 属性

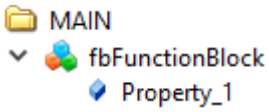
要求

该功能仅适用于基于 TwinCAT 3 的数据访问 [▶ 48]设备和 TMC [▶ 52] 符号文件的导入，以及在线符号。

自 TwinCAT 3.1 Build 4024 版本起，系统已支持在服务器命名空间中显示 PLC 属性。您只需在“Property（属性）”上设置 2 个特殊的 PLC 属性，即可将该属性导入命名空间，并作为 UA 属性在命名空间中显示。示例：

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

定义该属性的功能块必须包含用于启用符号的 PLC 常规属性。



在配置文件 TcUaDaConfig.xml 中，可以对 PLC 属性的处理方式进行全局定义。为此，我们使用了“ImportPlcProperties”标志。

值	描述
false	显示了 OPC UA 命名空间中同时设置了 OPC.UA.DA.Property 和监控属性的所有属性。
true	显示了 OPC UA 命名空间中已设置监控属性的所有属性。

4.11.10 StatusCode

要求

该功能仅适用于基于 TwinCAT 3 的数据访问 [▶ 48]设备和 TMC 符号文件 [▶ 52]的导入。

5.x 安装版本的新功能

下文将介绍 5.x 安装版本中 TwinCAT OPC UA Server 的功能。随后将介绍 [▶ 80] 4.x. 安装版本中 TwinCAT OPC UA Server 的功能。

TwinCAT OPC UA Server 支持为特定的 PLC 符号调整 OPC UA 状态代码。请执行以下步骤，以便在运行时更改符号的 StatusCode。

创建结构

为确保数据一致性，该概念基于一种结构（参见“”）。必须将要更改 StatusCode 的每个符号封装在一个结构中。

创建一个新结构，并在结构定义前添加以下编译指示。该结构包含符号本身和一个数据类型为 DINT 的变量，后者以十进制形式表示 StatusCode。

```
{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
```

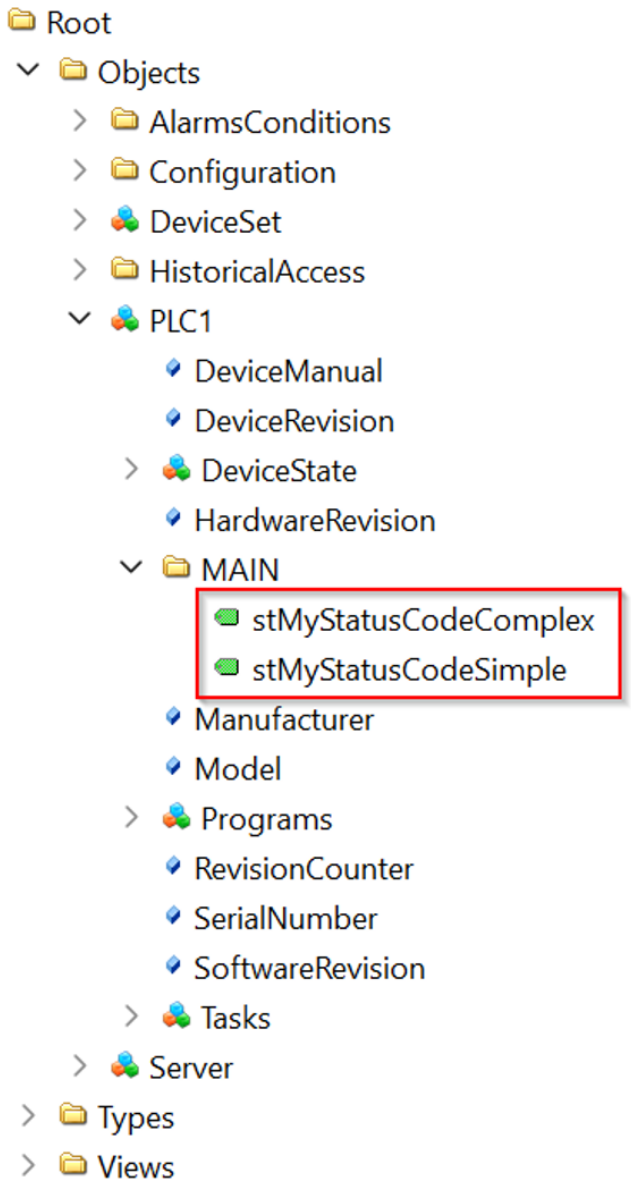
您也可以使用复杂符号，例如：

```
{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
```

现在，请在 MAIN 程序中创建该结构的实例，并添加编译指示，以便通过 OPC UA 启用该实例。

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

在 TwinCAT OPC UA Server 中，现在已为定义完成的结构实例提供了定义为“value（值）”的成员变量的数据类型。



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]

运行时更改 StatusCode

若要在运行时更改 StatusCode，只需编辑成员变量“status（状态）”的值即可。如果将其设置为“-2147155968”（见上文的示例），StatusCode 将更改为“BadCommunicationError”。

MAIN [Online]

TwinCAT_Project2.Untitled1.MAIN

Expression	Type	Value
stMyStatusCodeSimple	ST_StatusCodeSimple	
value	REAL	0
status	DINT	-2147155968
stMyStatusCodeComplex	ST_StatusCodeCom...	

#	Server	Node Id	Display Name	Value	Datatype	Source Timestamp	Server Timestamp	Statuscode
1	TcOpcUaServer...	NS4|String|MAIN.stMyStatusCodeSimple	stMyStatusCodeSimple	0	Float	11:24:44.038 AM	11:24:44.038 AM	BadCommunicationError

但是，如果将该值设为“0”，StatusCode 将更改为“Good（好）”。

MAIN [Online]

TwinCAT_Project2.Untitled1.MAIN

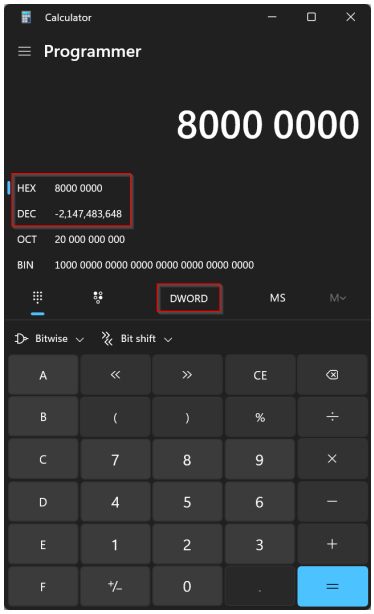
Expression	Type	Value
stMyStatusCodeSimple	ST_StatusCodeSimple	
value	REAL	0
status	DINT	0
stMyStatusCodeComplex	ST_StatusCodeCom...	

#	Server	Node Id	Display Name	Value	Datatype	Source Timestamp	Server Timestamp	Statuscode
1	TcOpcUaServer...	NS4|String|MAIN.stMyStatusCodeSimple	stMyStatusCodeSimple	0	Float	11:26:08.539 AM	11:26:08.539 AM	Good

此处使用的十进制值由 OPC UA 规范定义，并且映射了 StatusCode 的当前版本。下表列出了一些常见的 StatusCode 及其数字表示法。

StatusCode	十六进制	十进制
Good	0x00000000	0
Uncertain	0x40000000	1073741824
Bad	0x80000000	-2147483648
BadUnexpectedError	0x80010000	-2147418112
BadInternalError	0x80020000	-2147352576
BadCommunicationError	0x80050000	-2147155968
BadTimeout	0x800A0000	-2146828288
BadServiceNotSupported	0x800B0000	-2146762752

根据状态代码的十六进制表示法计算其十进制表示法时，请注意将您的计算机（例如 Windows 计算器）设置为 DWORD（“程序员”视图）。



在 4.x 安装版本中使用

在 4.x 安装版的旧版功能中，变量在 OPC UA 端不以单个变量的形式显示，而是以与 PLC 端相同的方式显示为一个结构。属性配置方面也存在差异。

```
{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
```

StatusCode 只能通过 OPC UA 在结构变量中使用。“Value（值）”和“Quality（质量）”值包含值和状态代码的值。

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

4.11.11 模拟项目类型

要求

该功能仅适用于基于 TwinCAT 3 的数据访问 [▶ 48]设备和 TMC 符号文件 [▶ 52]的导入。

AnalogItemType 是 OPC UA 规范的一部分，支持将单位等元信息附加到变量上。您可以在 TwinCAT 3 PLC 中以 PLC 属性的形式定义这些元信息项。

可以设置以下参数：

- EngineeringUnits：OPC UA 规范定义的单位
- EURange：变量的最大值范围
- InstrumentRange：变量的最大值范围
- WriteBehavior：在写入操作过程中超出数值范围时需采取的操作。

以下示例显示了如何将 fillLevel 变量配置为 AnalogItemType。设置了以下参数：

- Unit：20529（“百分比”，在 OPC UA 规范中定义）
- Max. value range：0 至 100
- Normal value range：10 至 90
- Write behavior：1（夹紧）

```
{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;
```

可以使用 OPC UA（OPC UA 规范第 8 部分）中指定的 ID 来配置 EngineeringUnits。这些 ID 是基于“联合国贸易便利化和电子商务中心”发布的被广泛使用和接受的“计量单位代码（第 20 号建议）”建立的。OPC UA 根据规范将指定三位字母数字单位 ID 的 CommonCode 转换为 Int32 值并进行引用（摘自 OPC UA 规范 v1.02，伪代码）：

```
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
}
```

写入操作

在写入 AnalogItemType 变量时，您可以定义 OPC UA 服务器应如何处理与值范围相关的新值。提供以下选项：

- 0：允许使用所有值，写入操作时也接受所有值。
- 1：根据值范围截断要写入的值。
- 2：如果要写入的值超出值范围，该值将会遭到拒绝。

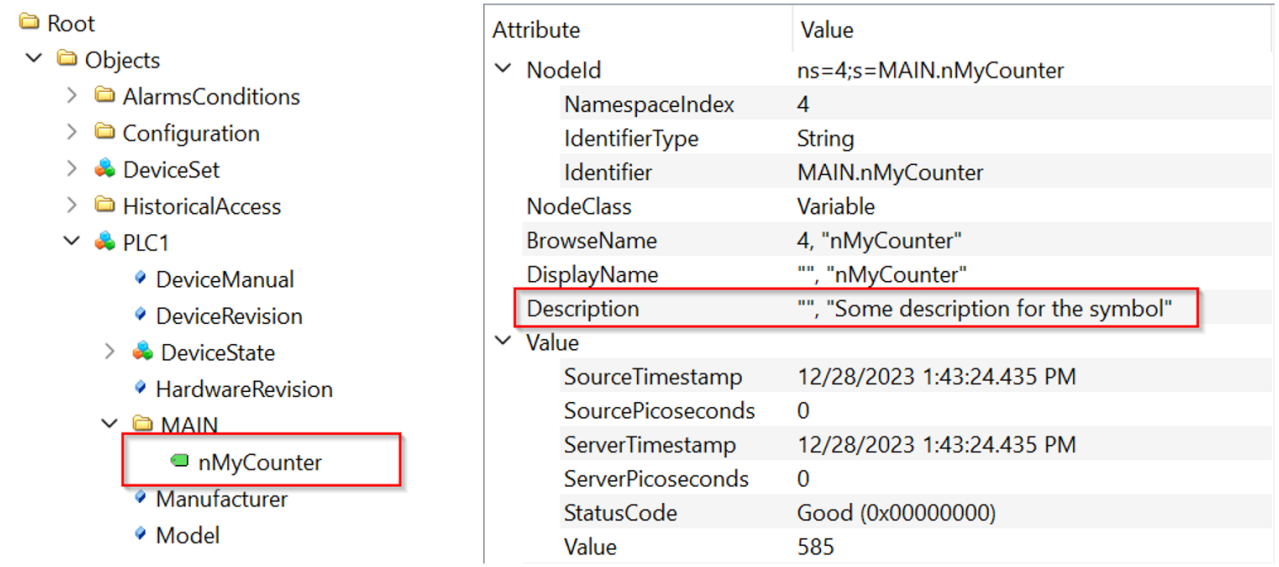
4.11.12 描述

要求

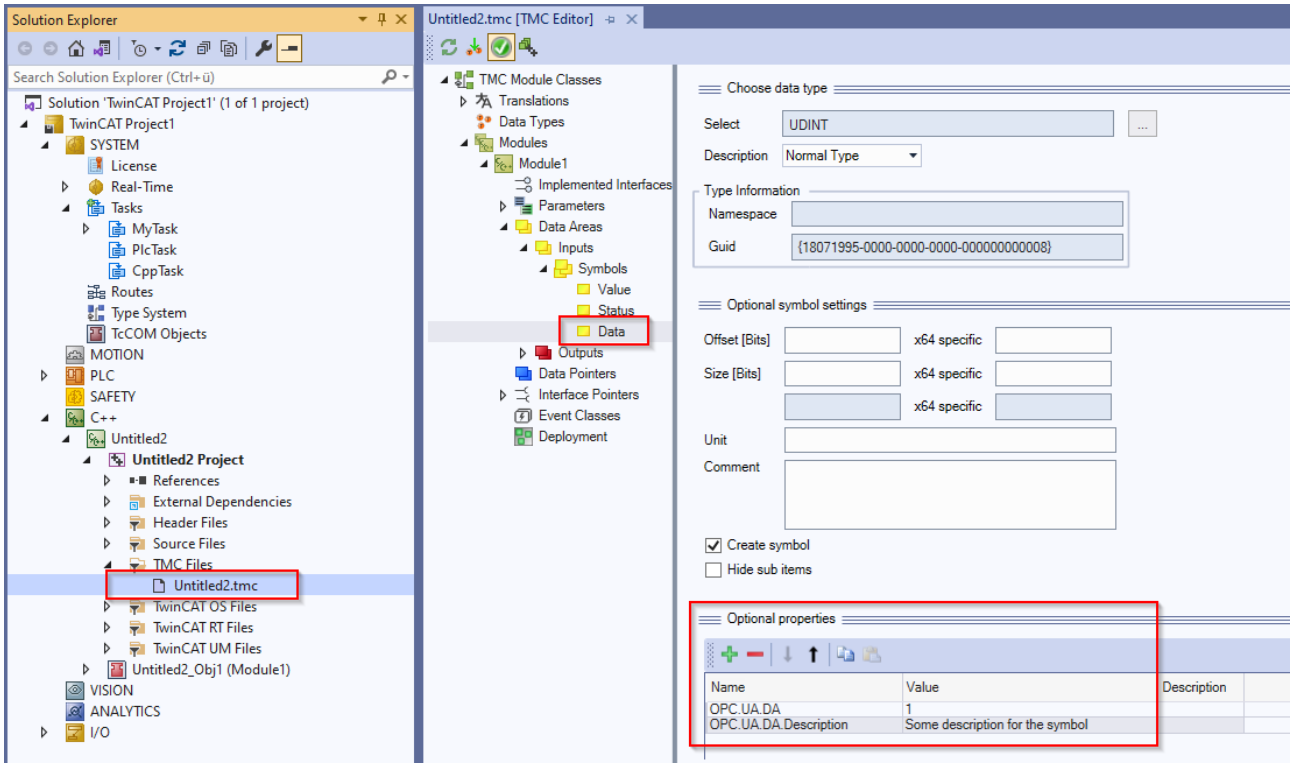
该功能仅适用于基于 TwinCAT 3 的数据访问 [▶ 48]设备和 TMC 与 TMI [▶ 52] 符号文件的导入，以及在线符号。

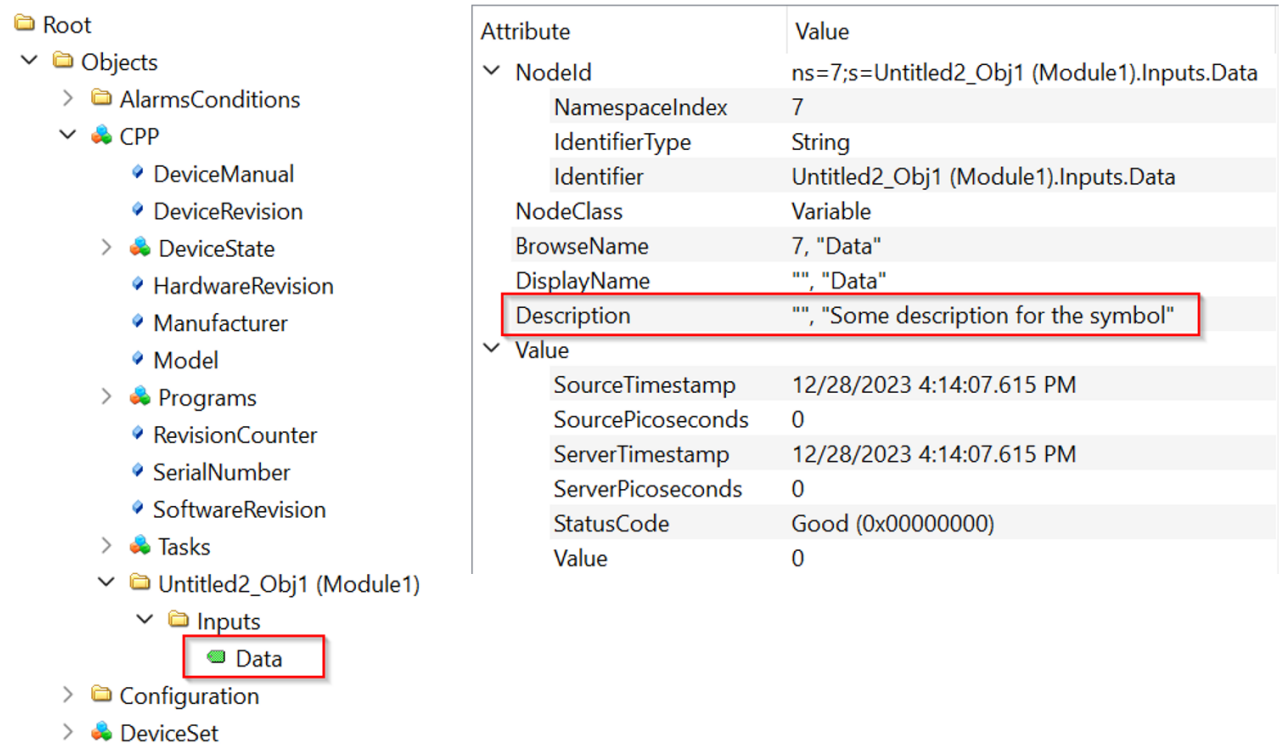
以下 TwinCAT 3 PLC [▶ 53] 编译指示可用于设置符号的 OPC UA 属性“Description（描述）”。

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



对于 TwinCAT 3 C++ [▶ 55]，您也可以在 TMC 代码生成器中使用该编译指示设置节点描述。请注意，只有在 TwinCAT 3 C++ 下使用在线符号 [▶ 67]时，才能使用该功能。





4.11.13 ReadOnly

要求

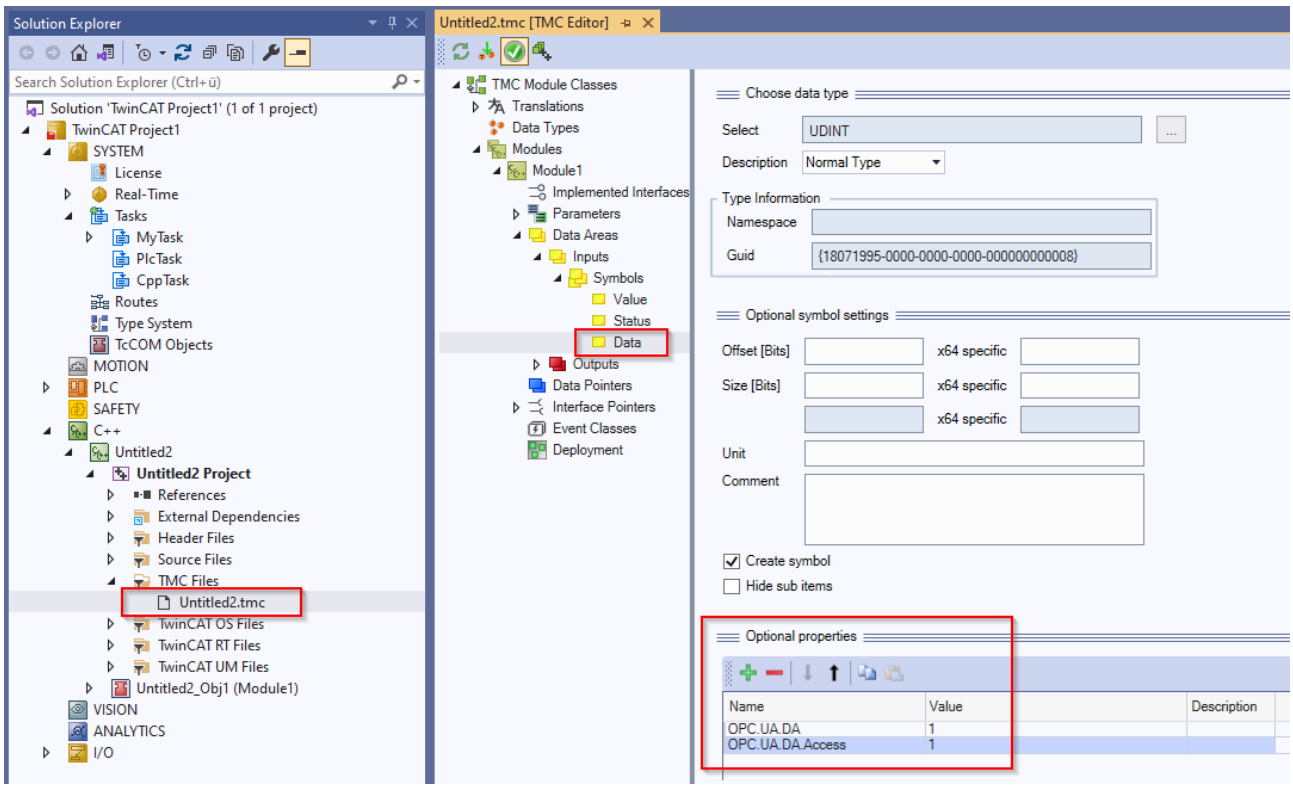
该功能仅适用于基于 TwinCAT 3 的数据访问 [▶ 48]设备和 TMC 符号文件 [▶ 52]的导入。

以下编译指示可用于将符号设置为只读模式。然后，服务器将通过 StatusCode 为 BadNotWriteable 的 OPC UA 客户端确认写入操作。

```
{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]

对于 TwinCAT 3 C++ [▶ 55]，您也可以在 TMC 代码生成器中使用该编译指示将节点标记为只读。请注意，只有在 TwinCAT 3 C++ 下使用在线符号 [▶ 67]时，才能使用该功能。



4.11.14 指针和引用

指针

服务器一般不会在命名空间中表示指针变量（如 POINTER TO）。如果指针变量位于某个结构中，并且该结构的配置为 Structured Data Type，那么该结构将不会显示为 Structured Data Type，而是显示为 FolderType。

引用

服务器在命名空间中将引用变量（REFERENCE TO）表示为单个变量，并且可以不受限制地读取这些变量。如果在结构内部进行引用，该结构在服务器中将无法再以 的形式使用，而只能以 FolderType 的形式使用。但是可以访问结构中的个别引用变量。

4.11.15 类型系统

要求

该功能仅适用于基于 TwinCAT 3 的数据访问 [▶ 48]设备和 TMC [▶ 52] 符号文件的导入，以及在线符号。

OPC UA 的最大优势之一在于元模型，它可以用来提供基本类型，也可以用自定义模型来扩展类型系统。我们还采用同样的机制来表示真实对象（节点），以便 OPC UA 客户端能够确定对象类型。

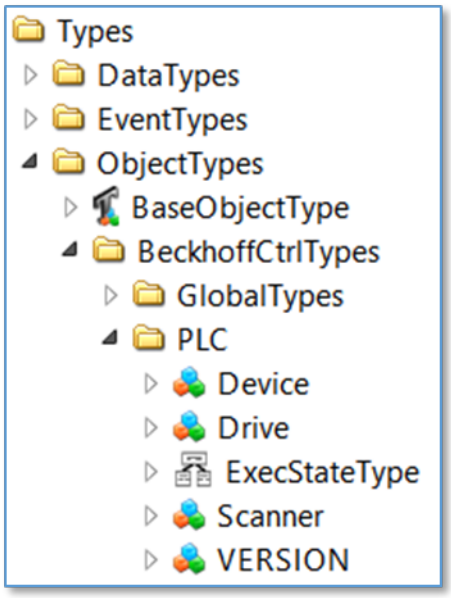
OPC UA 服务器在其命名空间中发布 IEC61131 提供的类型信息。其中不仅包括 BOOL、INT、DINT 或 REAL 等基本类型信息，还包括表示对象的当前类别（功能块）或结构等扩展类型信息。

类型信息

类型信息是 UA 命名空间的一部分。OPC UA 服务器对基本类型信息进行了如下扩展：

- 仅在 1 个运行时有效的本地类型信息与运行时符号存储在同一个命名空间中。
- 可在不同运行时有效的全局类型信息存储在单独的全局命名空间中。

类型系统也以虚拟形式呈现，可在 OPC UA 服务器的“Types（类型）”区域进行查看：

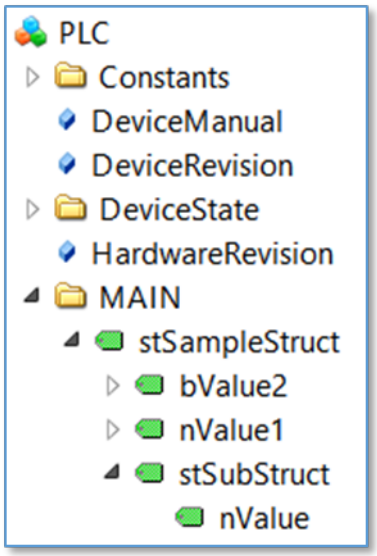


所有非标准数据类型均在 BeckhoffCtrlTypes 区域输入。

基本信息

假设 TwinCAT 3 PLC 由具有不同 STRUCT 的 PLC 程序组成。每个 STRUCT 在 UA 命名空间中均以节点的形式表示，结构中的每个元素都是一个下级节点。

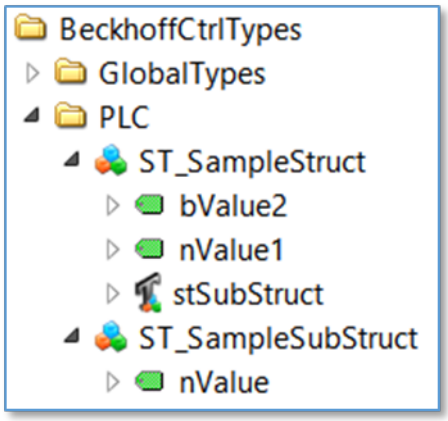
在该示例中，STRUCT stSampleStruct 包含 3 个下级元素：1 个 INT 类型的变量 nValue1、1 个 BOOL 类型的变量 bValue2 和 1 个 STRUCT stSubStruct，后者只包含 1 个下级元素（INT 类型的变量 nValue）。



结构本身是 UA 命名空间中的一个常规变量，数据类型为 ByteString。因此，客户端只能与根元素（结构本身）连接，可以通过解读 ByteString 来读取/写入根元素的值。为了简化对每个下级元素的解读，类型系统中包含了关于结构本身的更多信息，这些信息主要位于实例引用中：

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

此外，类型系统中还包含了关于 ST_SampleStruct 的更多信息：



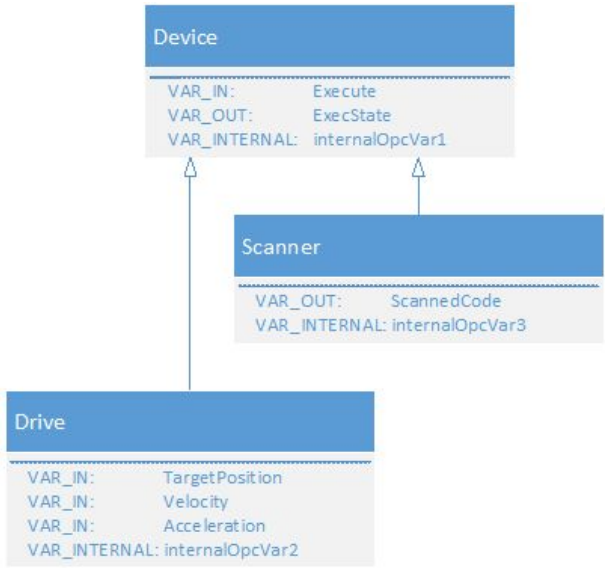
而在 ST_SampleStruct 的引用中：

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

总之，如果客户端想进一步解读该结构，类型系统可以提供非常有用的信息。

面向对象的扩展

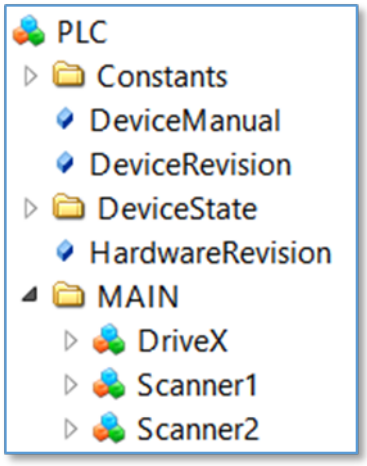
假设 TwinCAT 3 PLC 运行时包含一个 PLC 程序，其结构的外观显示如下：



Scanner 和 Drive 这两个功能块是通过使用 IEC61131-3 面向对象的扩展，从基类 Device 衍生而来。现在，MAIN 程序包含以下声明：

```
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
```

这三个对象都属于 Device 类型，但也属于其特殊的数据类型。OPC UA 服务器导入对象的方式如下：



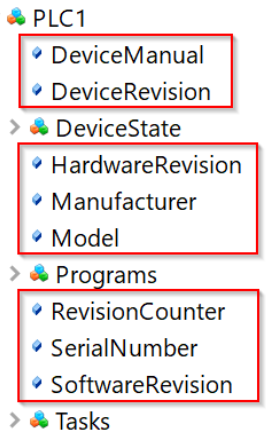
现在可以在每个对象的引用中确定基本数据类型，例如对象 Scanner1：

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

根据 IEC61131 基本程序，对象 Scanner1 属于 Scanner 数据类型，同时还显示了其中包含的变量以及变量的类型：输入、输出或内部（局部）变量。从上图可以看出，这里不仅显示了实际功能块的变量，还显示了基类功能块的衍生变量。整个 IEC61131-3 继承链均在 UA 命名空间中得以呈现。

4.11.16 DI 组件

服务器上的每个 PLC 命名空间都包含若干节点，可用于指定 PLC 的静态元信息。可在每个命名空间的 TcUaDaConfig.xml 中指定这些可选信息。



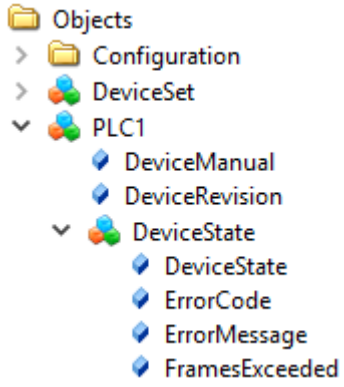
下表提供了有关这些节点的更多信息。各个节点的具体分配值在很大程度上可能与具体应用有关，因此服务器在交付状态下只使用示例值。服务器本身会在其命名空间中提供这些节点，但不会独立更改其分配值。

节点	描述
DeviceManual	可帮助您指定一个可以找到设备手册的地址，例如文件系统中的路径或网址。
DeviceRevision	包含硬件组件或整个设备的修订等级。

节点	描述
HardwareRevision	包含硬件的修订等级。
Manufacturer	包含设备制造商名称，通常为 FQDN（完全合格域名），如 beckhoff.com。
Model	包括“产品”名称（若适用）。
RevisionCounter	可包括设备配置更新次数的计数器。
SerialNumber	设备制造商分配的设备唯一序列号。
SoftwareRevision	包含软件组件、硬件组件的固件甚至设备固件的版本或修订等级。

4.11.17 DeviceState

TwinCAT OPC UA Server 中的每个命名空间均包含一个 DeviceState 对象。



该对象通过各种属性指示下级 ADS 设备的状态。

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

如果设备处于 ERROR 状态，ErrorCode 属性将返回以下值：

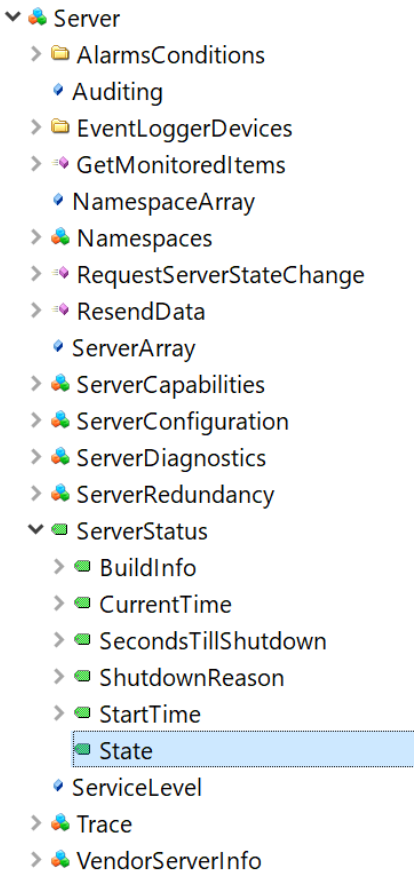
```
#define UA_DEVSTATE_INVALID_STATE 0x80EB0010
#define UA_DEVSTATE_CREATE_NS_FAILED 0x80EB0011
#define UA_DEVSTATE_LOAD_NS_FAILED 0x80EB0012
#define UA_DEVSTATE_INVALID_IO_SETTING 0x80EB0100
```

相应的错误消息会显示在 ErrorMessage 属性中。

4.11.18 ServerState

服务器命名空间中的 ServerState 变量可指示服务器的当前状态。下表对可能的变量值进行了概述。

变量值	描述
Running	已成功启动服务器。
Failed	在服务器配置文件中发现了一个问题，例如 TcUaSecurityConfig.xml 中的配置无效。
NoConfiguration	服务器尚未初始化。
Suspended	服务器尚未完全启动，即并非所有功能均可用。



4.12 历史访问

4.12.1 概述

本章将介绍在 TwinCAT OPC UA Server 命名空间中配置用于**历史访问**（HA）的变量所需的步骤。

历史访问是一项 OPC UA 功能，其中，（历史）变量值存储在存储器（如文件或数据库）中，并通过标准接口供客户端访问。在这里，您可以配置 TwinCAT OPC UA Server 如何读取和保存变量值。

要求

可针对任意运行时系统 [▶ 49]（如 TwinCAT PLC 或 TwinCAT 3 C++ 等）的变量进行该配置。前提条件是，必须首先为 OPC UA 启用相应的变量。您可以在我们的快速入门 [▶ 23]教程中了解如何操作。

存储器类型

在 TwinCAT OPC UA Server 中，我们将各个存储器类型配置为所谓的“HistoryAdapters”。可以通过 TwinCAT OPC UA Server 配置器执行此操作。例如，如果要在不同的数据存储器中存储各个变量的历史值，每个 HistoryAdapter（“主存”（易失）类型的适配器除外）都可以多次使用。

变量值采样

TwinCAT OPC UA Server 可以通过 2 种不同的方式接收变量值：

1. 使用自己的采样引擎接收变量值，以及
2. 从 OPC UA 客户端（通过所谓的“HistoryUpdate”功能）接收变量值

在单独的采样引擎中，我们为每个变量配置了从下级实时系统采样并存储在存储器中的采样率。我们将服务器从实时系统读取变量值的时间作为时间戳。

而使用 HistoryUpdate 功能时，服务器会从所连接的 OPC UA 客户端接收变量值和时间戳，因此不会自行采样。

4.12.2 支持的功能

使用历史访问功能需要满足以下几个前提条件：

- 必须为拟存储符号进行历史访问的运行时系统配置数据访问功能（且必须启用相应的变量）。
- 若要使用 SQL Compact 数据库作为存储介质，必须在运行 OPC UA 服务器的计算机上安装 SQL Compact Runtime 3.5 SP2。
- Windows CE 也支持 SQL Compact 数据库。
- Windows CE 不支持 SQL Server 数据库。
- 支持以下 MS SQL Server 版本：2017、2019
- 在使用相应的存储器类型时，应遵循以下建议：

  主存：条目数 < 5,000

  文件系统：条目数 < 10,000

  数据库：条目数 >= 10,000

存储器类型

支持以下存储器类型：

存储器类型	TF6100 Server 安装版本	操作系统	描述
主内存	4.x 和 5.x	Windows、Windows CE、TwinCAT/BSD	将值保存在 OPC UA 服务器运行设备的 RAM 中。无需提供其他参数。设备重启后，存储的值将不再可用。因此，存储介质不具备持久性。需遵循上述要求和建议。
文件系统	4.x	Windows、Windows CE	将值保存在多个文件中，文件的存储位置可由用户指定。针对该存储介质配置的每个符号，系统都会在该目录下为其创建自己的文件。此外，每个符号都有一个文件备份副本，当缓冲区达到一定大小时便会创建该文件备份副本。然后将数据文件的内容另存为备份副本，并创建一个新文件。需遵循上述要求和建议。
SQL Compact 数据库	4.x	Windows、Windows CE	将值保存到 SQL Compact 数据库，用户可以指定该数据库的位置。需遵循上述要求和建议。
MS SQL Server 数据库	4.x	Windows	将值保存在通过各种参数引用的 SQL Server 数据库中。需遵循上述要求和建议。
TwinCAT Analytics [▶ 93]	5.x	Windows、TwinCAT/BSD	将值保存在与 TwinCAT Analytics 存储格式相对应的文件格式中。因此，可以通过 TwinCAT Analytics Toolchain 进一步使用这些数据。自第 5.x

存储器类型	TF6100 Server 安装版本	操作系统	描述
			版 TF6100 起，该格式取代了旧版基于文件的存储格式（参见上文）。

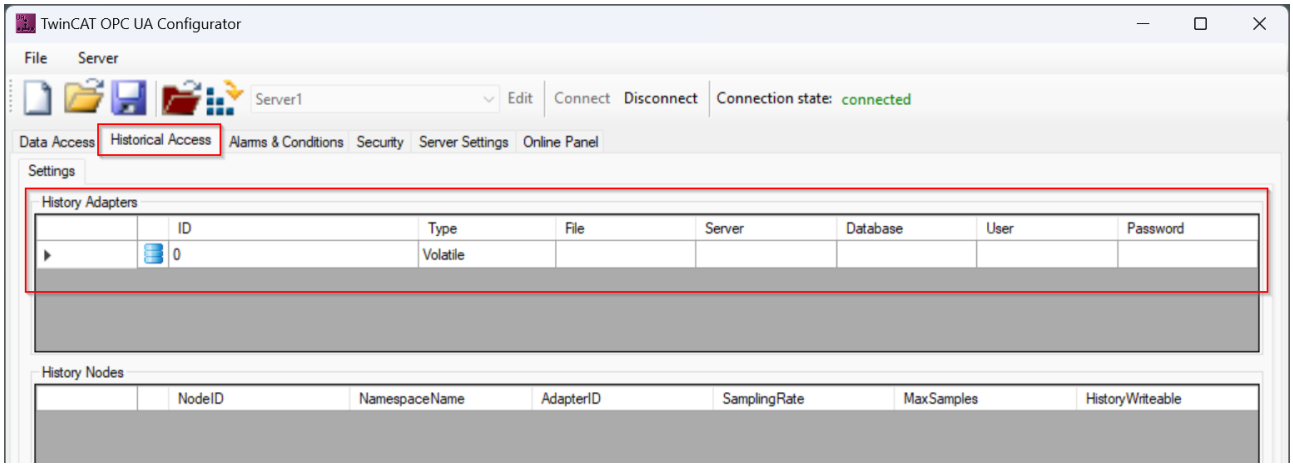
4.12.3 配置

您可以使用 TwinCAT OPC UA Configurator 来配置历史访问功能。
若要为历史访问功能配置符号，必须执行以下 3 个步骤：

1. 必须已启用该符号的数据访问功能（参见“启用符号 [▶ 52]”一章）。
2. 必须创建一个用于保存符号值的存储器类型。
3. 必须为该符号定义配置参数。

创建存储器类型

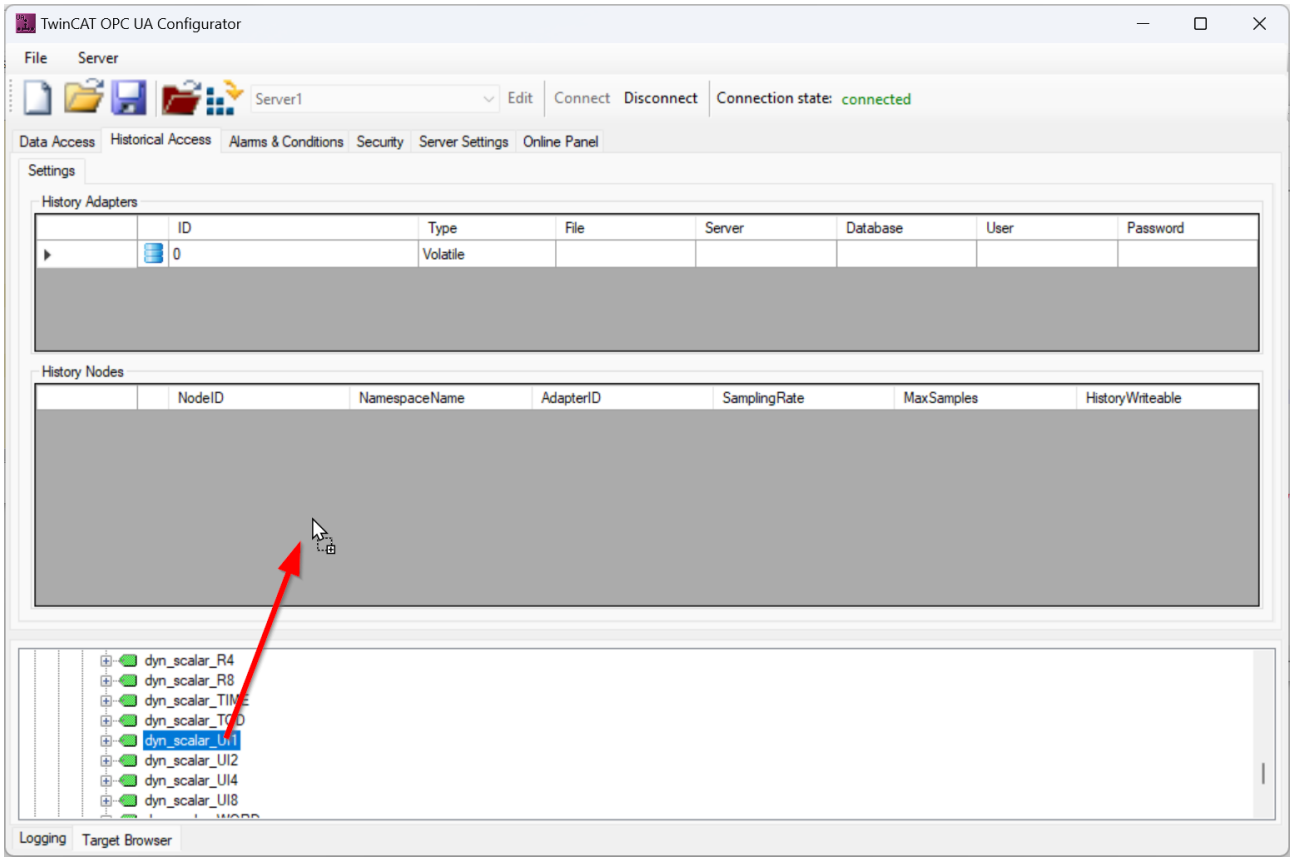
在独立版 TwinCAT OPC UA Configurator 中，您可以通过“**Historical Access（历史访问）**”选项卡来配置存储器类型（所谓的“历史适配器”）。例如，在下方的屏幕截图中，配置的存储器类型为“Main memory（主存）”（“易失”）。



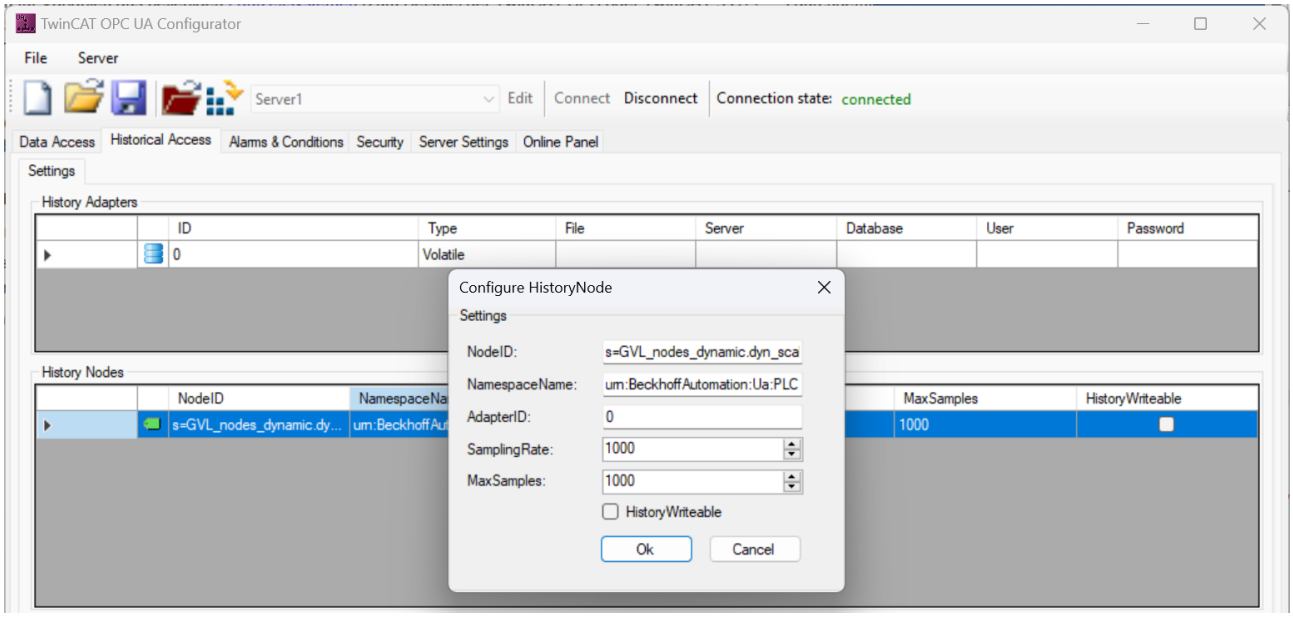
可通过上下文菜单创建其他存储器类型，或者编辑现有存储器类型或将其从配置中删除。

配置符号

在同一对话框的“History Nodes"（历史节点）”部分，您可以配置符号并将其与某一存储器类型相关联。您可以手动创建符号（假设您知道其“地址”或 NodeID），或者使用集成的 Target Browser 将现有符号拖放至配置区域。



然后为符号添加默认配置值。双击符号条目可修改这些配置值。在这里，您还可以创建存储器类型的链接，用户可通过该存储器类型的 ID 对其进行引用。



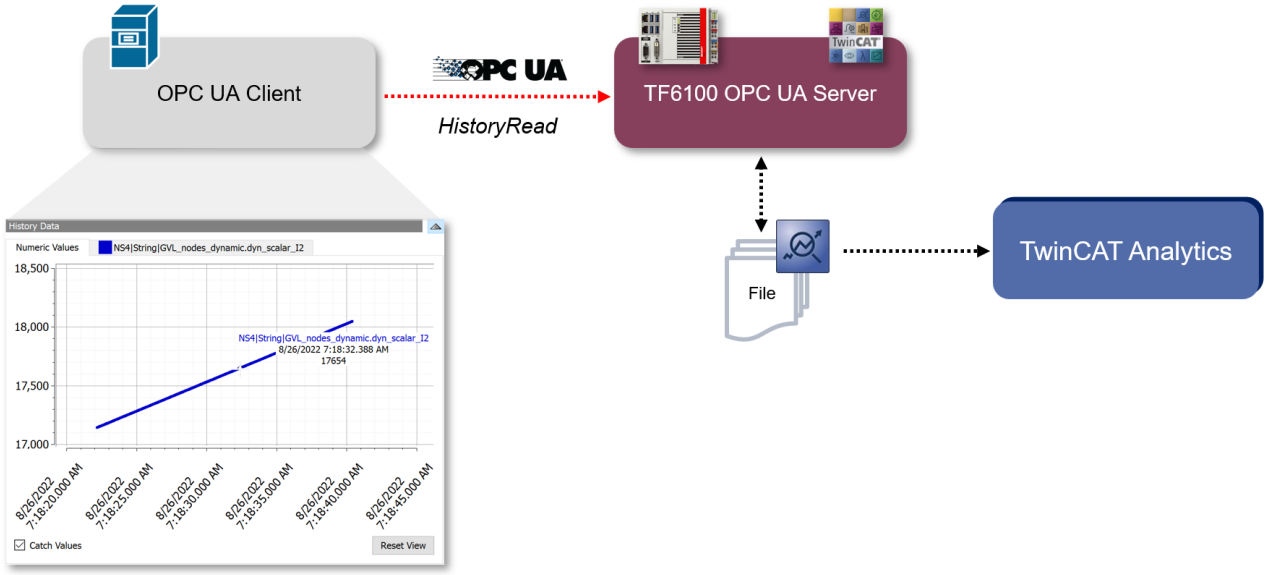
4.12.4 HistoryUpdate

使用 HistoryUpdate 功能时，服务器不会出于历史访问的目的独立采样并保存变量值。相反，服务器会从 OPC UA 客户端接收变量值和时间戳。

可以通过在 TwinCAT OPC UA Configurator 中设置属性 `HistoryWriteable="true"`，为该功能启用一个变量。在这种情况下，服务器不再对该变量自行进行采样，而是“等待”OPC UA 客户端提供数值和时间戳。例如，TwinCAT OPC UA Client 可以通过功能块 UA_HistoryUpdate 进行此类调用。

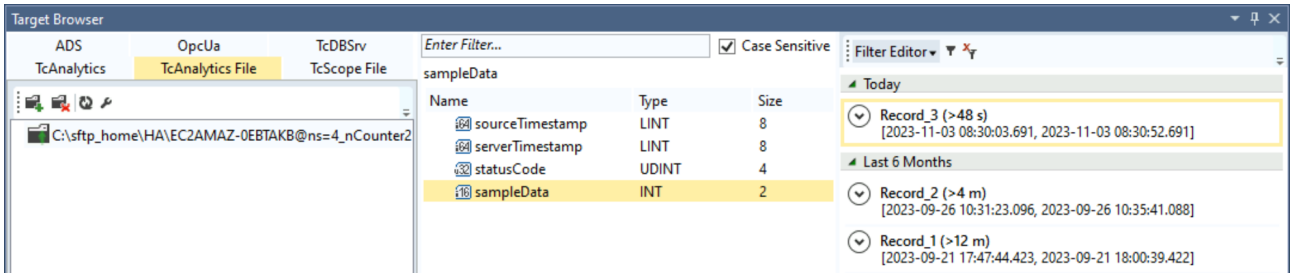
4.12.5 TwinCAT Analytics

使用“TwinCAT Analytics”存储器类型时，系统会写入一个二进制文件作为存储器，其格式与 TwinCAT Analytics 数据格式一致。虽然这与 OPC UA 客户端无任何关系（因为它通过 OPC UA 访问历史数据），但它确实为进一步使用历史数据提供了有趣的可能性，因为您可以将二进制文件读入 TwinCAT Analytics Toolchain 并在其中进行处理。
下图对这种关系进行了阐释。



TwinCAT Target Browser

如果您在历史访问配置中使用了 TwinCAT Analytics 存储格式，还可以通过 TwinCAT Analytics Toolchain（例如 TwinCAT Target Browser）获取存储值。为此，请在 TwinCAT Target Browser的“**TcAnalytics File（TcAnalytics 文件）**”选项卡中添加您在配置中定义的用于保存历史值的目录。保存的值随后将以 TwinCAT Analytics 形式显示为记录，并可进行进一步处理。



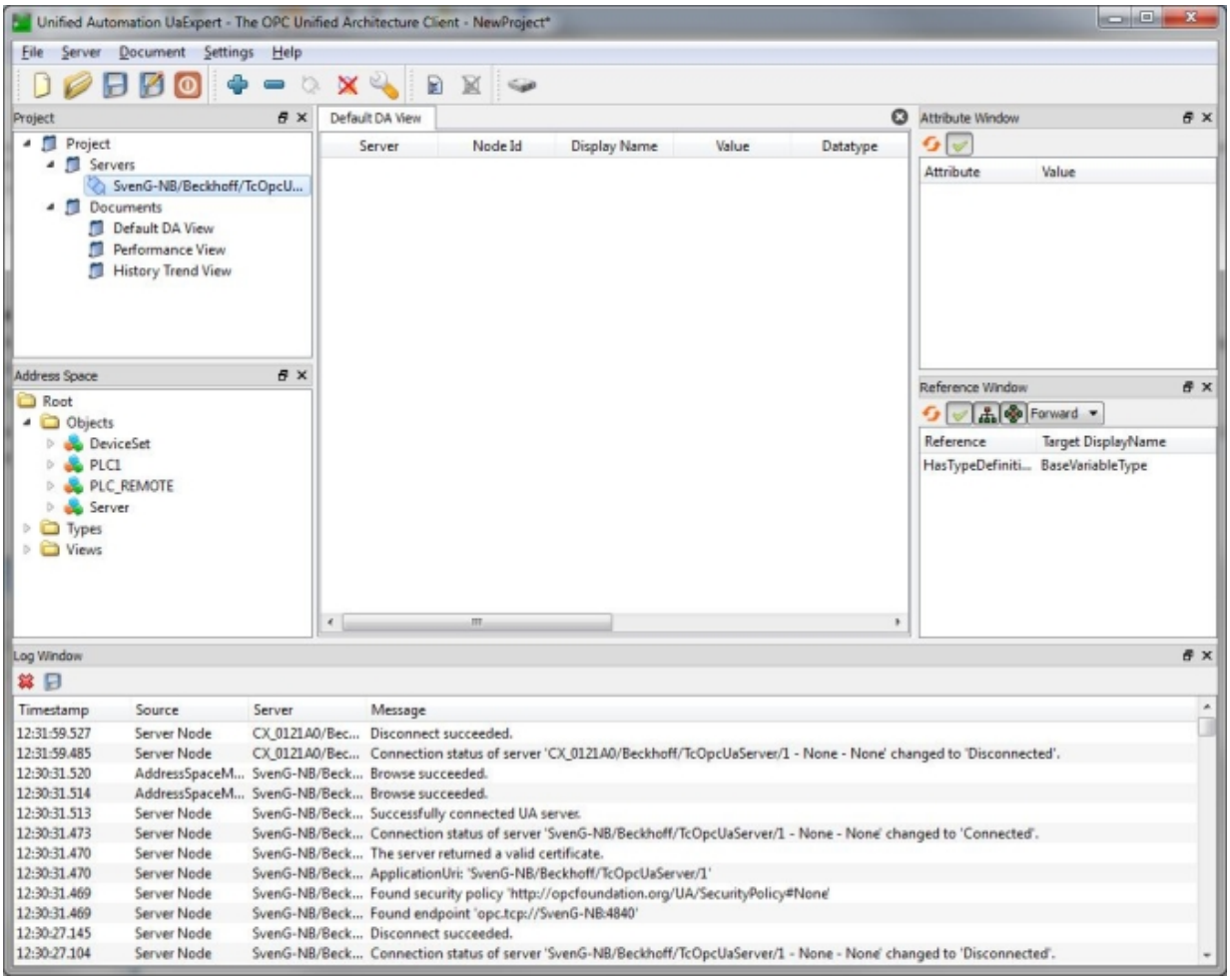
4.12.6 访问历史数据

您可以使用任何支持 OPC UA 历史访问功能的 OPC UA 客户端来获取历史数据。在下方的示例中，Unified Automation 公司的“UA Expert”软件用于从 TwinCAT OPC UA Server 读取并直观显示历史数据。

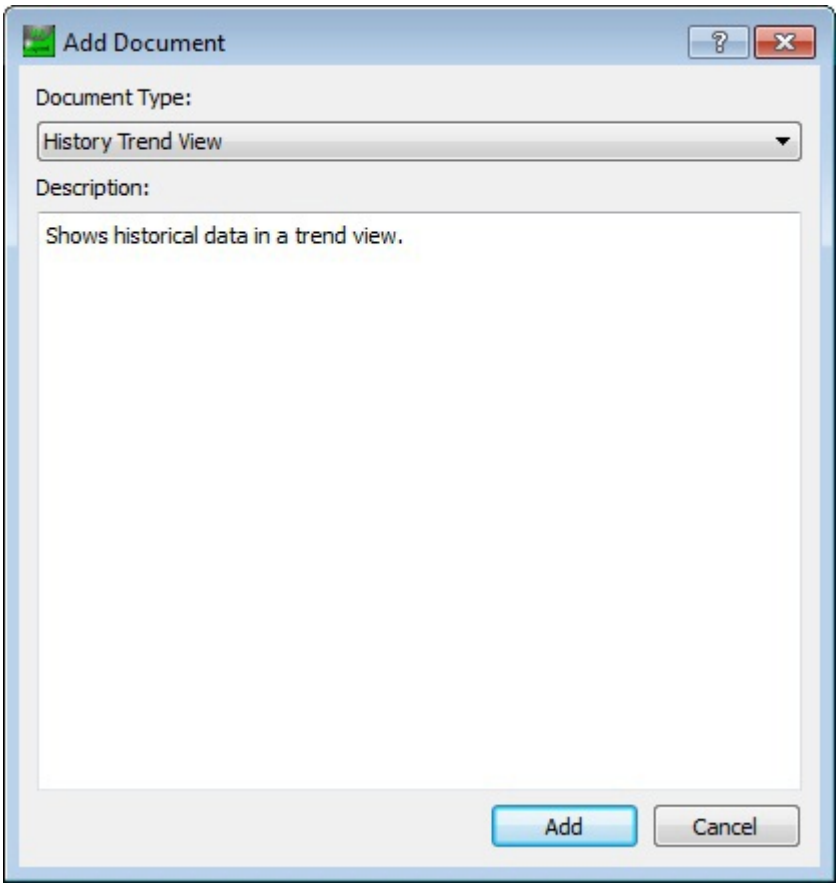
在 OPC UA 客户端中显示历史访问值

以下分步说明介绍了如何配置 UA Expert 软件以访问历史数据。

1. 启动 UA Expert 软件并连接 OPC UA 服务器。

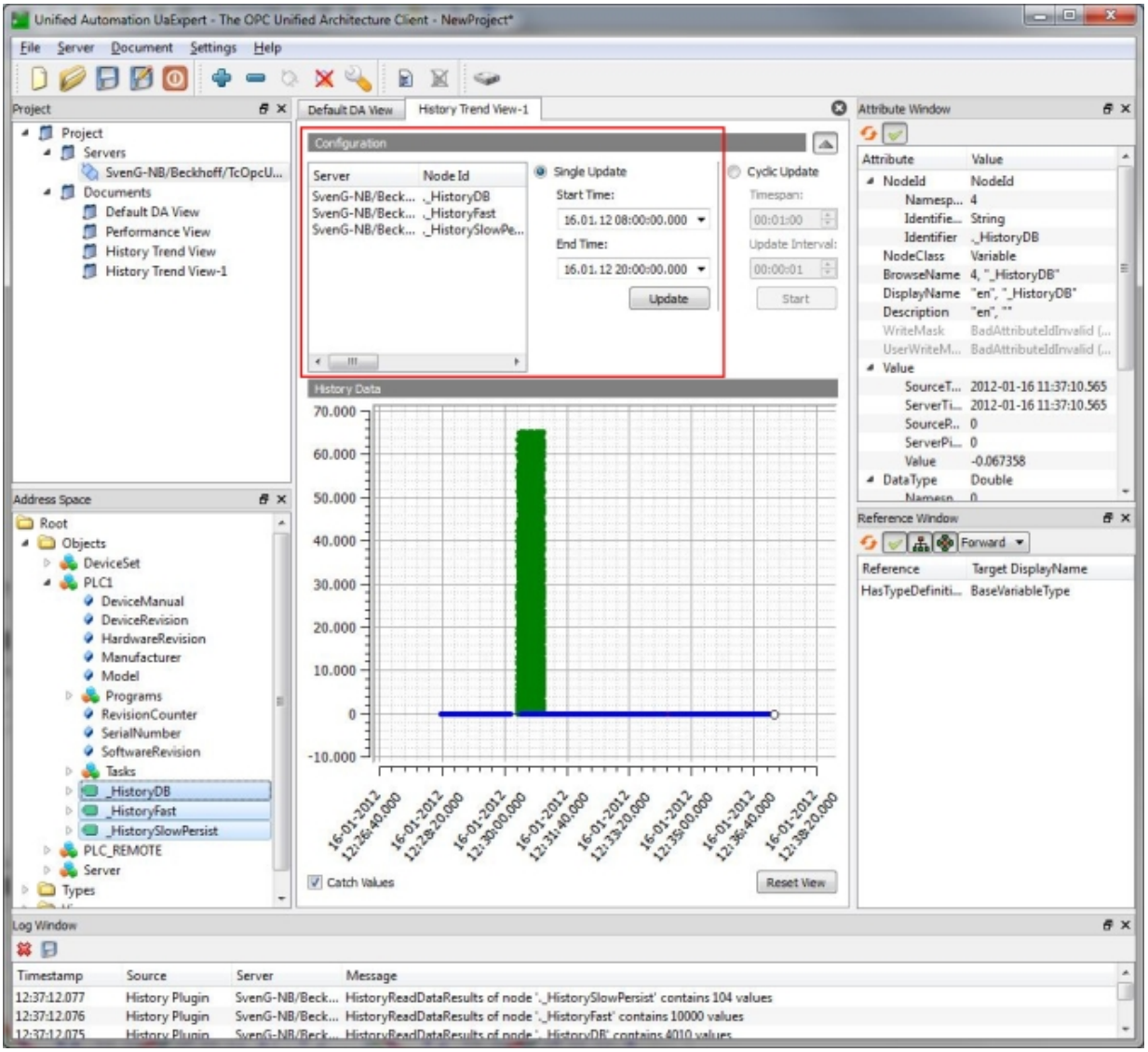


2. 添加新的**历史趋势视图**。



3. 浏览 PLC1 命名空间，使用拖放功能添加 PLC 变量 _HistoryDB、_HistoryDBcompact、_HistoryFast 和 _HistorySlowPersist。

⇨ 现在，您可以使用**开始时间**和**结束时间**控制键来指定显示符号值所需的时间段，也可以根据需要为这些变量启动**循环更新**。



4.13 报警与条件

4.13.1 概述

本章将介绍在 TwinCAT OPC UA Server 上使用 OPC UA **报警与条件**（A&C）的配置步骤。OPC UA A&C 描述了一个用于监控符号值并在符号值发生变化或超过阈值时发出报警和事件的模型。

要求

可针对任意运行时系统 [▶ 49]（如 TwinCAT PLC 或 TwinCAT 3 C++ 等）的变量进行该配置。前提条件是，必须首先为 OPC UA 启用相应的变量。您可以在我们的快速入门 [▶ 23]教程中了解如何操作。

4.13.2 支持的功能

下表列出了 TwinCAT OPC UA Server 的报警与条件功能所支持的标准 AlarmType。

AlarmType	描述
限制警报类型	让您可以为符号定义不同的阈值。如果超过阈值，将会触发可配置报警文本和严重程度的报警。
OffNormalAlarmType	可定义符号的“正常”和“非正常”阈值。如果符号值偏离定义值，将会触发报警。

BOOL 在 OffNormalAlarmType 中的使用

若要对 OffNormalAlarm 类型使用 BOOL，则必须使用 true 和 false 表示法。若为大写，则表示 TwinCAT OPC UA Server 假设为 STRING，而不再将该值作为 BOOL 进行解读。

除了报警与条件功能外，TwinCAT OPC UA Server 还提供与 TwinCAT EventLogger [▶ 108] 连接的选项，并将在其中接收到的报警或事件作为 OPC UA 报警或事件进行相应的转发。

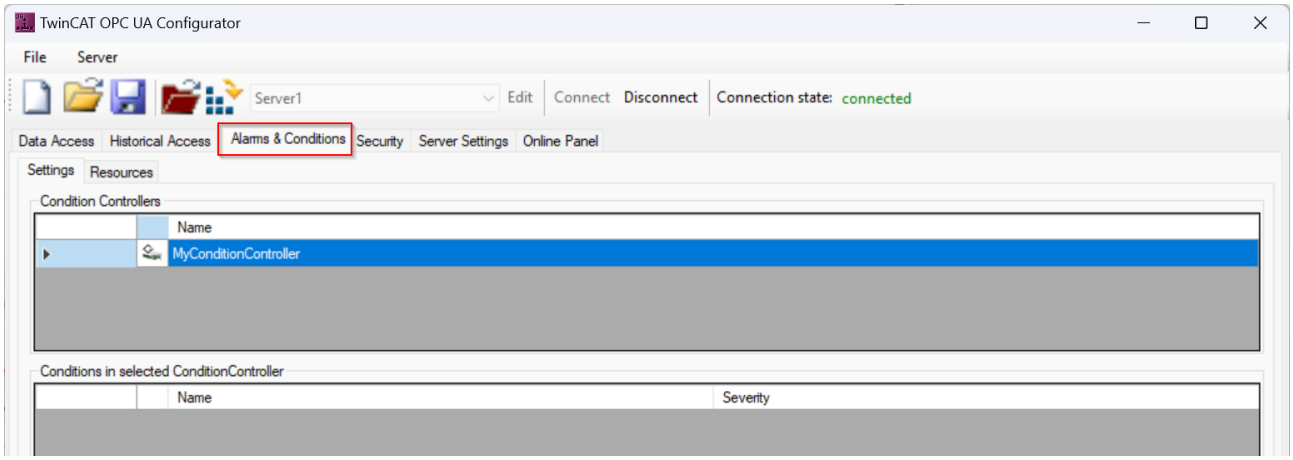
4.13.3 配置

您可以使用 TwinCAT OPC UA Configurator 来配置报警与条件功能。若要为报警与条件功能配置符号，必须执行以下 2 个步骤：

1. 必须已启用该符号的数据访问功能（参见“启用符号 [▶ 52]”一章）。
2. 必须创建一个 ConditionController，以便对报警和事件进行分组。
3. 必须创建报警和事件文本。
4. 必须为该符号定义配置参数。

创建 ConditionController

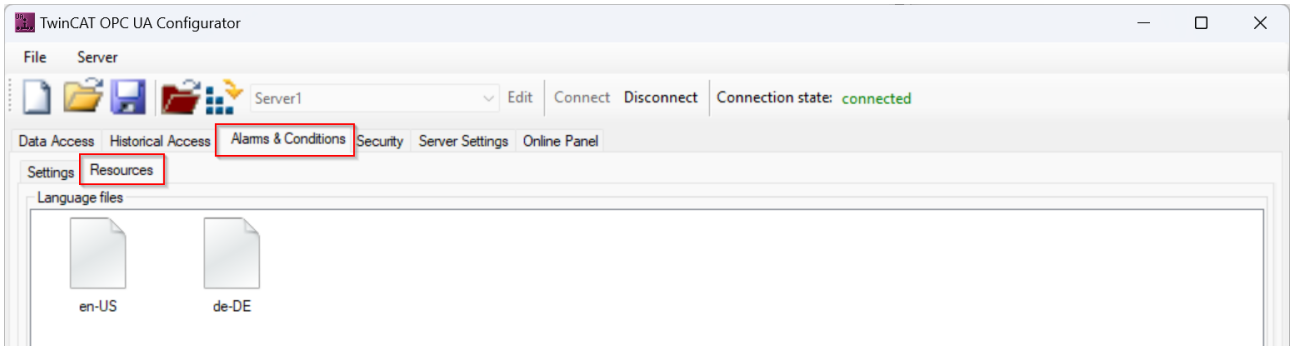
在独立版 TwinCAT OPC UA Configurator 中，您可以通过“**Alarms & Conditions**（报警与条件）”选项卡来配置 ConditionController。例如，在下方的屏幕截图中，我们配置了一个名为“MyConditionController”的 ConditionController。



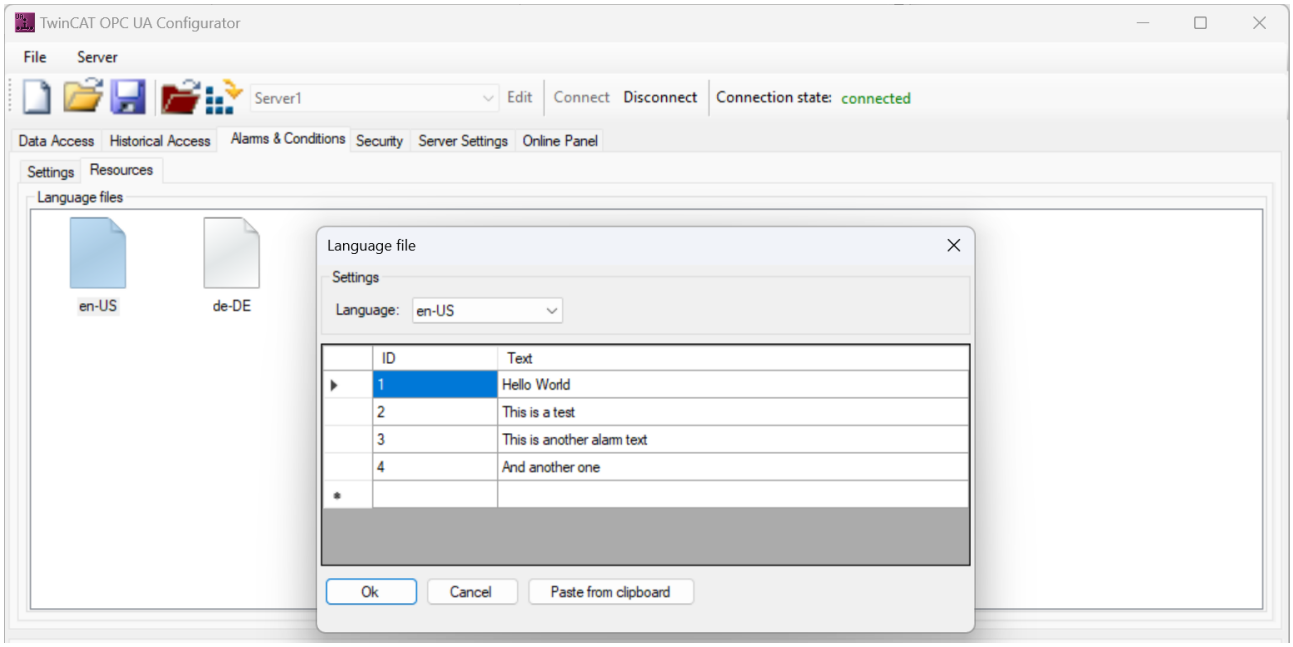
可通过上下文菜单创建其他 ConditionController，或者编辑现有 ConditionController 或将其从配置中删除。

创建报警和事件文本

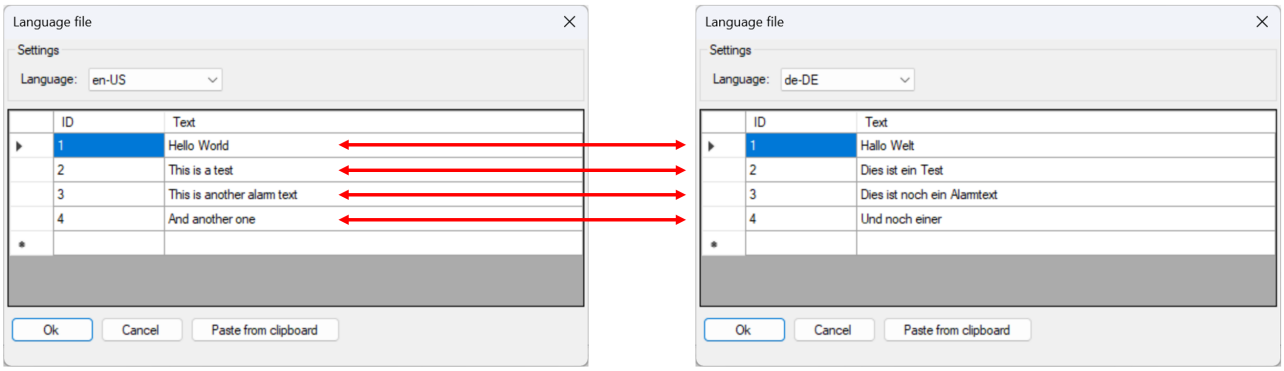
可将读取报警或事件时需使用的文本作为所谓的“resources（资源）”存储在 TwinCAT OPC UA Server 中。可以用多种语言定义这些文本，并通过相应的国家/地区代码来标识这些语言。在 TwinCAT OPC UA Configurator 中，我们通过“Alarms & Conditions（报警与条件）”配置中的“**Resources**（资源）”选项卡来添加这些资源。



您可以通过上下文菜单添加、删除或编辑作为资源的其他语言。在上方的屏幕截图中，我们配置了 2 种语言资源：de-DE（德语/德国）和 en-US（英语/美国）。文本以表格形式保存在语言中，也可从 Microsoft Excel 中复制和粘贴文本。



配置好的文本会通过其 ID 进行跨语言分配，例如：

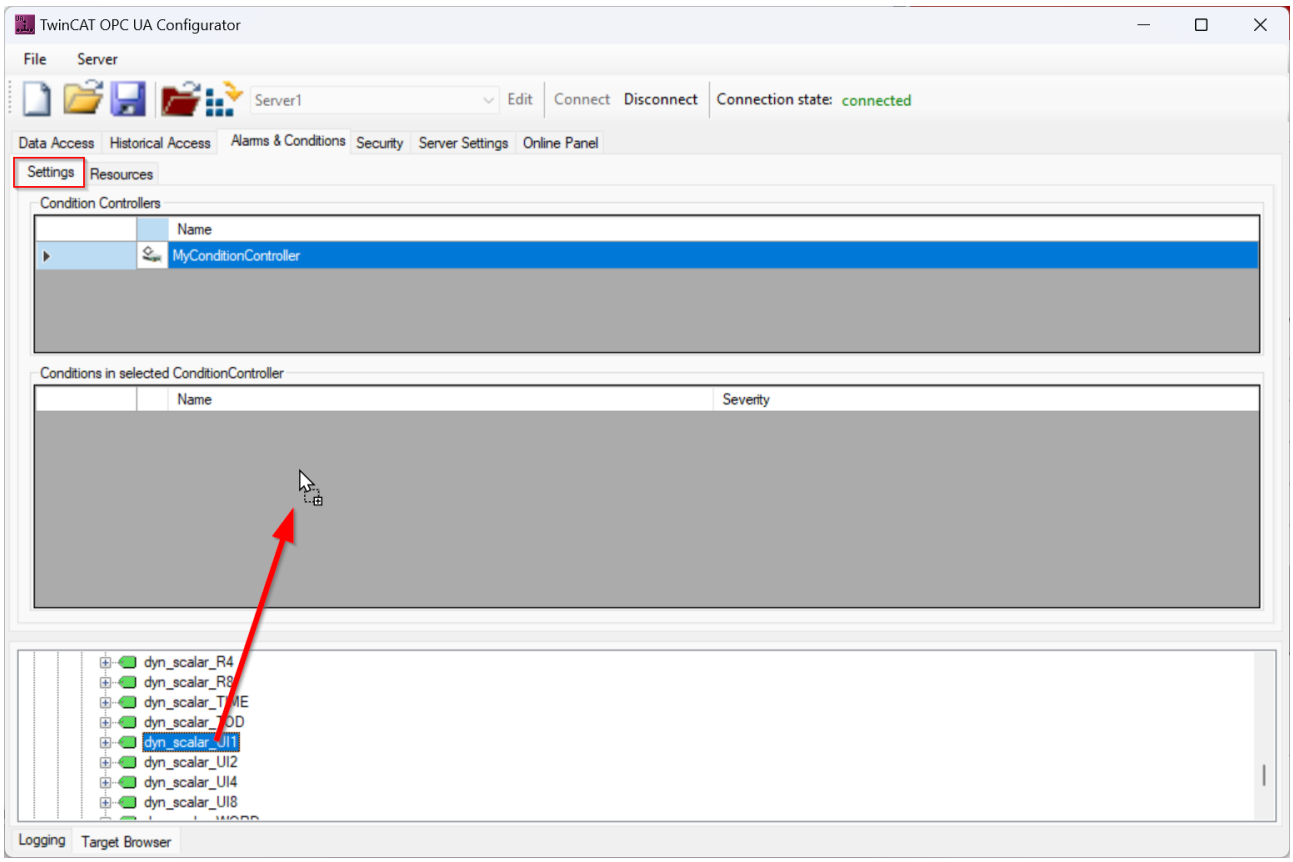


报警/事件所使用的国家语言

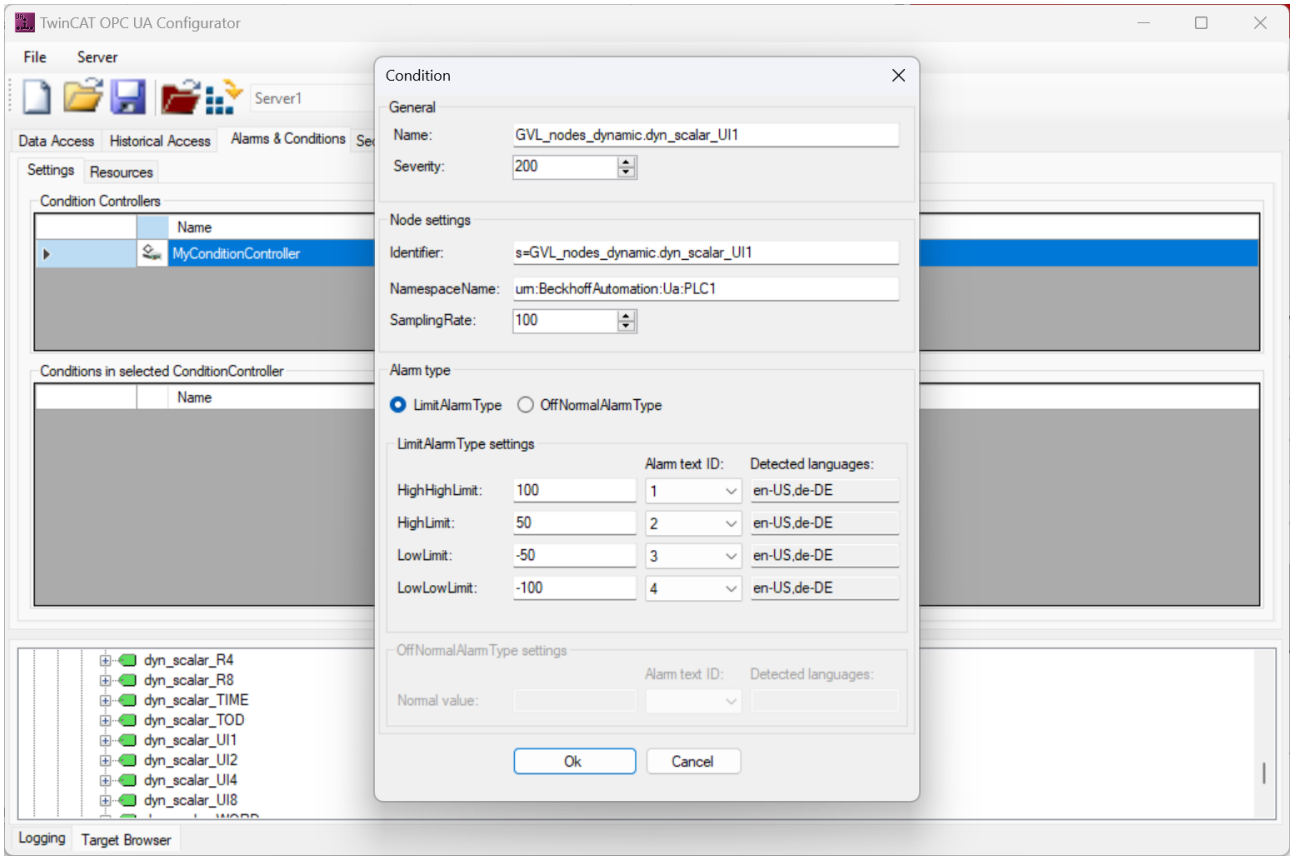
触发报警或事件时所使用的语言文本取决于 OPC UA 客户端与服务器连接时所使用的国家/地区代码。

配置符号

配置好报警和事件文本后，您还可以使用这些文本来配置符号。在“**Settings**（设置）”选项卡中进行符号配置。您可以手动配置图标，或者使用集成的 Target Browser 将其拖放至配置区域。



然后将符号添加至选定的 ConditionController 中。在下方的配置对话框中，您可以定义要配置的报警类型和阈值，还可以选择语言文本。

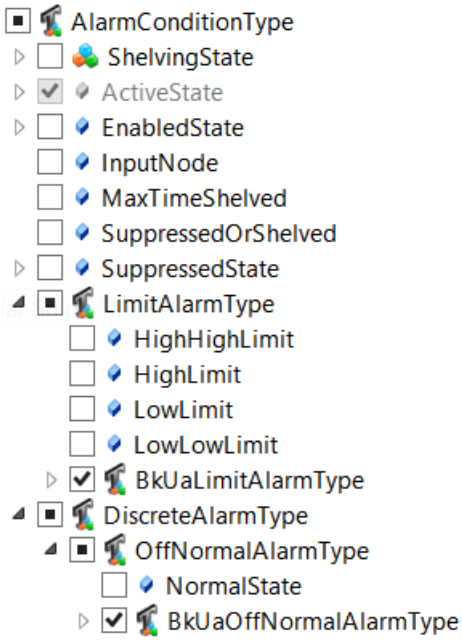


4.13.4 其他应用数据

如果您想将应用程序中的其他数据附加到报警中，可以使用以下几个特殊的 AlarmType 进行配置：

AlarmType	源自
BkUaLimitAlarmType	限制警报类型
BkUaOffNormalAlarmType	OffNormalAlarmType

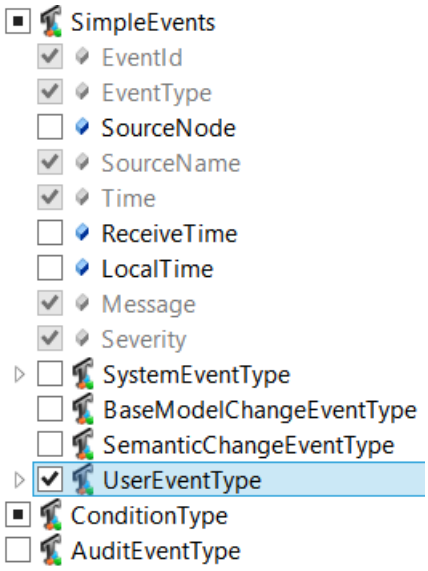
这些 AlarmType 中还定义了其他字段，您可以使用应用程序中的值来填充这些字段。若要使 OPC UA 客户端能够使用这些附加值，该客户端必须订阅相应的 AlarmType 并对其进行解读。



OPC UA 客户端随后会在传入报警的字段 BkUaEventData 和 BkUaEventValue 中接收附加应用数据，例如：

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

用户定义的 EventField 将作为“UserEventData”附加在上面。登录到 SimpleEventType“UserEventType”的 OPC UA 客户端可以接收到这些数据。



若要使用此功能，必须在 PLC 中定义一个结构，其中包含要监控的符号值和触发报警时需发送的附加值。必须对该结构进行如下定义：

```
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
```

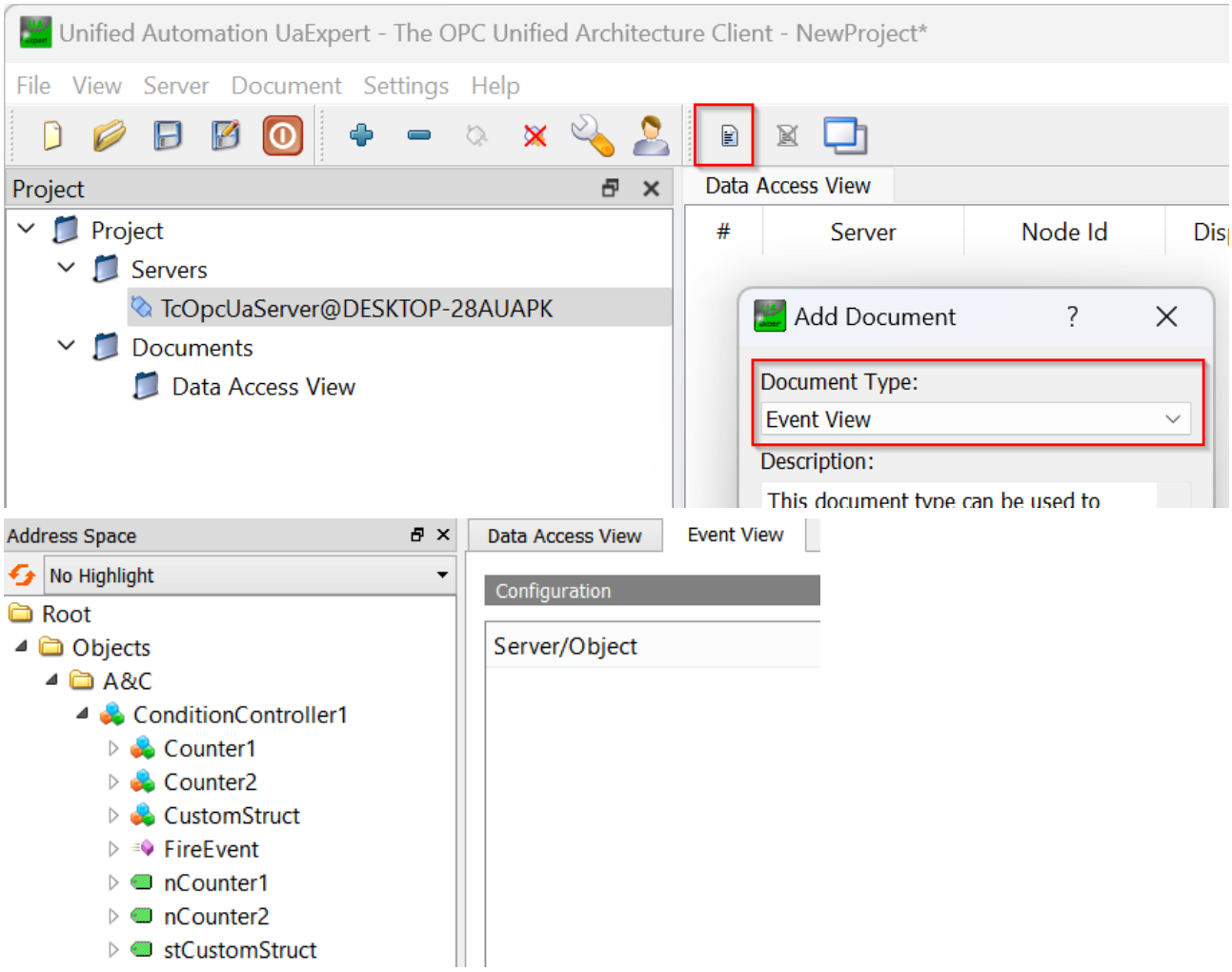
然后，结构 ST_CustomStruct 的实例将作为数据访问符号 [▶ 52]得以启用。还必须将该结构作为 StructuredType [▶ 74] 进行启用。

4.13.5 访问报警和事件

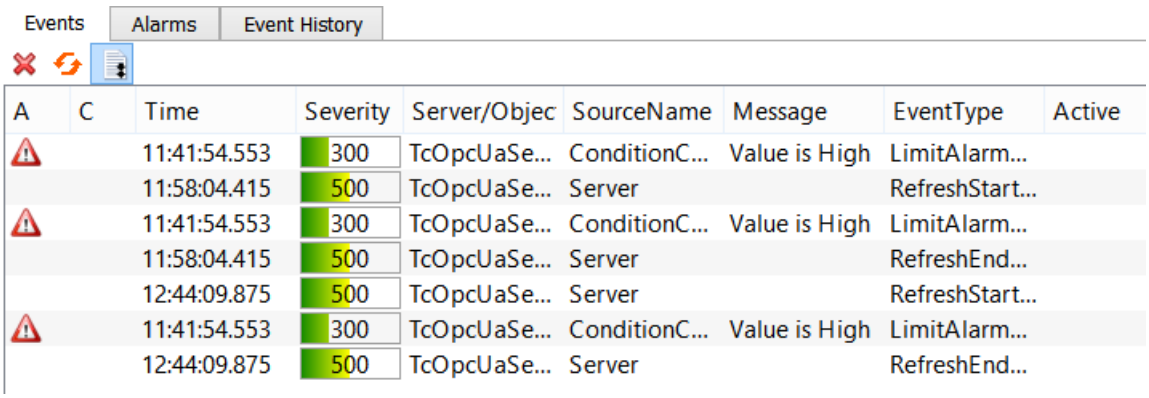
若要接收报警和事件，TwinCAT OPC UA Server 上的 OPC UA 客户端必须登录其中一个已配置完成的 ConditionController。然后，客户端便会收到在该 ConditionController 中配置的符号的报警和事件。

下方的示例显示了如何使用 Unified Automation 公司的 UA Expert 软件订阅 ConditionController，以便接收其发出的报警和事件。

在启动 UA Expert 并与 TwinCAT OPC UA Server 建立连接后，请在您的工作区中添加一个“Event View（事件视图）”类型的新文档视图。



在服务器的地址空间中，您可以在“A&C”节点下方找到所有已配置完成的 ConditionController。现在，您可以通过从“Event View（事件视图）”工作区拖放一个 ConditionController 至此处来进行订阅。然后，该 ConditionController 的报警和事件将显示在“Event（事件）”窗口中。



4.14 方法调用

4.14.1 概述

PLC 中有 2 种不同的概念可用于调用 OPC UA 方法：

1. RPC 方法 [▶ 106]和
2. Job 方法 [▶ 103]

这两种概念的根本区别在于 PLC 的处理环境和 PLC 代码的实现方式。
下表对这种关系进行了阐释。

类型	处理	PLC 代码
RPC 方法	在一个 PLC 周期内	OPC UA 方法由 PLC 方法表示。
Job 方法	在多个 PLC 周期内	OPC UA 方法由 PLC 功能块表示。

一般来说，可以这样进行定义：如果方法调用需要“较长”时间，即可能需要若干次 PLC 循环，则 Job 方法是首选方法。相比之下，RPC 方法的处理速度必须要“快”，否则存在循环超限的风险。**因此，我们通常会选择 Job 方法。**虽然在 PLC 中进行编程的工作要复杂一些，但与 PLC 周期脱钩可带来巨大的优势，因此有必要为此付出努力。

4.14.2 Job 方法

Job 方法的概念与普通方法调用之间有一个根本区别：OPC UA 方法不再 1:1 映射到 PLC 方法，而是映射到具有特定签名的功能块。从 PLC 应用程序的角度来看，这也让耗时超过 1 次循环的方法调用成为可能。

要求

该功能仅适用于基于 TwinCAT 3 的数据访问 [▶ 48]设备和 TMC [▶ 52] 符号文件的导入，以及在线符号。

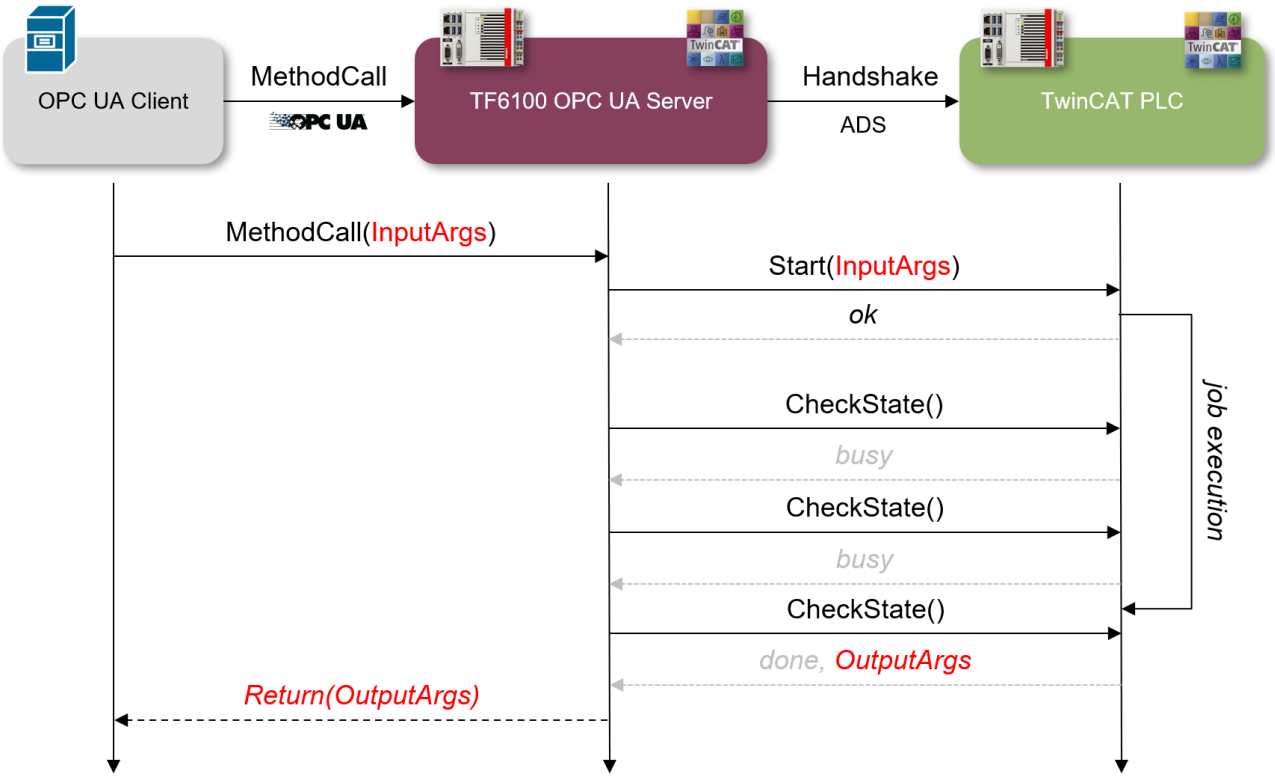
此类 Job 方法在 PLC 端的结构定义如下：有一个通过 PLC 属性被定义为 Job 方法的功能块。然后，该功能块包含各种 PLC 方法，TwinCAT OPC UA Server 采用握手机制访问这些方法，以便能够将其作为 OPC UA 方法供用户使用。

方法	描述
Start	当 OPC UA 客户端调用 OPC UA 方法时，服务器会立即调用该方法。包含作为 VAR_INPUT 的 OPC UA 方法调用输入参数。 该方法的 HRESULT 返回值可用于直接返回十进制的 OPC UA 状态代码，例如“0”表示状态代码“Good”。其值即为 OPC UA 规范中定义的状态代码数值。可在此处查看所有可用状态代码的定义： http://www.opcfoundation.org/UA/schemas/StatusCode.csv

方法	描述
	该方法通常返回“0”（Good）值。但是，PLC 开发人员也可以选择对输入参数等进行验证。如果出现错误，OPC UA 方法调用可能也会随之失败，例如，返回值为“2158690304”（BadInvalidArguments）。
CheckState	由服务器进行循环调用，以检查作业是否仍在处理中。只要作业仍在处理中，该方法就会返回值“Busy（忙）”，否则返回“Done（已完成）”。OPC UA 方法的输出参数在此处声明为 VAR_OUTPUT。 该方法的 HRESULT 返回值可用于直接返回十进制的 OPC UA 状态代码，例如“0”表示状态代码“Good”。其值即为 OPC UA 规范中定义的状态代码数值。可在此处查看所有可用状态代码的定义： http://www.opcfoundation.org/UA/schemas/StatusCode.csv 如果作业处理成功，该方法通常会返回值“0”（状态代码为“Good”）。但是，PLC 开发人员也可以决定，若出现错误，OPC UA 方法调用应失败，例如，返回值为“2151415808”（BadOutOfRange）或更通用的“2147483648”（Bad）。
Abort	如果作业必须中止，例如服务器关闭或重启，服务器会调用该方法。这样，PLC 开发人员便可以借此机会对其 PLC 代码进行相应的清理。

工作流程

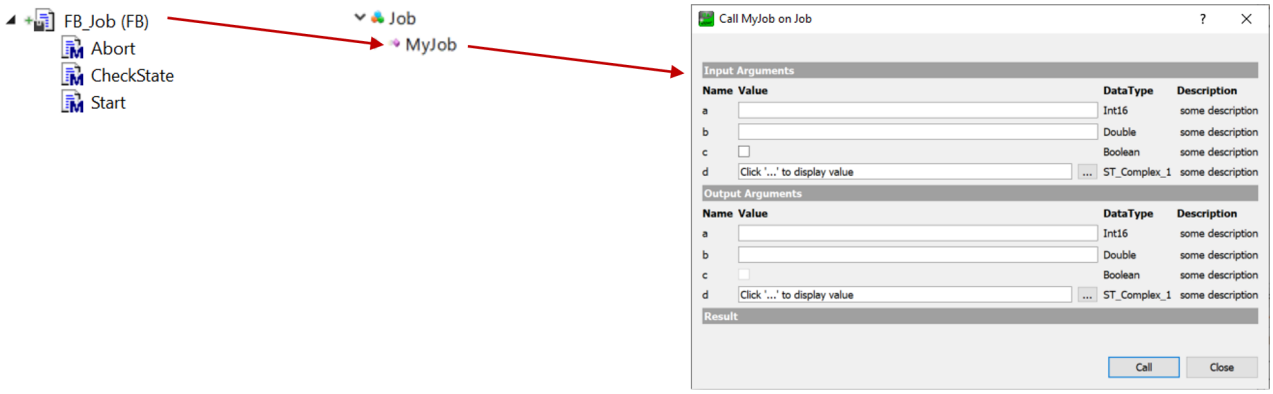
可以简化形式表示服务器与 PLC 之间的握手机制，具体方式如下：



示例

您也可以在“示例”部分以可执行形式获取以下示例。这部分文档旨在阐述系统的基本工作原理。该示例“模拟”了一次 Job 方法调用，该方法在 PLC 中的处理时间约为 4 秒。此次模拟是借助定时器实现的，该作业的功能块部分对该定时器进行了声明和使用。该功能块的状态机会延迟作业的完成（CheckState 握手方法返回“busy（忙）”），直至定时器超时（CheckState 握手方法返回“done（已完成）”），作业才会完成。

在本例中，我们针对名为“MyJob”的 OPC UA 方法创建了一个名为 FB_Job 的功能块，该功能块具备上文规定的签名。



功能块声明中包含属性 OPC.UA.DA.JobMethod，以及作为其值使用的 OPC UA 方法名称。

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

Abort、CheckState 和 Start 这三个方法在其声明中包含了属性 TcRpcEnable（这样，这些方法也能通过 ADS 进行调用）。示例：

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

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

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

在 PLC 方法 Start() 中，我们将 OPC UA 方法的输入参数声明为 VAR_INPUT。变量后的注释用于对相应的 OPC UA 输入参数进行描述。示例：

```
{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
```

在 PLC 方法 CheckState() 中，我们将 OPC UA 方法的输出参数声明为 VAR_OUTPUT。变量后的注释用于对相应的 OPC UA 输出参数进行描述。示例：

```
{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
```

(在本例中，为了便于说明，输入参数的值与输出参数的值保持 1:1 对应关系。因此，输入参数和输出参数完全相同）。

并发方法调用

可通过 PLC 应用程序来管理并发方法调用。为此，可以通过 Start() 方法的返回值向服务器返回一个句柄（输出变量 hdl），然后将其用作 CheckState() 方法所有后续调用的输入变量。这样，PLC 应用程序即可确定相应方法调用的执行上下文。总而言之，具体工作流程如下：

- Start() 方法确定执行上下文，并通过输出变量“hdl”向服务器返回一个句柄。
- 现在，服务器将使用此句柄循环调用 CheckState()，并告知 PLC 应用程序涉及的是哪个执行上下文。

4.14.3 RPC 方法

方法调用是 OPC UA 规范的基本组成部分。随着这些功能被引入 PLC 环境，TwinCAT 3 实现了在 IEC61131 环境下高效执行 RPC 调用，从而减少了设备间通信中经典握手模式的使用。

借助 RPC 方法的概念，TwinCAT OPC UA Server 可直接从 TwinCAT PLC/C++ 中导入一个方法，作为 OPC UA 方法。

要求

该功能仅适用于基于 TwinCAT 3 的数据访问 [▶ 48]设备和 TMC 与 TMI [▶ 52] 符号文件的导入，以及在线符号。

RPC 方法调用在 OPC UA 层级的处理方式如下：

- 如果 RPC 方法执行成功，服务器会返回状态代码“Good（好）”，作为对 OPC UA 方法调用的反馈。
- 如果无法调用 RPC 方法，服务器会根据发生的错误，以“Bad_***”形式返回错误消息。
- 如果 RPC 方法调用成功，但服务器无法读取响应，那么，服务器将返回状态代码“OpcUa_GoodPostActionFailed”。

处理上下文

TwinCAT PLC/C++ 方法在实时环境中执行，因此属于实时任务的处理环境。因此，TwinCAT 开发人员必须采取预防措施，使方法的执行时间与任务循环时间相匹配。

TwinCAT 3 PLC 中的方法

在 IEC61131 环境中，方法始终配置在功能块之下。从高级语言的角度来看，我们可以将功能块视为方法的所属类别。必须使用一个特殊的 PLC 属性来声明方法本身，以便 TwinCAT 系统知晓该方法将会激活，用于进行远程方法调用。

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

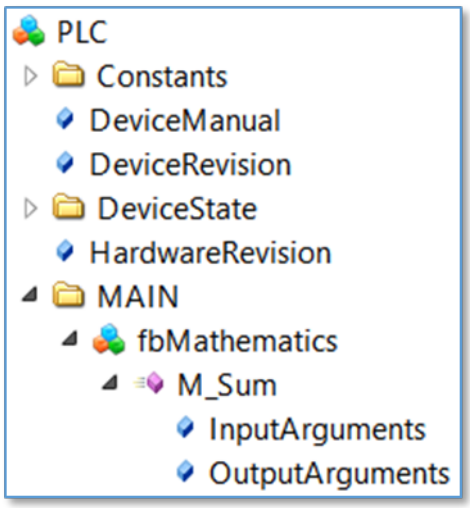
根据所用导入模式的不同，还必须启用周围的功能块，以便访问 OPC UA。通过使用常规 PLC 属性访问 OPC UA 即可完成此操作。

请注意，只有在使用过滤模式允许通过 OPC UA 访问 PLC 中的符号的情况下，才需要使用 PLC 属性。这是 OPC UA 服务器的默认设置。

示例：

方法 M_Sum 位于功能块 FB_Mathematics 中。功能块实例的声明使用了已启用该功能块以及 OPC UA 访问方法的 PLC 属性。

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



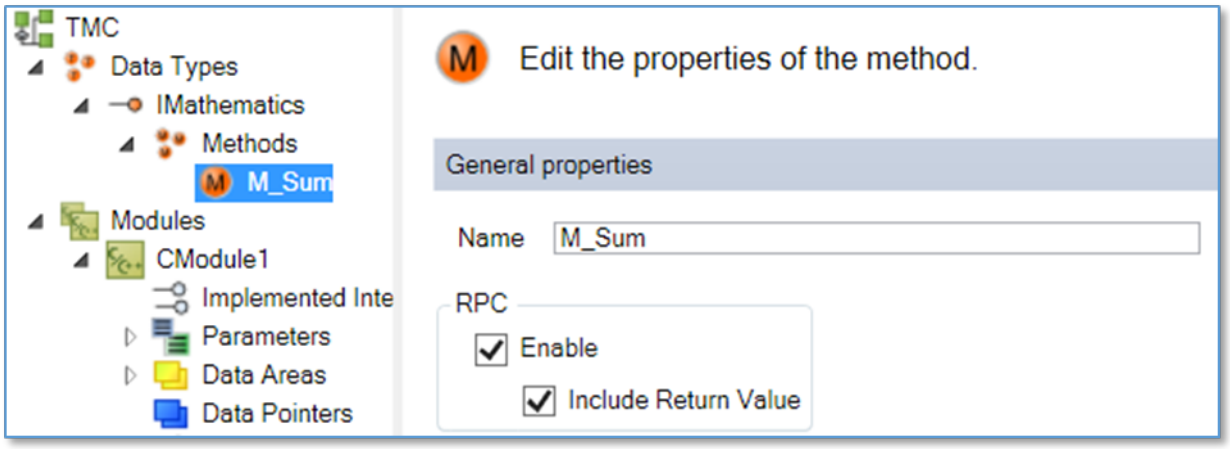
作为 VAR_IN_OUT 的指针变量

定义为 VAR_IN_OUT 的指针变量不会由 PLC 或 TwinCAT OPC UA Server 进行处理。系统会将相应的跟踪事件写入服务器跟踪日志。

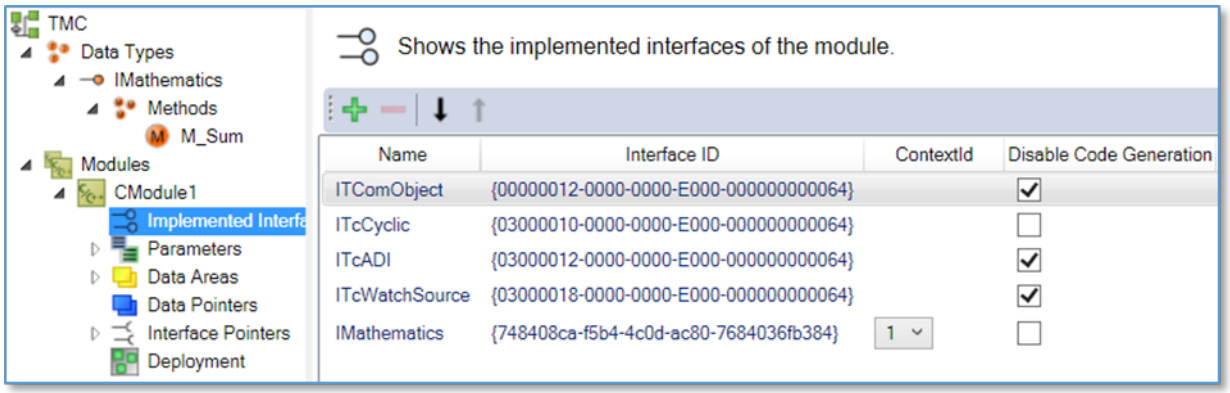
08:47:37.677Z|1|11A0* 在导入方法“METH_PArray”时出错：不允许使用指针变量 VAR_IN_OUT!

TwinCAT 3 C++ 中的方法

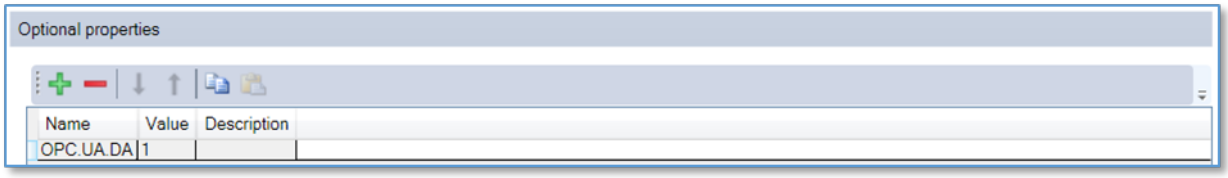
TwinCAT 模块可以实现包含预定义方法的接口（参见“TcCOM 模块”）。必须在定义时启用 RPC 调用方法（参见 TwinCAT 模块类向导文档），以便 OPC UA 服务器知晓该方法已准备就绪，随时可以执行。

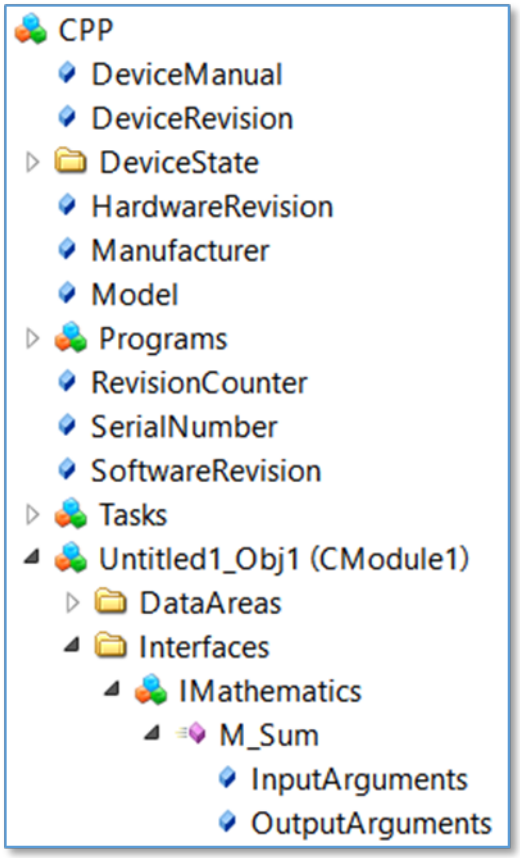


若要获取该方法的返回值，必须激活相应的选项“**Include Return Value**（包含返回值）”。请注意，必须采用创建该方法时所使用的接口。



根据所用导入模式的不同，您必须声明通过 OPC UA 进行访问的方法。通过使用 TMC 代码编辑器和通常作为可选属性的 OPC UA 属性即可完成此操作。





4.15 TwinCAT EventLogger

4.15.1 概述

TwinCAT OPC UA Server 可与 TwinCAT EventLogger 集成，用于发送报警和事件。报警/事件由 TwinCAT EventLogger 转换为 OPC UA 报警或事件。可以在 TwinCAT OPC UA Server 中配置来自多个 TwinCAT 设备的 EventLogger 消息。每个设备均由其 AMS Net ID 进行标识。

在 Windows CE 下运行的 TwinCAT OPC UA Server 无法集成 TwinCAT EventLogger。但是，您可以在非 CE 设备上安装 TwinCAT OPC UA Server，然后从 Windows CE 设备远程接收 TwinCAT EventLogger 消息。

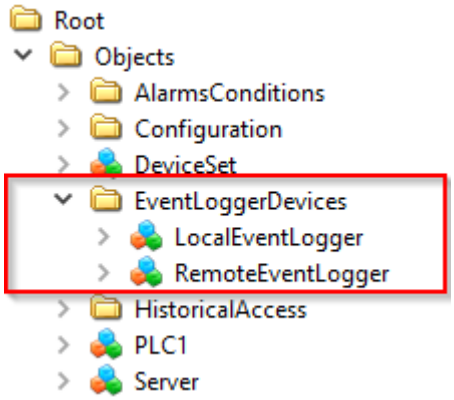
只有在与 TwinCAT 3 EventLogger 连接的情况下，才能集成 TwinCAT EventLogger。不支持使用 TwinCAT 2 EventLogger。

4.15.2 配置

若要配置与 EventLogger 连接的服务器，必须在服务器目录下创建一个名为“TcUaEventLogConfig.xml”的新配置文件。该文件包含通过 AMS NetID 进行标识的 TwinCAT EventLogger 设备的列表。配置结构如下：

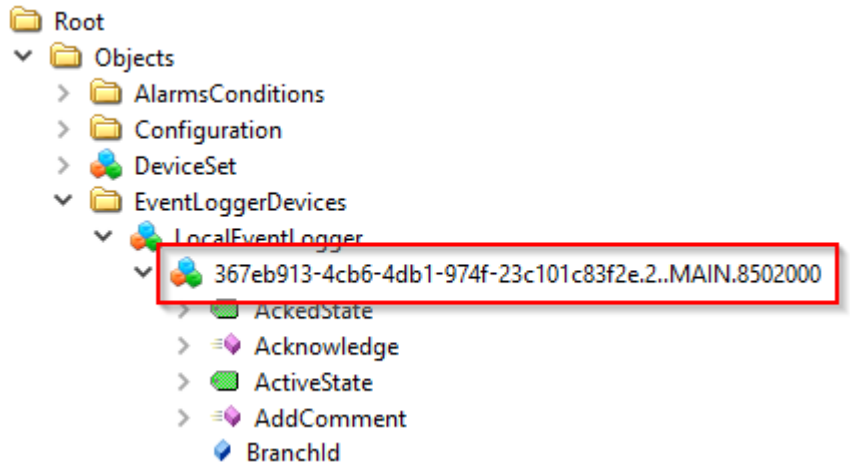
```
<TcUaEventLogConfig>
  <EventLoggerDevice Name="LocalEventLogger" AmsAddr="127.0.0.1.1.1"/>
  <EventLoggerDevice Name="RemoteEventLogger" AmsAddr="192.168.0.56.1.1"/>
</TcUaEventLogConfig>
```

然后，各个设备条目将显示在服务器命名空间的“EventLoggerDevices”文件夹中。



OPC UA 客户端现在可以订阅相应的对象，以便从相应的 TwinCAT EventLogger 设备接收事件和/或报警。

与事件不同的是，相应的 EventLogger 设备还会将报警显示为子元素。

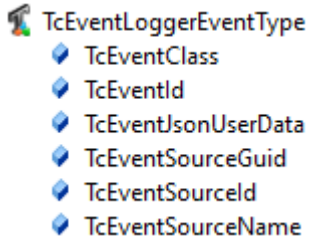


4.15.3 访问报警和事件

在订阅对象及接收报警或事件时，客户端必须考虑到所使用的对象类型属于特殊类型。各类型的定义如下：

TcEventLoggerEventType

TcEventLoggerEventType 源自 BaseEventType，并增加了 TwinCAT EventLogger 特有的属性：




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

TcEventLoggerAlarmConditionType

TcEventLoggerAlarmConditionType 源自 AlarmConditionType，并增加了 TwinCAT EventLogger 特有的属性：

TcEventLoggerAlarmConditionType
- TcEventClassGUID
- TcEventId
- TcEventJsonUserData
- TcEventSourceGUID
- TcEventSourceId
- TcEventSourceName

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

示例

以下示例基于 TwinCAT EventLogger 的标准代码示例，您可以从倍福信息系统中获取该标准代码示例。该示例包含 PLC 的代码片段，可用于触发事件和报警。

第 1 步：配置服务器

在本示例中，TwinCAT OPC UA Server 和 PLC 现于同一系统上本地运行。因此，系统还创建了 TcEventLogConfig.xml：

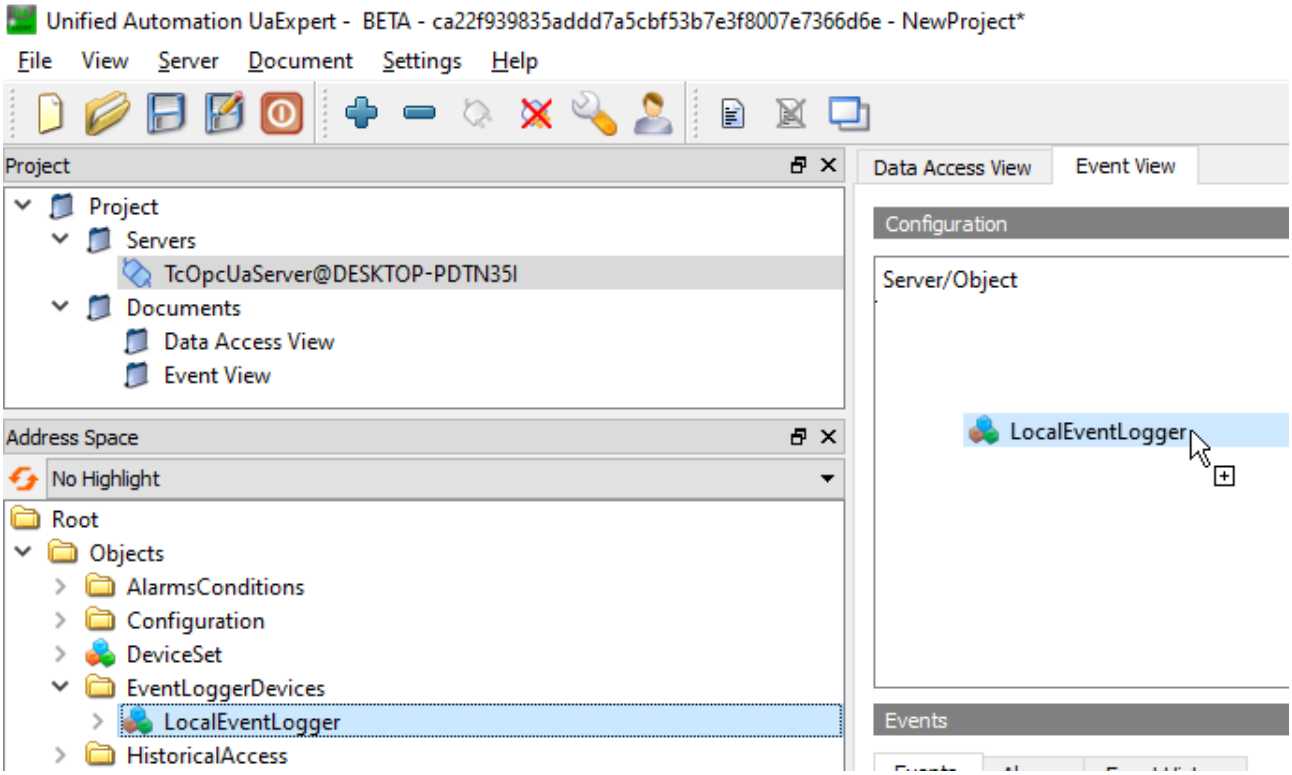
```
<TcUaEventLogConfig>
  <EventLoggerDevice Name="LocalEventLogger" AmsAddr="127.0.0.1.1.1"/>
</TcUaEventLogConfig>
```

第 2 步：激活 TwinCAT EventLogger 示例

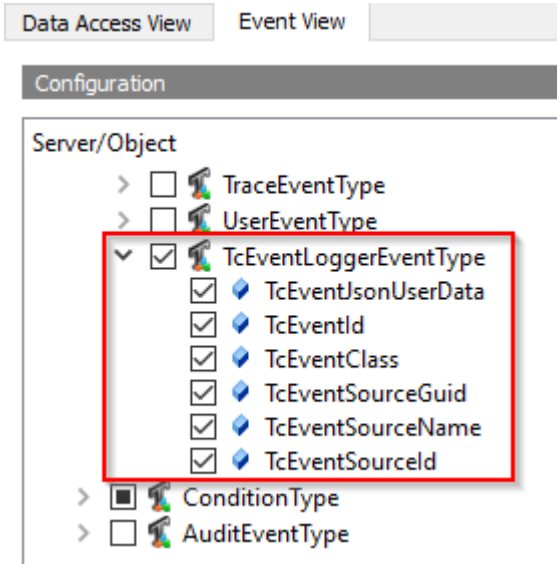
在启用 TwinCAT EventLogger 示例之前，我们首先要禁用事件或报警的自动触发功能。在本示例版本中，我们通过将 bSend 和 bAlmRaise 初始化为 FALSE 来完成此操作。然后在本地 PLC 运行时激活并执行该项目。

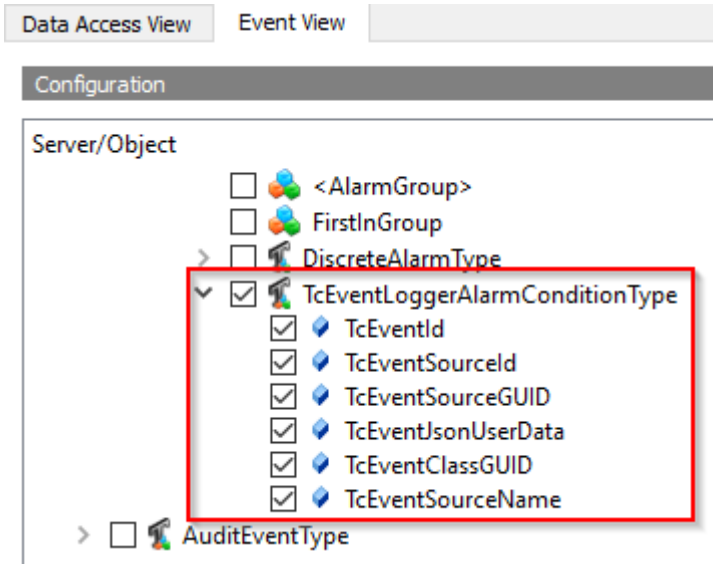
第 3 步：将 OPC UA 客户端与服务器连接

将 UA Expert 选为 OPC UA 客户端。与服务器建立连接后，您现在可以在 UA Expert 中添加一个新的“Event View（事件视图）”文档。然后将配置好的 TwinCAT EventLogger 设备拖放至事件视图中。



若要获取 TwinCAT EventLogger 特有的属性，UA Expert 必须对相应的 Event 或 AlarmType 进行过滤。您可以在事件视图的类型列表中找到 TcEventLoggerEventType 或 TcEventLoggerAlarmConditionType。示例：

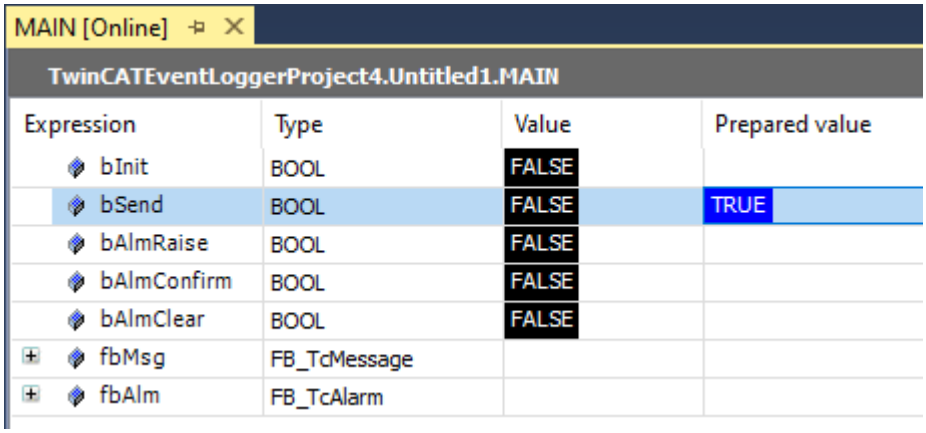


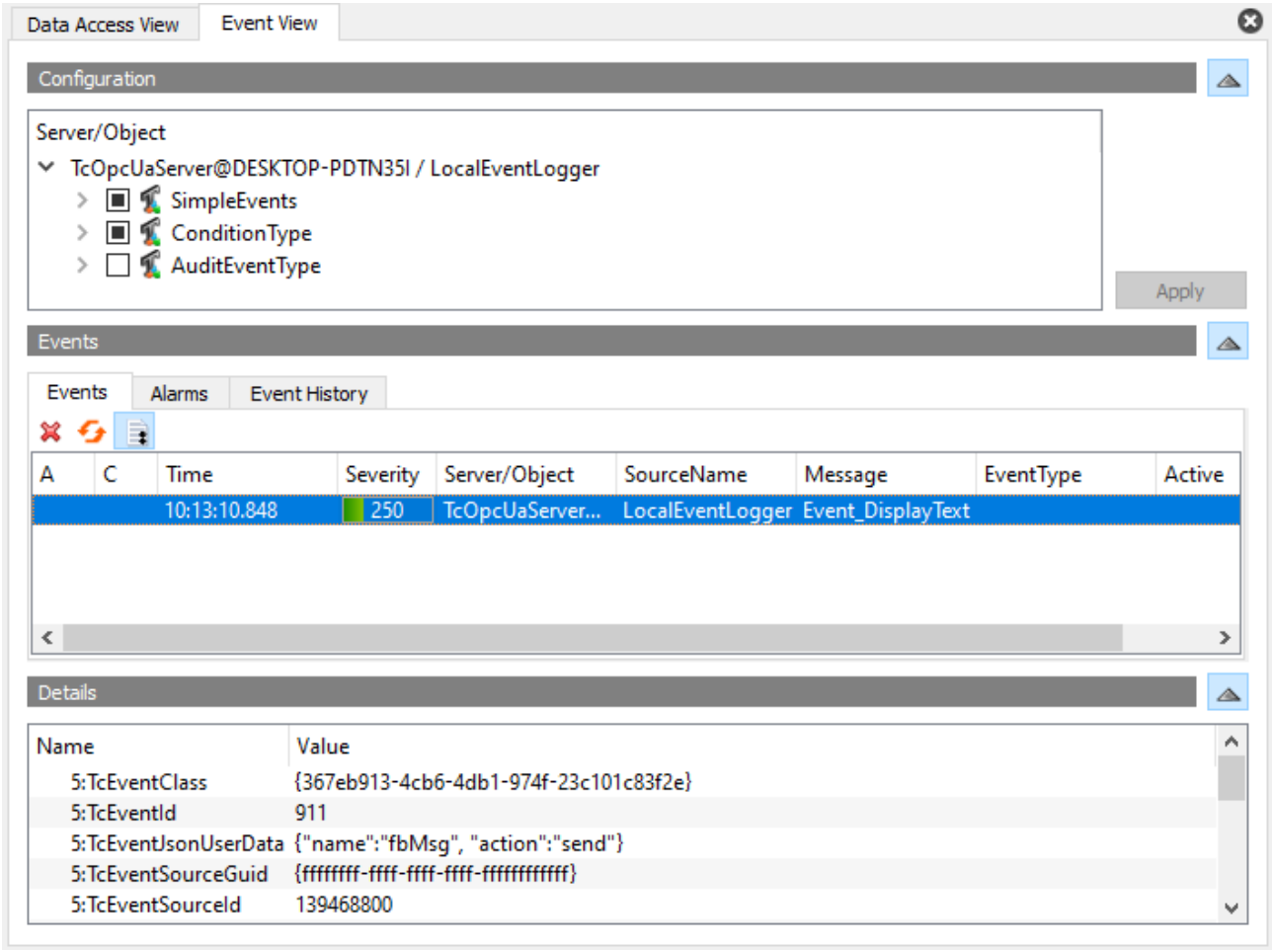


点击“**Apply**（应用）”按钮，即可应用这些筛选条件。

第 4 步：触发事件

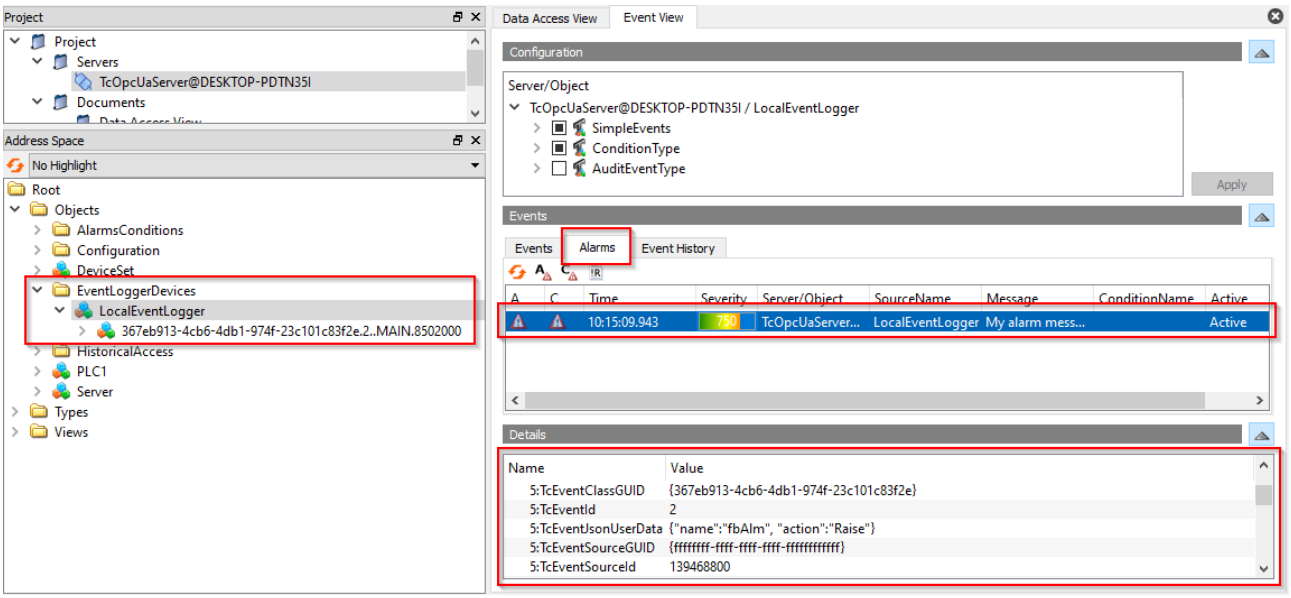
在 TwinCAT EventLogger 示例中，我们将变量 bSend 的值设置为 TRUE，以便触发事件。UA Expert 会自动接收该事件，并将其与 TwinCAT EventLogger 特有的属性一起显示在事件视图中。





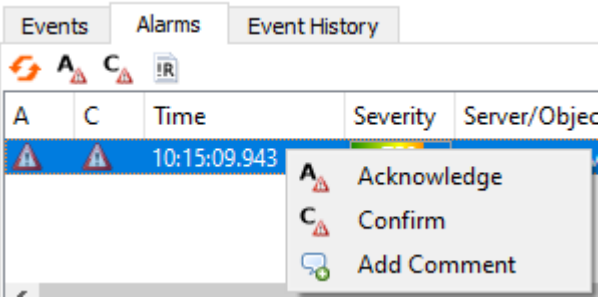
第 5 步：触发报警

在 TwinCAT EventLogger 示例中，我们将变量 bAlmRaise 的值设置为 TRUE，以便触发报警。UA Expert 会自动接收该报警，并将其与 TwinCAT EventLogger 特有的属性一起显示在事件视图中。此外，该报警还作为一个独立对象显示在 EventLogger 设备下方的命名空间中。



第 6 步：确认报警

除了接收报警外，还可以对报警进行确认。服务器也会向 TwinCAT EventLogger 报告该确认事宜。在 UA Expert 中，可以通过事件视图中的上下文菜单来确认报警。



确认

请注意，服务器仅会向 TwinCAT EventLogger 报告一个“Confirm（确认）”。由于 TwinCAT EventLogger 目前只提供“Confirm（确认）”的概念，因此有关“Acknowledge（确认）”的信息仍保留在服务器中。

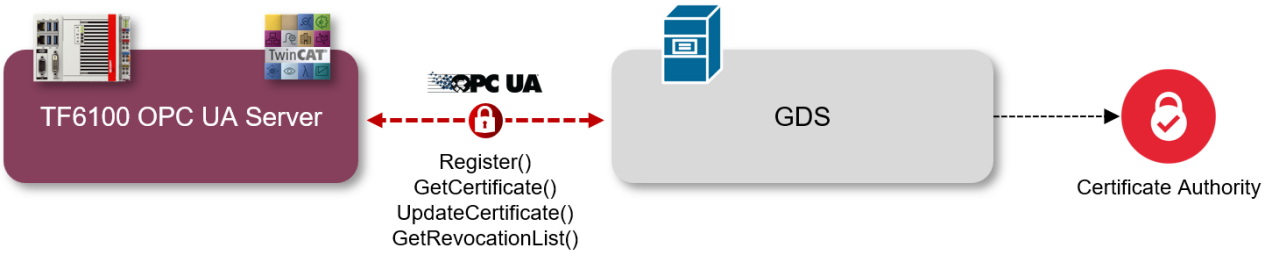
执行“Confirm（确认）”操作后，将会在 FB_TcAlarm 类型的 TwinCAT EventLogger 实例中设置相应的变量 eConfirmationState。

bRaised	BOOL	TRUE
eConfirmationState	TCEVENTCONFIRMA...	Confirmed

4.16 全局查找服务器

4.16.1 概述

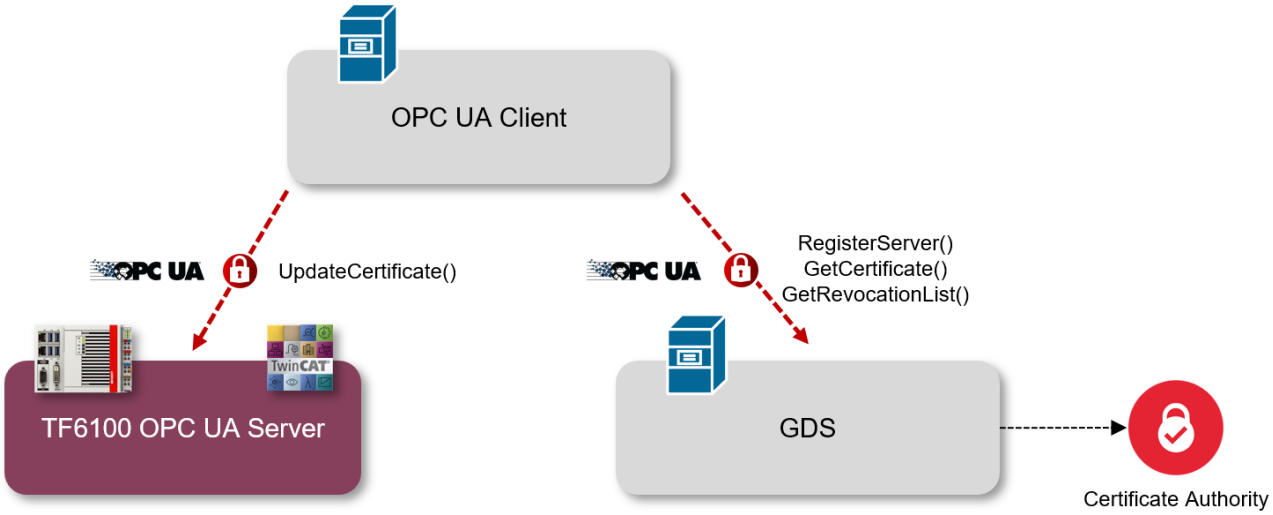
TwinCAT OPC UA Server 支持注册**全局查找服务器**（GDS）。GDS 提供了一个基于 OPC UA 的标准接口，用于注册 OPC UA 应用程序以及签发应用程序证书及相关的证书吊销列表信息。为此，GDS 与**证书**颁发**机构**（CA）建立了连接。使用标准化的 OPC UA 信息模型来注册应用程序并发放（和更新）证书。



在访问 GDS 时，TwinCAT OPC UA Server 支持推 [▶ 113]和拉 [▶ 114]这两种模型。

4.16.2 推模型

在该模型中，服务器包含一个标准接口，OPC UA 客户端（支持推模型）可以用它来代表服务器与全局查找服务器连接，在其中注册服务器应用程序，并申请包括当前**证书吊销列表**（CRL）在内的服务器证书。



使用该功能的前提条件是，OPC UA 客户端必须使用具有管理员权限的用户账户在服务器上进行身份验证。

OPC UA 客户端

任何支持推模型的客户端均可作为 OPC UA 客户端使用。各 OPC UA 工具套件制造商均提供相应的软件包。另外，OPC 基金会还在 Github 上提供了 GDS 示例客户端。

GDS 服务器

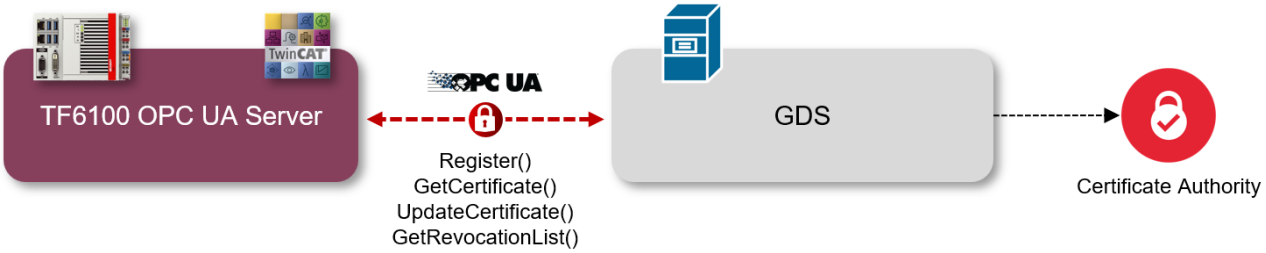
任何 GDS 均可作为全局查找服务使用。各 OPC UA 工具套件制造商均提供相应的软件包。另外，OPC 基金会还在 Github 上提供了 GDS 示例服务器。

服务器中的配置

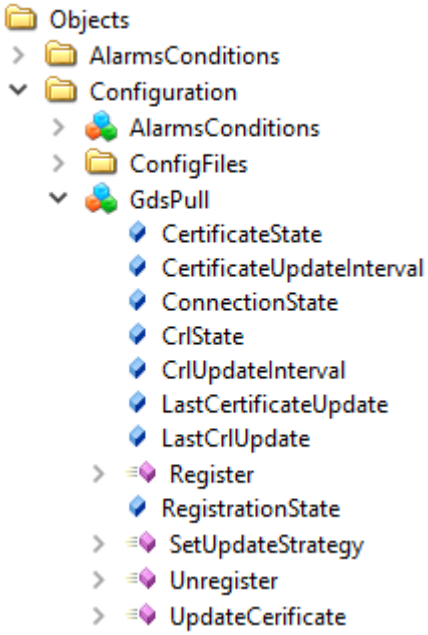
无需在 TwinCAT OPC UA Server 中进行其他特殊配置步骤，即可使用该功能。相应接口均已默认激活，可供具有管理员权限的用户使用。在初始化 [▶ 29]过程中配置的用户拥有所有必要的权限。

4.16.3 拉模型

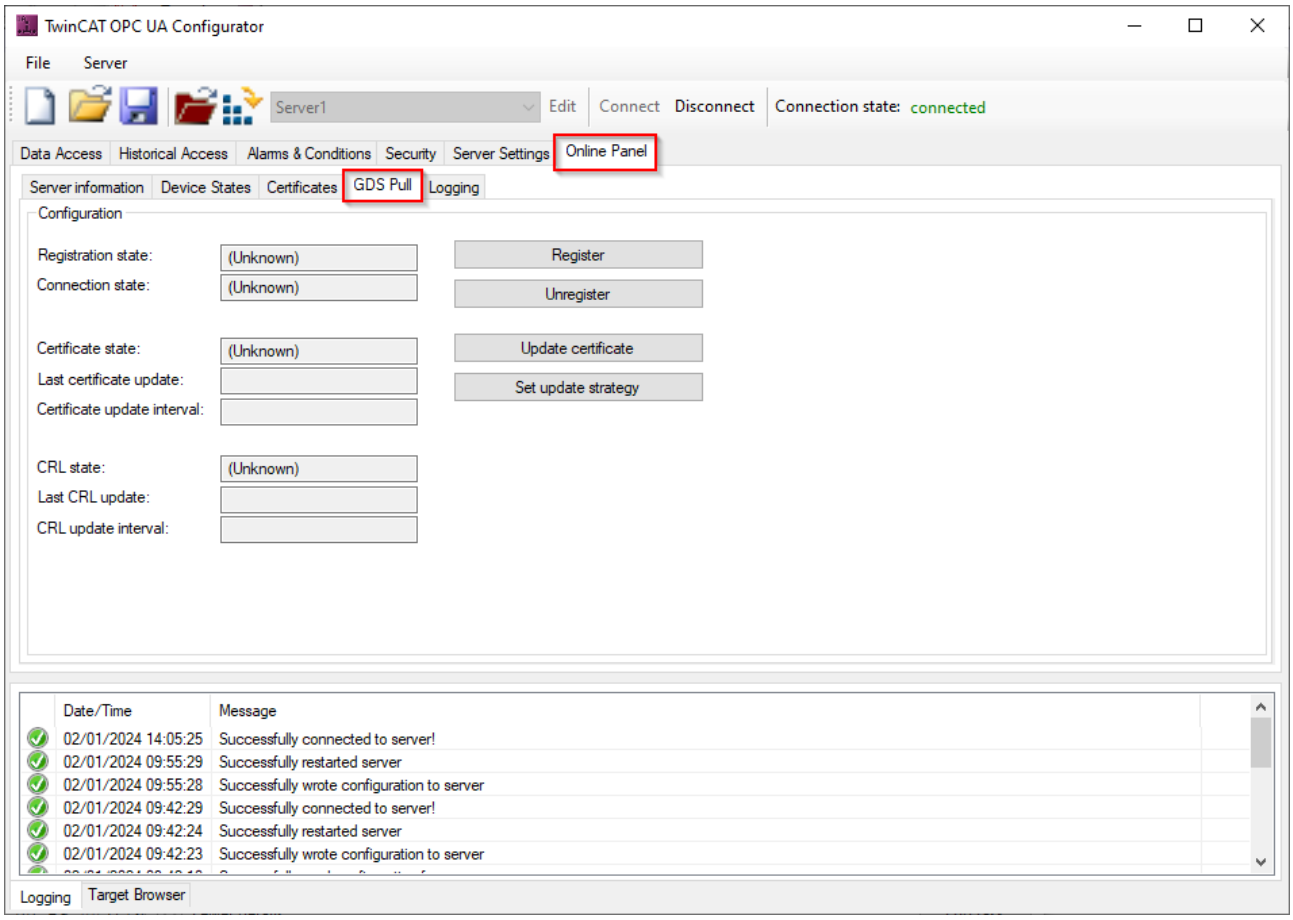
在该模型中，服务器会独立连接全局查找服务器，在其中注册为服务器应用程序，并获取合适的证书。



TwinCAT OPC UA Server 提供了通过其配置命名空间启用和配置 GDS 拉取功能的选项。



另外，也可以使用 TwinCAT OPC UA Configurator 在服务器中配置该功能。可使用相应的用户界面执行此操作。



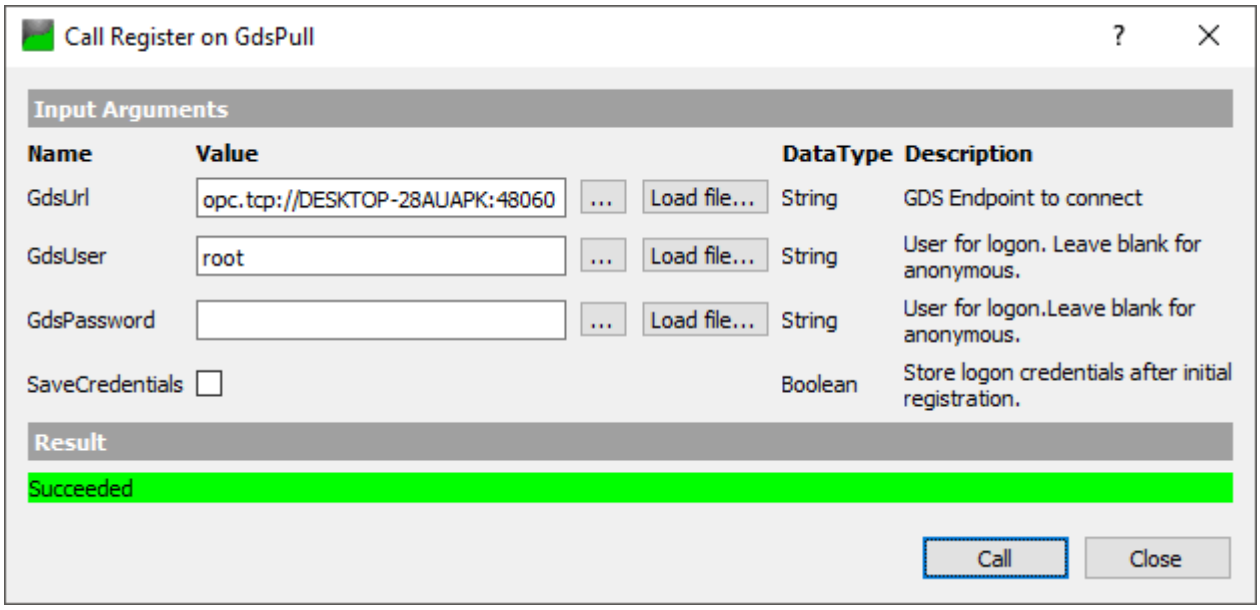
通过 OPC UA 客户端进行配置

下文将介绍如何使用通用 OPC UA 客户端通过服务器地址空间进行配置。将 Unified Automation 公司的 UA Expert 用作客户端软件。

第一步，必须在 GDS 将 TwinCAT OPC UA Server 注册为应用程序。通过 Register() 方法来完成此操作。在 GDS 注册完成后，会自动为服务器应用程序申请服务器证书。根据 GDS 应用程序的执行情况，将自动发放证书，或在经管理员批准后手动发放证书。可使用 RegistrationState 和 CertificateState 这两个变量检查服务器是否已注册全局查找服务，是否已获得该服务器发放的证书。变量 CrlState 表示证书吊销列表的状态以及是否能从 GDS 获取该列表。

#	Server	Node Id	Display Name	Value	Datatype	ource 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

该方法需要使用以下输入参数：



输入参数	含义
GdsUrl	以 opc.tcp://hostnameOrIpAddress:port 形式显示的全局查找服务 URL
GdsUser	有权注册新应用程序的 GDS 用户的用户名。
GdsPassword	用户密码
SaveCredentials	在 TwinCAT OPC UA Server 的配置文件中保存用户密码。出于安全考虑，不建议使用此设置。此参数专为全局查找服务开发，该服务强制要求进行用户名/密码身份验证。通常只需要在注册应用程序时进行一次用户名/密码身份验证。随后，所有与 GDS 的后续连接均需使用所发放的服务器证书。

服务器端点重新初始化

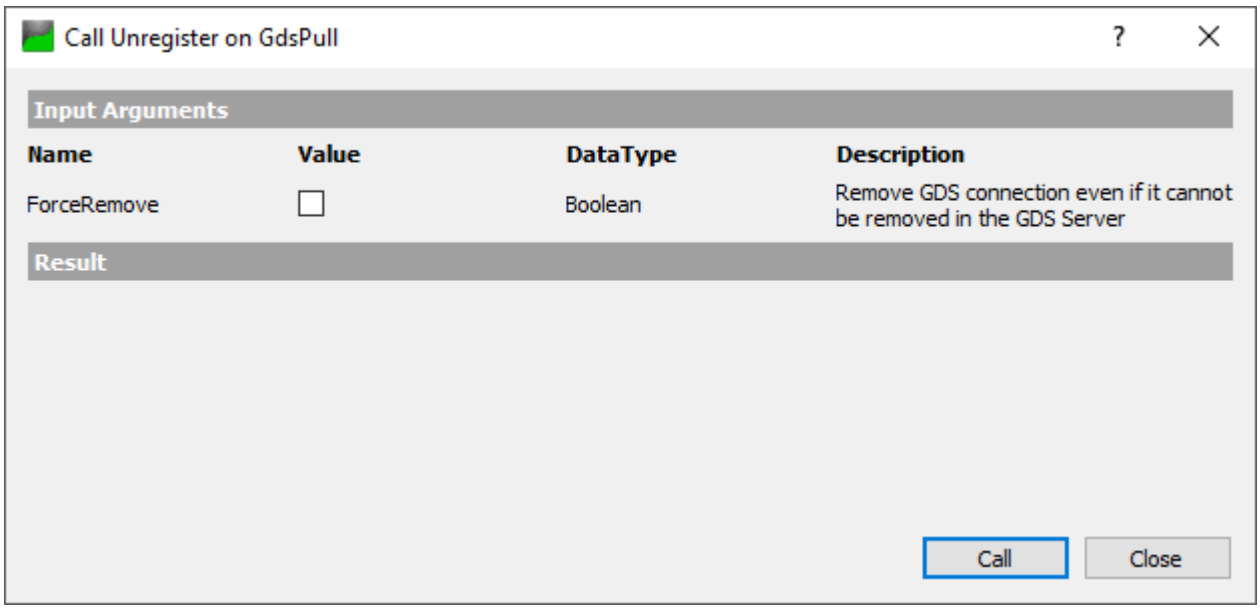
在向全局查找服务注册服务器应用程序并获取服务器证书后，服务器会对其端点重新进行一次初始化，导致已连接的客户端暂时断开连接。

服务器应用程序在全局查找服务注册完成后，将在 TwinCAT OPC UA Server 的安装目录下创建一个名为“TcUaGdsClientConfig.xml”的新文件。该文件包含已配置的 GDS 的连接信息和从中获取的注册信息，以及服务器应用程序的时间戳信息，例如证书和 CRL 最近一次更新的时间。

取消在全局查找服务的注册

可采用 Unregister() 方法来取消 TwinCAT OPC UA Server 应用程序在 GDS 的注册。成功执行该方法后，将会取消服务器应用程序在 GDS 上的注册，并删除 TcUaGdsClientConfig.xml 文件的内容。

该方法需要使用以下输入参数：



输入参数	含义
ForceRemove	如果无法再与全局查找服务建立连接，可以通过设置此输入参数来删除与 GDS 的连接。TwinCAT OPC UA Server 将从其配置中删除 GDS。

已发放的服务器证书和 CRL 在 TwinCAT OPC UA Server 与 GDS 解耦后仍然有效。若要删除它们，并使用拥有自签名证书的服务器，则必须删除服务器 PKI 目录中的相应文件，然后重启服务器。然后，服务器会另行创建一个自签名证书。

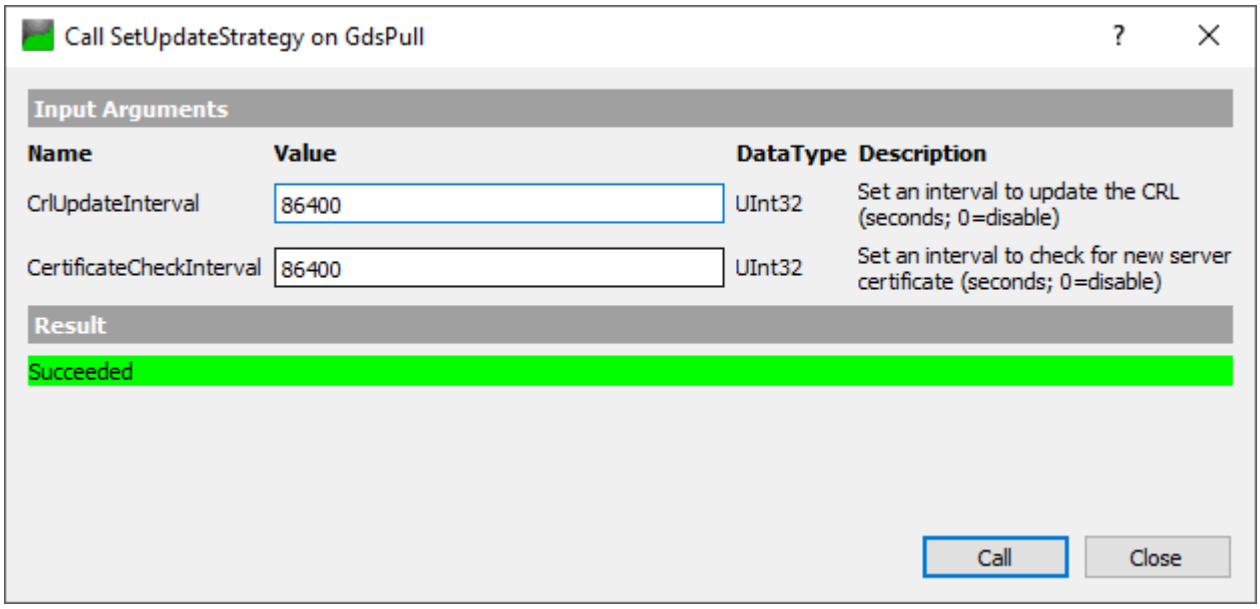
更新服务器证书

可通过执行 UpdateCertificate() 方法，在常规更新间隔时间之外申请 GDS 更新服务器证书。该方法不需要使用任何其他输入参数。

设置服务器证书和 CRL 的更新间隔时间

可通过执行 SetUpdateStrategy() 方法来设置服务器证书和证书吊销列表的更新间隔时间。

该方法需要使用以下输入参数：



输入参数	含义
CrlUpdateInterval	设置证书吊销列表的更新间隔时间（秒）。
CertificateCheckInterval	设置服务器证书的更新间隔时间（秒）。

4.17 安全

4.17.1 概述

OPC UA 作为一种通信技术如此成功的原因之一是其集成式安全机制。基于 OPC UA 的数据通信可以在 2 个层面上确保安全：传输层和应用层。在与服务器连接时，客户端首先要选择一个端点，该端点指定了将使用的安全功能。

端点

服务器会向客户端提供一个不同端点 [▶ 118]的列表，客户端可以与之连接。端点描述的是通过该端点进行的通信连接应实现的安全功能（如“消息安全”模式、“安全策略”和可用“身份令牌”）。例如，端点可能需要对数据包进行签名和加密（传输层），以及根据用户名/密码对客户端进行额外的身份验证（应用层）。

传输层

基于 OPC UA 的通信连接可以在传输层得到保护。这是通过使用客户端/服务器证书以及客户端和服务器应用程序之间的相互信任关系来实现的。在这里，客户端必须信任服务器证书，反之亦然，这样才能建立通信连接。这需要相互交换证书 [▶ 119]。

应用层

除了传输层，还可以在应用层确保通信连接的安全性。为此，可以使用服务器端点提供的各种身份验证机制 [▶ 121]。

4.17.2 端点

TwinCAT OPC UA Server 通过默认端口 4840/tcp 为 OPC UA 客户端提供各种端点。端点定义了客户端与服务器之间的连接类型，以及是否应确保其连接的安全性。

标准端口

请注意，标准端口 4840 可能会被其他 OPC UA 服务器占用，例如 OPC 基金会的**局部查找服务器**（LDS），一些供应商会将其与 OPC UA 软件包配合使用。

信任关系

请注意，若要使用安全端点，必须在服务器与客户端之间建立信任关系，而这种关系通常是通过两者的证书来实现的。接下来将介绍如何在服务器端配置这种信任关系 [▶ 119]。

已弃用的端点

请注意，随着时间的推移，端点中当前可用的安全配置文件可能会被归类为潜在不安全文件，并由新版文件所取代。在这种情况下，建议更新 TwinCAT OPC UA Server。可以使用配置开关（<AllowDeprecatedSecurityPolicies>）来重新激活已弃用且被列为不安全文件的安全策略。
出于安全考虑，倍福建议禁用该配置开关。

端点列表

以下列表对 TwinCAT OPC UA Server 的端点进行了汇总。其中包括已经停止使用的端点。默认情况下，TwinCAT OPC UA Server 仅提供目前公认安全的端点。自安装版本 4.3.28 起，出于安全和认证原因，未加密端点也将默认被禁用。

安全配置文件	安全模式	简介
None	None	该端点不对消息进行加密或签名。相反，可以进行身份验证。

安全配置文件	安全模式	简介
Basic128Rsa15（已弃用）	签名 签名和加密	从安全角度来看，该端点已被列为弃用端点，并且默认处于禁用状态。如有必要，可以再次启用该端点。
Basic256（已弃用）	签名 签名和加密	从安全角度来看，该端点已被列为弃用端点，并且默认处于禁用状态。如有必要，可以再次启用该端点。
Basic256Sha256	签名 签名和加密	服务器中当前存在的端点，用于进行安全签名和加密。可以进行额外的身份验证。
Aes256_Sha256_RsaPss	签名 签名和加密	服务器中当前存在的端点，用于进行安全签名和加密。可以进行额外的身份验证。
Aes256_Sha256_RsaOaep	签名 签名和加密	服务器中当前存在的端点，用于进行安全签名和加密。可以进行额外的身份验证。

可以通过服务器配置启用或禁用列表中的所有端点。下图中的所有端点均已启用。

4.17.3 证书交换

若要通过安全端点 [▶ 118]在传输层建立安全通信连接，必须在客户端与服务器之间建立相互信任。默认情况下，TwinCAT OPC UA Server 和 TwinCAT OPC UA Client 在首次启动时会生成一个机器特定的自签名密钥对，该密钥对由一个公钥和一个私钥组成，且包含具有有效期的相应的证书。

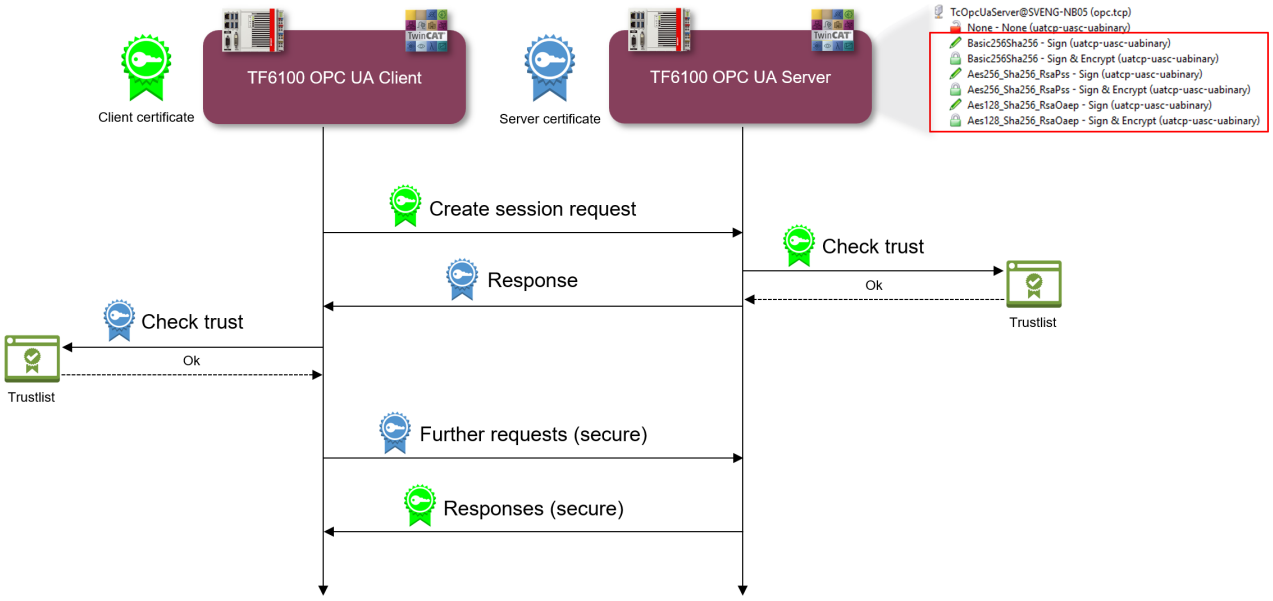
证书有效期

自签名证书的默认有效期为 20 年。这是服务器的特定设置，可根据需要进行调整。

除了使用自签名证书外，您也可以持有并使用任何认证机构发放的证书。但是，我们建议使用全局查找服务器 [▶ 113]，以减少行政管理工作量。

若要在 OPC UA 客户端与 TwinCAT OPC UA Server 之间建立信任关系，需要使用客户端证书的公钥。服务器必须信任该证书。服务器在应用程序目录 [▶ 46]的一个子目录中管理客户端证书的信任设置。

下图说明了建立安全通信连接时客户端和服务器证书之间的关系：



客户端随 CreateSession 请求一起传输其公钥。然后，服务器可选择检查信任关系。如果服务器信任客户端，便会在响应中传输自己的公钥。因此，客户端还可以选择检查与服务器之间的信任关系。

如果确保互信，则启动通信连接。服务器的公钥用于加密客户端向服务器发出的请求。然后，服务器对客户端的响应将使用客户端的公钥进行加密。通信双方都可以选择用自己的私钥对收到的消息进行解密。

消息经过反向签名：用发件人的私钥对消息进行签名。由于收件人能识别发件人的公钥，因此可以验证签名。

通过文件系统配置信任关系

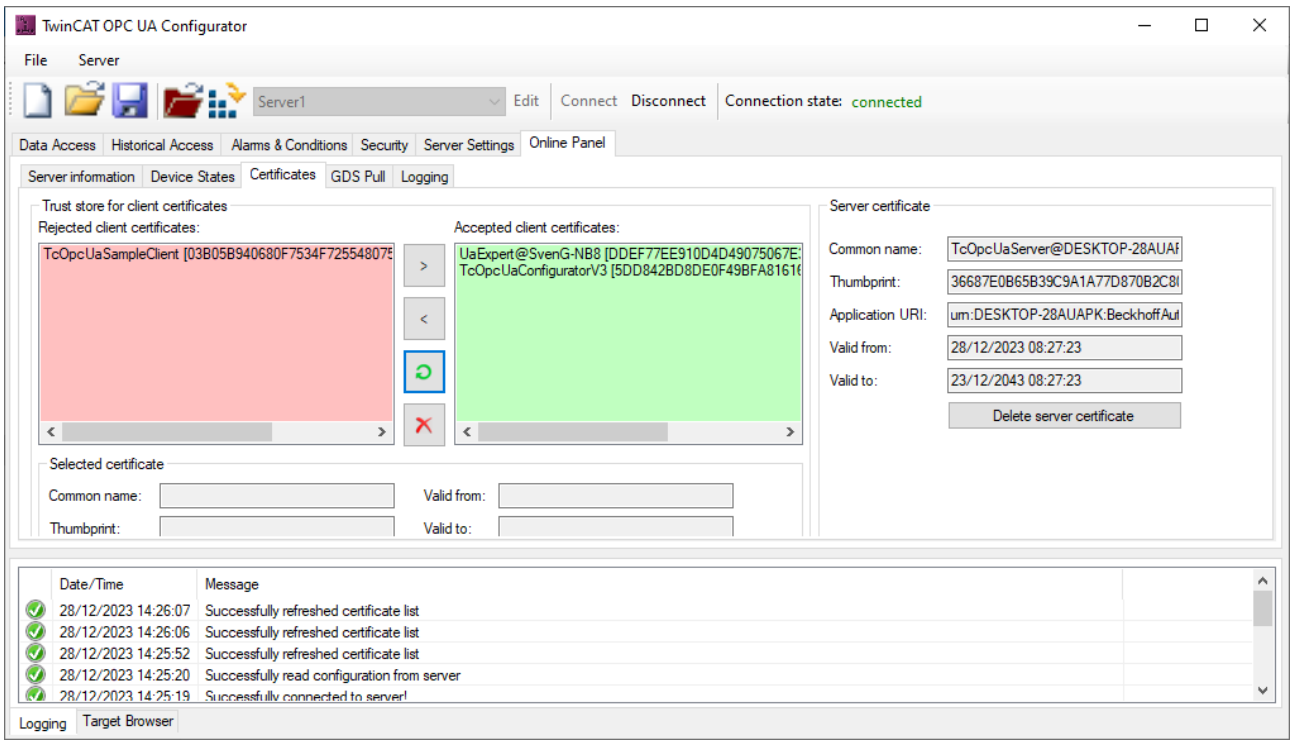
通过在信任/拒绝目录之间移动客户端证书，可以相应地调整信任设置。客户端证书的公钥会在客户端首次尝试与安全端点连接时自动存储在被拒绝证书的目录中。通过随后将公钥移至受信任证书的目录，客户端在下一次尝试连接时就会受到信任，可以进行连接。

AutomaticallyTrustAllClientCertificates

如果在服务器上启用了该配置选项，它会自动信任所有客户端证书。在这种情况下，服务器不会将这些证书列在上述任何目录中。

使用配置器配置信任关系

您还可以通过 配置器 [▶ 40] 调整信任设置。TwinCAT OPC UA Configurator 包括一个用于配置信任设置的图形用户界面。



4.17.4 身份验证

OPC UA 客户端应用程序可以通过各种 IdentityToken 向 TwinCAT OPC UA Server 进行身份验证。支持使用以下 IdentityToken：

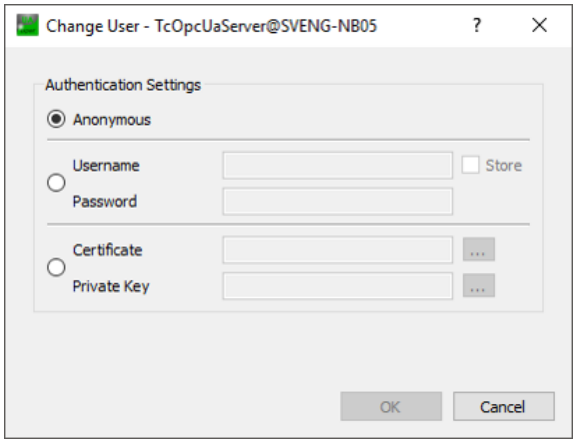
- Anonymous（匿名）
- Username/Password（用户名/密码）
- User certificate（用户证书）

交付状态

“Anonymous（匿名）” IdentityToken 在服务器交付时已启用；但是，需要对服务器进行一次性初始化才能将其启动。然后，该 IdentityToken 将被禁用，客户端应用程序必须使用服务器上的用户名进行身份验证。

Anonymous（匿名）

此类身份验证允许任何 OPC UA 客户端与服务器应用程序连接。不需要指定用户身份，这意味着无法在服务器上定义访问权限。倍福建议在调试服务器后禁用此类身份验证。可以通过 TwinCAT OPC UA Configurator 来完成此操作。以下是 OPC UA 客户端应用程序“UA Expert”的屏幕截图示例：



Username/Password（用户名/密码）

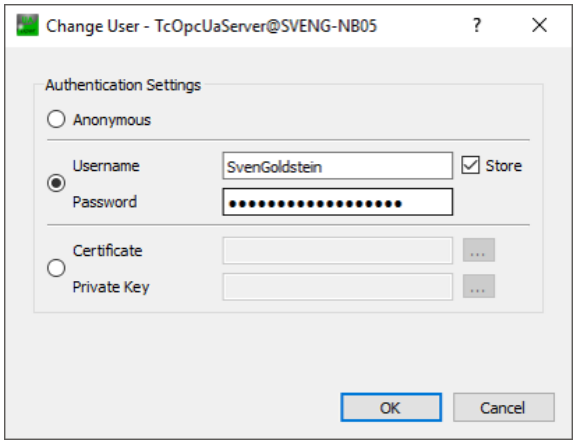
此类身份验证使用用户名/密码组合对客户端进行身份验证，以访问服务器应用程序。然后即可在服务器上为相应的用户身份定义访问权限。可在不同等级定义用户身份：

- 在服务器中定义用户身份
- 用户身份来自下级操作系统（如本地 Windows 用户）
- 用户身份来自活动目录（例如，如果工业 PC 是 Windows 域名的一部分）

使用 UserIdentityToken 时的建议

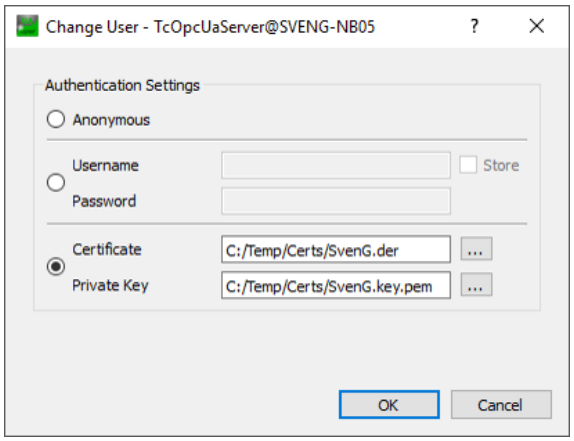
若要使用 UserIdentityToken 对客户端应用程序进行身份验证，倍福建议不要在 OPC UA 服务器中定义用户身份。相反，应在操作系统或活动目录（若适用）中定义用户账户。

以下是 OPC UA 客户端应用程序“UA Expert”的屏幕截图示例：



User certificate（用户证书）

此类身份验证使用证书对服务器应用程序进行身份验证。服务器端对用户证书的处理与传输层对证书的使用完全相同，即服务器必须信任（用户）证书，客户端才能成功地通过证书向服务器进行身份验证。为此，可在服务器上创建一个用于管理用户证书的独立目录（“pkiuser”）。以下是 OPC UA 客户端应用程序“UA Expert”的屏幕截图示例：



此外，必须通过 TwinCAT OPC UA Configurator 创建一个 X.509 用户，才能使用该身份验证方法。用户的姓名必须与证书的通用名一致。

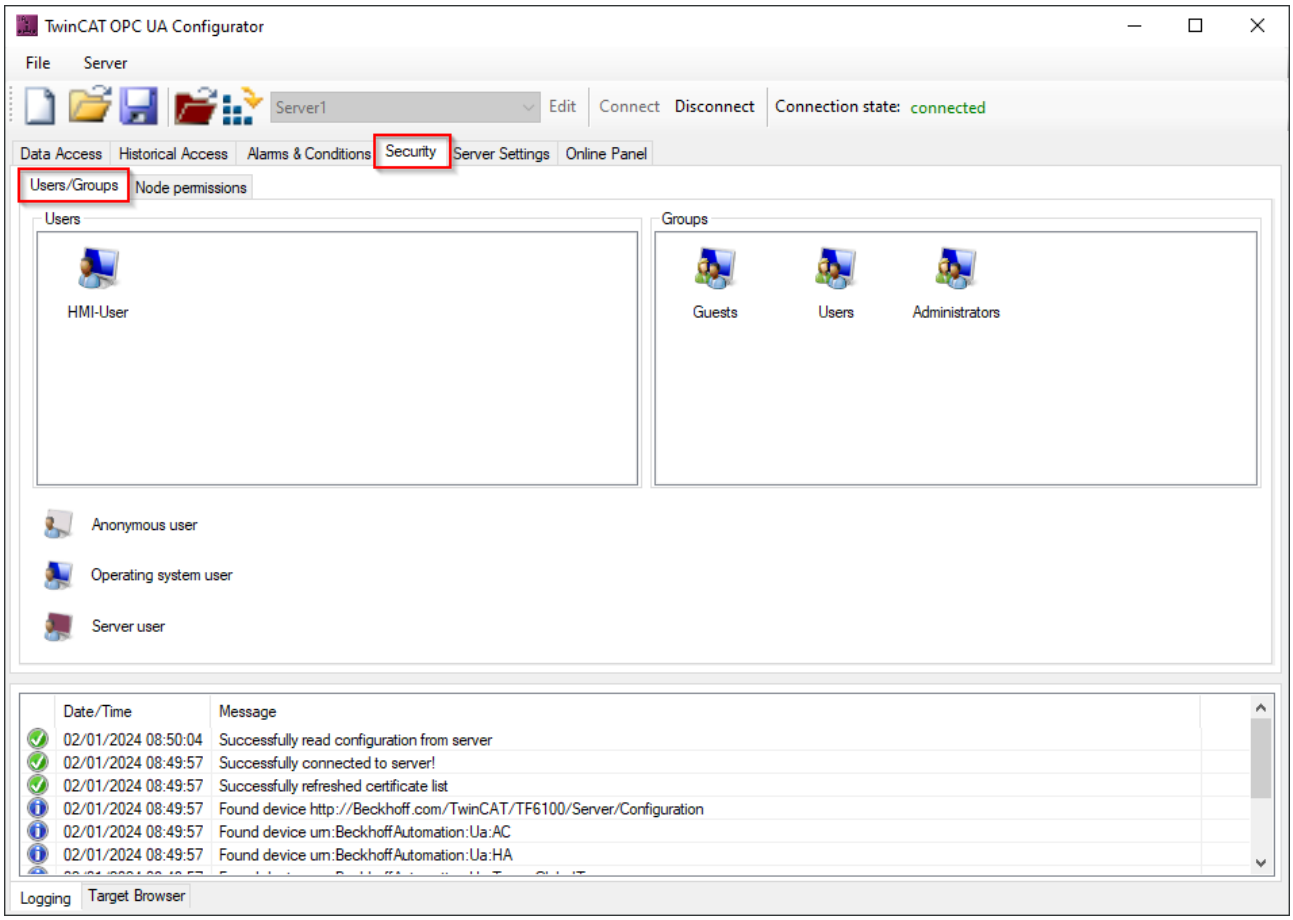
注意

身份验证和服务器证书

在结合身份验证使用未加密端点时，TwinCAT OPC UA Client 仍然需要 OPC UA 服务器证书中的公钥，以便在传输过程中对密码进行加密。为此，必须在 TwinCAT OPC UA Client 中信任该证书（参见“证书交换 [▶ 119]”）。

配置

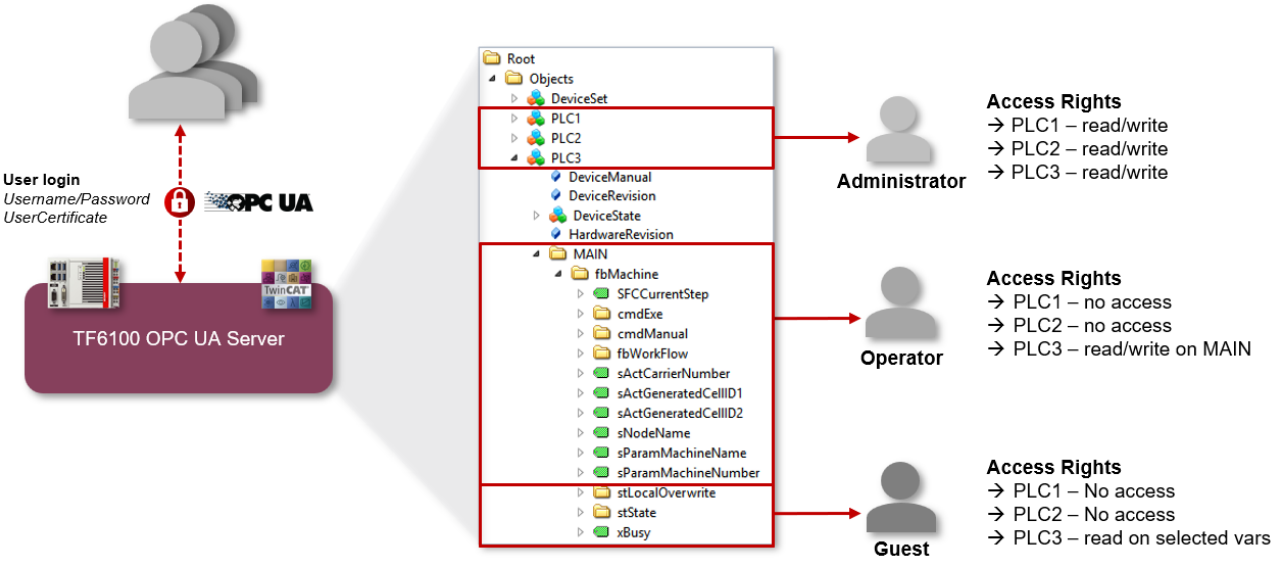
IdentityToken 通常通过 TwinCAT OPC UA Configurator 进行配置。该配置器，例如独立配置器，有一个用于启用 IdentityToken 的图形用户界面：



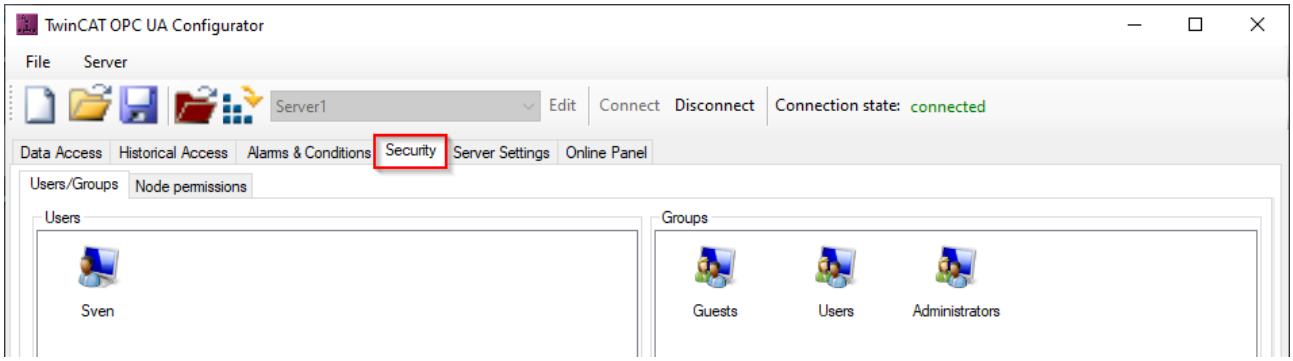
4.17.5 访问权限

TwinCAT OPC UA Server 可以为特定已通过身份验证的用户 [▶ 121]配置访问权限。既可为整个命名空间配置这些访问权限，也可为独立节点配置这些访问权限。这样，用户既可以访问 ADS 设备（例如，访问不同的 PLC 运行时系统），也可以精细设置各个符号。

这些安全设置适用于可在服务器命名空间中显示的所有数据访问 [▶ 48]设备。

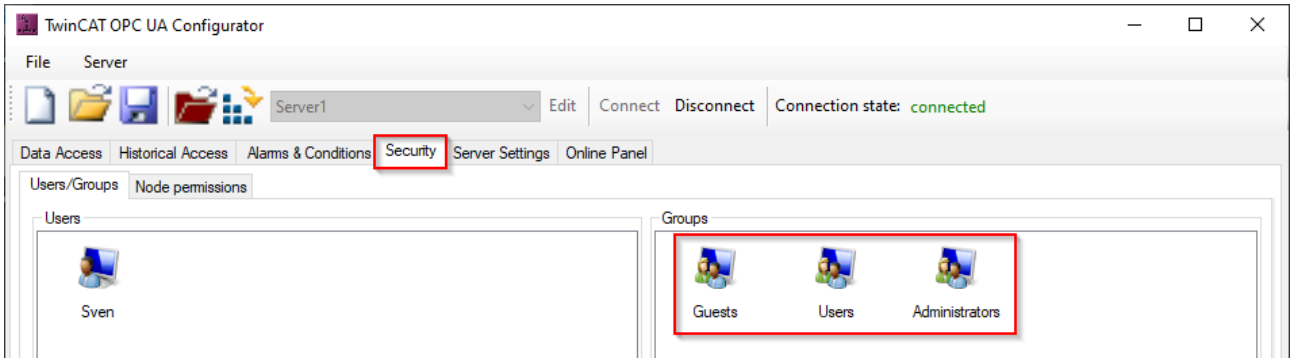


通过 TwinCAT OPC UA Configurator 配置该功能。在独立版配置器中，您可以在“**Security**（安全）”选项卡下找到相应的配置界面。



配置命名空间的访问权限

始终基于已配置的用户组来配置用户对各个命名空间的访问权限。您可以在用户组设置中管理各个命名空间的相应访问权限。有些群组已预先配置完毕，您可以使用其配置参数作为参考。



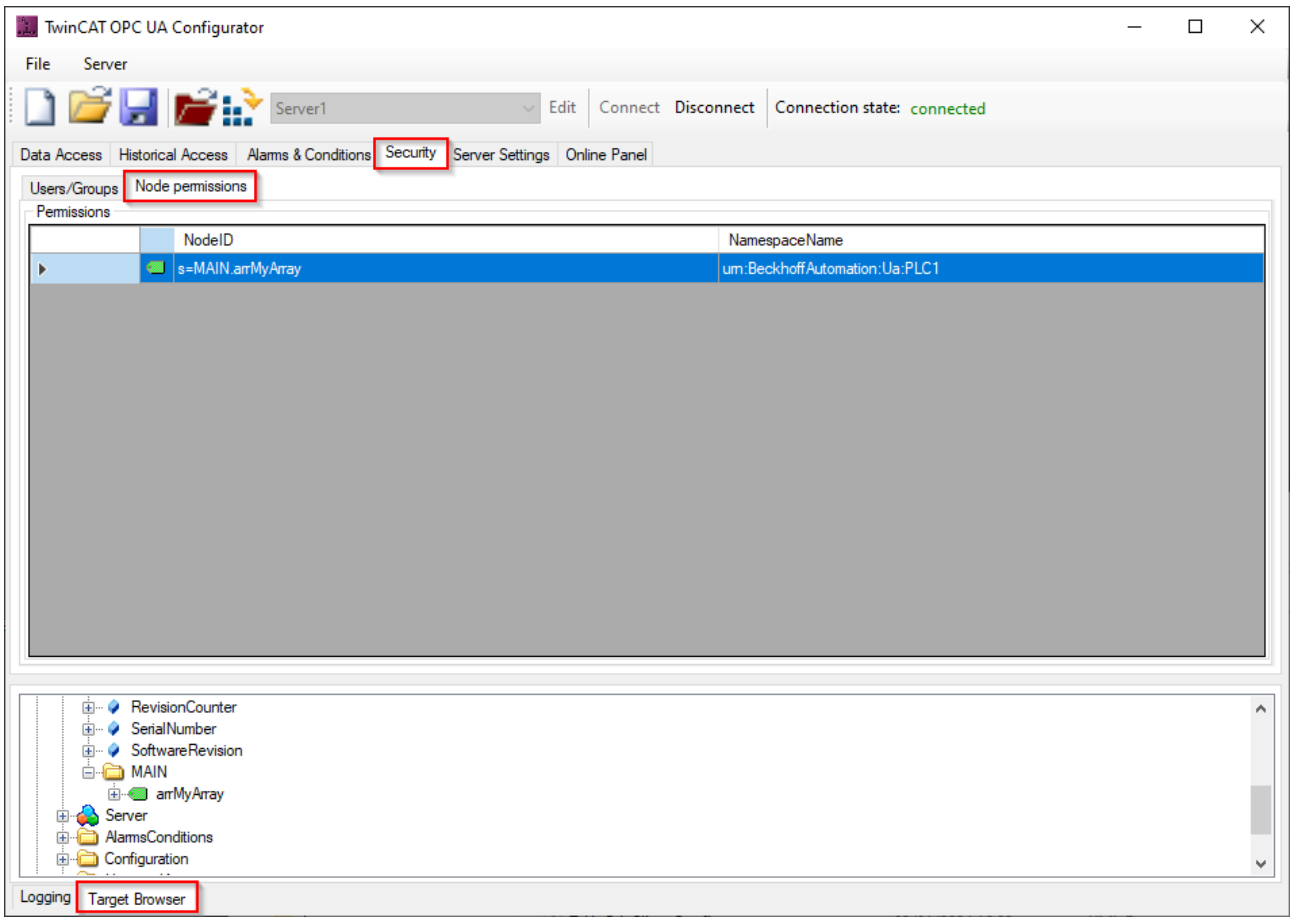
下表概述了在交付时为每个用户组定义的权限。

用户组	描述
管理员	为服务器管理员预定义的用户组。该用户组拥有所有命名空间的完全访问权限。
访客	为访客用户预定义的用户组。该用户组对服务器的访问会受到限制，拥有 ReadAttribute、ReadValue 和 Browse 权限的用户只能访问默认命名空间“0”。
用户	为普通用户预定义的用户组。该用户组拥有对所有命名空间的扩展访问权限，特别是对预配置数据访问 [▶ 48]设备的命名空间拥有完全访问权限。

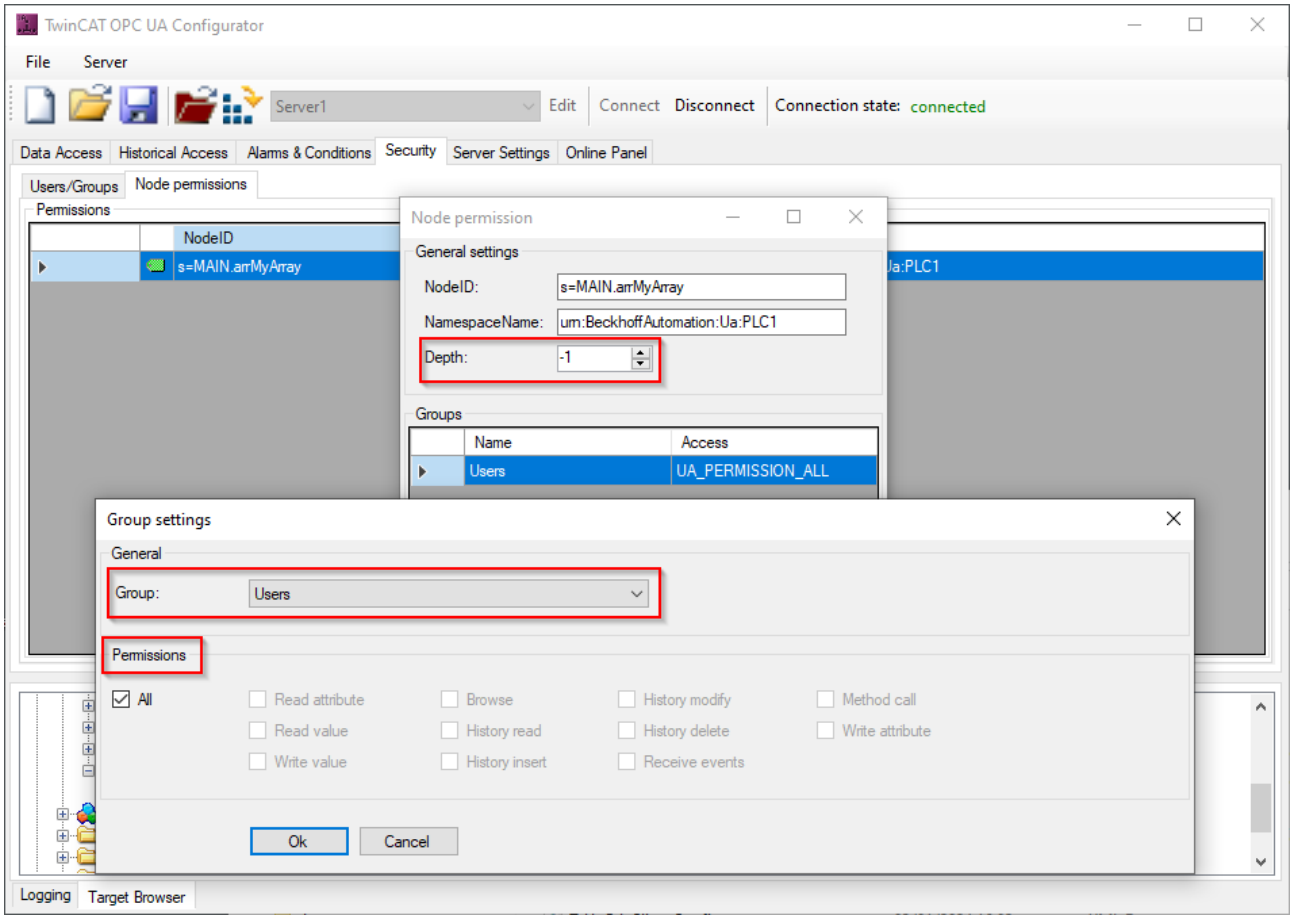
将用户添加至用户组后，他们会自动从该用户组继承相应的权限。

在节点层面配置访问权限

可使用 “**Node permissions**（节点权限）”选项卡在节点层面配置扩展和超精细权限。子元素可以继承这些权限。您可以使用 Target Browser，通过拖放操作将要配置的节点从服务器转移到配置中。



可将节点上的各个权限与配置的用户组相关联。

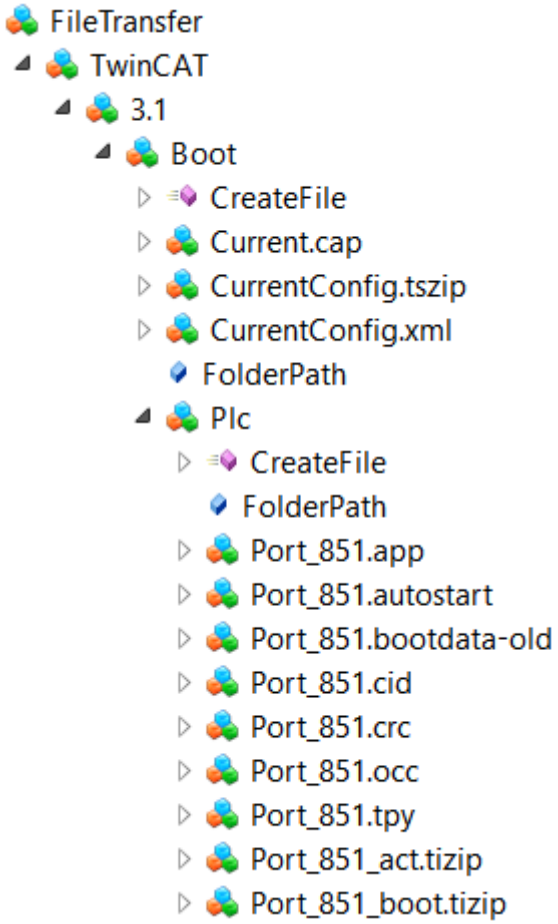


4.18 文件传输

4.18.1 概述

OPC UA 规范包含了用于传输文件的特殊数据类型。这个名为“FileType”的特殊 ObjectType 描述了数据传输的信息模型。可将文件作为 OPC UA 中的简单变量用 ByteStrings 进行建模。FileType 是一个文件，其中包含了访问该文件的方法。OPC UA 规范提供了关于 FileType 的更多信息，以及在 OPC UA 命名空间访问文件的基本方法和属性的结构和处理。

倍福已经实现了一种将文件和文件夹从本地硬盘加载到 OPC UA 命名空间的通用方法。每个文件由一个 FileType 表示，并允许对该文件进行读写操作。此外，每个文件夹都包含一个用来在硬盘上创建新文件的 CreateFile() 方法，以及一个用来指定 OPC UA 服务器上文件夹实际路径的单独 FolderPath。



OPC UA 服务器设备管理器中的 FileTransfer

只有倍福设备管理器（IPC 诊断）的 OPC UA 服务器具备此功能。TwinCAT OPC UA Server 也提供部分文件传输功能。然而，能够公开所有文件和文件夹的一般功能仅在 OPC UA 服务器中可用，它是每个倍福工业 PC 或嵌入式控制器上自动提供的设备管理器的一部分。更多信息请参见设备管理器文档。

4.18.2 配置

FileType 对象在一个名为“FileTransfer”的独立命名空间中创建。使用 一个 XML 文件（files.xml）来配置该命名空间，并通过 OPC UA 选择可用文件和文件夹。该文件必须与 OPC UA 服务器的可执行文件在同一目录中。系统必须重新启动，以激活配置。XML 文件包含关于文件夹路径和搜索掩码的信息，搜索掩码定义了在 OPC UA 命名空间中发布的文件：

```
<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>
```

使用 OPC UA 客户端读取文件

常规文件处理方法详见 OPC UA 规范附录 C。
因此，通过 OPC UA 读取文件可以分为以下几个步骤：

- 调用文件的 Open 方法。该方法会返回一个必须保存的文件句柄，以便日后进行访问。该模式定义了是对文件执行读取还是写入操作（参见“文件模式”）。
- 通过属性“Size”确定文件大小。这样，在调用 Read 方法时便可读取整个文件。
- 调用 Read 方法。插入文件句柄和文件大小作为输入。选择方法调用后保存文件内容的目标文件夹。
- 调用 Close 方法来启用文件句柄。

文件模式

下表显示了所有可用文件模式：

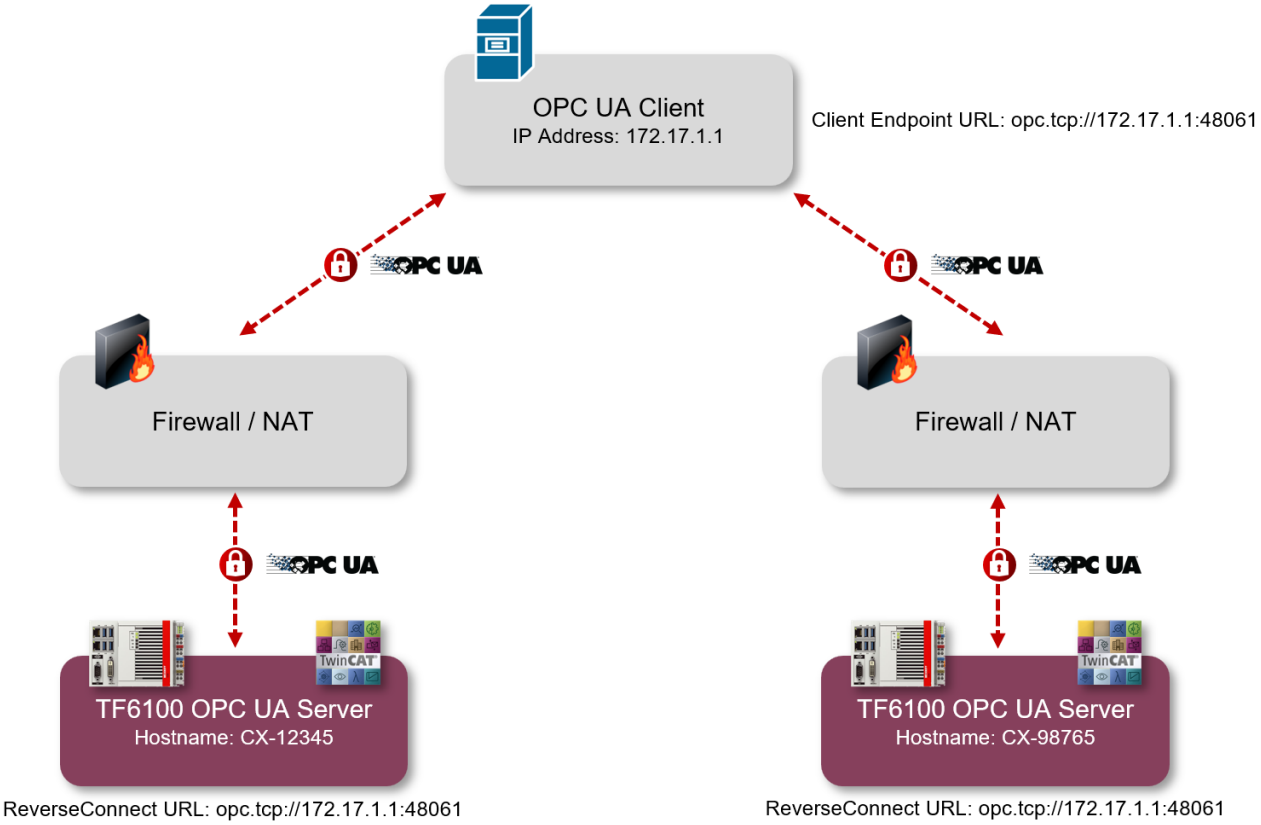
字段	位	描述
Read	1	文件已打开，可以读取该文件。如果未对该位进行设置，则不能执行读取操作。
Write	4	文件已打开，可以写入该文件。如果未对该位进行设置，则不能执行写入操作。
EraseExisting	6	现有文件内容已删除，并提供了一个空文件。
Append	10	文件已打开并位于末尾，否则会将其移至开头。可以通过 SetPosition 更改该位置。

通用行为

并行打开的文件数量原则上不受限制，仅受底层操作系统的限制。但是，文件的打开有 60 秒的超时限制。超时后，打开的文件不会立即自动关闭。相反，它们会被标记为“to close（待关闭）”。在此期间，如果使用了相应的 FileHandle 进行读取/写入操作，超时将被重置，FileHandle 仍然有效。如果在此期间对同一文件执行了打开操作，将会启用旧版 FileHandle。如果 OPC UA 客户端与服务器断开连接，但仍有文件处于打开状态，那么，属于此会话的所有 FileHandle 均将自动关闭。

4.19 反向连接

TwinCAT OPC UA Server 支持 OPC UA 的 ReverseConnect 功能，以在服务器与客户端之间建立反向通信连接。若要激活该功能，必须在服务器中存储客户端地址列表。然后，服务器会为列表中的每个客户端建立 OPC UA TCP 连接。当 OPC UA 客户端需要与位于防火墙或 NAT 设备后面的服务器建立连接时，通常会使用此类连接设置。下图阐释了这种关系。



本例中有一个 OPC UA 客户端需要连接 2 个服务器，这两个服务器均在防火墙后面。防火墙会屏蔽所有传入的通信流量，并且不会打开内部网络的任何端口。现在，客户端已完成 ReverseConnect 配置，并在客户端端点 URL opc.tcp://172.17.1.1:48061 下打开了其专属的网络端口。各下级服务器也已完成 ReverseConnect 设置，并输入了客户端端点 URL 作为 ReverseConnect URL。现在，服务器会使用该 URL 打开并维护与客户端的 TCP 连接。OPC UA 客户端与服务器之间实际通信是通过该 TCP 连接进行隧道传输的。从防火墙的角度来看，由于最初建立的是 TCP 连接，因此该通信属于“传出”通信。只需在防火墙中启用传出通信端口（本例中为 48061）即可。

客户端兼容性

请注意，OPC UA 客户端也必须支持该功能，并可通过其客户端端点 URL 进行访问。

服务器中的配置

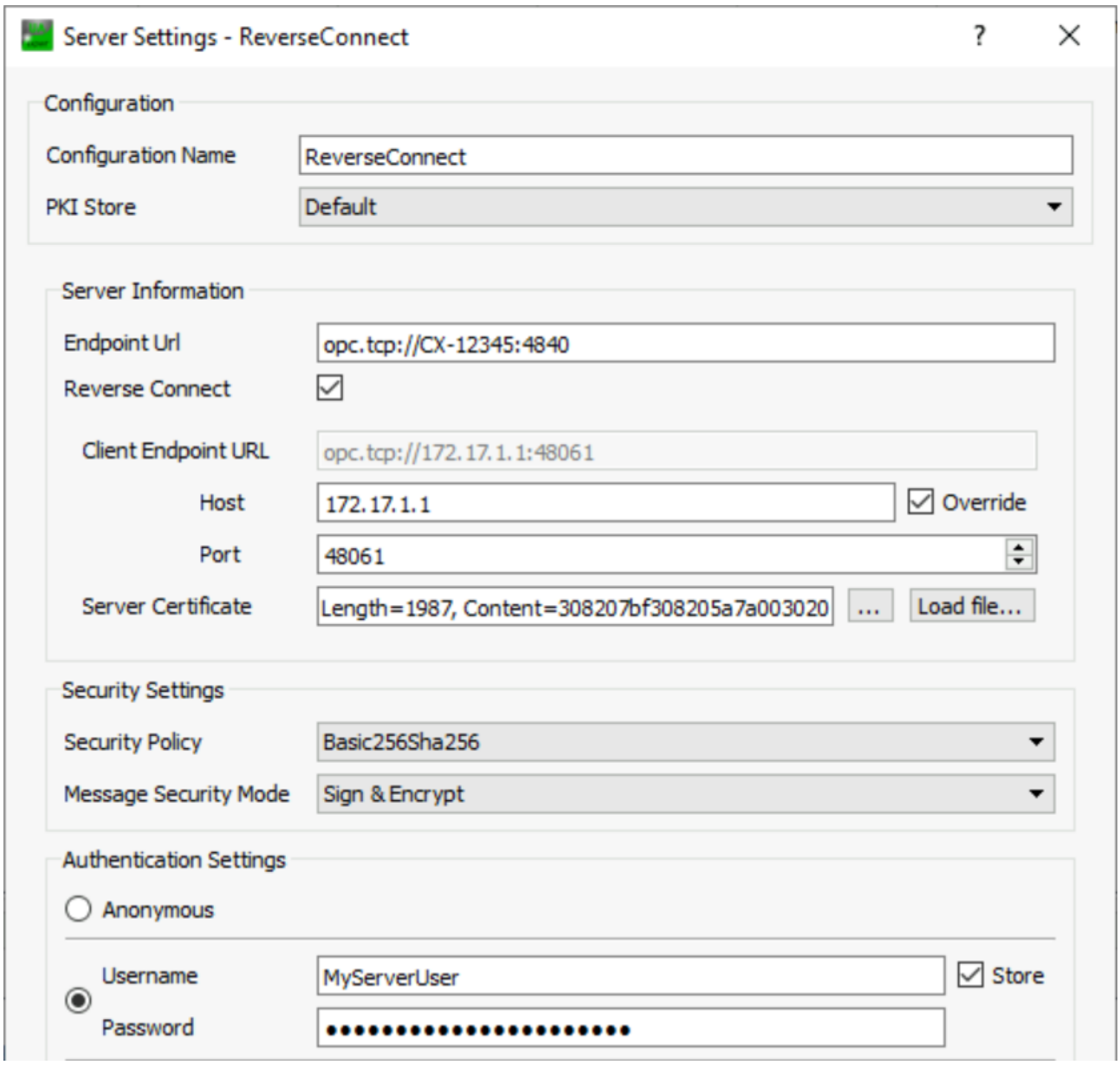
可在 <UaEndpoint> 内的 TcUaServerConfig.xml 中配置 OPC UA 客户端列表。在此处输入可以访问相应客户端的客户端端点 URL。

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

OPC UA 客户端中的配置

下方的截图显示了在 Unified Automation 公司的 OPC UA 客户端软件“UA Expert”中配置 ReverseConnect 连接的示例。在连接设置中启用 ReverseConnect 并输入客户端端点 URL，服务器可通过该 URL 访问客户端。在 EndpointURL 字段输入服务器端点 URL，客户端应在服务器建立 ReverseConnect TCP 连接后使用该 URL。如上图所示，此处输入的是 opc.tcp://CX-12345:4840 或 opc.tcp://CX-98765:4840（示例）。然后，还需要进行安全设置、消息安全模式和身份验证参数的设置，以建立该连接。

请注意，您可能仍需要导入服务器证书。它是服务器的公钥，存储在相应的应用程序目录 [▶ 46]中。



通信历史记录

下方的 Wireshark® Trace 显示了基于 ReverseConnect 进行的连接设置的示例。与上图唯一不同的是，该记录中使用的客户端配置为客户端端点 URL opc.tcp://172.30.3.86:48061。服务器位于 NAT 设备后面，而 NAT 设备的 IP 地址为 178.200.200.59。

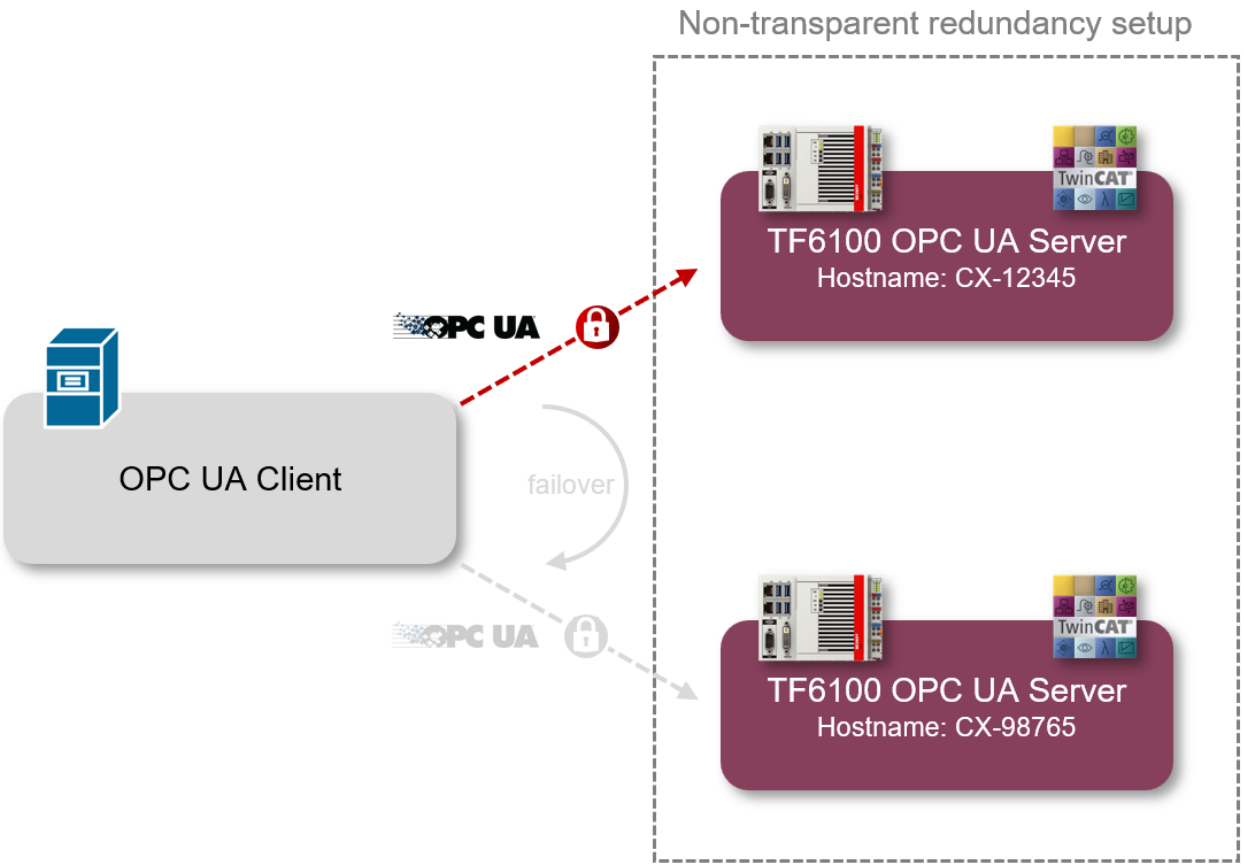
No.	Time	Source	Destination	Protocol	Length	Info
608	5.228616	178.200.200.59	172.30.3.86	OpcUa	154	Reverse Hello message
609	5.228927	172.30.3.86	178.200.200.59	OpcUa	116	Hello message
610	5.249617	178.200.200.59	172.30.3.86	OpcUa	82	Acknowledge message
612	5.344295	172.30.3.86	178.200.200.59	OpcUa	3039	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

在服务器与客户端之间建立 TCP 连接时，系统会发送所谓的“Reverse Hello（反向问候）”消息。在这种情况下，服务器会通知客户端它可以通过哪个服务器端点 URL 访问该服务器。该服务器端点 URL 与您在客户端配置的服务器端点 URL 相同（参见上文）。客户端会使用该服务器端点 URL 与服务器进行进一步连接。

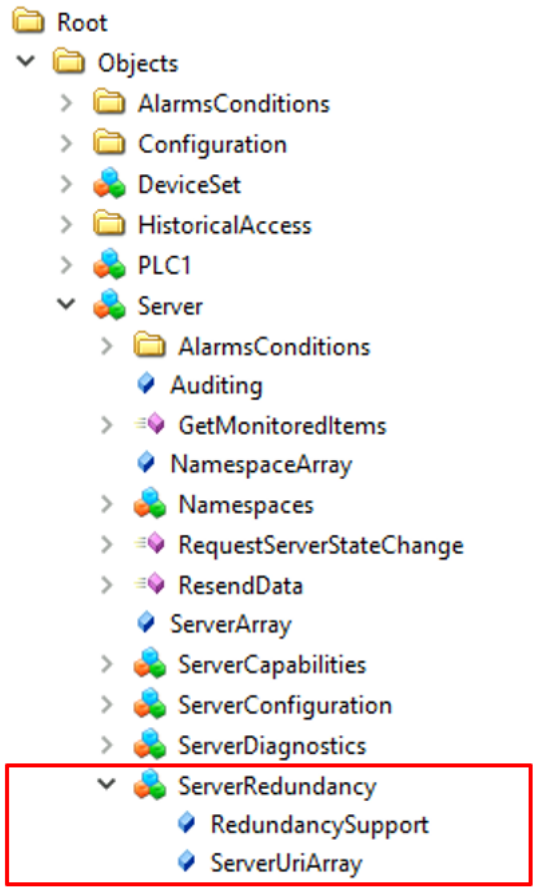
4.20 Redundancy

OPC UA 规范定义了 2 种类型的服务器冗余：透明冗余和非透明冗余。在采用透明冗余的情况下，将功能从一台服务器转移到另一台服务器的过程对客户端完全透明，即客户端不知道发生了故障转移过程，因此无法控制故障转移行为。此外，客户端无需执行任何操作即可继续发送或接收数据。在采用非透明冗余的情况下，从一个服务器到另一个服务器的故障转移过程以及为继续发送或接收数据而采取的操作均由客户端执行。客户端必须了解服务器的冗余配置，并执行必要的操作，以便在发生故障转移时能够充分利用服务器冗余功能。非透明冗余定义了各种故障转移模式，除其他功能外，这些模式还指定了有多少台服务器主动参与通信，或在发生故障转移时作为纯被动系统使用。

TwinCAT OPC UA Server 支持“冷”、“暖”和“热”运行模式下的非透明冗余。这三种运行模式的区别详见下文。



对于非透明冗余，OPC UA 提供了多种数据结构，客户端可以通过这些数据结构识别冗余服务器组中可用的服务器，以及可向客户端传达服务器支持哪些故障转移模式的服务器信息。根据这些信息，客户端可以确定需要采取哪些措施来实现故障转移。为此，ServerRedundancy 对象存在于服务器的地址空间中，并指示服务器（即冗余配置中包含的所有服务器）支持的冗余模式和冗余集。



#	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

故障转移模式

下表列出了非透明冗余支持的所有故障转移模式。

模式	描述
冷	在冷故障转移模式下，一次只能激活 1 台服务器。这可能意味着，冗余服务器不可用（未开启），或者冗余服务器可用，但未运行（PC 在运行，但应用程序未启动）。
暖	在暖故障转移模式下，备份服务器可以处于激活状态，但不得与实际数据点建立连接（通常是底层设备仅限建立单个连接的系统）。PLC 等底层设备的资源可能有限，只允许连接一台服务器。因此，只有 1 台服务器能够检索数据。
热	在热故障转移模式下，所有服务器均已开启并且随时可以运行。在服务器从 PLC 等下游设备检索数据的情况下，会有 1 台或多台服务器处于激活状态，并与下游设备并行连接。这些服务器对所在群组的其他服务器知之甚少，均独立工作。如果服务器出现故障或严重问题，其 ServiceLevel 将会下降（参见下文）。恢复后，服务器将以适当的 ServiceLevel 返回冗余服务器集，以表明其可供用户使用。

ServiceLevel

ServiceLevel 向客户端提供有关服务器状态及其数据提供能力的信息。ServiceLevel 是一种数据类型为字节的节点，取值范围为 0 至 255。TwinCAT OPC UA Server 目前将 ServiceLevel 静态设置为值 255。此外，还可以通过服务器中的 ADS 接口来影响 TwinCAT OPC UA Server 的 ServiceLevel。有关更多信息，请参见子章节“ServiceLevel [▶ 133]”。

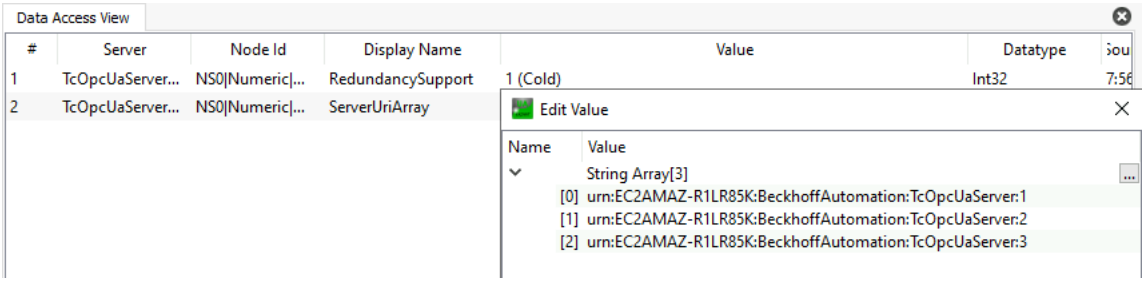
配置

可通过 TwinCAT OPC UA Configurator 来配置服务器冗余。为此需要采取 2 个步骤：

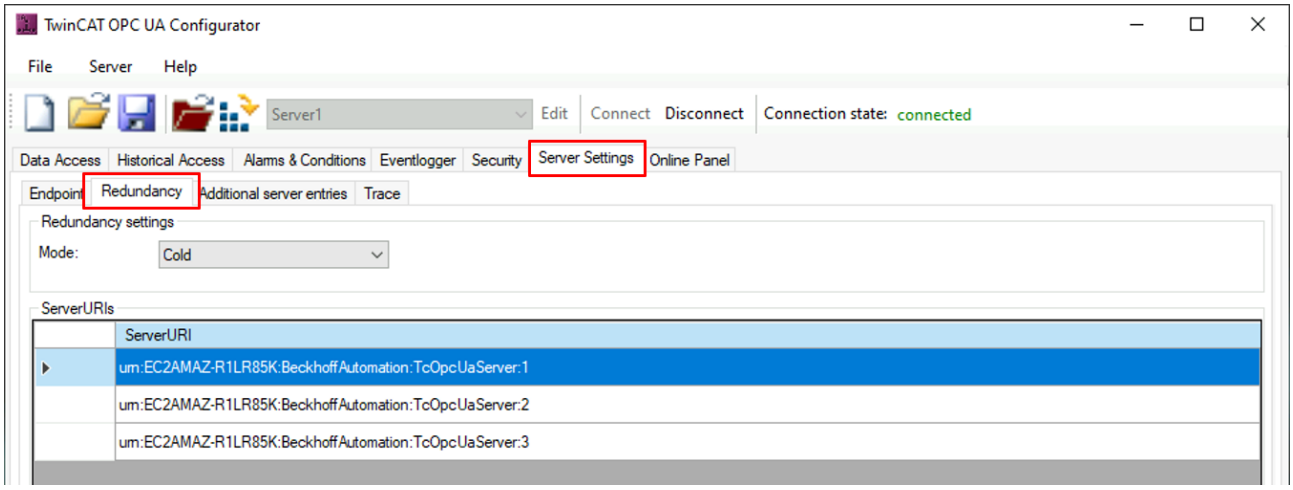
- 配置冗余
- 将冗余集中的服务器添加为 AdditionalServerEntry

配置冗余

在配置服务器冗余时，您需要定义服务器提供哪种故障转移模式，以及哪些其他服务器应包含在冗余配置中。通过其 ServerURI 来识别这些服务器。然后，服务器会将故障转移模式和 ServerURI 列表存储在其 ServerRedundancy 中（参见上文）。



可以通过 TwinCAT OPC UA Configurator 在“Server Settings（服务器设置）”选项卡中配置这些设置：



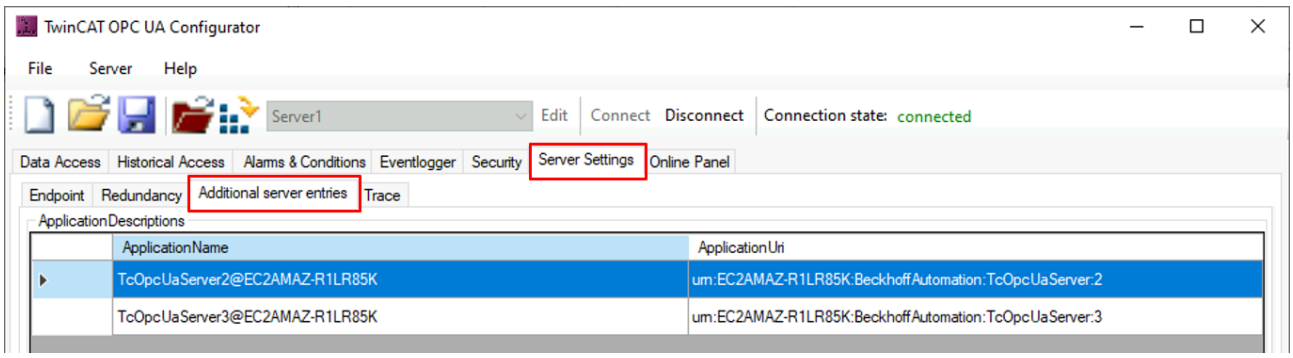
由于 ServerURI 本身尚不包含任何连接信息（如服务器或查找 URL），因此必须在其他位置向客户端提供这些信息。通过所谓的 AdditionalServerEntry 配置来完成此操作。这样，客户端即可通过 FindServer 调用功能从服务器中检索所有必要信息，并识别特定 ServerURI 的连接信息。该配置步骤的说明详见下文。

将冗余集中的服务器添加为 AdditionalServerEntry

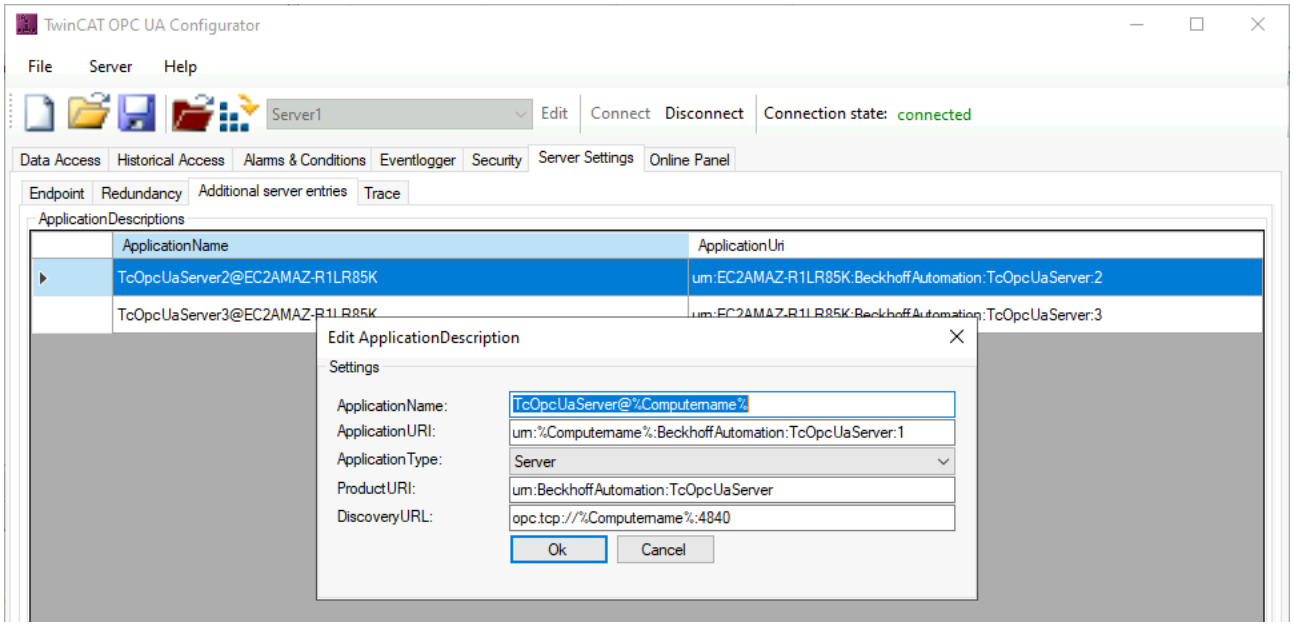
在 AdditionalServerEntry 配置下，将其他 TwinCAT OPC UA Server 提供的信息添加至服务器的 FindServer 调用中。这样，客户端即可根据冗余配置定位所有服务器，并与之连接。下方的屏幕截图显示了在本地 TwinCAT OPC UA Server 上调用 FindServer 的示例。该调用操作会返回 3 台 TwinCAT OPC UA Server，它们也是冗余配置的一部分。

客户端使用冗余配置中的 ServerURI 来识别服务器。

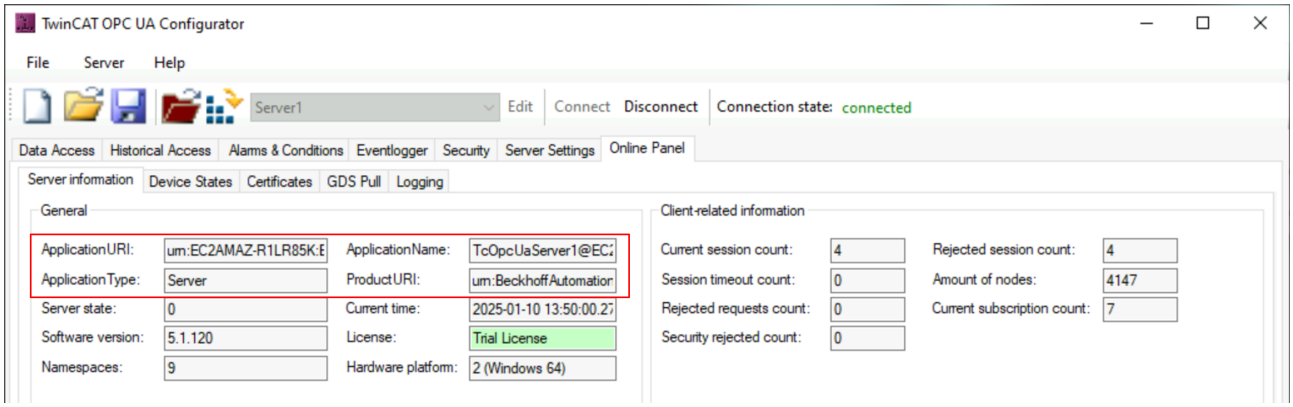
可通过 TwinCAT OPC UA Configurator 在“**Server Settings**（服务器设置）”选项卡中配置 AdditionalServerEntries，例如：



添加新条目时，某些字段已使用模板填写完毕。请根据您的配置调整这些条目。



ProductUri、ApplicationName、ApplicationUri 和 ApplicationType 的具体值请参见 TwinCAT OPC UA Configurator 的 OnlinePanel：



DiscoveryURL 与您通常用来连接相应服务器的 ServerURL 相一致。

还请参阅有关此

DeviceState [▶ 88]

4.20.1 ServiceLevel

ServiceLevel 向客户端提供有关服务器状态及其数据提供能力的信息。ServiceLevel 是一种数据类型为字节的节点，取值范围为 0 至 255。高值代表可用性或服务质量高，低值代表可用性有限。

客户端可以使用 ServiceLevel 来评估服务器的当前状态，或对冗余或分布式环境中的服务器进行比较。应将 ServiceLevel 应视为一般状态指标，不能取代对个别错误原因的详细诊断。

TwinCAT OPC UA Server 目前将 ServiceLevel 静态设置为值 255。此外，还可以通过服务器中的 ADS 接口来影响 TwinCAT OPC UA Server 的 ServiceLevel。使用该功能时，使用或不使用所谓的 Watchdog（看门狗）均可。

使用 Watchdog（看门狗）监控

在配置的时间范围内，必须重新循环写入已写入的 ServiceLevel。如果没有更新，则会自动设置所配置的备用值。为此使用了功能块“FB_OpcUaServer_SetServiceLevel [▶ 147]”。

在使用 Watchdog（看门狗）的情况下，2 个可配置值如下：

设置	含义
ServiceLevelWatchdogTimeout	如果在此时间（以 ms 为单位）内未再次向服务器提交 ServiceLevel，将会触发 Watchdog（看门狗）。
ServiceLevelWatchdogFail	这是在 Watchdog（看门狗）遇到超时问题时设置的 ServiceLevel 值。

不使用 Watchdog（看门狗）监控

写入的 ServiceLevel 由功能块“FB_OpcUaServer_SetServiceLevel [▶ 147]”进行一次性设置。任何激活的 Watchdog（看门狗）均将停用。该值保持不变，直至再次写入时才会发生更改。若要重新激活 Watchdog（看门狗），必须在使用 Watchdog（看门狗）监控的情况下再次执行该方法。

还请参阅有关此

FB_OpcUaServer_SetServiceLevel [▶ 147]

4.21 日志记录

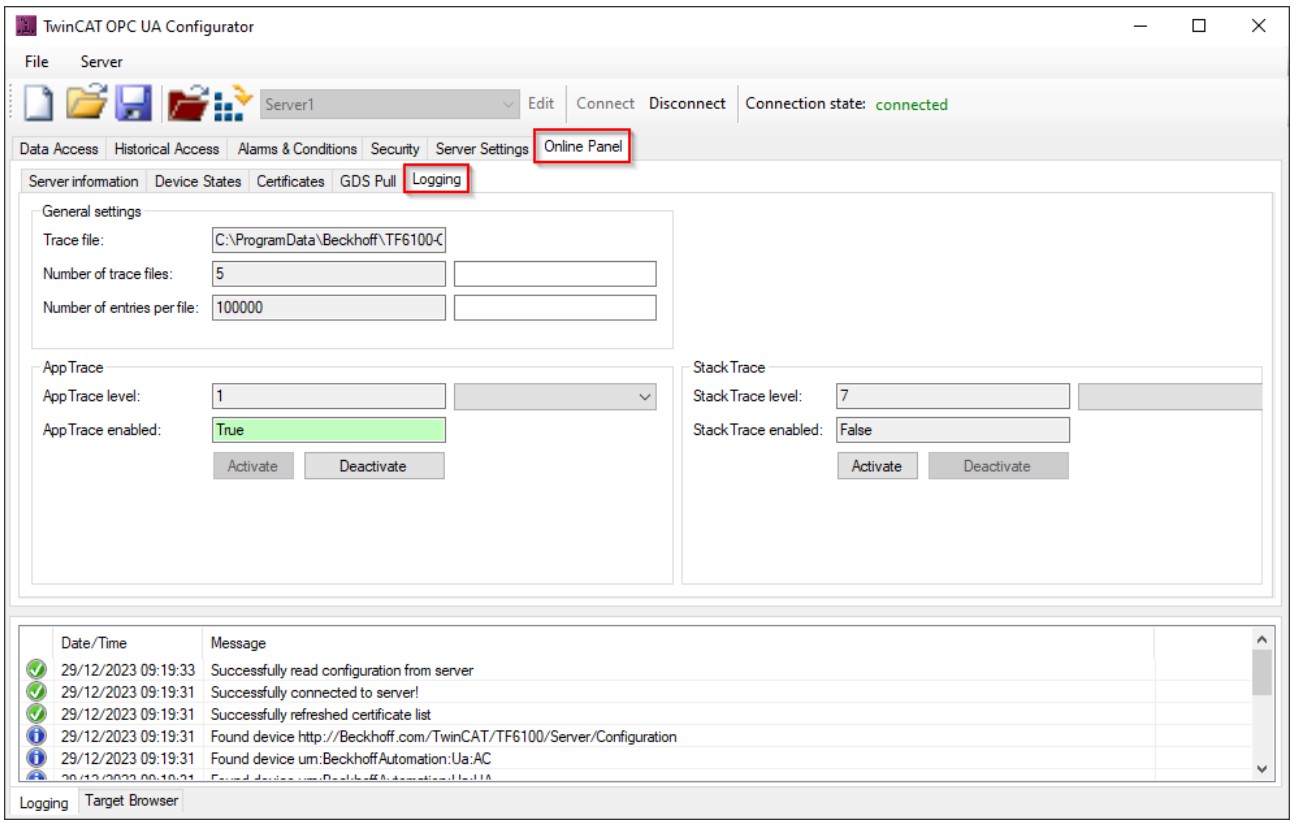
您可以激活服务器中的日志文件，以便进行扩展诊断，然后根据不同的日志等级记录各种信息。

日志记录对运行特性的影响

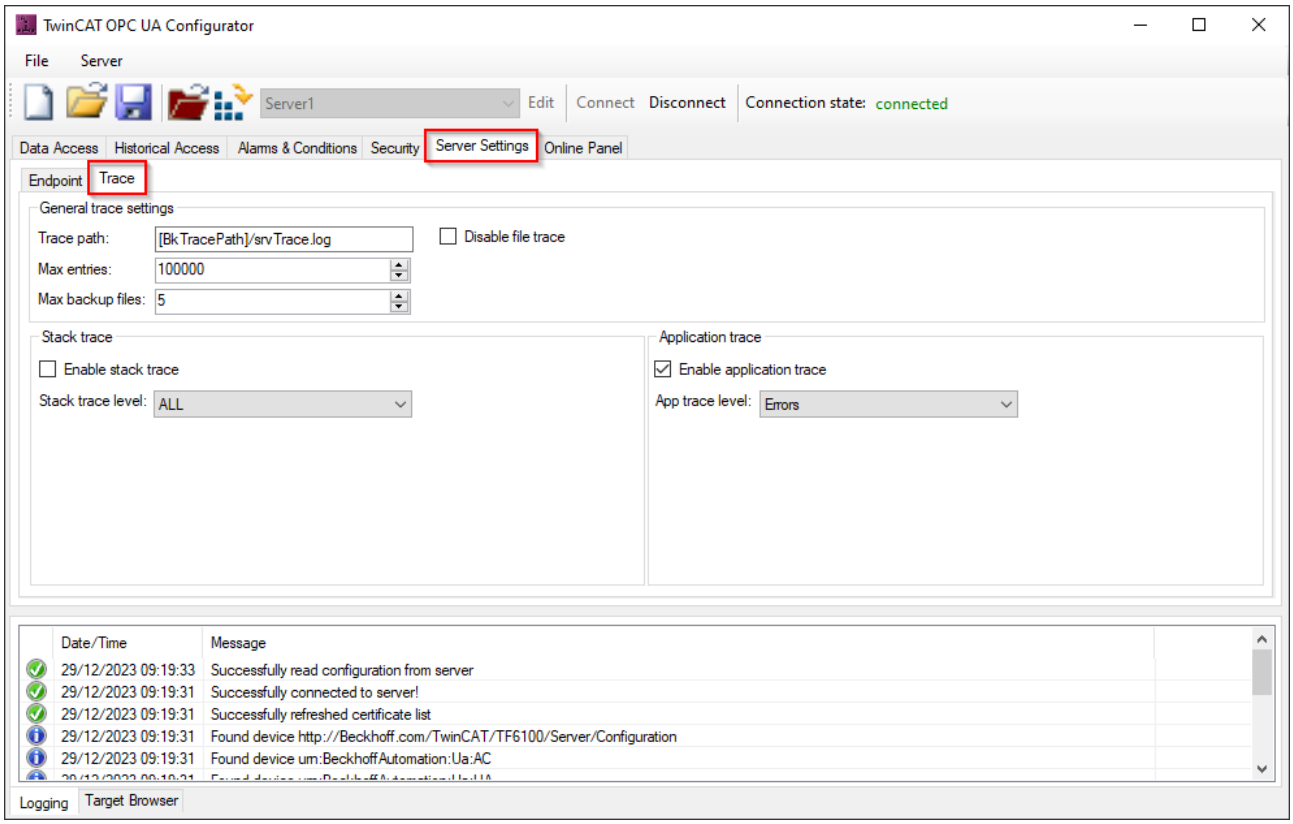
请注意，启用日志文件可能会对 TwinCAT OPC UA Server 的速度和运行特性产生负面影响。

创建日志文件的默认路径取决于所使用的操作系统，详情请参见“应用程序目录 [▶ 46]”一章。

通常通过 TwinCAT OPC UA Configurator 来启用/禁用日志记录。您可以使用相应的配置文件在服务器上在线和离线启用/禁用日志记录，这意味着，只有在服务器重启后才会启用日志记录。下方的屏幕截图显示了在线日志记录的配置界面：

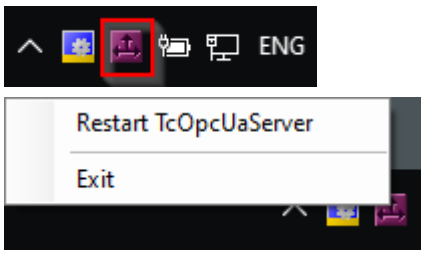


下方的屏幕截图显示了离线日志记录的配置界面：



4.22 System Tray

TwinCAT OPC UA Server 中包含一个应用程序，可以将其作为图标在 Windows System Tray 中进行调用。该应用程序可触发服务器重启。可通过上下文菜单调用相应的功能。


ENG
Restart TcOpcUaServer
Exit

5 PLC API

5.1 Tc2_OpcUa

5.1.1 数据类型

5.1.1.1 ST_OpcUAServerInfo

ST_OpcUAServerInfo 包含 TwinCAT OPC UA 服务器的会话信息。

语法

```
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
```

参数

名称	类型	描述
nReserved	UDINT	占位符。
nCummulatedSessionCount	UDINT	自服务器启动以来的客户端会话总数。
nCurrentSessionCount	UDINT	当前客户端会话总数。
nRejectedSessionCount	UDINT	被服务器拒绝的会话总数。
nSecurityRejectedSessionCount	UDINT	出于安全原因（如用户名和密码组合错误）而被服务器拒绝的会话总数。
nSessionTimeoutCount	UDINT	超时的会话总数。
nCurrentSubscriptionCount	UDINT	服务器中当前订阅的总数。
nRejectedRequestCount	UDINT	请求失败总数。
nSecurityRejectedRequestCount	UDINT	因安全原因而失败的请求总数。

5.1.1.2 E_OpcUAServerOption

E_OpcUAServerOption 决定向 TwinCAT OPC UA 服务器发送哪条命令。

语法

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

参数

名称	描述
eOPCUAServerOption_None	枚举的初始状态。

名称	描述
eOPCUAServerOption_Restart	该选项会触发服务器 OPC UA 接口的重启。
eOPCUAServerOption_Shutdown	该选项会触发服务器 OPC UA 接口的关闭。由于上述重启选项通过 OPC UA 运行，因此使用该选项后，在服务器完全重启之前，该选项将不再可用。
eOPCUAServerOption_RefreshCfg	该选项目前没有任何功能。
eOPCUAServerOption_ServerInfo	该选项可查询 ST_OpcUAServerInfo [▶ 137] 中的服务器信息。

5.1.1.3 E_OpcUAServerStatus

E_OpcUAServerStatus 表示 TwinCAT OPC UA 服务器的运行时状态。

语法

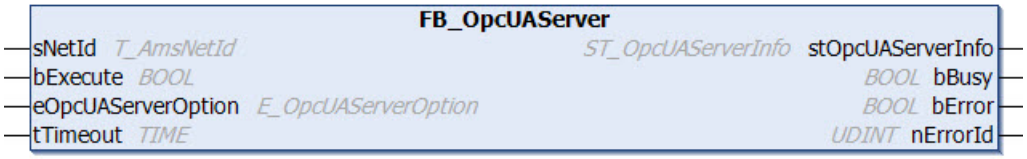
```
TYPE E_OpcUAServerStatus
(
    eOPCUAServerStatus_None,
    eOPCUAServerStatus_Alive,
    eOPCUAServerStatus_NotResponding
);
END_TYPE
```

参数

名称	描述
eOPCUAServerStatus_None	枚举的初始状态。
eOPCUAServerStatus_Alive	可以访问 TwinCAT OPC UA 服务器的 ADS 接口。
eOPCUAServerStatus_NotResponding	TwinCAT OPC UA 服务器的 ADS 接口无法访问。

5.1.2 功能块

5.1.2.1 FB_OpcUAServer



通过该功能块可以读出状态信息并重新启动 TwinCAT OPC UA 服务器。

语法

定义：

```
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
```

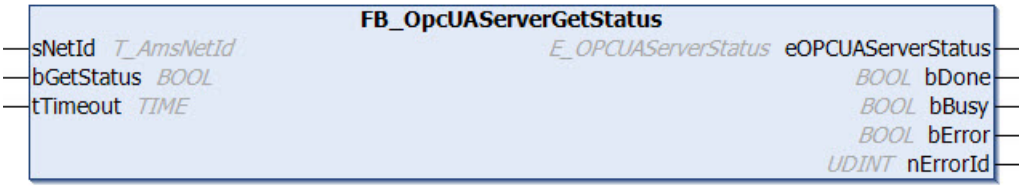
输入

名称	类型	描述
sNetId	T_AmsNetId	运行 TwinCAT OPC UA 服务器的系统的 AmsNetId。
bExecute	BOOL	上升沿会激活功能块的处理。
eOpcUAServerOption	E_OpcUAServerOption [▶ 137]	指定要执行的操作。
tTimeout	TIME	ADS 超时

输出

名称	类型	描述
stOpcUAServerInfo	ST_OpcUAServerInfo [▶ 137]	当在 eOpcUAServerOption 输入中选择 ServerInfo 时，包含来自服务器的状态信息。
bBusy	BOOL	只要功能块的处理还在进行，就为 TRUE。
bError	BOOL	一旦出现情况，则变为 TRUE。
nErrorId	UDINT	包含发生错误 (bError) 时的错误代码。

5.1.2.2 FB_OpcUAServerGetStatus



通过该功能块可以读取 TwinCAT OPC UA 服务器的当前状态（正在运行、未响应）。需要注意的是，该功能块此时处理的是 OPC UA 服务器的 ADS 接口。如果重新启动或关闭 OPC UA 服务器，则仍可访问服务器的 ADS 接口。ADS 接口只能通过终止服务器进程来关闭。

语法

定义：

```
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
```

输入

名称	类型	描述
sNetId	T_AmsNetId	运行 TwinCAT OPC UA 服务器的系统的 AmsNetId。
bGetStatus	BOOL	上升沿会激活功能块的处理。
tTimeout	TIME	ADS 超时

输出

名称	类型	描述
eOPCUAServerStatus	E_OpcUAServerStatus [▶ 138]	包含服务器的状态信息。
bDone	BOOL	功能块处理完成时为 TRUE。
bBusy	BOOL	只要功能块的处理还在进行，就为 TRUE。
bError	BOOL	一旦出现情况，则变为 TRUE。
nErrorId	UDINT	包含发生错误 (bError) 时的错误代码。

5.2 Tc3_OpcUa

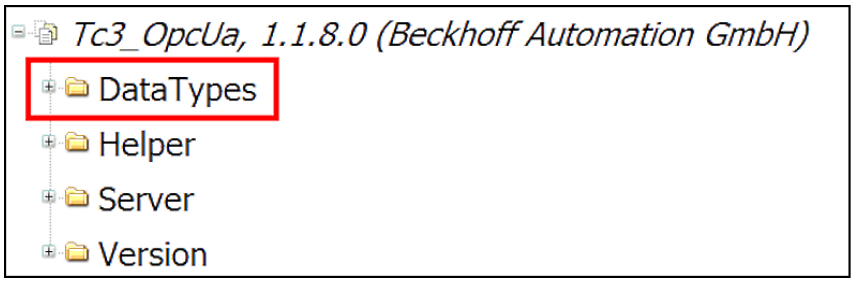
该 PLC 库包含与 TwinCAT OPC UA 配合使用的数据类型、功能块等，其顶层由以下文件夹组成：

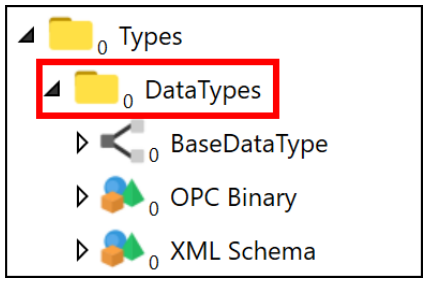
表 1: *文件夹结构*

顺序名称	内容描述
DataTypes	包含可映射 OPC UA 数据类型（UA_DataType）的 PLC 类型。
Helper	包含支持使用 OPC UA 数据类型的 PLC 类型。
服务器	包含与 TwinCAT OPC UA Server（TF6100）配合使用的 PLC 类型（以及更多）。
版本	包含 PLC 库中的版本信息。

5.2.1 数据类型

该文件夹（左图）包含可映射 OPC UA 数据类型（UA_DataType）的 PLC 类型，代表命名空间http://opcfoundation.org/UA/（右图）中的节点“DataTypes”[i=90]。





5.2.1.1 BaseDataType

该文件夹表示“BaseDataType”节点 [i=24]，包含可映射其子类型的 PLC 类型。BaseDataType 本身没有对应的 PLC 类型。

5.2.1.1.1 UA_Boolean

此 PLC 数据类型可映射节点“Boolean”[i=1]：reference.opcfoundation.org
此内置数据类型定义了 TRUE 或 FALSE 值。

语法

```
TYPE UA_Boolean : BOOL;
END_TYPE
```

5.2.1.1.2 UA_Byte

此 PLC 数据类型可映射节点“Byte”[i=3]：reference.opcfoundation.org
此内置数据类型定义了 TRUE 或 FALSE 值。

语法

```
TYPE UA_Byte : USINT;
END_TYPE
```

5.2.1.1.3 UA_ByteString

此 PLC 数据类型可映射节点“ByteString”[i=1]：reference.opcfoundation.org
此内置数据类型定义了表示字节值序列的值。

语法

```
TYPE UA_ByteString :
STRUCT
    nLength : UDINT;
    aData : ARRAY [0..255] OF BYTE;
END_STRUCT
END_TYPE
```

参数

名称	类型	描述
nLength	UDINT	aData 中使用的字节数
aData	ARRAY [0..255] OF BYTE	字节字符串的用户数据

5.2.1.1.4 UA_DateTime

此 PLC 数据类型可映射节点“DateTime”[i=13]：reference.opcfoundation.org
此内置数据类型定义了公历中的日期和时间。

语法

```
TYPE UA_DateTime : DT;
END_TYPE
```

5.2.1.1.5 UA_Double

此 PLC 数据类型可映射节点“Double”[i=11]：reference.opcfoundation.org
此内置数据类型根据 ISO/IEC/IEEE 60559:2020 定义定义了具有双精度（64 位）的浮点值。

语法

```
TYPE UA_Double : LREAL;
END_TYPE
```

5.2.1.1.6 UA_Float

此 PLC 数据类型可映射节点“Float”[i=10]：reference.opcfoundation.org
此内置数据类型根据 ISO/IEC/IEEE 60559:2020 定义定义了具有单精度（32 位）的浮点值。

语法

```
TYPE UA_Float : REAL;
END_TYPE
```

5.2.1.1.7 UA_Int16

此 PLC 数据类型可映射节点“Int16”[i=4]：reference.opcfoundation.org
此内置数据类型定义了介于 -32,768 至 32,767 之间的有符号整数。

语法

```
TYPE UA_Int16 : INT;
END_TYPE
```

5.2.1.1.8 UA_Int32

此 PLC 数据类型可映射节点“Int32”[i=6]：reference.opcfoundation.org
此内置数据类型定义了介于 -2,147,483,648 至 2,147,483,647 之间的有符号整数。

语法

```
TYPE UA_Int32 : DINT;
END_TYPE
```

5.2.1.1.9 UA_Int64

此 PLC 数据类型可映射节点“Int64”[i=8]：reference.opcfoundation.org
此内置数据类型定义了介于 -9,223,372,036,854,775,808 至 9,223,372,036,854,775,807 之间的有符号整数。

语法

```
TYPE UA_Int64 : LINT;
END_TYPE
```

5.2.1.1.10 UA_LocaleId

此 PLC 数据类型可映射节点“LocaleId”[i=295]:reference.opcfoundation.org
这种简单的数据类型是一个字符串，由符合 RFC 5646 标准的语言代码和可选国家/地区代码（如“de-DE”）组成。

语法

```
TYPE UA_LocaleId : STRING(6);
END_TYPE
```

5.2.1.1.11 UA_LocalizedText

此 PLC 数据类型可映射节点“LocalizedText”[i=21]:reference.opcfoundation.org
此内置数据类型定义了一个结构，该结构由字符串/文本和本地化翻译的本地标识符组成。

语法

```
{attribute 'pack_mode' := '1'}
TYPE UA_LocalizedText :
STRUCT
    sLocaleId : UA_LocaleId;
    {attribute 'TcEncoding' := 'UTF-8'}
    sText : STRING(256);
END_STRUCT
END_TYPE
```

参数

名称	类型	描述
sLocaleId	Tc3_OpcUa.UA_LocaleId	本地标识符
sText	STRING(256)	本地化字符串/文本

5.2.1.1.12 UA_NodeId

此 PLC 数据类型可映射节点“NodeId”[i=17]:reference.opcfoundation.org
此内置数据类型由 3 个元素（命名空间索引、ID 类型、标识符）组成，可作为服务器内某节点的唯一标识符。

语法

```
{attribute 'pack_mode' := '1'}
TYPE UA_NodeId :
STRUCT
    xml : UA_String;
END_STRUCT
END_TYPE
```

参数

名称	类型	描述
xml	Tc3_OpcUa.UA_String	以字符串作为编码的完整 NodeId，例如：ns=5;i=1234

5.2.1.1.13 UA_SByte

此 PLC 数据类型可映射节点“SByte”[i=2]：reference.opcfoundation.org
此内置数据类型定义了介于 -128 至 127 之间的有符号整数。

语法

```
TYPE UA_SByte : SINT;
END_TYPE
```

5.2.1.1.14 UA_String

此 PLC 数据类型可映射节点“String”[i=12]:reference.opcfoundation.org
此内置数据类型定义了一个 Unicode 字符串，该字符串不应包含除空格外的任何控制字符。

语法

```
TYPE UA_String : STRING;
END_TYPE
```

5.2.1.1.15 UA_UInt16

此 PLC 数据类型可映射节点“UInt16“[i=5]：reference.opcfoundation.org
此内置数据类型定义了介于 0 至 65,535 之间的无符号整数。

语法

```
TYPE UA_UInt16: UINT;
END_TYPE
```

5.2.1.1.16 UA_UInt32

此 PLC 数据类型可映射节点“UInt32“[i=7]：reference.opcfoundation.org
此内置数据类型定义了介于 0 至 4,294,967,295 之间的无符号整数。

语法

```
TYPE UA_UInt32 : UDINT
END_TYPE
```

5.2.1.1.17 UA_UInt64

此 PLC 数据类型可映射节点“UInt64“[i=9]：reference.opcfoundation.org
此内置数据类型定义了介于 0 至 18,446,744,073,709,551,615 之间的无符号整数。

语法

```
TYPE UA_UInt64 : ULINT
END_TYPE
```

5.2.1.1.18 UA_UtcTime

此 PLC 数据类型刻映射节点“UtcTime”[i=294]:reference.opcfoundation.org
这种简单的数据类型是用于表示协调世界时（UTC）的 DateTime 值，专门用于进行 OPC UA 通信。

语法

```
TYPE UA_UtcTime : UA_DateTime
END_TYPE
```

5.2.1.1.19 结构

该文件夹表示“Structure”节点 [i=22]，包含可映射其子类型的 PLC 类型。“Structure”本身没有对应的 PLC 类型。可将从“Structure”衍生的具体 OPC UA 结构类型映射为 PLC 数据类型：

将 OPC UA 结构类型映射为 PLC 数据类型：

要求

1. PLC 中的数据结构与 OPC UA 数据结构相匹配。特别是：
   - 字段顺序
   - 数据类型
2. PLC 类型定义包含作为属性使用的 OPC UA 类型信息。
   TwinCAT OPC UA Server 需要使用这些属性，以便向 OPC UA 数据类型分配正确的 PLC 数据类型。简单的 PLC 数据类型不需要具备这些属性。

可以在实例或类型声明上方手动写入所需的属性，也可以通过 TwinCAT OPC UA Nodeset Editor 检索并复制这些属性。

逐步操作 Nodeset Editor：

1 – 查找并打开结构类型。

5 – 在 TwinCAT 中创建 DUT。

2 – 选择并复制字段的所有名称和 DataType BrowseName

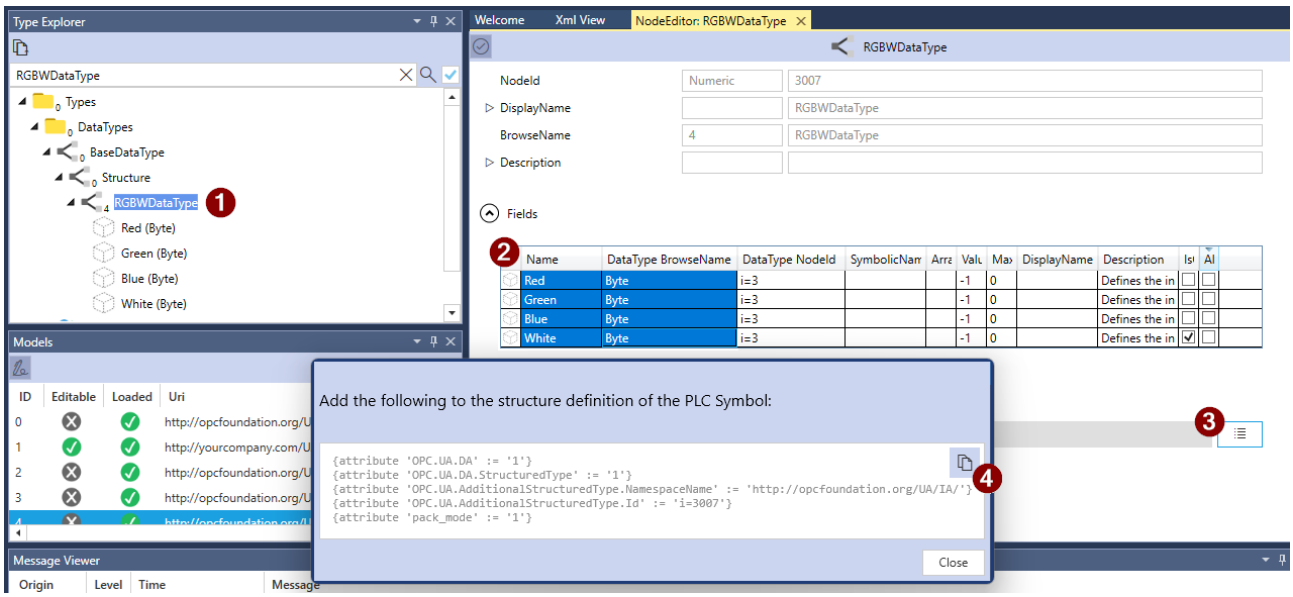
5 – 在 TwinCAT 中的 STRUCT 与 END_STRUCT 之间插入上述内容

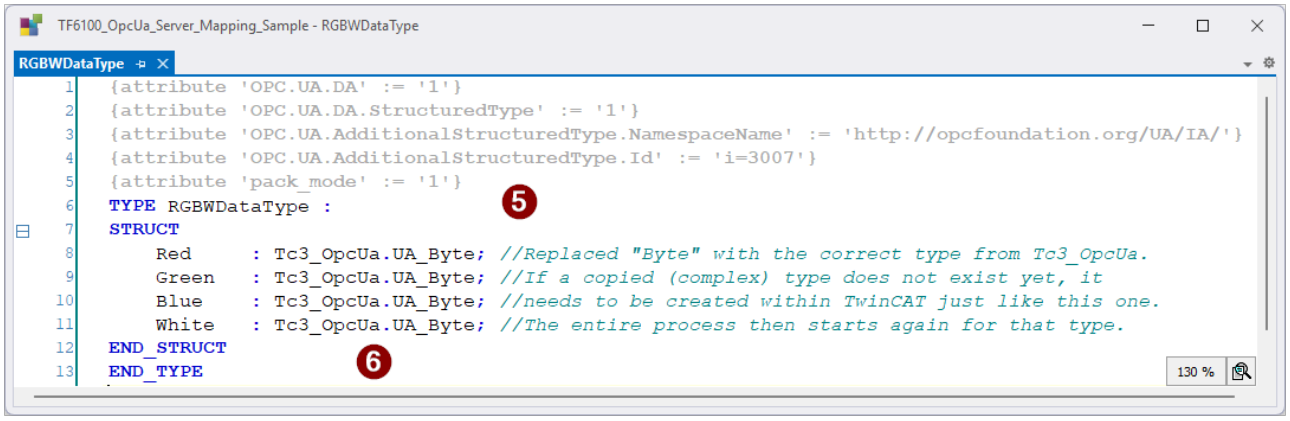
3 – 打开该类型的编译指示

4 – 复制这些编译指示

5 – 在 TwinCAT 中 TYPE 的上方插入这些编译指示

6 – 将 DataType BrowseName 替换为相应的 PLC 类型
（例如 Boolean => “Tc3_OpcUa.UA_Boolean”或“BOOL”）。
可能还需要创建缺失的（结构）类型。





属性的含义：

- **OPC.UA.DA**
  激活通过 OPC UA 提供的数据类型和任何子元素。
- **OPC.UA.DA.StructuredType**
  将类型标识为结构化 OPC UA 数据类型。
- **OPC.UA.AdditionalStructuredType.NamespaceName**
  指定节点集中该数据类型的 NamespaceUri。
- **OPC.UA.AdditionalStructuredType.Id**
  包含 OPC UA 数据类型的 NodeId，在本例中，i=3007。
- **pack_mode := '1'**
  确保已定义结构的内存布局且该内存布局具有兼容性。

5.2.1.1.19.1 UA_EUInformation

此 PLC 数据类型可映射节点“EUInformation”[i=887]：reference.opcfoundation.org
此结构化数据类型包含计量单位的相关信息（如命名空间 URI、单位 ID、显示名称和描述）。

语法

```
{attribute 'pack_mode' := '1'}
TYPE UA_EUInformation :
STRUCT
    sNamespaceURI:  UA_String;
    nUnitId:        UA_Int32;
    sDisplayName:   UA_LocalizedText;
    sDescription:   UA_LocalizedText;
END_STRUCT
END_TYPE
```

参数

名称	类型	描述
sNamespaceURI	Tc3_OpcUa.UA_String	定义该计量单位的组织的 URI
nUnitId	Tc3_OpcUa.UA_Int32	用于进行程序化评估的数字标识符
sDisplayName	Tc3_OpcUa.UA_LocalizedTExt	通常为计量单位的缩写
sDescription	Tc3_OpcUa.UA_LocalizedTExt	计量单位的全称（完整写入）

5.2.1.1.19.2 UA_OptionSet

此 PLC 数据类型可映射“OptionSet”节点 [i=12755]：reference.opcfoundation.org
这种抽象的数据类型是所有表示 64 位以上位掩码或必须标明各位y有效期的数据类型的基本类型。

语法

```
TYPE UA_OptionSet :
STRUCT
END_STRUCT
END_TYPE
```

使用 UA_OptionSet

若要定义特定的 OptionSet 并将其与 TF6100（服务器）和 TE6100（节点集编辑器）配合使用，必须由 Tc3_OpcUa.UA_OptionSet 进行衍生。

特定选项集的实施示例：

```
TYPE UA_OptionSet_Sample EXTENDS UA_OptionSet :
STRUCT
    BitOne : UA_OptionSet_Option;
    BitTwo : UA_OptionSet_Option;
END_STRUCT
END_TYPE
```

5.2.1.1.19.2.1 *UA_OptionSet_Option*

此 PLC 数据类型是定义您自己的 OptionSet 数据类型时所需使用的辅助数据类型。
该数据类型可映射 OptionSet（位掩码）中的一个选项（位），包括有效性规范。

语法

```
TYPE UA_OptionSet_Option :
STRUCT
    value : BOOL;
    valid : BOOL;
END_STRUCT
END_TYPE
```

参数

名称	类型	描述
value	BOOL	该选项的值
Valid	BOOL	该选项的有效性规范

5.2.1.1.19.3 UA_Range

此 PLC 数据类型可映射“Range”节点 [i=884]：reference.opcfoundation.org
此结构化数据类型定义了一个包含下限值（低值）和上限值（高值）的双精度值的取值范围。

语法

```
{attribute 'pack_mode' := '1'}
TYPE UA_Range :
STRUCT
    fLow : UA_Double;
```

```
    fHigh: UA_Double;
END_STRUCT
END_TYPE
```

参数

名称	类型	描述
fLow	Tc3_OpcUa.UA_Double	取值范围的下限
fHigh	Tc3_OpcUa.UA_Double	取值范围的上限

5.2.1.1.19.4 UA_TimeZoneDataType

此 PLC 数据类型可映射节点“TimeZoneDataType”[i=8912]:reference.opcfoundation.org
此结构化数据类型定义了当地时间，考虑或不考虑夏令时均可。

语法

```
{attribute 'pack_mode' := '1'}
TYPE UA_TimeZoneDataType :
STRUCT
    nOffset                 : UA_Int16;
    bDaylightSavingInOffset : UA_Boolean;
END_STRUCT
END_TYPE
```

参数

名称	类型	描述
nOffset	Tc3_OpcUa.UA_Int16	相对于 UTC 时间的时间偏移量，以分钟为单位
nDaylightSavingInOffset	Tc3_OpcUa.UA_Boolean	指示偏移量是否包括夏令时校正

5.2.2 功能块

该文件夹包含可与 TwinCAT OPC UA 配合使用的 PLC 功能块。

5.2.2.1 服务器

该文件夹包含可与 TwinCAT OPC UA Server（TF6100）配合使用的 PLC 功能块。

5.2.2.1.1 FB_OpcUaServer_SetServiceLevel

TwinCAT OPC UA Server 目前将 ServiceLevel 静态设置为值 255。该功能块可通过服务器中的 ADS 接口设置 ServiceLevel。是否与 Watchdog（看门狗）配合使用，两者之间是有区别的。

有关服务等级的更多信息，请参见 TwinCAT OPC UA Server 的 ServiceLevel 文档。

语法

```
FUNCTION_BLOCK FB_OpcUaServer_SetServiceLevel
VAR_INPUT
    tAmsNetId               : T_AmsNetId := '127.0.0.1.1.1';
    tTimeout                : TIME := T#5s;;
    bExecute                : BOOL;
    bWatchdog               : BOOL;
    nServiceLevel           : BYTE;
END_VAR
```

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

输入

名称	类型	描述
tAmsNetId	T_AmsNetId	运行 TwinCAT OPC UA Server 的系统的 AmsNetId。如果未提供信息，将使用本地主机。
tTimeout	TIME	执行 ADS 命令的最长时间。默认设置为 5 秒。
bExecute	BOOL	可选参数，用于指定 WebSocket 开启握手的 URI。
bWatchdog	BOOL	决定在设置 ServiceLevel 时是否使用 Watchdog（看门狗）。
nServiceLevel	BYTE	在 TwinCAT OPC UA Server 中设置的 ServiceLevel。

输出

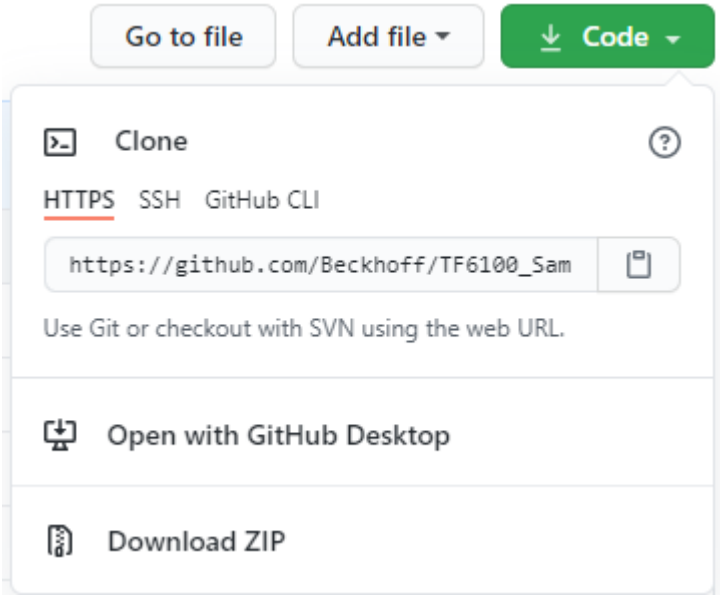
名称	类型	描述
bBusy	BOOL	只要功能块正在忙于处理，即为 TRUE。
bError	BOOL	一旦出现错误情况，即变为 TRUE。
nErrorId	UDINT	ADS 错误代码。

要求

开发环境	目标平台	要包括的 PLC 库
TwinCAT v3.1.4026.x	IPC 或 CX（x86、x64、Arm®）	Tc3_OpcUa (1.1.8 或更高版本)

6 示例

本产品的示例代码和配置可从 GitHub 上的相应存储库获取：https://github.com/Beckhoff/TF6100_Samples。您可以克隆存储库或下载包含示例的 ZIP 文件。

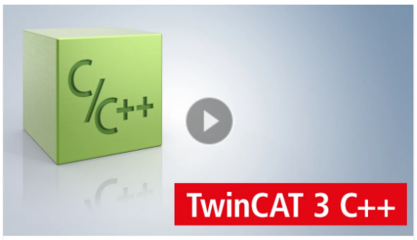


有以下示例：

名称	TwinCAT 版本	描述
TF6100_OpcUa_Client_Sample	TwinCAT 3	该示例包含 TwinCAT OPC UA 客户端（PLCOpen 功能块）各种功能的示例代码。其中包括 Browse、Connect、HistoryUpdate、MethodCall、Read 和 Write。还包括用于访问的服务器示例。
TF6100_OpcUa_Server_Sample	TwinCAT 3	该示例包含一个可为 TwinCAT OPC UA 服务器（OPC UA 数据访问）广泛提供 PLC 数据的 PLC。
TS6100_OpcUa_Client_Sample	TwinCAT 2	该示例包含 TwinCAT OPC UA 客户端（PLCOpen 功能块）各种功能的示例代码。其中包括 MethodCall、Read 和 Write。
TS6100_OpcUa_Server_Sample	TwinCAT 2	该示例演示了如何使用 PLC 注释通过 TwinCAT OPC UA 服务器释放变量。

7 教程

可在我们的网站 https://www.beckhoff.com/tutorials 上查看本产品的视频教程。视频教程提供了对产品和产品各个方面的快速入门。



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 附录

8.1 属性和注释

下表概述了可在服务器中配置的所有编译指示和注释。您可以在 TwinCAT 的各种实时环境中定义这些内容，以实现不同的功能。有关属性和注释用途的详细说明，请参见“启用符号 [▶ 52]”一章。

在 PLC 中使用标签

在 PLC 中使用编译指示时，请确保将键值和值置于引号中。您可以在“启用符号\PLC [▶ 53]”一章中查看示例。

TwinCAT 3

键值	值	含义
OPC.UA.DA	0	明确屏蔽 OPC UA 的某个符号。
OPC.UA.DA	1	启用 OPC UA 的某个符号。
OPC.UA.DA	2	启用 OPC UA 的某个符号。对于结构和 StructuredDataType，不会将成员变量作为独立节点加载到服务器的地址空间。
OPC.UA.DA.Access	1	将节点设为只读 [▶ 83]。
OPC.UA.DA.Access	2	将节点设为只写。
OPC.UA.DA.Access	3	启用对节点的读取/写入访问（若未进行设置，则采用默认值）。
OPC.UA.DA.Alias	<string>	为节点定义不同的名称（别名）。
OPC.UA.DA.Description	<string>	定义 OPC UA 属性“Description [▶ 81]”的内容。
OPC.UA.DA.StructuredType	0	禁用结构 [▶ 74]的 StructuredDataType。
OPC.UA.DA.StructuredType	1	为结构 [▶ 74]启用 StructuredDataType。
OPC.UA.DA.Status	quality	在 OPC UA 命名空间中手动定义符号的 StatusCode [▶ 77]。

TwinCAT 2

PLC 注释	含义
(*~ (OPC:0:not available) *)	锁定 OPC UA 的变量，此后该变量在 UA 命名空间中不再可见。
(*~ (OPC:1:available) *)	启用 OPC UA 的某个变量，此后该变量在 UA 命名空间中将变为可见状态。如果您想使用 UA 的某个变量，务必设置该标签。
(*~ (OPC_PROP[0005]:1:read-only) *)	为某个变量设置写入保护。必须与 (*~ (OPC: 1: available) *) 配合使用。
(*~ (OPC_UA_PROP[5100] : x: Alias name) *)	在 UA 命名空间中将 x 指定为节点名称，即所谓的别名映射。
(*~ (OPC_UA_PROP[5000]:x:Storage media) *)	启用某个变量，以便进行“历史访问”。必须与 (*~ (OPC: 1: available) *) 配合使用。x 定义了用于存储数据值的存储介质： 1 = RAM 2 = 文件 3 = SQL Compact 数据库

PLC 注释	含义
	4 = SQL Server 数据库
(*~ (OPC_UA_PROP[5000][1]:x:SamplingRate) *)	根据参数“x”确定存储变量值的采样率，单位为 [ms]。
(*~ (OPC_UA_PROP[5000][2]:x:Buffer) *)	根据参数“x”定义数据存储器中保留的值的最大数量。

8.2 32 位和 64 位进程

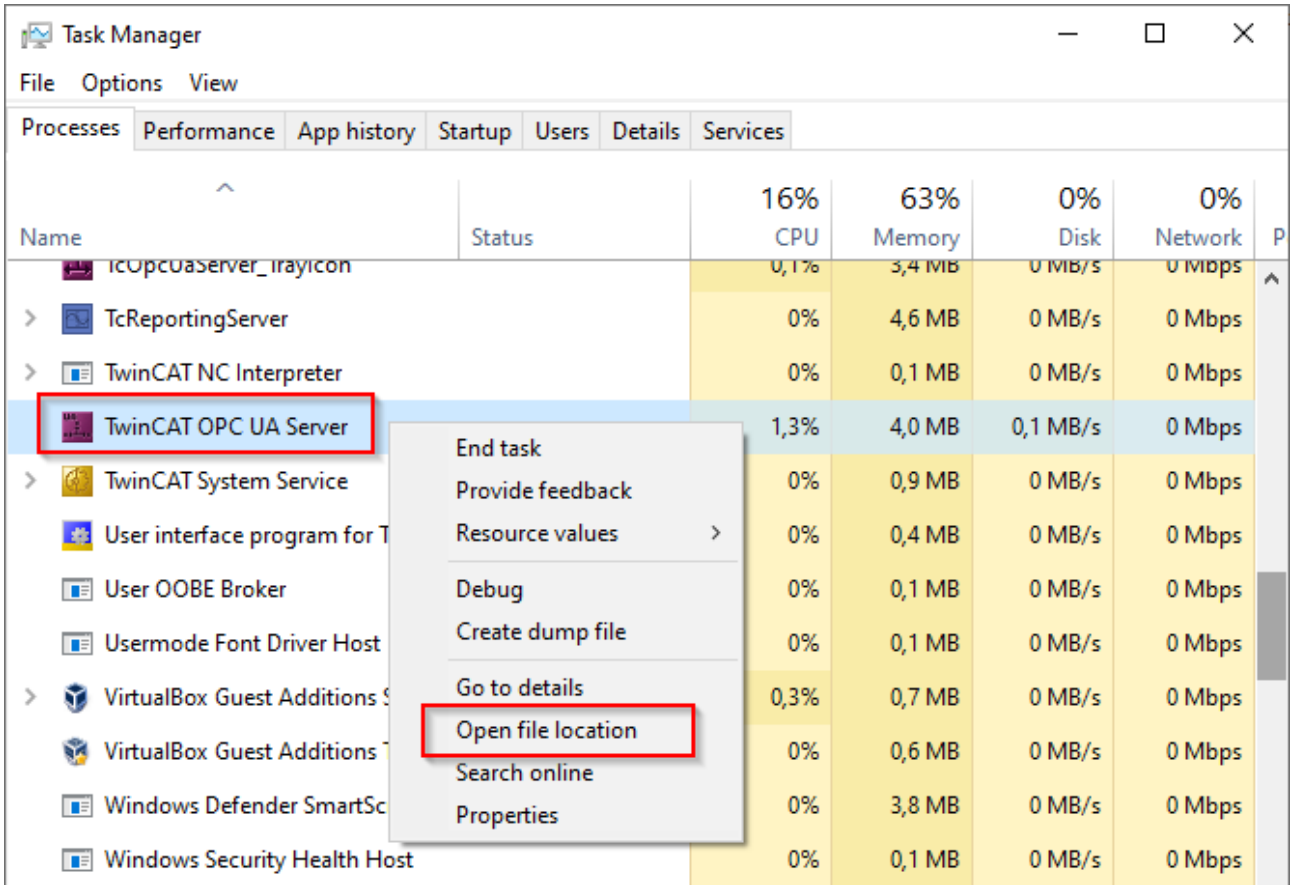
TwinCAT OPC UA Server 有 32 位和 64 位进程可供选择。相应的文件会通过设置或软件包自动进行安装。

您的系统启用的是哪一版本？

✓ 通过 Windows 任务管理器中的一个小技巧，您可以轻松识别系统中当前启用的是哪一版本。

1. 打开任务管理器。
2. 导航至“TwinCAT OPC UA Server”进程。
3. 通过上下文菜单打开文件定位。

⇨ 现在，您可以在文件资源管理器的地址栏中查看系统中当前注册和启用的 TcOpcUaServer.exe 版本：



注册/取消注册

若要注册或取消注册其他版本的 TwinCAT OPC UA Server，请按照以下步骤操作。

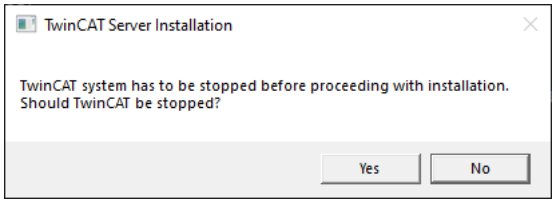
> 在切换版本之前，请务必先结束正在运行的进程。

从 32 位进程切换至 64 位进程：

- 在任务管理器中结束 TcOpcUaServer.exe 文件的运行进程。
- 以管理员身份打开 Windows 命令提示符，并切换至 32 位版本的安装目录 [▶ 46]。
- 输入以下命令以取消注册该进程：

```
TcOpcUaServer.exe /UnRegTcServer2
```

将显示一个对话框，提示您必须重启 TwinCAT 系统服务才能取消注册。点击“**Yes**（是）”确认该对话框。



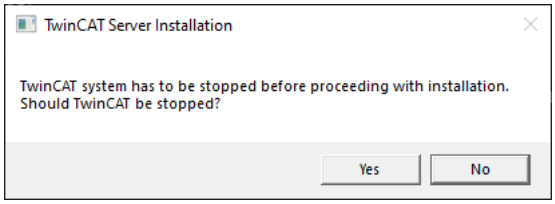
执行该命令后，TwinCAT OPC UA Server 不再随 TwinCAT 自动启动。

接下来，您将了解如何在 TwinCAT 系统服务中注册 64 位版本的服务器。

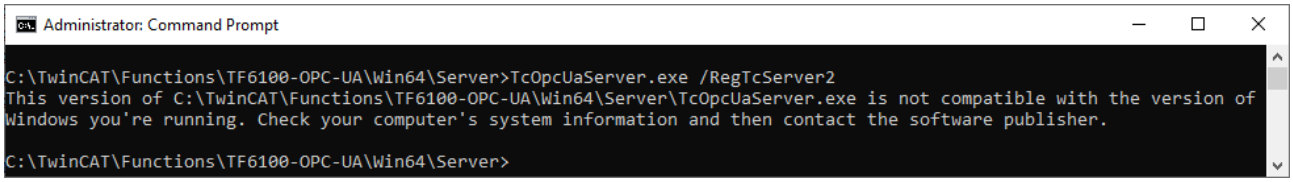
- 切换至 64 位版本服务器的安装目录 [▶ 46]，输入以下命令以注册该进程：

```
TcOpcUaServer.exe /RegTcServer2
```

将再次显示一个对话框，提示您必须重启 TwinCAT 系统服务才能取消注册。点击“Yes（是）”确认该对话框。



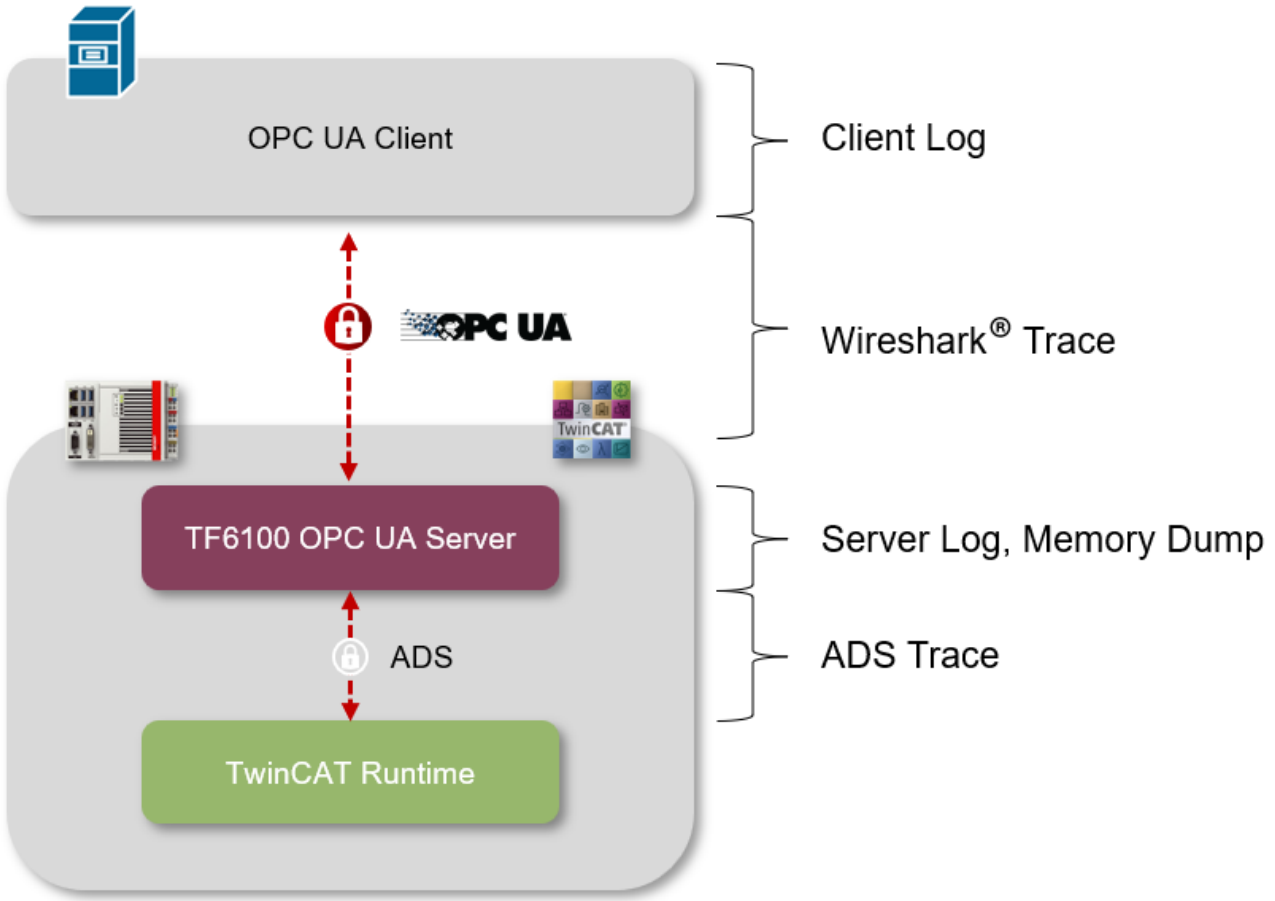
执行该命令后，TwinCAT OPC UA Server 在 TwinCAT 系统服务中即完成注册，将与 TwinCAT 一同自动启动。如果在输入此命令时收到错误消息，请确保您并非无意中尝试在 32 位系统上注册 64 位进程。



反之，32 位进程当然也可以在 64 位系统上进行注册。

8.3 错误诊断

如果出现不理想的运行特性，可能需要进行更全面的诊断。根据具体情况，诊断措施可包括：ADS 跟踪、Wireshark® 跟踪、日志文件、内存转储。客户端也可以使用日志记录功能，而且该功能非常实用。操作正确与否在很大程度上取决于运行特性的表象。仔细查看 TwinCAT OPC UA Server 的软件架构，并与运行特性进行比较。下图说明了各项措施（更详细的说明请参见附表）在哪些方面会有所帮助。



措施	用例	链接
客户端日志	用于记录客户端的扩展协议功能。	有关客户端日志记录功能的更多信息，请联系客户端制造商。
Wireshark® 跟踪	用于检查客户端与服务器之间的通信。	https://www.wireshark.org/
服务器日志	用于记录高级服务器端日志记录功能。	请参见本文档中的“日志记录 [▶ 134]”一章。
内存转储	可用于在 TwinCAT OPC UA Server 意外终止的情况下创建内存转储。所需的具体工具取决于操作系统。	Microsoft Debug Diagnostics
ADS 跟踪	用于检查服务器与 PLC 之间的通信。	Beckhoff Download Finder（TF6010 TC3 ADS Monitor）

支持

倍福支持部门 [▶ 159]很乐意协助您采取适当的措施。

下表概述了可能的意外运行特性及其解决措施。

行为	操作
OPC UA 客户端不显示 PLC 命名空间。	TF6100 安装版本 3.x 及更早版本：此状态表示缺少授权。检查是否激活了有效的 TF6100 授权。
在读取节点时，系统向 OPC UA 客户端分配了 StatusCode 0x810e0000。	TF6100 安装版本 4.x：此状态表示缺少授权。检查是否激活了有效的 TF6100 授权。
通过注释/属性启用的变量未显示在 OPC UA 服务器中。	检查是否将符号文件正确传输至控制器（例如 PLC 项目中的复选框），验证其是否存在于启动目录中，以及服务器配置文件中的符号文件路径是否指向正确的符号文件。您还可

行为	操作
	以使用相应命名空间中的 DeviceState 节点来检查可能出现的任何错误消息。如果未找到符号文件，将会在此处添加一条记录。 此外，还要检查注释/属性的拼写是否正确。
服务器不符合 OPC UA 客户端要求的采样率/发布间隔时间。	OPC UA 客户端/服务器并非实时协议，也就是说，无法保证服务器始终 100% 满足客户端要求的采样率或发布间隔时间。可在服务器配置文件中查看可用的采样率和发布间隔时间，并根据需要进行修改（<AvailableSamplingRates> 和 <MinPublishingInterval>）。
虽然服务器显示在 Windows 任务管理器中，但 OPC UA 客户端依然无法连接服务器。显示了错误消息“Host unreachable（无法访问主机）”（或类似消息）。	检查防火墙设置是否阻止了与服务器的通信。服务器端口必须对传入的 TCP 通信开放，这样客户端才能与之连接。
OPC UA 客户端显示服务器的端点，但与这些端点的连接依然失败，并显示错误消息“Host unreachable（无法访问主机）”。	检查网络中的名称解析是否正常运行，是否可以通过其主机名访问服务器。即使 OPC UA 客户端表面上与服务器的 IP 地址（如 opc.tcp://192.168.0.1:4840）连接来访问服务器的端点，服务器也始终在其端点中返回自己的主机名。如果客户端直接与其中一个端点连接，将会再次使用服务器的主机名。如果名称解析失败，连接也会失败。
OPC UA 客户端显示服务器的端点，但与安全端点的连接依然失败。显示了错误消息“BadSecurityChecksFailed”。	检查服务器是否信任客户端证书。所需的配置步骤请参见“”部分。在这种情况下，还必须确保使用 SHA2 组（SHA256、SHA364、SHA512）的签名哈希算法对客户端证书进行签名。如果使用的是 SHA1 等已弃用的算法，TwinCAT OPC UA Server 将不允许进行连接。
在使用 SQL 服务器存储历史访问信息时，这些值并未添加到 SQL 数据库中。	检查 SQL 服务器的访问数据，并验证在网络中是否可以访问 SQL 服务器。此外，还请确保您在 TwinCAT OPC UA Server 上使用的是“Big Windows”操作系统，因为在 Windows CE 下无法使用 SQL 服务器进行历史访问（但 SQL Compact 可以）。
OPC UA 客户端在读取节点时收到错误消息“BadDeviceFailure”。	这表示由于未启动 PLC 程序等原因，无法访问相关的 ADS 设备。检查是否已连接 ADS 设备，并确保相应的运行时处于激活状态。
OPC UA 客户端在读取节点时收到错误消息“BadLicenseExpired”。	这表示系统中未激活 TF6100 授权或授权已过期。请确保您已在安装服务器的设备上激活了 TF6100 授权。
数组未以全分辨率显示在命名空间中。	默认情况下，简单数据类型的数组在命名空间中并不以展开形式显示。但是，仍可使用 OPC UA 的 IndexRange 功能来处理各个数组索引。因此，OPC UA 客户端应支持该功能。如果情况并非如此，可以使用服务器配置文件中的单选按钮以展开形式显示数组，以便可将每个单独的数组元素作为单独的节点来处理。有关该单选按钮的相关说明，详见“数组 [▶ 58]“一章。
错误日志中反复出现以下消息：“CBkUaPlcSamplingEngine_Poll : !! CRITICAL!! Sampling exceeded n times in a row, leaving no time left to rest.（连续采样超过 n 次，没有时间进行其余步骤）”。	该消息表示服务器无法以请求的采样率处理客户端添加到订阅中的变量。原因通常是数据量过大，例如在对大型数据结构（StructuredType [▶ 74]）进行采样时。 请注意，该消息并不一定代表出现问题。在许多 OPC UA 客户端应用程序中，采样率设置是通用的，适用于所有变量，即使它们的采样速度可以更慢。 在任何情况下，进行优化的第一步应始终包括检查并根据需要降低采样率。如果使用的是大型结构化类型，还应检查是否可以减少数据量。 升级为速度更快的 CPU 也会有所帮助。但是，经验表明，降低采样率和数据量足以取得初步成功。

行为	操作
Job 方法 [▶ 103]调用失败，状态代码为“BadResourceUnavailable”	如果 Job 方法返回此状态代码，则表明代表该 Job 方法的功能块的签名设置不正确。首先检查 Start() 方法是否存在。 另请注意：在这种情况下，您会在 ADS 跟踪过程中看到该方法的相关调用（AdsReadWriteRequest）返回的结果为 0x710（“未找到符号”）。
在 UMRT 系统上对 TOFU 进行初始化时，显示了“Access denied (not root)（拒绝访问（非 root））”的错误消息。	在授权有限的 UMRT 系统上启动 TwinCAT OPC UA Server 时，可能会导致 TOFU 用户无法进入用户数据库。在这种情况下，请按照我们“初始化 [▶ 34]”一文中的说明，自行将 TOFU 用户添加到用户数据库中。

8.4 ADS 返回代码

错误代码分组：
全局错误代码：0x0000 [▶ 156]... （0x9811_0000 ...）
路由器错误代码：0x0500 [▶ 157]... （0x9811_0500 ...）
一般 ADS 错误：0x0700 [▶ 157]... （0x9811_0700 ...）
RTime 错误代码：0x1000 [▶ 158]... （0x9811_1000 ...）

全局错误代码

Hex	Dec	HRESULT	名称	描述
0x0	0	0x98110000	ERR_NOERROR	无错误。
0x1	1	0x98110001	ERR_INTERNAL	内部错误。
0x2	2	0x98110002	ERR_NORTIME	不具有实时性。
0x3	3	0x98110003	ERR_ALLOCLOCKEDMEM	分配已锁定 – 内存错误。
0x4	4	0x98110004	ERR_INSERTMAILBOX	邮箱已满 – 无法发送 ADS 消息。减少每个周期的 ADS 消息数量将有所帮助。
0x5	5	0x98110005	ERR_WRONGRECEIVEHMSG	HMSG 错误。
0x6	6	0x98110006	ERR_TARGETPORTNOTFOUND	未找到目标端口 – ADS 服务器未启动或无法访问。
0x7	7	0x98110007	ERR_TARGETMACHINENOTFOUND	未找到目标计算机 – 未找到 AMS 路由。
0x8	8	0x98110008	ERR_UNKNOWNCMDID	未知命令 ID。
0x9	9	0x98110009	ERR_BADTASKID	任务 ID 无效。
0xA	10	0x9811000A	ERR_NOIO	No IO。
0xB	11	0x9811000B	ERR_UNKNOWNAMSCMD	未知 AMS 命令。
0xC	12	0x9811000C	ERR_WIN32ERROR	Win32 错误。
0xD	13	0x9811000D	ERR_PORTNOTCONNECTED	端口未连接。
0xE	14	0x9811000E	ERR_INVALIDAMSLENGTH	AMS 长度无效。
0xF	15	0x9811000F	ERR_INVALIDAMSNETID	AMS Net ID 无效。
0x10	16	0x98110010	ERR_LOWINSTLEVEL	安装级别 – TwinCAT 2 授权错误。
0x11	17	0x98110011	ERR_NODEBUGINTAVAILABLE	无调试可用。
0x12	18	0x98110012	ERR_PORTDISABLED	端口已禁用 – TwinCAT 系统服务未启动。
0x13	19	0x98110013	ERR_PORTALREADYCONNECTED	端口已连接。
0x14	20	0x98110014	ERR_AMSSYNC_W32ERROR	AMS Sync Win32 错误。
0x15	21	0x98110015	ERR_AMSSYNC_TIMEOUT	AMS Sync 超时。
0x16	22	0x98110016	ERR_AMSSYNC_AMSERROR	AMS Sync 错误。
0x17	23	0x98110017	ERR_AMSSYNC_NOINDEXINMAP	不存在适用于 AMS Sync 的索引映射。
0x18	24	0x98110018	ERR_INVALIDAMSPORT	AMS 端口无效。
0x19	25	0x98110019	ERR_NOMEMORY	无内存。
0x1A	26	0x9811001A	ERR_TCPSEND	TCP 发送错误。
0x1B	27	0x9811001B	ERR_HOSTUNREACHABLE	主机无法访问。
0x1C	28	0x9811001C	ERR_INVALIDAMSFRAGMENT	AMS 片段无效。
0x1D	29	0x9811001D	ERR_TLSSEND	TLS 发送错误 – ADS 安全连接失败。
0x1E	30	0x9811001E	ERR_ACCESSDENIED	拒绝访问 – 拒绝 ADS 安全访问。

路由器错误代码

Hex	Dec	HRESULT	名称	描述
0x500	1280	0x98110500	ROUTERERR_NOLOCKEDMEMORY	无法分配锁定的内存。
0x501	1281	0x98110501	ROUTERERR_RESIZEMEMORY	路由器内存大小无法更改。
0x502	1282	0x98110502	ROUTERERR_MAILBOXFULL	邮箱已达到最大消息数。
0x503	1283	0x98110503	ROUTERERR_DEBUGBOXFULL	调试邮箱已达到最大消息数。
0x504	1284	0x98110504	ROUTERERR_UNKNOWNPORTTYPE	端口类型未知。
0x505	1285	0x98110505	ROUTERERR_NOTINITIALIZED	路由器未初始化。
0x506	1286	0x98110506	ROUTERERR_PORTALREADYINUSE	端口号已分配。
0x507	1287	0x98110507	ROUTERERR_NOTREGISTERED	端口未注册。
0x508	1288	0x98110508	ROUTERERR_NOMOREQUEUES	已达到最大端口数。
0x509	1289	0x98110509	ROUTERERR_INVALIDPORT	端口无效。
0x50A	1290	0x9811050A	ROUTERERR_NOTACTIVATED	路由未激活。
0x50B	1291	0x9811050B	ROUTERERR_FRAGMENTBOXFULL	邮箱已达到最大片段消息数。
0x50C	1292	0x9811050C	ROUTERERR_FRAGMENTTIMEOUT	发生片段超时。
0x50D	1293	0x9811050D	ROUTERERR_TOBEREMOVED	端口已移除。

一般性 ADS 错误代码

Hex	Dec	HRESULT	名称	描述
0x700	1792	0x98110700	ADSERR_DEVICE_ERROR	一般性设备错误。
0x701	1793	0x98110701	ADSERR_DEVICE_SRVNOTSUPP	服务器不支持该服务。
0x702	1794	0x98110702	ADSERR_DEVICE_INVALIDGRP	索引组无效。
0x703	1795	0x98110703	ADSERR_DEVICE_INVALIDOFFSET	索引偏移量无效。
0x704	1796	0x98110704	ADSERR_DEVICE_INVALIDACCESS	不允许读取或写入。 可能有几种原因。例如，创建路由时输入了错误的密码。
0x705	1797	0x98110705	ADSERR_DEVICE_INVALIDSIZE	参数大小不正确。
0x706	1798	0x98110706	ADSERR_DEVICE_INVALIDDATA	无效数据值。
0x707	1799	0x98110707	ADSERR_DEVICE_NOTREADY	设备尚未准备好运行。
0x708	1800	0x98110708	ADSERR_DEVICE_BUSY	设备正忙。
0x709	1801	0x98110709	ADSERR_DEVICE_INVALIDCONTEXT	操作系统上下文无效。这可能是由于在不同的任务中使用 ADS 函数块造成的。可以通过在 PLC 中进行多任务同步解决这个问题。
0x70A	1802	0x9811070A	ADSERR_DEVICE_NOMEMORY	内存不足。
0x70B	1803	0x9811070B	ADSERR_DEVICE_INVALIDPARM	参数值无效。
0x70C	1804	0x9811070C	ADSERR_DEVICE_NOTFOUND	未找到（文件…）。
0x70D	1805	0x9811070D	ADSERR_DEVICE_SYNTAX	文件或命令中存在语法错误。
0x70E	1806	0x9811070E	ADSERR_DEVICE_INCOMPATIBLE	对象不匹配。
0x70F	1807	0x9811070F	ADSERR_DEVICE_EXISTS	对象已存在。
0x710	1808	0x98110710	ADSERR_DEVICE_SYMBOLNOTFOUND	符号未找到。
0x711	1809	0x98110711	ADSERR_DEVICE_SYMBOLVERSIONINVALID	符号版本无效。这可能是由于联机更改造成的。创建新句柄。
0x712	1810	0x98110712	ADSERR_DEVICE_INVALIDSTATE	设备（服务器）处于无效状态。
0x713	1811	0x98110713	ADSERR_DEVICE_TRANSMODENOTSUPP	不支持 AdsTransMode。
0x714	1812	0x98110714	ADSERR_DEVICE_NOTIFYHNDINVALID	通知句柄无效。
0x715	1813	0x98110715	ADSERR_DEVICE_CLIENTUNKNOWN	通知客户端未注册。
0x716	1814	0x98110716	ADSERR_DEVICE_NOMOREHDLS	没有更多的句柄可用。
0x717	1815	0x98110717	ADSERR_DEVICE_INVALIDWATCHSIZE	通知大小过大。
0x718	1816	0x98110718	ADSERR_DEVICE_NOTINIT	设备未初始化。
0x719	1817	0x98110719	ADSERR_DEVICE_TIMEOUT	设备超时。
0x71A	1818	0x9811071A	ADSERR_DEVICE_NOINTERFACE	接口查询失败。
0x71B	1819	0x9811071B	ADSERR_DEVICE_INVALIDINTERFACE	请求的接口错误。

Hex	Dec	HRESULT	名称	描述
0x71C	1820	0x9811071C	ADSERR_DEVICE_INVALIDCLSID	Class ID 无效。
0x71D	1821	0x9811071D	ADSERR_DEVICE_INVALIDOBJID	Object ID 无效。
0x71E	1822	0x9811071E	ADSERR_DEVICE_PENDING	请求待定。
0x71F	1823	0x9811071F	ADSERR_DEVICE_ABORTED	请求已中止。
0x720	1824	0x98110720	ADSERR_DEVICE_WARNING	警告信号。
0x721	1825	0x98110721	ADSERR_DEVICE_INVALIDARRAYIDX	数组索引无效。
0x722	1826	0x98110722	ADSERR_DEVICE_SYMBOLNOTACTIVE	符号未激活。
0x723	1827	0x98110723	ADSERR_DEVICE_ACCESSDENIED	拒绝访问。 有几种可能的原因。例如，单向 ADS 路由用于相反方向。
0x724	1828	0x98110724	ADSERR_DEVICE_LICENSENOTFOUND	缺少授权。
0x725	1829	0x98110725	ADSERR_DEVICE_LICENSEEXPIRED	授权已过期。
0x726	1830	0x98110726	ADSERR_DEVICE_LICENSEEXCEEDED	超出授权。
0x727	1831	0x98110727	ADSERR_DEVICE_LICENSEINVALID	授权无效。
0x728	1832	0x98110728	ADSERR_DEVICE_LICENSESYSTEMID	授权问题：系统 ID 无效。
0x729	1833	0x98110729	ADSERR_DEVICE_LICENSENOTIMELIMIT	授权不受时间限制。
0x72A	1834	0x9811072A	ADSERR_DEVICE_LICENSEFUTUREISSUE	授权问题：未来的时间。
0x72B	1835	0x9811072B	ADSERR_DEVICE_LICENSETIMETOLONG	授权期限太长。
0x72C	1836	0x9811072C	ADSERR_DEVICE_EXCEPTION	系统启动异常。
0x72D	1837	0x9811072D	ADSERR_DEVICE_LICENSEDUPLICATED	授权文件读取了两次。
0x72E	1838	0x9811072E	ADSERR_DEVICE_SIGNATUREINVALID	签名无效。
0x72F	1839	0x9811072F	ADSERR_DEVICE_CERTIFICATEINVALID	证书无效。
0x730	1840	0x98110730	ADSERR_DEVICE_LICENSEOEMNOTFOUND	OEM未知公钥。
0x731	1841	0x98110731	ADSERR_DEVICE_LICENSERESTRICTED	此系统 ID 授权无效。
0x732	1842	0x98110732	ADSERR_DEVICE_LICENSEDEMODENIED	演示授权已禁止。
0x733	1843	0x98110733	ADSERR_DEVICE_INVALIDFNCID	无效函数 ID。
0x734	1844	0x98110734	ADSERR_DEVICE_OUTOFRANGE	超出有效范围。
0x735	1845	0x98110735	ADSERR_DEVICE_INVALIDALIGNMENT	无效对齐。
0x736	1846	0x98110736	ADSERR_DEVICE_LICENSEPLATFORM	无效平台级别。
0x737	1847	0x98110737	ADSERR_DEVICE_FORWARD_PL	上下文 – 转到被动级别。
0x738	1848	0x98110738	ADSERR_DEVICE_FORWARD_DL	上下文 – 转到调度级别。
0x739	1849	0x98110739	ADSERR_DEVICE_FORWARD_RT	上下文 – 转到实时。
0x740	1856	0x98110740	ADSERR_CLIENT_ERROR	客户端错误。
0x741	1857	0x98110741	ADSERR_CLIENT_INVALIDPARM	服务包含无效参数。
0x742	1858	0x98110742	ADSERR_CLIENT_LISTEMPTY	轮询列表为空。
0x743	1859	0x98110743	ADSERR_CLIENT_VARUSED	Var 连接已在使用中。
0x744	1860	0x98110744	ADSERR_CLIENT_DUPLINVOKEID	调用的 ID 已在使用中。
0x745	1861	0x98110745	ADSERR_CLIENT_SYNCTIMEOUT	已发生超时 – 远程终端在指定的 ADS 超时中未响应。远程终端的路由设置可能配置不正确。
0x746	1862	0x98110746	ADSERR_CLIENT_W32ERROR	Win32 子系统中发生错误。
0x747	1863	0x98110747	ADSERR_CLIENT_TIMEOUTINVALID	客户端超时值无效。
0x748	1864	0x98110748	ADSERR_CLIENT_PORTNOTOPEN	端口未打开。
0x749	1865	0x98110749	ADSERR_CLIENT_NOAMSADDR	无 AMS 地址。
0x750	1872	0x98110750	ADSERR_CLIENT_SYNCINTERNAL	Ads sync 中发生内部错误。
0x751	1873	0x98110751	ADSERR_CLIENT_ADDHASH	哈希表溢出。
0x752	1874	0x98110752	ADSERR_CLIENT_REMOVEHASH	未在表格中找到密钥。
0x753	1875	0x98110753	ADSERR_CLIENT_NOMORESYM	缓存中没有符号。
0x754	1876	0x98110754	ADSERR_CLIENT_SYNCRESINVALID	收到无效响应。
0x755	1877	0x98110755	ADSERR_CLIENT_SYNCPORTLOCKED	同步端口已锁定。
0x756	1878	0x98110756	ADSERR_CLIENT_REQUESTCANCELLED	请求被取消。

RTime 错误代码

Hex	Dec	HRESULT	名称	描述
0x1000	4096	0x98111000	RTERR_INTERNAL	实时系统中发生内部错误。
0x1001	4097	0x98111001	RTERR_BADTIMERPERIODS	计时器值无效。

Hex	Dec	HRESULT	名称	描述
0x1002	4098	0x98111002	RTERR_INVALIDTASKPTR	任务指针提供了无效值 0（零）。
0x1003	4099	0x98111003	RTERR_INVALIDSTACKPTR	堆栈指针提供了无效值 0（零）。
0x1004	4100	0x98111004	RTERR_PRIOEXISTS	请求任务优先级已分配。
0x1005	4101	0x98111005	RTERR_NOMORETCB	没有可用的空闲 TCB（任务控制块）。TCB 的最大数量为 64。
0x1006	4102	0x98111006	RTERR_NOMORESEMAS	没有可用的空闲信号。信号的最大数量为 64。
0x1007	4103	0x98111007	RTERR_NOMOREQUEUES	队列中没有可用的空闲空间。队列中的最大位置数为 64。
0x100D	4109	0x9811100D	RTERR_EXTIRQALREADYDEF	已应用外部同步中断。
0x100E	4110	0x9811100E	RTERR_EXTIRQNOTDEF	未应用外部同步中断。
0x100F	4111	0x9811100F	RTERR_EXTIRQINSTALLFAILED	外部同步中断应用失败。
0x1010	4112	0x98111010	RTERR_IRQLNOTLESSOREQUAL	在错误的上下文中调用服务函数
0x1017	4119	0x98111017	RTERR_VMXNOTSUPPORTED	不支持 Intel® VT-x 扩展。
0x1018	4120	0x98111018	RTERR_VMXDISABLED	BIOS 中未启用 Intel® VT-x 扩展。
0x1019	4121	0x98111019	RTERR_VMXCONTROLSMISSING	Intel® VT-x 扩展中缺少函数。
0x101A	4122	0x9811101A	RTERR_VMXENABLEFAILS	Intel® VT-x 激活失败。

具体的正向 HRESULT 返回代码：

HRESULT	名称	描述
0x0000_0000	S_OK	无错误。
0x0000_0001	S_FALSE	无错误。 示例：处理成功，但有一个负面或不完整的结果。
0x0000_0203	S_PENDING	无错误。 示例：处理成功，但还没有结果。
0x0000_0256	S_WATCHDOG_TIMEOUT	无错误。 示例：处理成功，但发生了超时。

TCP Winsock 错误代码

Hex	Dec	名称	描述
0x274C	10060	WSAETIMEDOUT	发生连接超时 - 在创建连接时发生错误，因为远程终端在特定时间后未正确响应，或者所建立连接因所连接主机未响应而无法维持。
0x274D	10061	WSAECONNREFUSED	连接遭到拒绝 - 无法建立连接，因为目标计算机明确拒绝了该连接。此错误通常由尝试连接到外部主机上处于非活动状态的服务（即没有运行服务器应用程序的服务）导致。
0x2751	10065	WSAEHOSTUNREACH	不存在至主机的路由 - 套接字操作引用了不可用的主机。
更多 Winsock 错误代码：Win32 错误代码			

8.5 技术支持和服务

倍福公司及其合作伙伴在世界各地提供全面的技术支持和服务，对与倍福产品和系统解决方案相关的所有问题提供快速有效的帮助。

下载搜索器

我们的下载搜索器包含我们供您下载的所有文件。您可以通过它搜索我们的应用案例、技术文档、技术图纸、配置文件等等。

可供下载的文件格式多种多样。

倍福分公司和代表处

若需要倍福产品的本地支持和服务，请联系倍福分公司或代表处！

倍福遍布世界各地的分公司和代表处地址可在倍福官网上找到：http://www.beckhoff.com.cn

该网页还提供更多倍福产品组件的文档。

倍福技术支持

技术支持部门为您提供全面的技术援助，不仅帮助您应用各种倍福产品，还提供其他广泛的服务：

- 技术支持
- 复杂自动化系统的设计、编程和调试
- 以及倍福系统组件的各种培训课程

热线电话：	+49 5246 963-157
电子邮箱：	support@beckhoff.com

倍福售后服务

倍福服务中心提供所有售后服务：

- 现场服务
- 维修服务
- 备件服务
- 热线服务

热线电话：	+49 5246 963-460
电子邮箱：	service@beckhoff.com

倍福公司总部

Beckhoff Automation GmbH & Co. KG

Huelshorstweg 20
33415 Verl
Germany

电话：	+49 5246 963-0
电子邮箱：	info@beckhoff.com
网址：	www.beckhoff.com

Trademark statements

Beckhoff®, ATRO®, EtherCAT®, EtherCAT G®, EtherCAT G10®, EtherCAT P®, MX-System®, Safety over EtherCAT®, TC/BSD®, TwinCAT®, TwinCAT/BSD®, TwinSAFE®, XFC®, XPlanar® and XTS® are registered and licensed trademarks of 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.

DALI, DALI-2, D4i, DALI+, and DiiA are trademarks in various countries in the exclusive use of the Digital Illumination Interface Alliance.

Docker is a trademark or registered trademark of Docker, Inc. in the United States and/or other countries. Docker, Inc. and other parties may also have trademark rights in other terms used herein.
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.

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

MAX®, Stratix®, Cyclone®, Altera®, Agilex™, Arria®, 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.

Wireshark is a registered trademark of Sysdig, Inc.

OPC UA is a registered trademark of the OPC Foundation.

更多信息:
www.beckhoff.com/tf6100

Beckhoff Automation GmbH & Co. KG
Hülshorstweg 20
33415 Verl
Germany
电话号码: +49 5246 9630
info@beckhoff.com
www.beckhoff.com