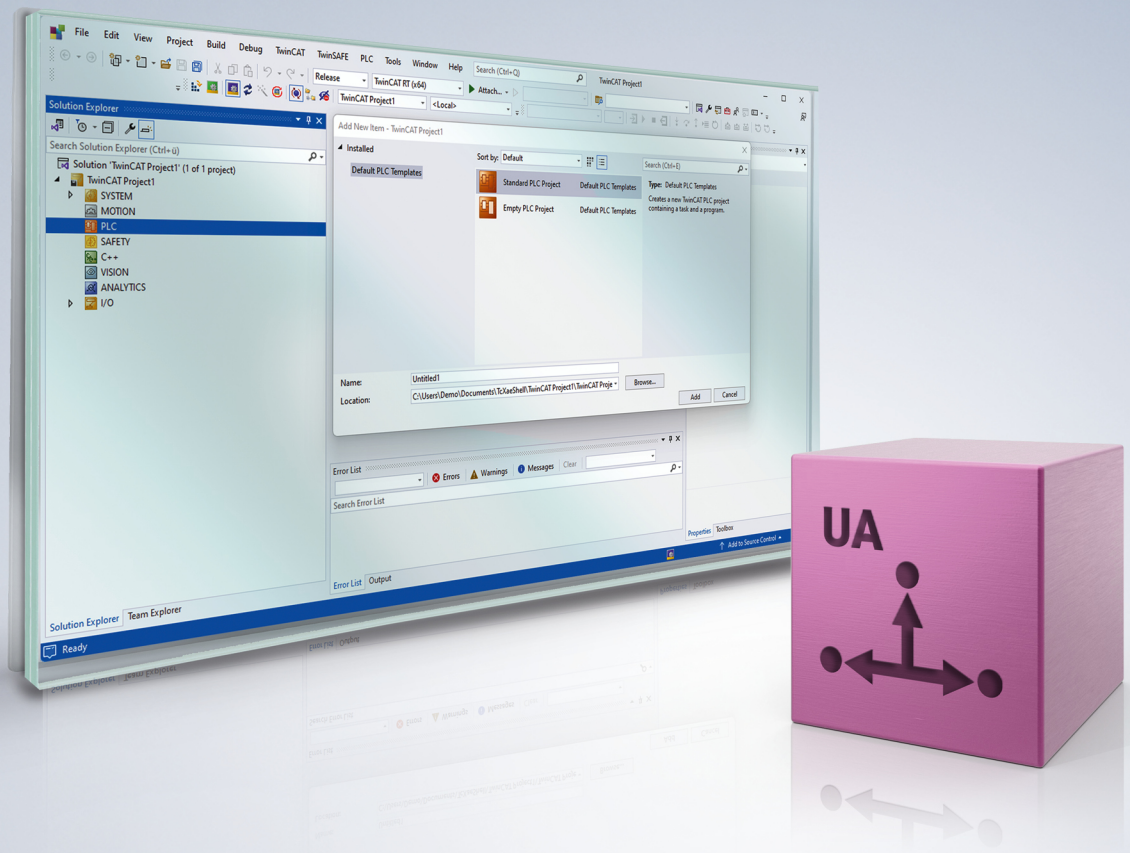
BECKHOFF New Automation Technology

手册 | ZH

TF6100

TwinCAT 3 | OPC UA Client



2026-04-08 | 版本： 1.3.0

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
1.3.x	**新增：** 安装 – TwinCAT 3 Usermode Runtime [▶ 12] 版本概览 [▶ 24] 数据类型 [▶ 28] ApplicationURI [▶ 57] 使用说明 [▶ 62]
1.2.x	**新增：** TS6100_Installation [▶ 11] TwinCAT 2 下的授权 [▶ 14] 技术简介/日志记录 [▶ 55]

版本	修改
	参数列表 [▶ 95]
1.1.x	**新增：** 支持的功能 快速入门 (TwinCAT 2) [▶ 21] 安全 [▶ 52] 教程 [▶ 97]

2 概述

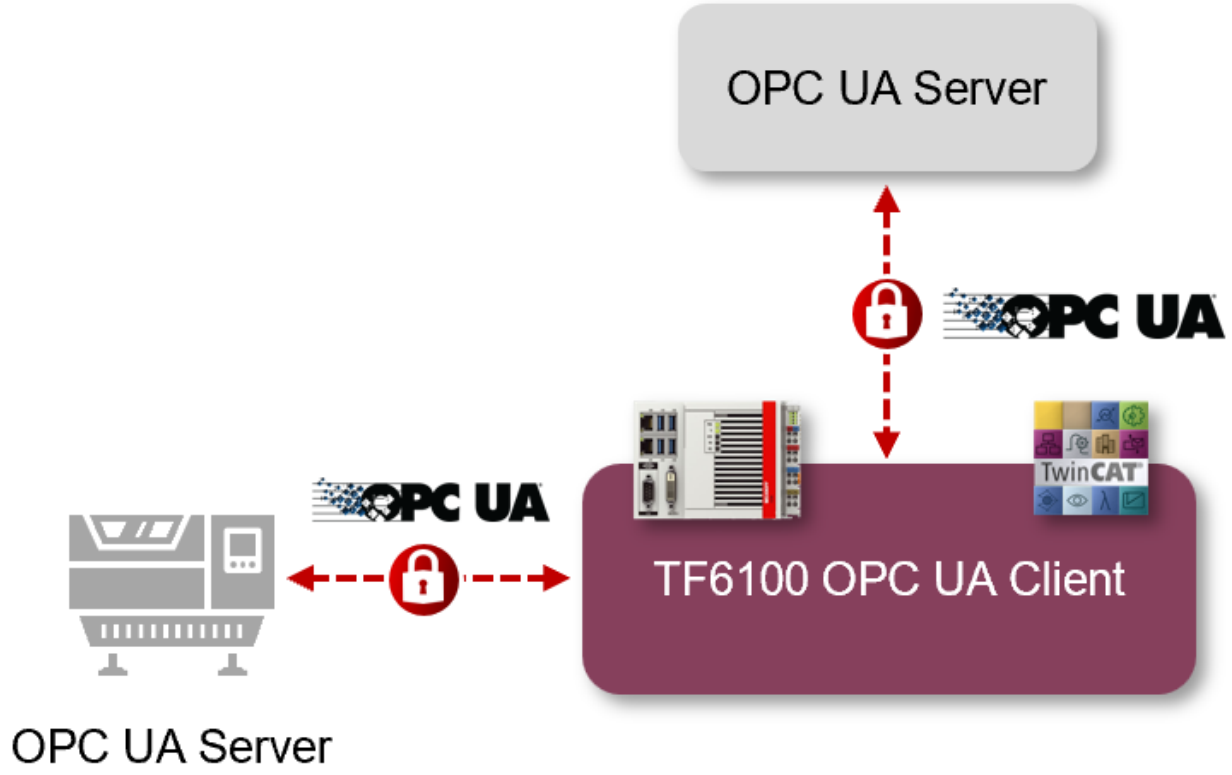
OPC 统一架构 (OPC UA) 是熟知的 OPC 标准的下一代产品。此为全球标准化通信协议，无论制造商和平台如何，都可以通过它交换机器数据。OPC UA 已经在协议中直接集成了通用安全标准。与 OPC 常规标准相比，OPC UA 的另一大优势是独立于 COM/DCOM 系统。

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。

本文档介绍的是 TwinCAT 3 OPC UA 客户端，此为 TwinCAT 运行时环境提供 OPC UA 客户端接口的软件组件。因此，TwinCAT 3 OPC UA 客户端可用于启动与 OPC UA 服务器的连接，以便与其交换数据。



技术用例的范围从基于 TCP 的（因此不具备实时能力）机器间通信，到要连接的 OPC UA 服务器位于云中时的机器到云通信。

TwinCAT OPC UA 客户端在技术上有两种不同的型号可供选择：

1. 作为 TwinCAT I/O 设备
2. 作为 PLC 功能块

更多信息

- 有关功能差异的概述，我们推荐我们的文章 支持的功能。
- 请注意该产品的 系统要求 [▶ 10]。
- 如需快速了解该产品，我们推荐我们的文章 安装 [▶ 11] 和 快速入门 [▶ 18]。

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。

客户端 v2.x

该客户端版本是 TwinCAT 3.1 Build 4026 的设置版本 2.x 的一部分。

技术数据	描述
操作系统	Windows 10 Windows 11 Windows Server 2022 TwinCAT/BSD TwinCAT/Beckhoff RT Linux®
目标平台	PC 架构（x86、x64、Arm®）
.NET Framework	---
最低 TwinCAT 版本	TwinCAT 3.1 Build 4026.19
最低 TwinCAT 安装级别	TwinCAT 3 XAE、XAR
所需 TwinCAT 授权	TF6100 TC3 OPC UA

客户端 v1.x

该客户端版本是 TwinCAT 3.1 Build 4024 的设置版本 4.4.x 和 TwinCAT 3.1 Build 4026 的工作负载版本 1.x 的一部分。

技术数据	描述
操作系统	Windows 10 Windows 11 Windows CE 6/7 Windows Server 2022
目标平台	PC 架构（x86、x64、Arm®）
.NET Framework	---
最低 TwinCAT 版本	TwinCAT 3.1
最低 TwinCAT 安装级别	TwinCAT 3 XAE、XAR
所需 TwinCAT 授权	TF6100 TC3 OPC UA

SampleServer

SampleServer 是所有设置和工作负载的一部分。

技术数据	描述
操作系统	Windows 10 Windows Server 2022
目标平台	PC 架构（x86，x64）
.NET Framework	4.7.2
最低 TwinCAT 版本	---
最低 TwinCAT 安装级别	---
所需 TwinCAT 授权	---

工程设计与运行时之间的版本兼容性

下表概述了使用客户端时工程设计与运行时系统之间支持的版本组合。

开发	运行环境	支持
客户端 v1	客户端 v1	是
客户端 v1	客户端 v2	是
客户端 v2	客户端 v1	否
客户端 v2	客户端 v2	是

要全面了解版本差异，我们推荐阅读以下文章：版本概览 [▶ 24]、软件架构 [▶ 25]。

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

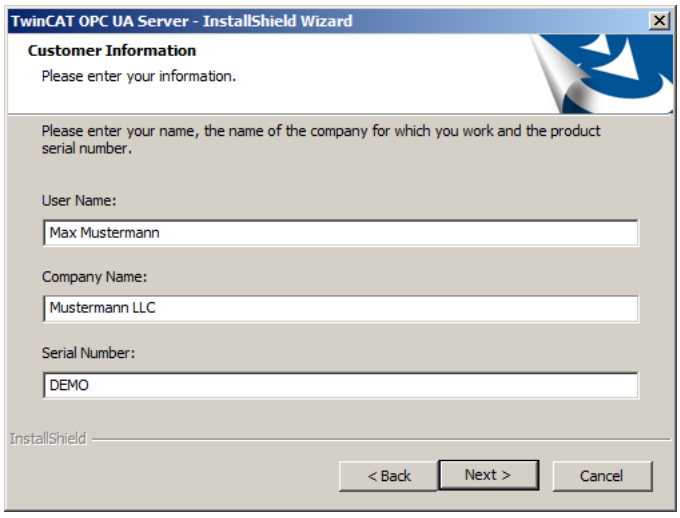
更新安装

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 TF610x.OpcUaClientPubSub.XAE
tcpkg install TF610x.OpcUaClientPubSub.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：

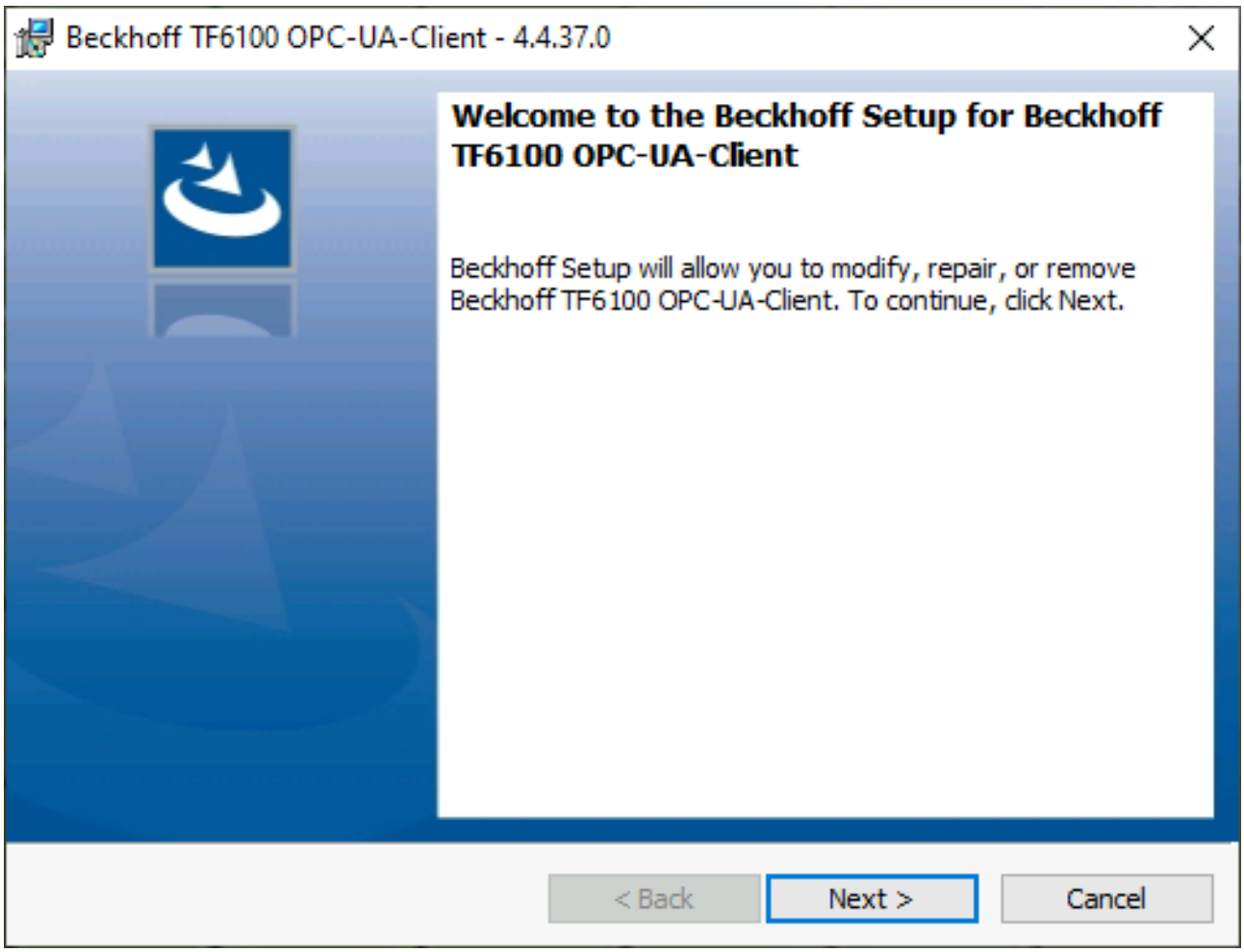
- 要启动的可执行文件的安装路径
- 启动 TwinCAT 3 功能组件所需的参数

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。

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。

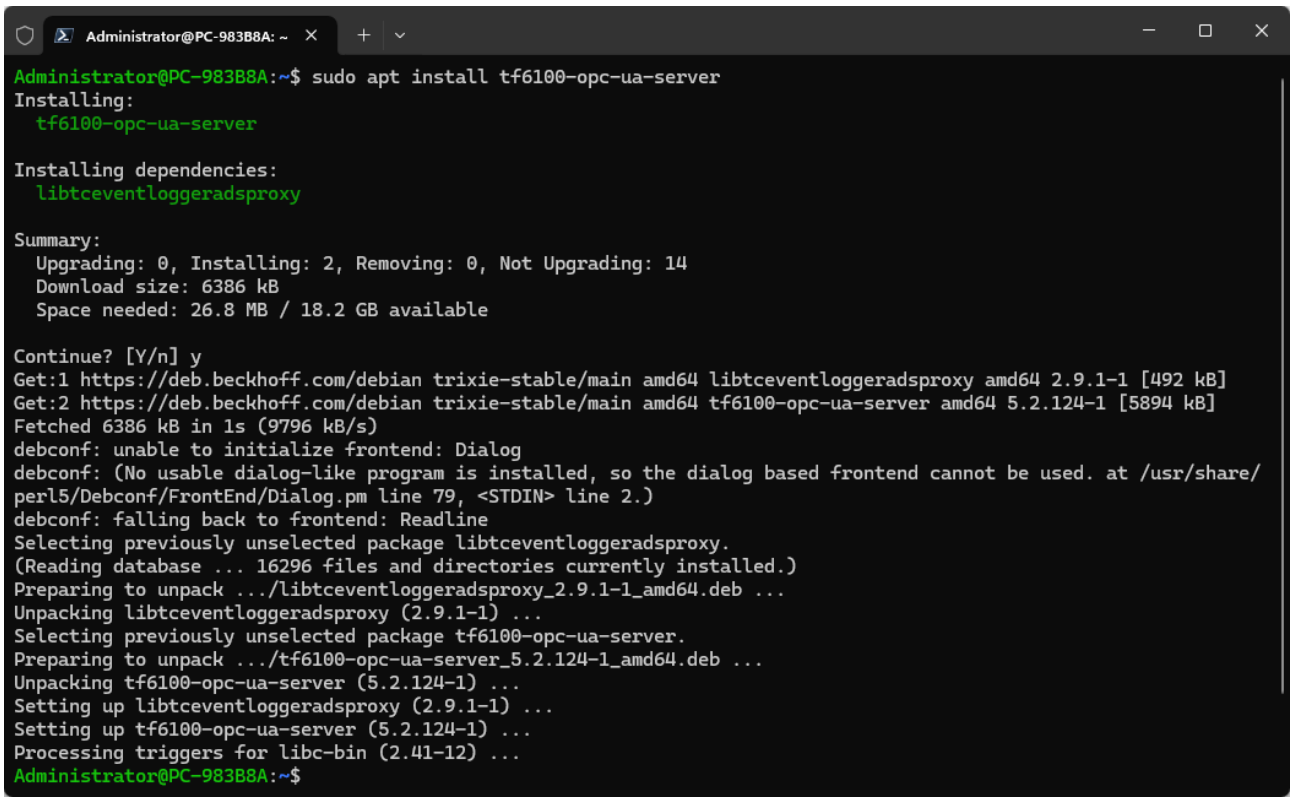
```
sudo apt update
```

1. 搜索现有的 TF6100 软件包。

```
sudo apt search tf610x
```

1. 通过 Package Manager 安装 TF6100 OPC UA 服务器。

```
sudo apt install tf610x-opc-ua-client-pubsub
```



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 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。

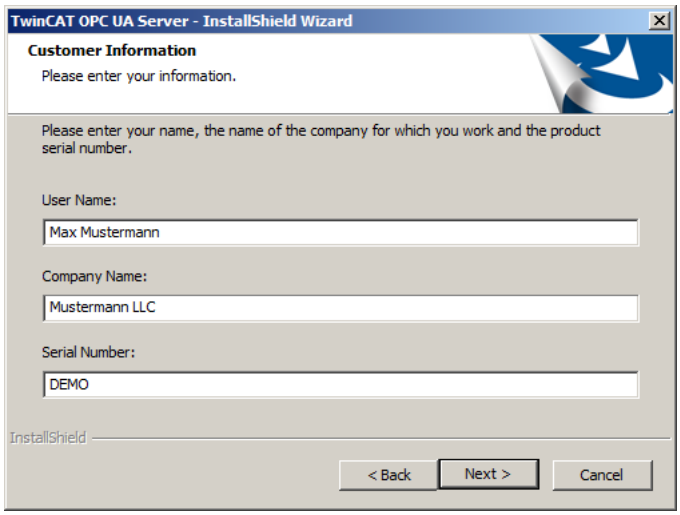
有关授权的更多信息，请参见以下章节：

- 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 授权更多信息，请参见倍福信息系统的文档（“TwinCAT 3 授权”）。

TwinCAT 2 下的授权

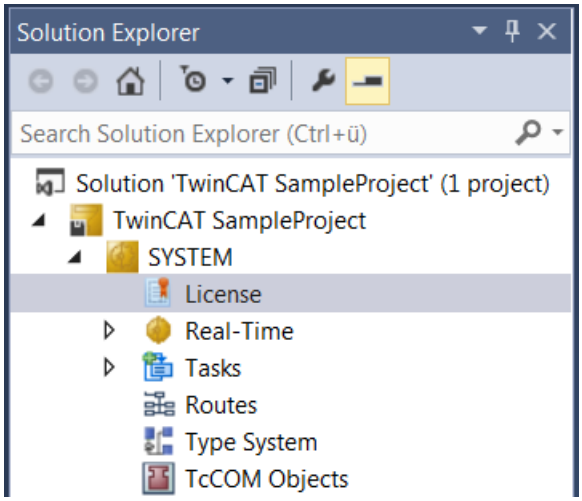
在安装过程中，该产品已获得 TwinCAT 2 授权；您可以在相应的文本字段中输入授权密钥。要评估该产品，您可以将其激活 30 天。为此，请输入“Demo”作为授权密钥。



授权一个 TwinCAT 3 功能的 7 天测试版本

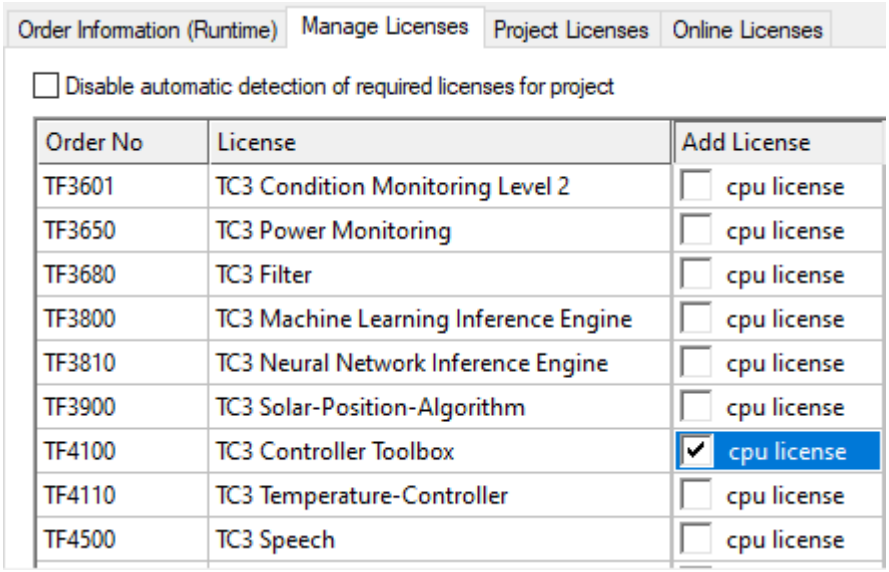
对于TwinCAT 3 许可证加密狗无法启用 7 天测试版本。

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**。

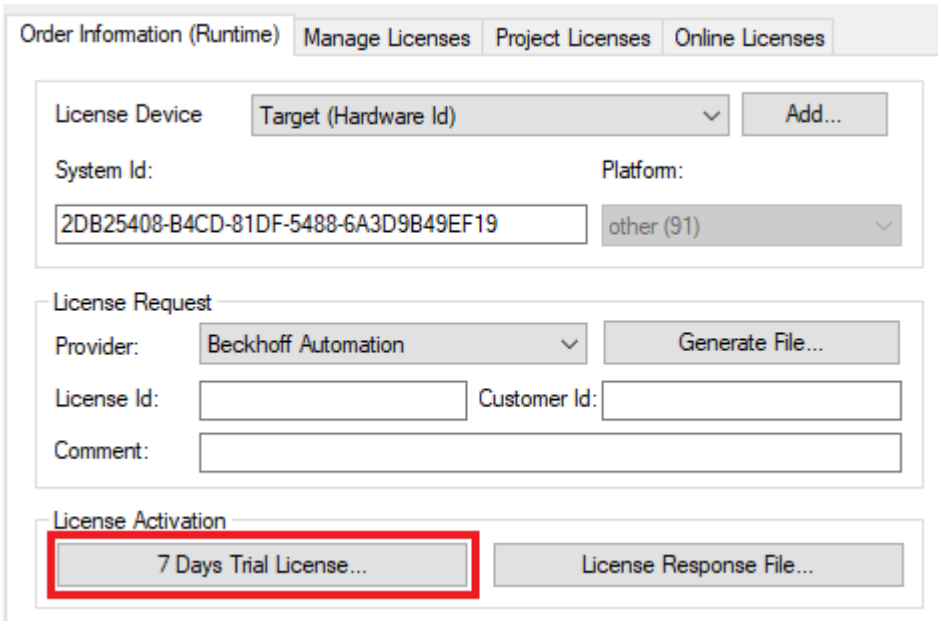


⇨ TwinCAT 3 许可证管理器打开。

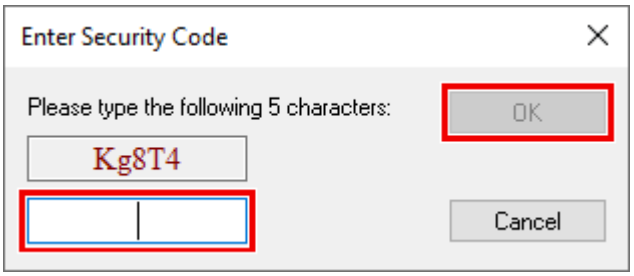
5. 打开**管理许可证**选项卡。在**添加许可证**栏中，选中希望添加到项目的许可证的复选框（例如“TF4100 TC3 控制器工具箱”）。



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

下一章介绍的是 TwinCAT OPC UA I/O 客户端的快速入门。在这些说明中，设置了与示例 OPC UA 服务器的连接，该服务器在其命名空间中提供了一些变量。这些变量被添加到 TwinCAT OPC UA I/O 客户端的过程映像中，然后链接到 PLC 变量。

OPC UA 服务器示例

OPC UA 服务器示例与 TF6100 OPC UA 客户端安装程序一起交付，位于客户端的安装目录 [▶ 28] 中。另外，您也可以使用 TwinCAT OPC UA 服务器代替 OPC UA 服务器示例，通过 OPC UA 提供一些变量。

下面按执行顺序介绍这些步骤：

- 启动 OPC UA 服务器示例
- 创建 TwinCAT 项目
- 读取 OPC UA 变量
- 开始生成代码

启动 OPC UA 服务器示例

1. 在 Windows Explorer 中，导航至 TF6100 FunctionSample 的安装目录，然后导航至子目录“SampleServer”。
2. 以管理员身份启动文件 TcOpcUaSampleServer.exe。

⇨ 服务器在控制台窗口中启动，然后可以通过以下 OPC UA URL 进行访问：

```
opc.tcp://localhost:48030
```

您可以确认有关受限运行时的信息。

创建 TwinCAT 项目

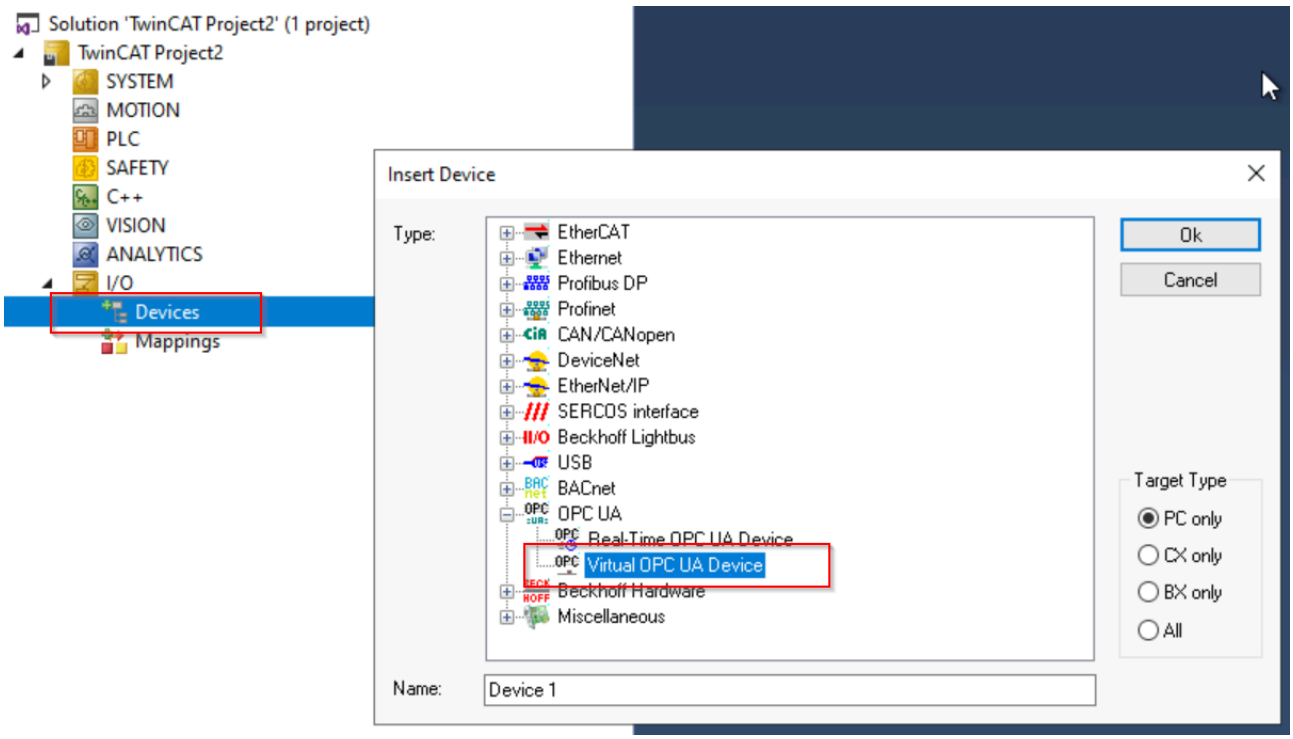
1. 打开 TwinCAT XAE Shell。
2. 在 **File**（文件）菜单中，选择命令 **New > Project**（新建 > 项目）。
3. 在项目中添加 PLC 项目。

⇨ 创建了一个新的 TwinCAT 项目，其中包括 PLC 项目。

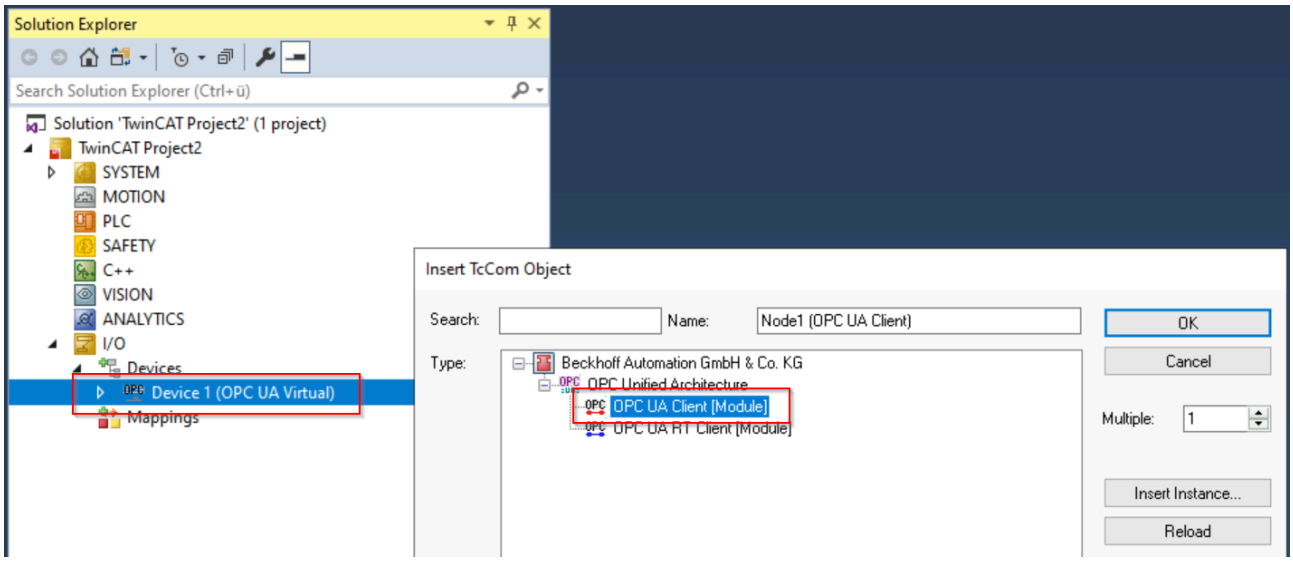
读取 OPC UA 变量

✓ 在此步骤中，TwinCAT OPC UA 客户端用于建立与服务器的连接，并读取其中可用的变量。

1. 在 TwinCAT 项目中添加一个新的 I/O 设备。
   使用“OPC UA 虚拟设备”作为设备类型

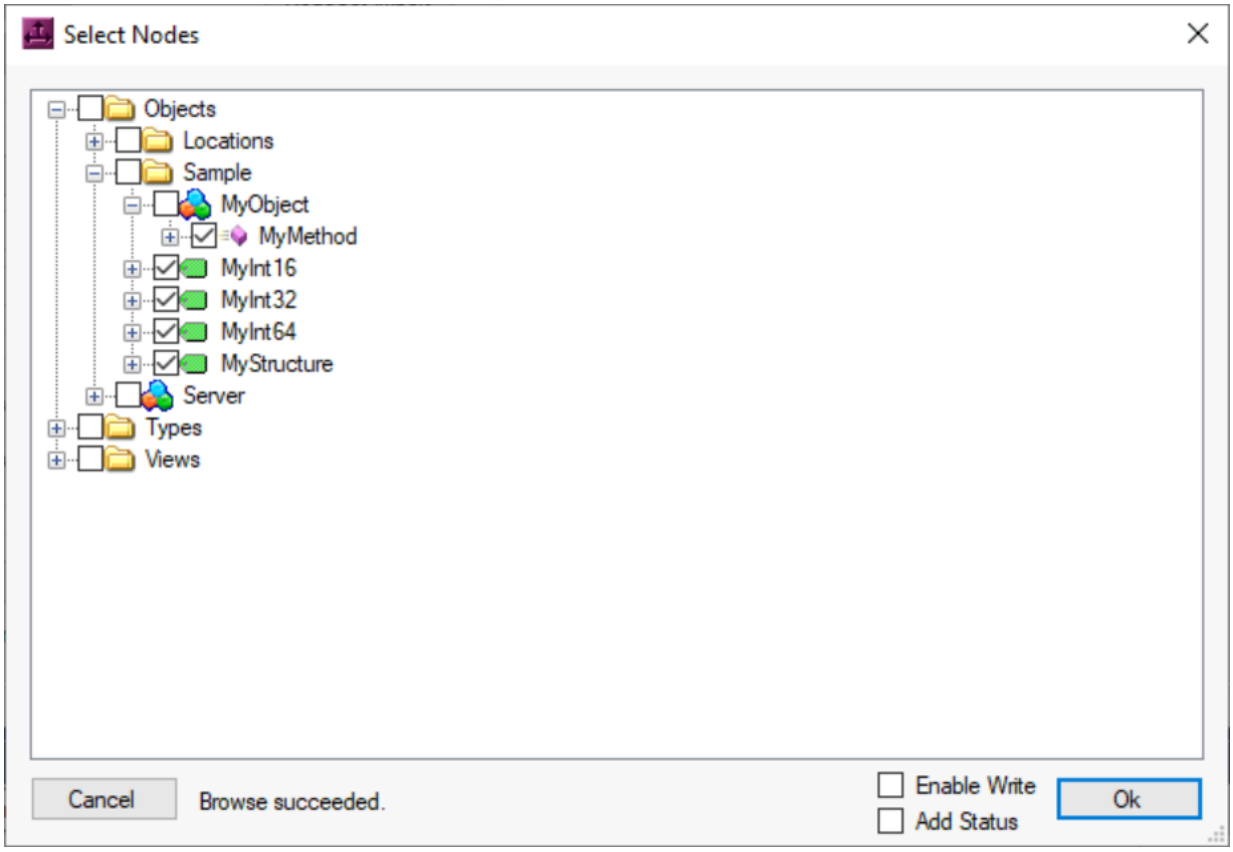


2. 为设备添加 OPC UA 客户端



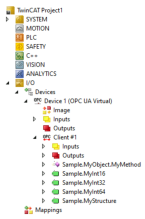
3. 双击客户端，打开 OPC UA 客户端的设置。
4. 导航至 **Settings**（设置）选项卡。输入 OPC UA 服务器的服务器 URL。在本示例中为“opc.tcp://localhost:48030”。

5. 点击**Add Nodes**（添加节点）。建立与服务器的连接后，服务器的地址空间将显示在一个单独的对话框中。



6. 选择上图所示的节点，然后点击 **OK**（确定）按钮。

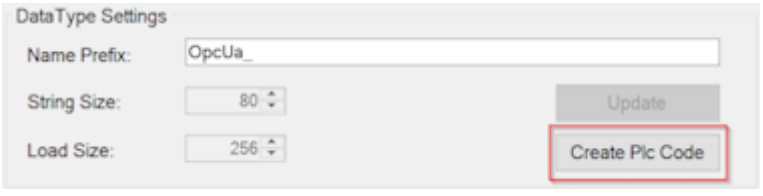
⇨ 已添加客户端过程映像的变量和方法。



开始生成代码

✓ 应使用自动生成代码功能生成 PLC 变量，以匹配添加的 OPC UA 节点。生成的 PLC 变量会自动关联到节点。或者，您也可以手动执行映射。

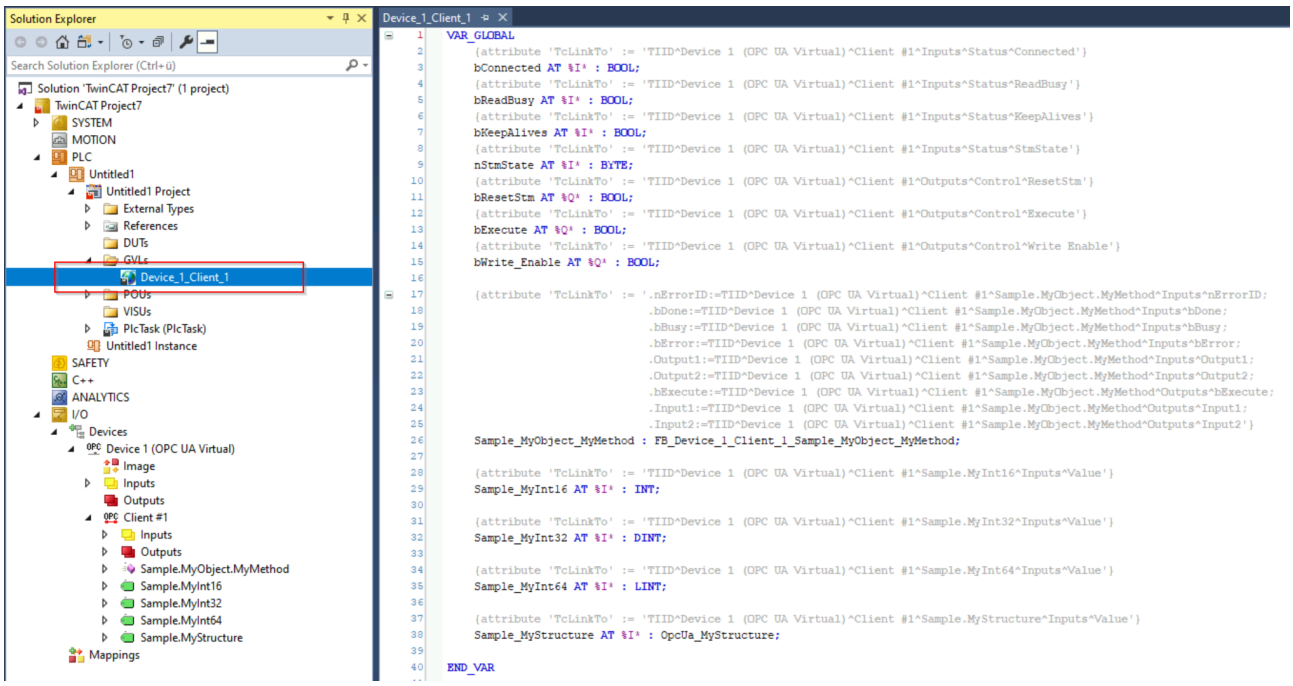
1. 双击 OPC UA 客户端。
2. 在 **Settings**（设置）选项卡中，选择 **DataType Settings**



（数据类型设置）部分。

3. 点击 **Create Plc Code**（创建 Plc 代码）按钮，开始生成代码。

⇨ 代码生成器会在现有 PLC 项目中创建一个 GVL，其名称源自 OPC UA 客户端设备名称。现在，PLC 变量已在 GVL 中自动创建，并通过“TcLinkTo”编译关联到 I/O 设备过程映像中的相应节点。



4. 激活配置。

⇨ OPC UA 节点的数值通过映射从服务器读取并写入 PLC 变量。

关于调用该方法的更多信息

调用该方法需要进一步的 PLC 逻辑。为此，已通过生成代码功能创建了一个合适的功能块，并提供了相应的输入/输出参数，请参见 方法调用 [▶ 32] 章节。

4.2 快速入门 (TwinCAT 2)

下一章提供了有关 TwinCAT OPC UA 客户端的快速入门。在这些说明中，PLC [▶ 96] 示例用于建立与 OPC UA 服务器的连接，并读出一个变量。任何服务器都可以用作 OPC UA 服务器，例如 TwinCAT OPC UA 服务器示例 [▶ 18]，它也可以作为可执行文件在 TwinCAT 2 系统上启动。或者，您也可以使用 OPC 基金会的 .NET 标准服务器示例，该服务器可从 OPC 基金会的 GitHub 存储库中获取。

以下示例基于 TwinCAT OPC UA 服务器示例。不一定要启动服务器才能执行各个步骤。本快速入门旨在让您熟悉 PLC 示例，并了解您需要根据操作环境调整示例的哪些部分。

下文将按执行顺序详细介绍这些行动步骤：

- OPC UA 服务器示例的地址空间
- 识别要读取的 OPC UA 节点
- 为 TwinCAT OPC UA 客户端调整 PLC 示例

TwinCAT OPC UA I/O 客户端

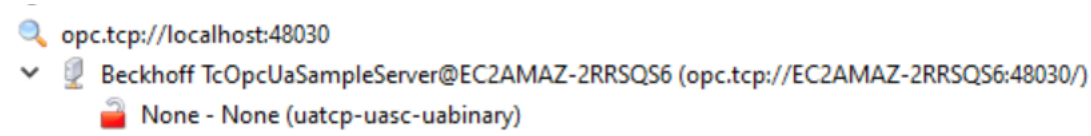
TwinCAT OPC UA I/O 客户端仅适用于 TwinCAT 3。在 TwinCAT 2 下，我们仅通过 PLC 库 Tc2_PLCopen_OpcUa 实现通信。本文就是为使用该库而撰写的。

OPC UA 服务器示例的地址空间

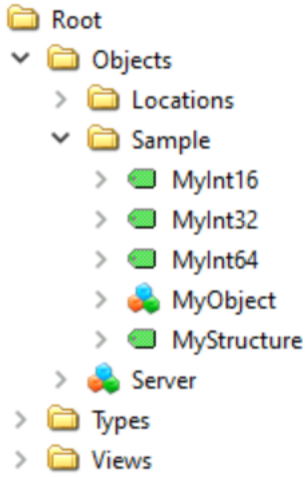
启动服务器后，其地址空间会包含一些节点，客户端可以读取这些节点。OPC UA 客户端需要以下信息才能建立连接：

- 服务器 URL
- 端点
- 使用的 IdentityToken 信息

在本示例中，可以通过以下服务器 URL 和端点“无/无”访问服务器。“匿名”用作 IdentityToken。以下截图是根据“UA Expert”软件制作的。



建立连接后，此处使用的服务器地址空间如下：

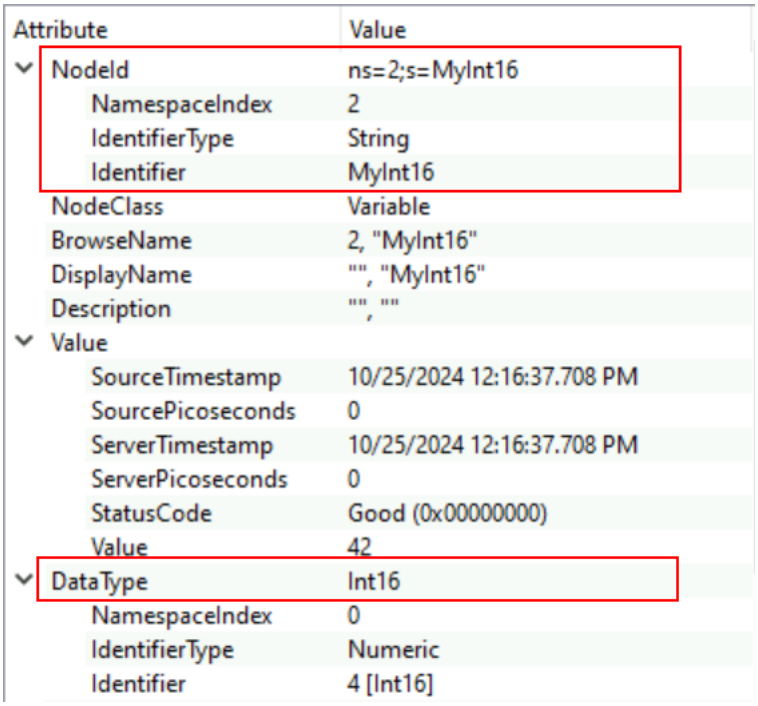


识别要读取的 OPC UA 节点

要读出一个节点，需要特定的地址信息，这些信息以所谓“属性”的形式存在。读取节点需要以下属性：

- 节点 ID（由 NamespaceIndex 或 NamespaceName 以及 IdentifierType 和 Identifier 组成）
- 节点的数据类型

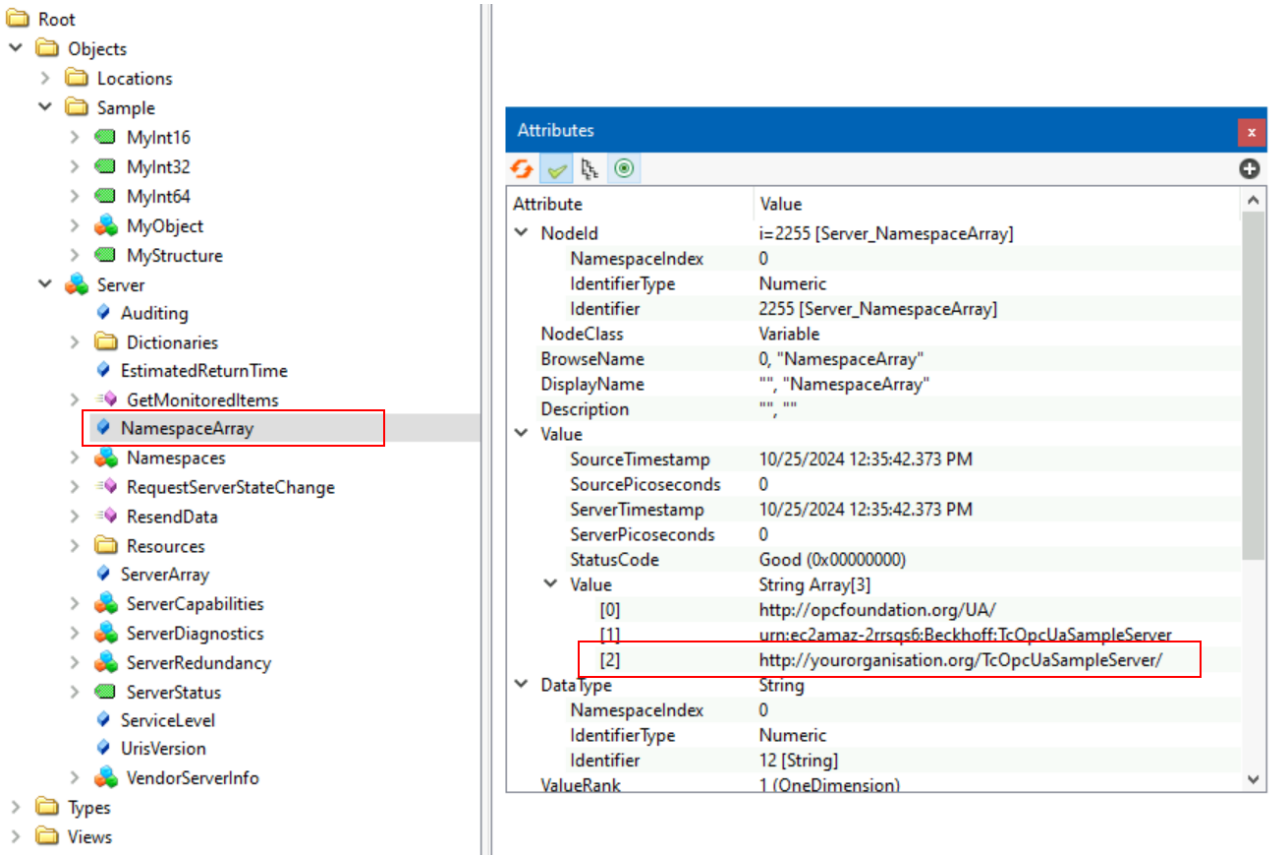
在本示例中，我们要读取“MyInt16”节点。上述属性如下：



Attribute	Value
NodeId	ns=2;s=MyInt16
NamespaceIndex	2
IdentifierType	String
Identifier	MyInt16
NodeClass	Variable
BrowseName	2, "MyInt16"
DisplayName	"", "MyInt16"
Description	"", ""
Value	
SourceTimestamp	10/25/2024 12:16:37.708 PM
SourcePicoseconds	0
ServerTimestamp	10/25/2024 12:16:37.708 PM
ServerPicoseconds	0
StatusCode	Good (0x00000000)
Value	42
DataType	Int16
NamespaceIndex	0
IdentifierType	Numeric
Identifier	4 [Int16]

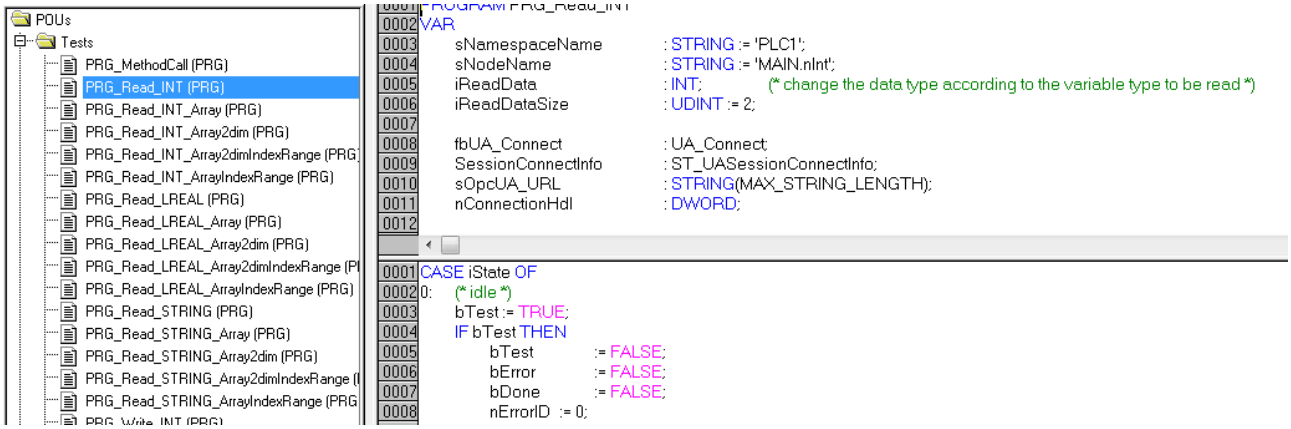
PLCopen 映射规定，OPC UA 节点必须从数据类型“Int16”映射到 PLC 数据类型“INT”。我们必须在下一步中考虑到这一点。

您可以使用 NamespaceIndex 进行通信。然而，由于 NamespaceIndex 可能会发生动态变化，因此最佳做法是使用相关的 NamespaceName。另一方面，NamespaceName 是静态的，可以在运行时在 NamespaceIndex 中进行解析。您可以在“服务器”对象下方的所谓“NamespaceArray”中找到 NamespaceName。然后，NamespaceName 就是数组中的第 x 个条目，即此处的第二个条目。



为 TwinCAT OPC UA 客户端调整 PLC 示例

在 TwinCAT PLC 控制中打开 PLC 示例，并打开 PRG_Read_INT 程序。该程序已包含一个现成的状态机，它使用 Tc2_PLCopen_OpcUa 库中的功能块来连接 OPC UA 服务器并读取节点。



建立连接

现在可能需要调整以下变量值以建立连接：

- sOpcUA_URL：将变量值设置为服务器的 ServerURL，本例中为“opc.tcp://localhost:48030”
- SessionConnectInfo.eSecurityMode：在这种情况下，我们使用的服务器会使用 SecurityMode“无”。因此，该变量值必须设置为“eUASecurityMsgMode_None”。
- SessionConnectInfo.eSecurityPolicyUri：在这种情况下，我们使用的服务器会使用 SecurityPolicy“无”。因此，该变量值必须设置为“eUASecurityPolicy_None”。

- 由于所用服务器使用“匿名”作为 IdentityToken，因此无需为 SessionConnectInfo.stUserIdentTokenType 变量指定任何值。

因此，PLC 程序的该部分如下所示：

```
(* Parameterize *)
sOpcUA_URL                          := 'opc.tcp://localhost:48030';     (* OpcUA Url of the device with ip address *)

SessionConnectInfo.tConnectTimeout      := T#1S;
SessionConnectInfo.tSessionTimeout      := T#1S;
SessionConnectInfo.sApplicationName     := '';
SessionConnectInfo.eSecurityMode        := eUASecurityMsgMode_None;
SessionConnectInfo.eSecurityPolicyUri   := eUASecurityPolicy_None;
SessionConnectInfo.eTransportProfileUri := eUATransportProfileUri_UATcp;
```

解析 NamespaceName

下一步，我们必须将 NamespaceName 解析为相应的 NamespaceIndex。为此，我们会使用功能块 UA_GetNamespaceIndex。该功能块接收由我们确定为输入的 NamespaceName。在 PLC 示例中，将 sNamespaceName 变量的值设置为“http://yourorganisation.org/TcOpcUaSampleServer/”。

调整地址信息

现在，我们可以调整 PLC 示例中节点的其他地址信息（IdentifierType 和 Identifier）。为此，我们将变量 sNodeName 和 NodeID.eIdentifierType 设置为之前确定的值。因此，程序的声明部分如下：

```
PROGRAM PRG_Read_INT
VAR
    sNamespaceName          : STRING := 'http://yourorganisation.org/TcOpcUaSampleServer/';
    sNodeName               : STRING := 'MyInt16';
```

以及实现部分：

```
(* Get Node Handle *)
NodeID.eIdentifierType := eUAIdentifierType_String;
NodeID.nNamespaceIndex := nNamespaceIndex;
NodeID.sIdentifier := sNodeName;
fbUA_NodeGetHandle(
    Execute             := TRUE,
    ConnectionHdl       := nConnectionHdl,
    NodeID              := NodeID,
    NodeHdl             => nNodeHdl
);
```

完成这些调整后，便可以激活程序。然后，服务器会读取 OPC UA 节点的变量值，并将其存储在 PLC 变量 iReadData 中。

4.3 版本概览

下表概述了各主要版本 TwinCAT OPC UA 客户端的功能。特别是对两种可用的使用型号（PLCopen 功能块和 I/O 客户端）进行了区分。

功能	PLCopen 功能块	I/O 客户端 v1.x.x	I/O 客户端 v2.x.x
轮询	x	x	x
订阅	-	x	x
方法调用	x	x	x
符合 IEC61131 标准的基本数据类型	x	x	x
符合 IEC61131 标准的基本数据类型数组	x	x	x
结构	-	x	x
结构数组	-	x	x
固定长度的数组	x	x	x
动态长度的数组	-	-	x
通过 X.509 证书（自签名 + CA）保证传输层的安全性	x	x	x

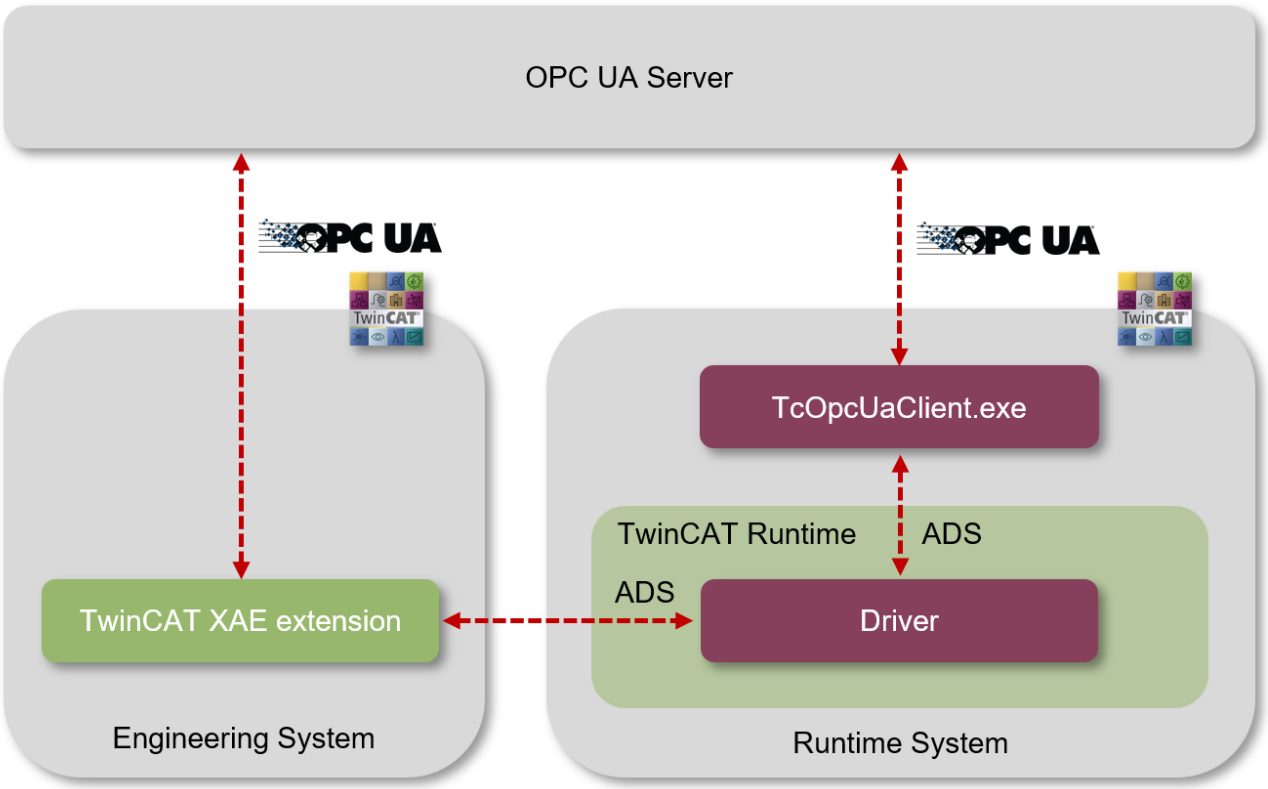
功能	PLCopen 功能块	I/O 客户端 v1.x.x	I/O 客户端 v2.x.x
通过用户名/密码保证应用层的安全性	x	x	x
通过 X.509 证书保证应用层的安全性	x	x	x
与“无/无”端点通信	x	x	x
与 Basic128 端点通信 (Sign&Encrypt)	x	x	x
与 Basic128Rsa15 端点通信 (Sign&Encrypt)	x	x	x
与 Basic256 端点通信 (Sign&Encrypt)	x	x	x
与 Basic256Sha256 端点通信 (Sign&Encrypt)	x	x	x
代码生成	-	x	x
支持 TwinCAT 3 Usermode Runtime	-	x	x
支持 Microsoft Windows	x	x	x
支持 TwinCAT/BSD	x	-	x
支持 Beckhoff RT Linux®	x	-	x
通过目标系统建立工程设计连接（见 软件架构 [▶ 25]）	-	-	x

4.4 软件架构

先前版本的 TwinCAT OPC UA 客户端（版本 1.x.x）由多个组件组成，彼此之间保持紧密互联。其中包括：

- 操作系统中的一个进程 (TcOpcUaClient.exe)
- 实时通信驱动程序 (TcIoOpcUa.sys)
- TwinCAT XAE 扩展软件

下图介绍了这些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（TcOpcUaClient.exe）负责 OPC UA 协议功能，并通过 ADS 服务器提供这些功能，以便 TwinCAT 实时组件（如驱动程序或 PLC 库）可以访问这些功能。

实时通信驱动程序

实时驱动程序负责 I/O 设备与操作系统中的进程之间的通信。它会将配置的过程数据转换成 ADS 报文，与流程交换，以便转换成 OPC UA 命令。TwinCAT XAE 中有一个工程设计组件，用于配置通信连接和选择数据点。

TwinCAT XAE 扩展软件

TwinCAT XAE 中的工程设计组件为 I/O 设备提供了图形化配置界面。这意味着可以从 OPC UA 服务器读取变量，并将其添加到过程映像中，以便稍后在运行时对其进行处理。

PLC 库

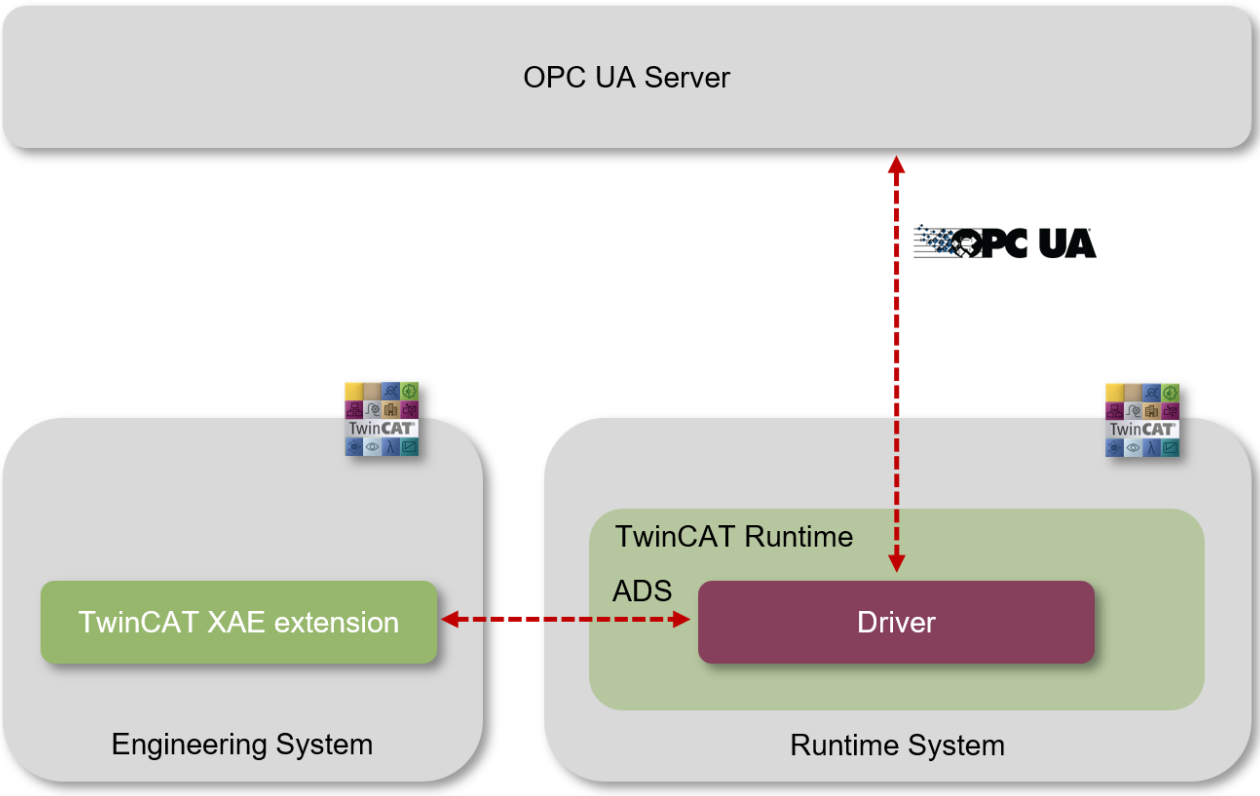
PLC 库可为 PLC 逻辑提供操作系统中流程的 OPC UA 功能。与驱动程序类似，PLC 库也会通过 ADS 与进程进行通信，以访问 OPC UA 功能。

TwinCAT OPC UA 客户端（版本 2.x.x 及以上）

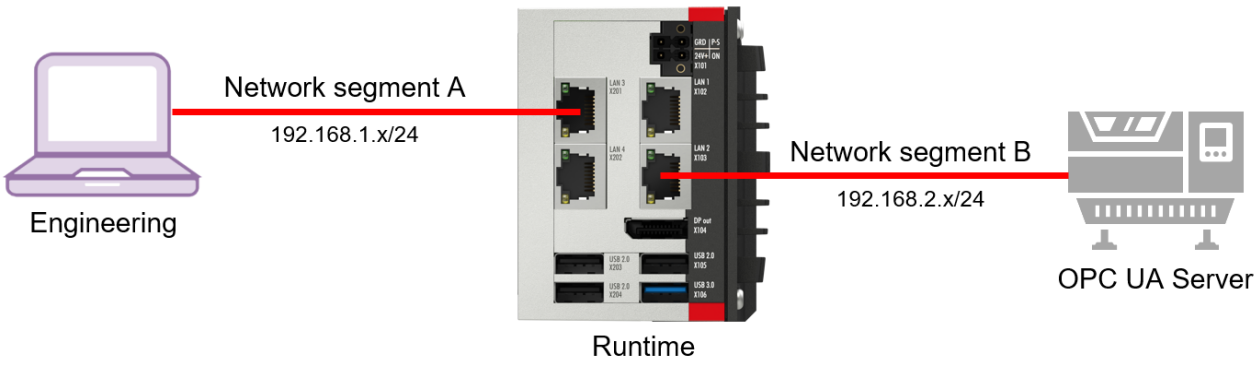
较新版本的 TwinCAT OPC UA 客户端只包括一个工程设计组件和一个运行时组件（即所谓的“驱动程序”）。省略操作系统中的进程（TcOpcUaClient.exe）。

驱动程序可提供完整的 OPC UA 通信堆栈，然后供工程设计和 TwinCAT 运行时使用。这样不仅可以减少复杂性，还能与服务器建立标准化的集中式通信连接。工程设计不再自行与 OPC UA 服务器建立通信连接，而是使用所选目标设备的驱动程序。（当然，目标设备也可以是本地系统。）

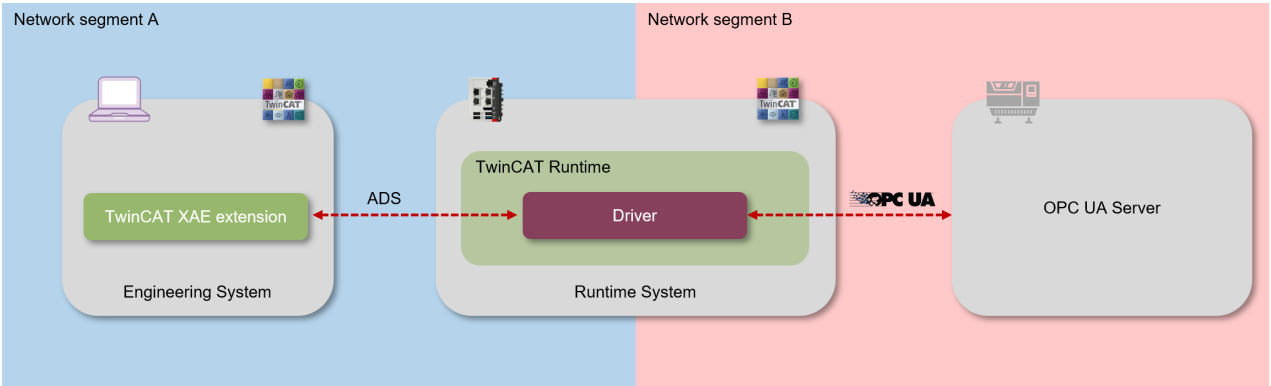
下图再次阐释了这种关系。



该程序可降低证书管理的复杂性，因为只需管理一个证书存储空间，即驱动程序的存储空间。此外，还支持网络环境，可以从运行时系统访问 OPC UA 服务器，但不能从工程设计系统访问。在操作环境中，控制器通常会将不同的网段分割到专用的网络接口卡上，例如“机器”网段和“维护”网段，后者用于在维护时连接工程设计笔记本电脑。下图再次阐释了这种关系。



下图以略微不同的形式再次阐释了这种关系。



4.5 数据类型

符合 IEC61131 标准的基本数据类型

为了读写数据，必须将 OPC UA 节点的数据类型分配给 TwinCAT 数据类型（映射）。基本数据类型的分配在标准化信息模型“PLCopen OPC UA Information Model for IEC 61131-3”中进行了描述，如下所示。您可以将此映射应用于 PLCopen 功能块和 TwinCAT OPC UA I/O 客户端。

PLC 数据类型	OPC UA 数据类型
BOOL	布尔类型
SINT	SByte
USINT	字节
INT	Int16
DINT	Int32
STRING	字符串
BYTE	USint
REAL	浮动
LREAL	双精度浮点型
UINT	UInt16
UDINT	UInt32
LINT	Int64
ULINT	UInt64
DT	DateTime
TIME	Int64
LTIME	Int64

4.6 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。

客户端版本 1.x.x

安装目录

应用程序的基本安装目录与 TwinCAT 安装目录相关。

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

然后，应用程序将安装在该目录下的以下目录中：

```
%TcInstallDir%\Functions\TF6100-OPC-UA\Win32\Client
```

TwinCAT OPC UA I/O 客户端的工程设计扩展软件安装在以下目录中：

```
%TcInstallDir%\Functions\TF6100-OPC-UA\Build_4026
```

TwinCAT OPC UA I/O 客户端的驱动程序安装在以下目录中：

```
%TcInstallDir%\3.1\Driver\Autoload
```

证书目录

用于建立安全连接的证书文件存储在以下目录中：

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

日志文件

日志文件 [▶ 55]存储在以下目录中：

```
%ProgramData%\Beckhoff\TF6100-OPC-UA\TcOpcUaClient\Logs
```

客户端版本 2.x.x

客户端版本 2.x. 的软件架构 [▶ 25]与先前版本有本质区别。这样会产生不同的应用程序目录。

安装目录

应用程序的基本安装目录与 TwinCAT 安装目录相关。

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

TwinCAT OPC UA I/O 客户端的工程设计扩展软件安装在以下目录中：

```
%TcInstallDir%\Functions\TF6100-OPC-UA\Build_4026
```

TwinCAT OPC UA I/O 客户端的驱动程序安装在以下目录中：

```
%TcInstallDir%\3.1\Driver\Autoload
```

证书目录

用于建立安全连接的证书文件存储在以下目录中：

```
%TcInstallDir%\3.1\Target\OpcUa\Client\PKI
```

日志文件

日志文件 [▶ 55]的处理方法将在另一篇文章中进行说明。

4.7 读取变量

可通过 TwinCAT OPC UA I/O 客户端从 OPC UA 服务器读取变量值。可以使用各种机制对数值进行采样，下文将详细介绍这些机制。

此上下文中的各种设置是在 I/O 设备的配置页面中进行的。**过程数据配置**区域会显示各种参数，用于设置从服务器采集数据的不同模式。

TwinCAT OPC UA I/O 客户端会提供三种不同的数据采集模式：

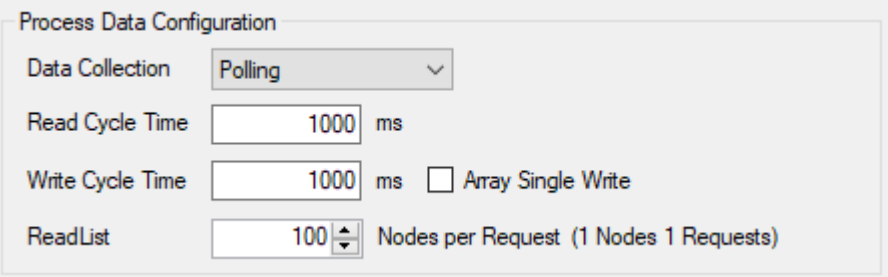
- 轮询（循环读取/写入）
- 订阅
- OnTrigger

轮询（循环读取/写入）

其中一种可能的数据采集方式是循环读取和写入。为读取和写入都定义了时间间隔。您还可以指定一条读取命令要读取多少个变量。

在轮询和订阅模式下写入变量

写入时请注意，只有当数值发生变化时才会写入。如果在循环结束时，配置变量的值没有发生变化，则不会写入新值。



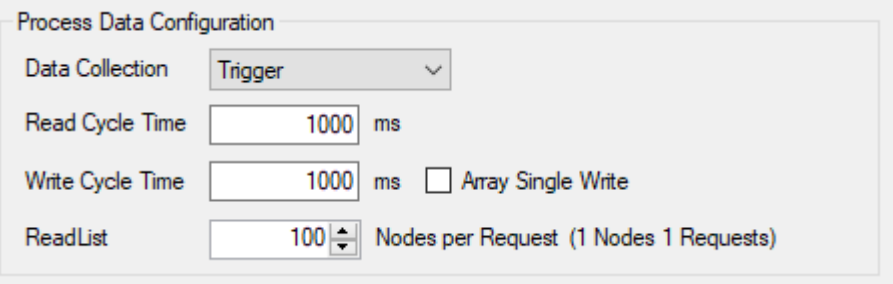
参数	描述
读取循环时间	指定循环读取变量的速度。

参数	描述
写入循环时间	定义在 OPC UA 通道上触发写入命令的频率。如果变量值在特定循环时间内多次变化，则只会将最后一个值写入 OPC UA 通道。如果在循环时间内没有改变配置值，则不会触发写入命令。
ReadList	为了节省带宽，OPC UA 通道上的读取命令被捆绑在一起。该参数会指定在 OPC UA 通道上的单个读取命令中收集多少个变量。后面的标签表示当前配置产生了多少条读取命令。
数组单次写入	如果数组中的某个值被更改，只有在激活时才会在 OPC UA 通道上对该值执行写入操作。若未激活，则始终会写入整个数组。

OnTrigger

此外，还可以选择通过触发变量来触发读取和写入。每个 OPC UA 客户端设备都有一个触发变量（可在“输出/控制/执行”下找到），可与 PLC 的变量连接，并根据需要进行设置。例如，如果只有在 PLC 发生特定事件时才从 OPC UA 服务器读取数据，则适合使用该选项。如果触发变量保持永久设置，则数据采集类型的行为方式与循环配置相同。

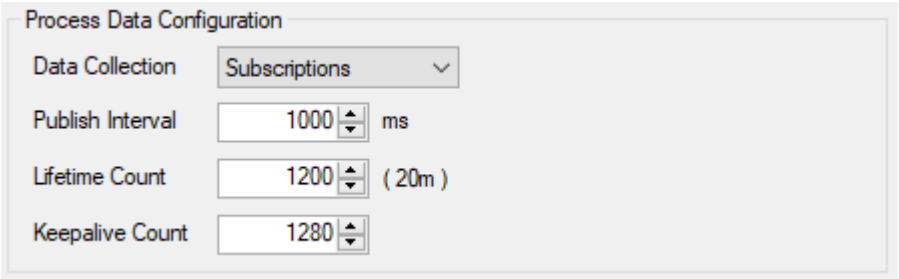
而写入时，如果已设置触发变量，则每个循环都会写入一个值。这里不考虑值的变化。



订阅

第三种也是最后一种数据采集方式是订阅。I/O 客户端会向相连 OPC UA 服务器注册订阅。可为“发布间隔”、“生命周期计数”和“保持连接计数”指定以下参数。

订阅模式主要用于读取变量。如果以这种模式写入数值，则其行为与循环写入相同（见上文）。



参数	描述
发布间隔	在指定时间后，相连 OPC UA 服务器会检查客户端是否有新的通知包。如果在一个发布间隔内发生多次数值变化，则只会传输最后一个数值。
生命周期计数	OPC UA 客户端负责向服务器发送 PublishRequest。在 PublishResponse 中，服务器会返回相应的通知包。生命周期计数表示在客户端发出多少次失败的 PublishRequest 后，服务器会删除订阅。括号中显示的是计算得出的时长（示例中 1200 乘以 1000 毫秒 = 20 分钟）。
保持连接计数	如果服务器未向客户端发送新的通知包，则不会返回任何数据。保持连接计数表示服务器在错过多少条信息后会向客户端发送一条空信息，以表明服务器仍然处于活动状态，订阅仍然有效。

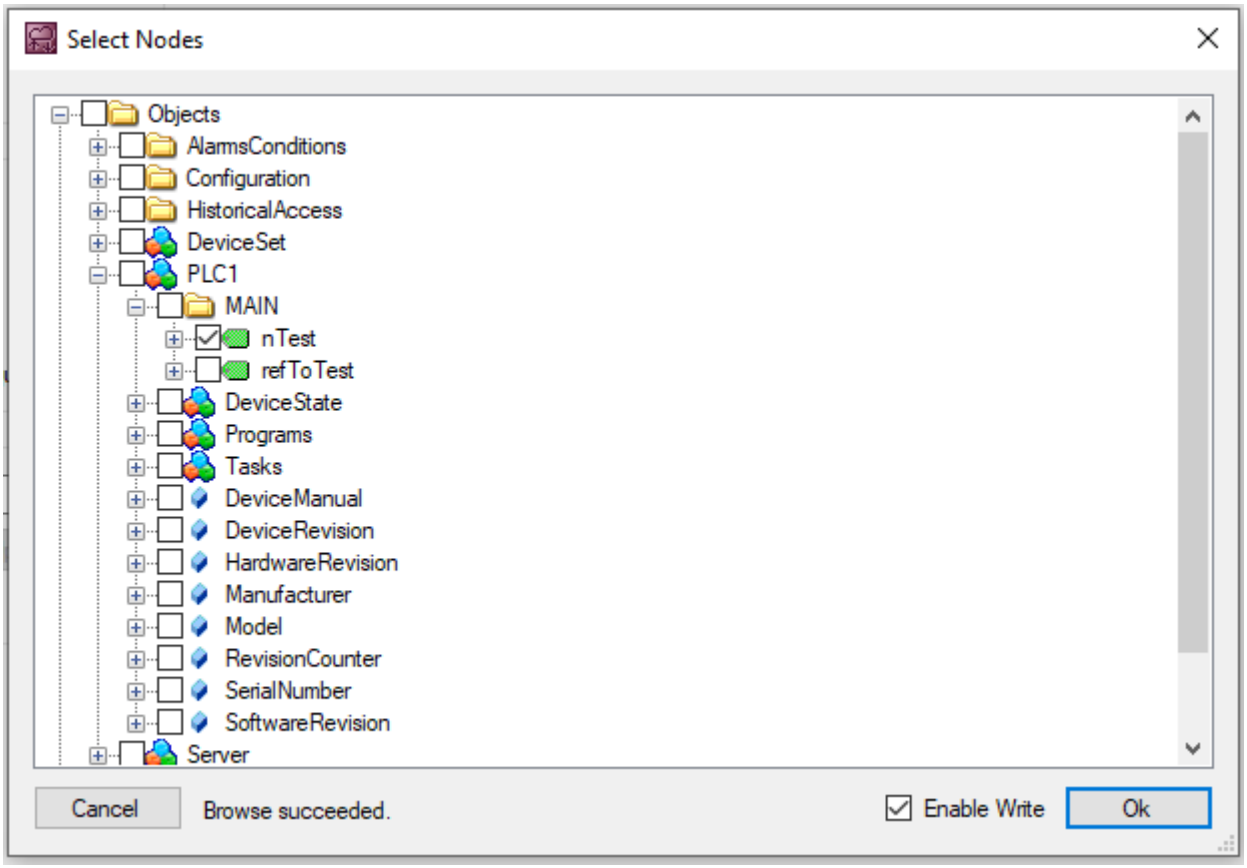
4.8 写入变量

要实现写入变量，必须满足几个条件：

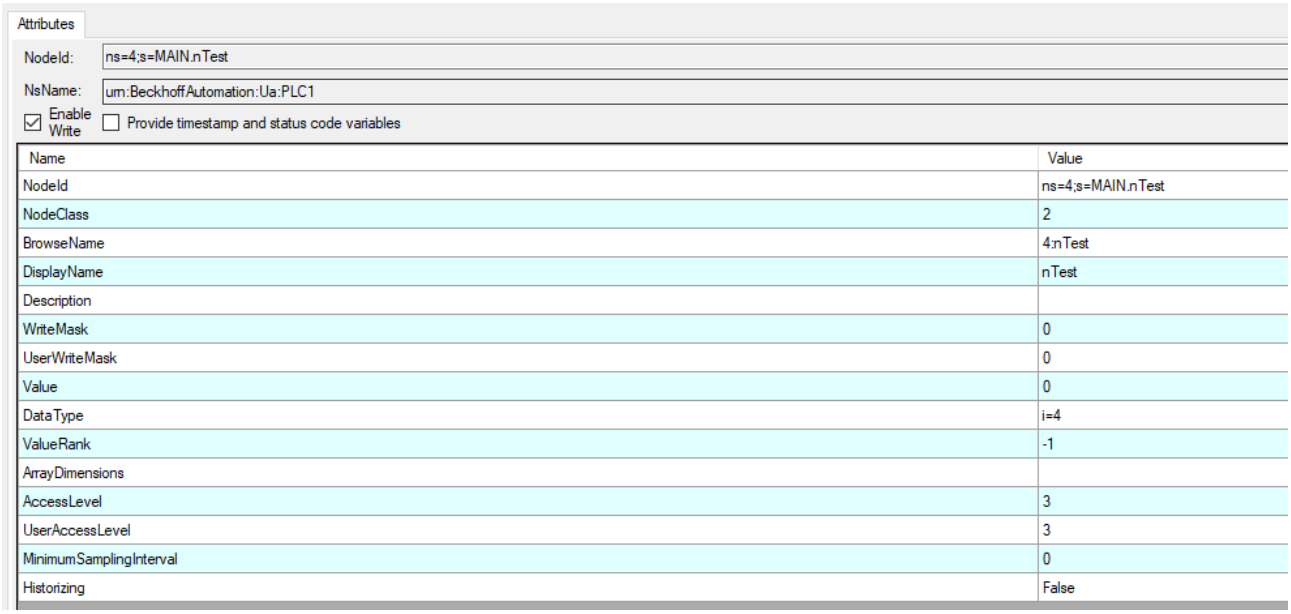
1. 必须为变量设置“启用写入”标志。可以在添加过程中通过 **Add Nodes**（添加节点）按钮完成，也可以在之后的变量参数设置中完成。
2. 在执行写入命令前，必须全局启用 I/O 客户端的“写入启用”输出。只有这样，才会生成写入命令。
3. 在“轮询”和“订阅”模式下，只有在 I/O 客户端内的值发生变化后才会写入。这对服务器重启尤为重要。服务器重启后，在这些模式下写入一次的值不会自动再次写入，因为另一个 OPC UA 客户端可能会在此期间写入一个新值，然后该值会被“旧”值覆盖。

为变量设置“启用写入”功能

为了要在过程映像中为变量不仅添加输入（读取）元素，还要添加输出（写入）元素，必须明确启用。例如，可以在添加变量时使用 **Add Nodes**“添加节点”对话框：

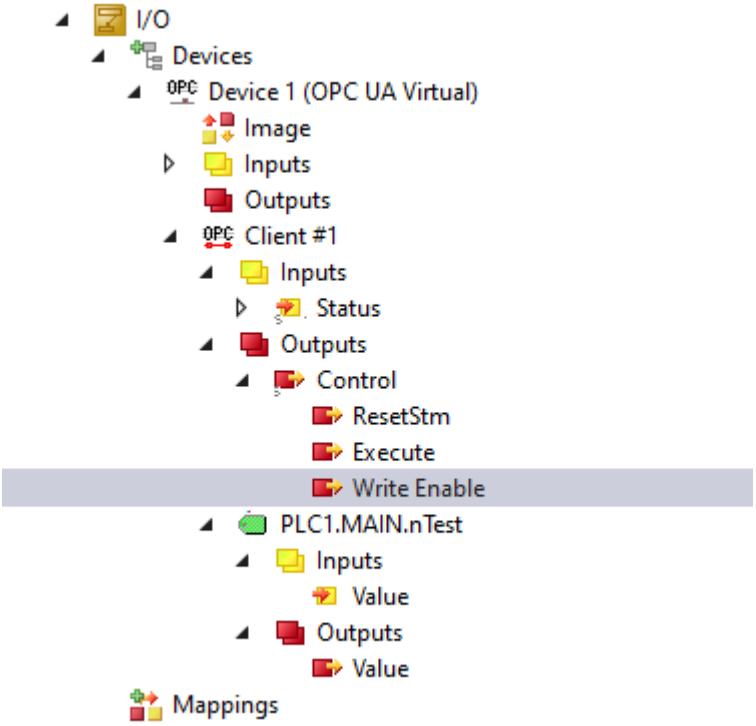


或者，也可以在稍后阶段通过过程映像中变量的配置参数来启用/禁用该设置。



全局启用写入访问

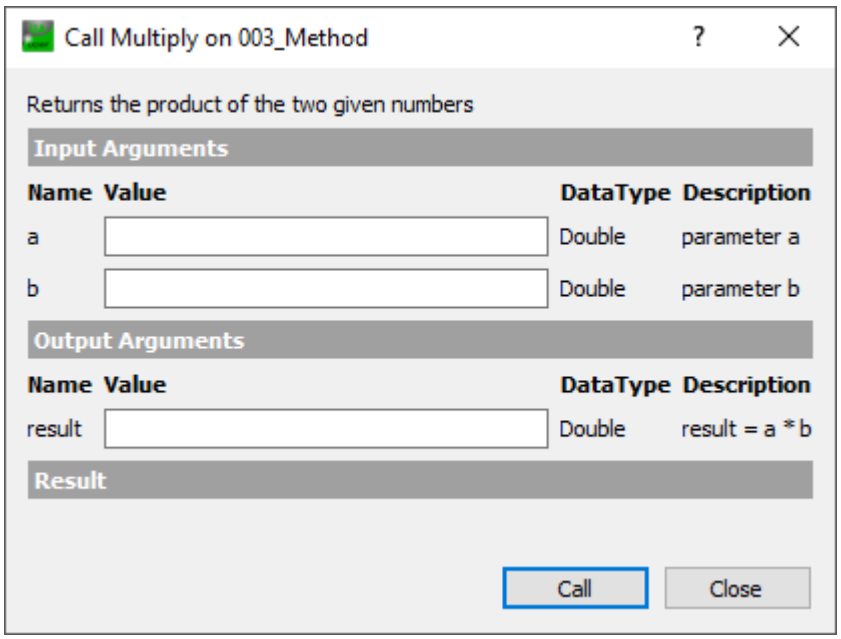
在发送写入命令之前，必须在全局范围内启用这些命令。可以通过设置 I/O 客户端的输出变量“写入启用”来实现：



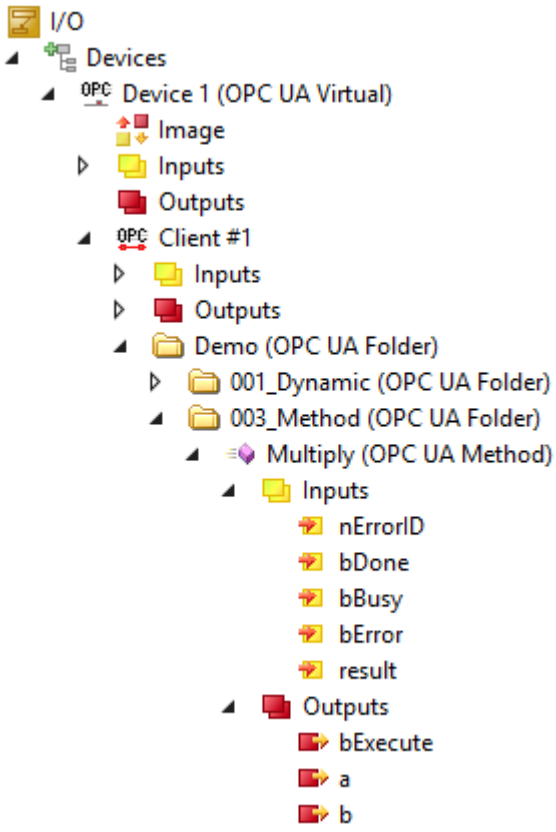
4.9 方法调用

TwinCAT OPC UA I/O 客户端支持调用服务器方法。您可以像任何其他变量一样为过程映像添加方法。这样，方法的“输入参数”便可以作为过程映像中的输出变量，而“输出参数”则作为输入变量添加。额外的输入和输出变量（如 bExecute、bBusy、bError）会添加到过程映像中，以便调用该方法。

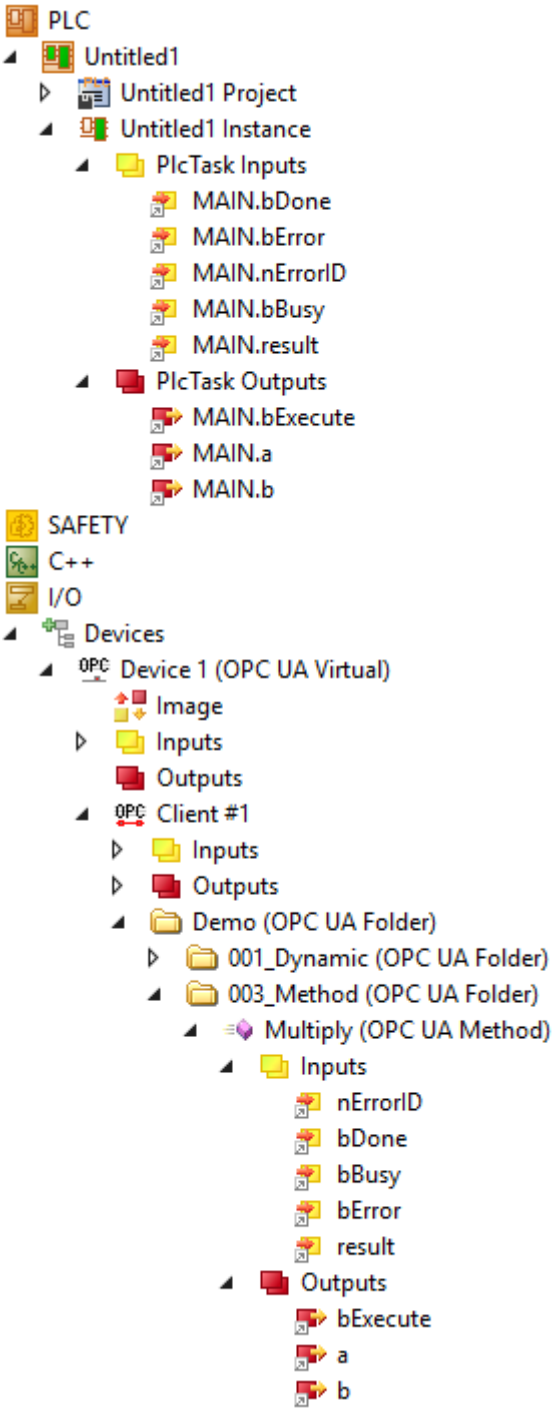
示例：服务器上的方法



示例：添加过程映像后的方法



然后，您便可以在输入/输出变量和 PLC 变量之间创建映射。



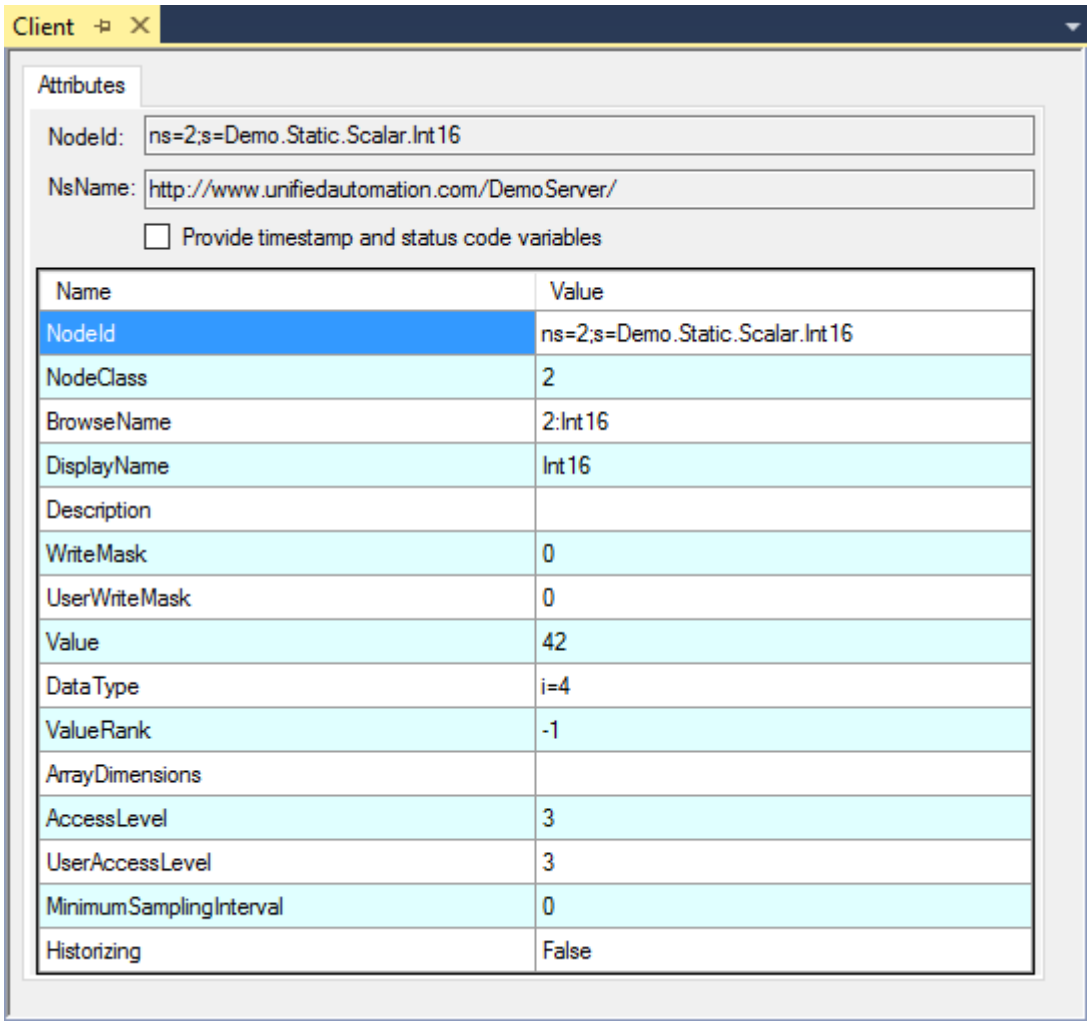
方法的调用

要调用方法，请将输出变量 bExecute 设为 TRUE。您可以通过输入变量 nErrorID、bDone、bBusy 和 bError 检查方法调用是否已完成以及是否成功。

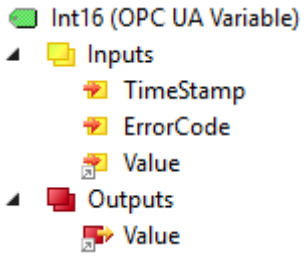
a	LREAL	3
b	LREAL	42
result	LREAL	126
bExecute	BOOL	TRUE
nErrorID	DINT	0
bDone	BOOL	TRUE
bError	BOOL	FALSE
bBusy	BOOL	FALSE

4.10 时间戳和 StatusCode

如果双击过程映像中的节点，您便可以看到 UA 属性的当前值，与打开窗口时的值一样。



使用复选框**提供时间戳和状态代码变量**可将更多可用于诊断的变量添加到过程映像中。



4.11 结构

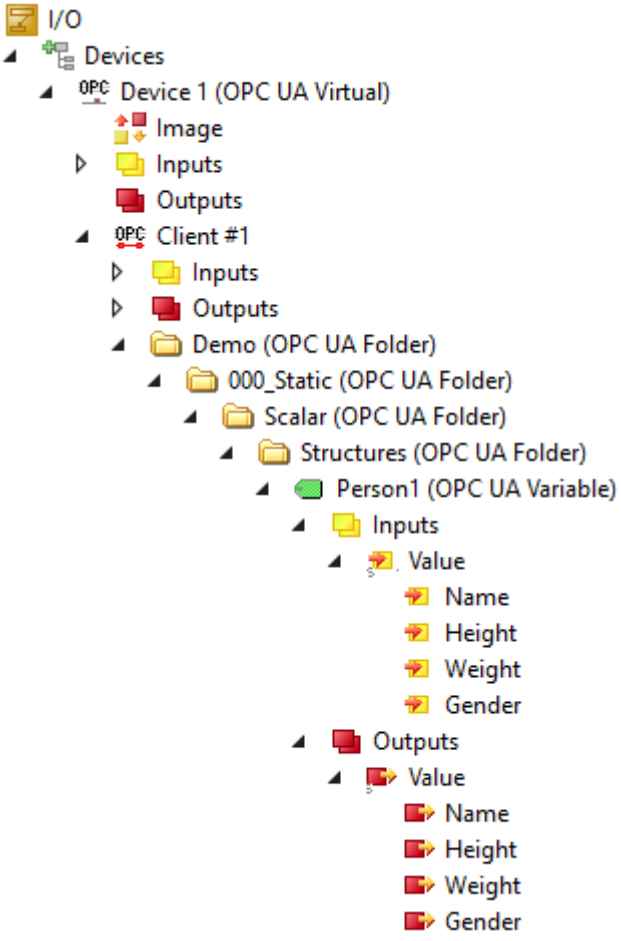
OPC UA I/O 客户端支持结构化数据类型 (StructuredType) 的读取/写入程序。您也可以像添加任何其他变量一样，将 StructuredType 添加到过程映像中。在过程映像中添加 StructuredType 时，会将待解析类型添加到 TwinCAT 类型系统中，这样，例如，PLC 应用程序便可以简单地使用该类型。

示例：服务器上的 StructuredType

Value	
SourceTimestamp	17.12.2021 12:18:50.450
SourcePicoseconds	0
ServerTimestamp	17.12.2021 12:18:52.887
ServerPicoseconds	0
StatusCode	Good (0x00000000)
Value	Person
Name	John Wayne
Height	193
Weight	77
Gender	0 (Male)
DataType	Person
NamespaceIndex	2
IdentifierType	Numeric
Identifier	543210

在本示例中，服务器包含一个结构化数据类型为“人”的节点，其中包含多个成员变量（姓名、身高、体重、性别）。

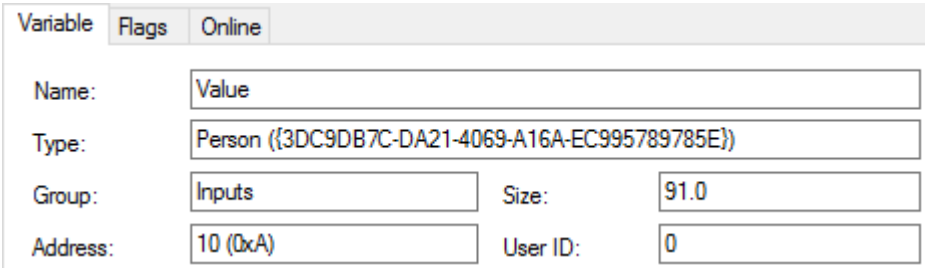
示例：过程映像中的 StructuredType



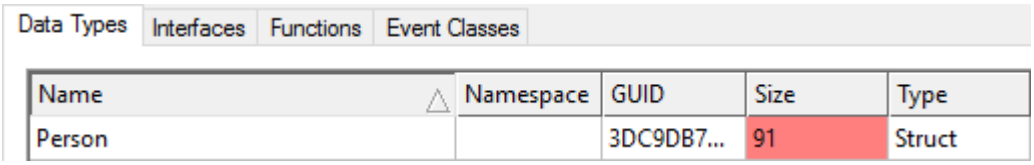
将节点添加到过程映像中后，过程映像就包含了该节点以及该类型的结构信息，例如，是否应读取或写入节点的各个成员变量。

TwinCAT 类型系统中的 StructuredType

会将数据类型添加到 TwinCAT 的类型系统中。这样，“值”树项便具有这种数据类型。



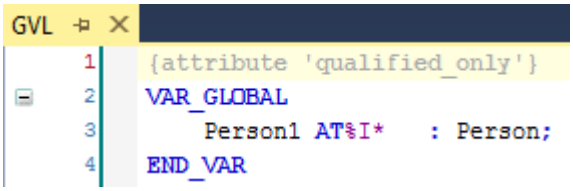
您还可以在 **SYSTEM（系统） > Type System（类型系统）** 下查看 TwinCAT 类型系统中的数据类型。

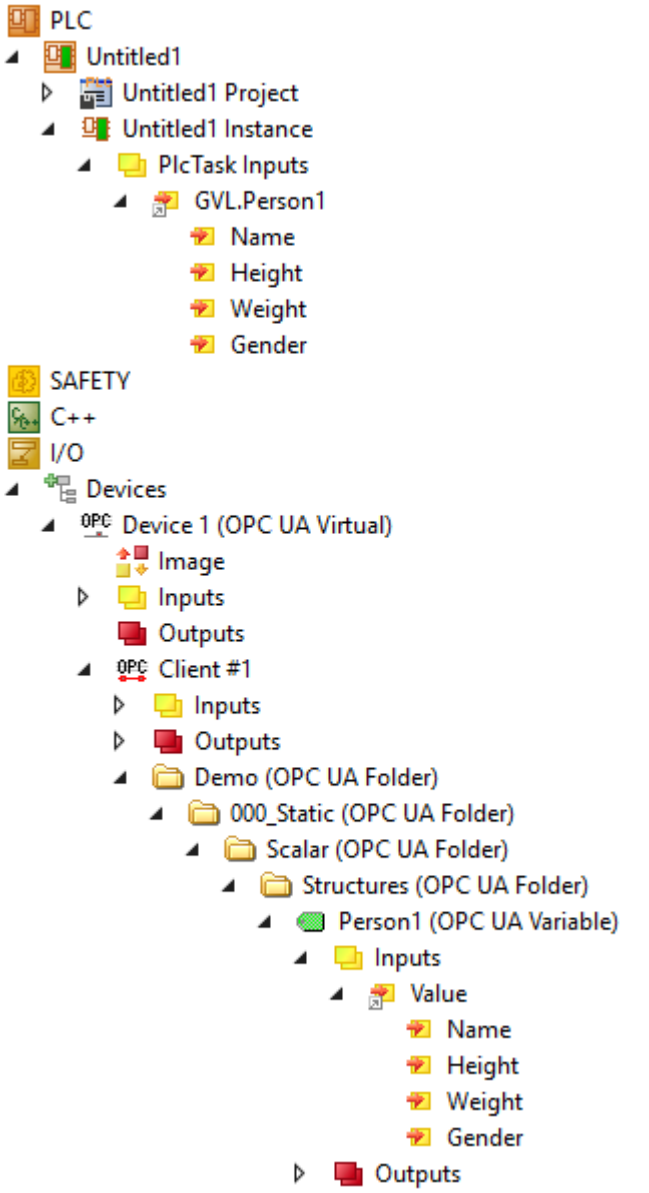


为了将该数据类型与其他数据类型区分开来，可以在 OPC UA 客户端的设置中添加一个前缀。

映射 StructuredType

由于已将每个 StructuredType 添加到 TwinCAT 类型系统中，因此变量的映射非常简单。创建该数据类型的输入/输出变量，随后创建映射。





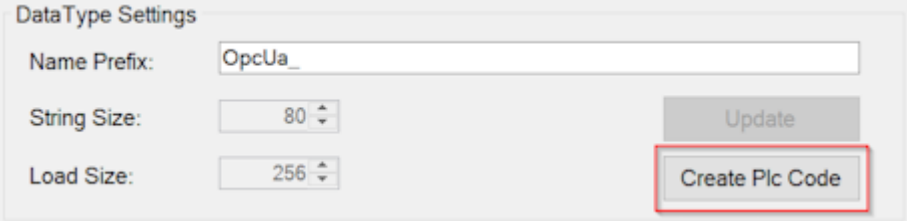
MAIN [Online]

TwinCAT_Project5.Untitled1.MAIN

Expression	Type	Value
Person1	OpcUa_Person	
Name	STRING	'John Wayne'
Height	UINT	193
Weight	REAL	77
Gender	OPCUA_GENDER	Male

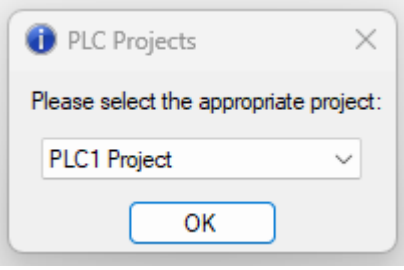
4.12 代码生成

借助自动生成代码功能，可以快速、轻松地生成 I/O 过程映像中配置的 OPC UA 节点的 PLC 变量，并自动与之关联。该功能可在 I/O 客户端的配置对话框中通过 **Create Plc Code**（创建 Plc 代码）按钮使用。该功能需要当前解决方案中已有的 PLC 项目。

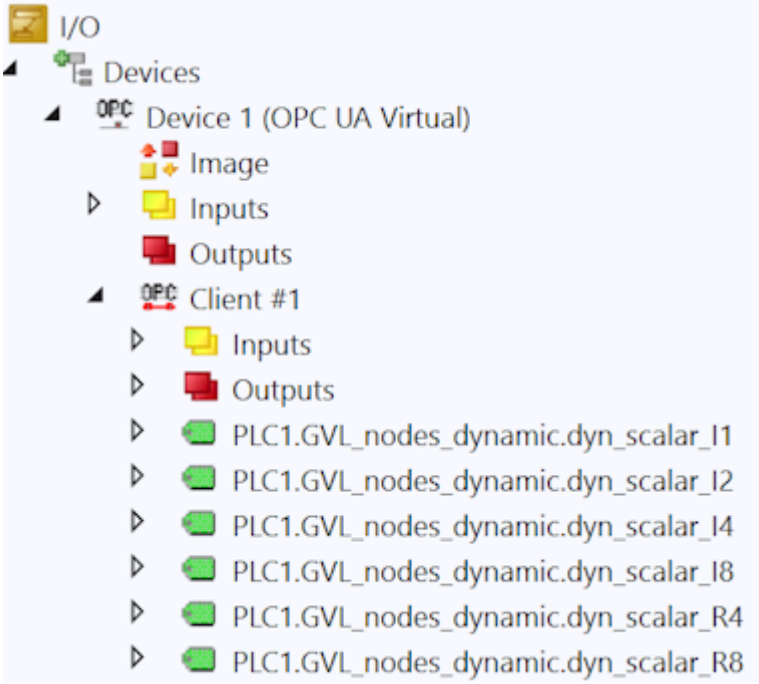


调用该功能后，将在 PLC 项目中创建一个带有 I/O 客户端名称的全新**全**局**变**量**列**表 (GVL)。随后，读取所有 OPC UA 节点，并在 GVL 中创建相应的变量。每个变量都会收到 TcLinkTo 属性，以便自动关联 I/O 过程映像中的相应变量。

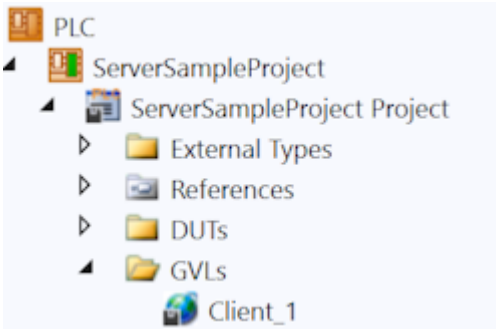
如果有多个 PLC 项目可用，则会出现选择对话框。在此选择对话框中，您可以指定应将 GVL 集成到哪个 PLC 项目中。



示例：在 TwinCAT XAE 的 I/O 部分创建了一个名为“客户端 1”的 TwinCAT OPC UA I/O 客户端，并在其中添加了来自服务器的各种 OPC UA 节点。



调用代码生成功能后，在（已经存在的）PLC 项目中创建了名为“客户端 1”的新 GVL。其中包含各个节点的相应 PLC 变量，然后通过 TcLinkTo 属性自动关联起来。



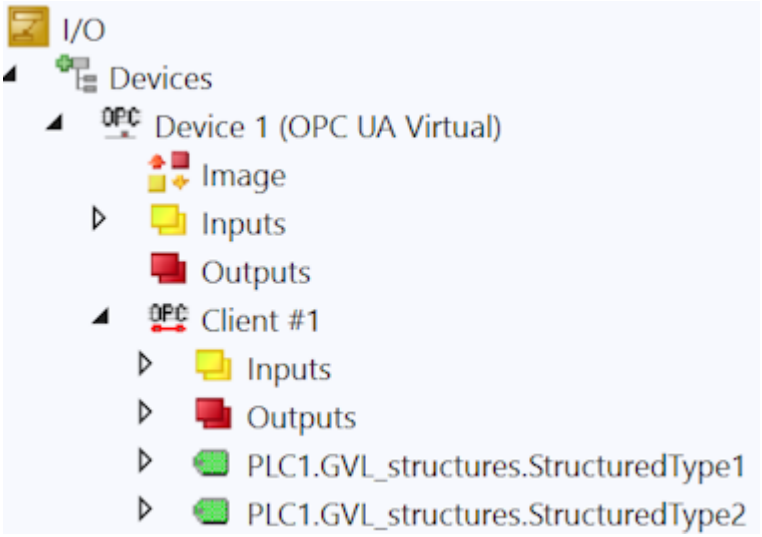
```
VAR_GLOBAL

    {attribute 'TcLinkTo' := 'TIID^Device 1 (OPC UA Virtual)^Client #1^PLC1.GVL_nodes_dynamic.dyn_scalar_I1^Inputs^Value'}
    PLC1_GVL_nodes_dynamic_dyn_scalar_I1_Client_1_Device_1 AT %I* : SINT;

    {attribute 'TcLinkTo' := 'TIID^Device 1 (OPC UA Virtual)^Client #1^PLC1.GVL_nodes_dynamic.dyn_scalar_I2^Inputs^Value'}
    PLC1_GVL_nodes_dynamic_dyn_scalar_I2_Client_1_Device_1 AT %I* : INT;

    {attribute 'TcLinkTo' := 'TIID^Device 1 (OPC UA Virtual)^Client #1^PLC1.GVL_nodes_dynamic.dyn_scalar_I4^Inputs^Value'}
    PLC1_GVL_nodes_dynamic_dyn_scalar_I4_Client_1_Device_1 AT %I* : DINT;

    {attribute 'TcLinkTo' := 'TIID^Device 1 (OPC UA Virtual)^Client #1^PLC1.GVL_nodes_dynamic.dyn_scalar_I8^Inputs^Value'}
    PLC1_GVL_nodes_dynamic_dyn_scalar_I8_Client_1_Device_1 AT %I* : LINT;

    {attribute 'TcLinkTo' := 'TIID^Device 1 (OPC UA Virtual)^Client #1^PLC1.GVL_nodes_dynamic.dyn_scalar_R4^Inputs^Value'}
    PLC1_GVL_nodes_dynamic_dyn_scalar_R4_Client_1_Device_1 AT %I* : REAL;

    {attribute 'TcLinkTo' := 'TIID^Device 1 (OPC UA Virtual)^Client #1^PLC1.GVL_nodes_dynamic.dyn_scalar_R8^Inputs^Value'}
    PLC1_GVL_nodes_dynamic_dyn_scalar_R8_Client_1_Device_1 AT %I* : LREAL;

END_VAR
```

此外，相应 I/O 客户端的控制变量也会作为变量在 GVL 中创建和关联（上方截图中未显示）。

结构代码生成

代码生成也会考虑到代表所谓 StructuredDataType 的 OPC UA 节点，并在 GVL 中创建相应的变量。由于 StructuredDataType 是在 TwinCAT 类型系统中作为本地数据类型创建的，因此可以像处理普通结构一样对其进行处理。

示例：在 I/O 客户端的过程映像中添加了来自服务器的两个 StructuredDataType。服务器上 StructuredDataType 的数据类型是 ST_Complex1 和 ST_Complex2（在下方截图中看不到）。



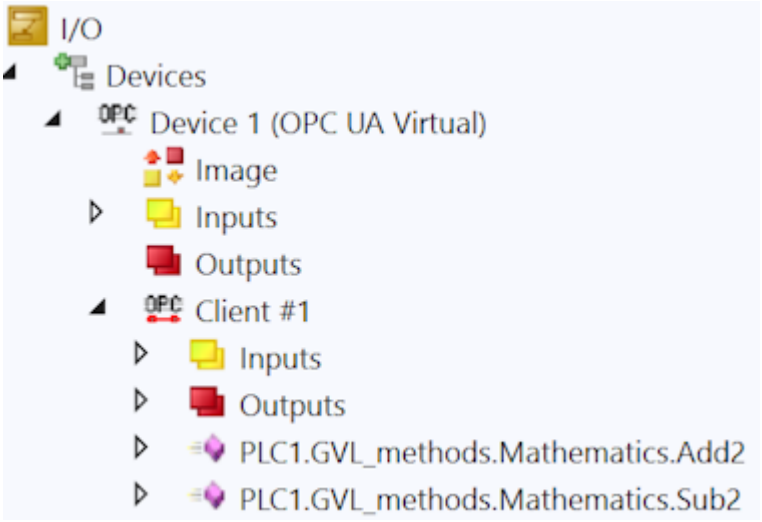
代码生成功能现在已创建一个相应的 GVL，其中包含两个来自各自自动生成的 TwinCAT 数据类型的变量，它们与各自的 StructuredDataType 相对应。

```
VAR_GLOBAL

    {attribute 'TcLinkTo' := 'TIID^Device 1 (OPC UA Virtual)^Client #1^PLC1.GVL_structures.StructuredType1^Inputs^Value'}
    PLC1_GVL_structures_StructuredType1_Client_1_Device_1 AT %I* : OpcUa_ST_Complex_1;

    {attribute 'TcLinkTo' := 'TIID^Device 1 (OPC UA Virtual)^Client #1^PLC1.GVL_structures.StructuredType2^Inputs^Value'}
    PLC1_GVL_structures_StructuredType2_Client_1_Device_1 AT %I* : OpcUa_ST_Complex_2;

END_VAR
```

方法代码生成

OPC UA 方法有输入和输出参数，这些参数会相应地传入方法或从中传出。此外，方法是独立的调用；必须由客户端显式启动。这种方法会相应地映射到方法处 I/O 客户端的过程映像中，在代码生成过程中也会考虑到这一点。对于方法而言，与普通变量或结构不同的是，要创建一个单独的功能块，然后在 GVL 中引用该功能块。

示例：在 I/O 客户端的过程映像中添加了来自服务器的两个方法。



现在，代码生成功能已创建相应的 GVL 以及两个功能块，分别代表相应方法的输入/输出和控制变量。

```
VAR_GLOBAL

    {attribute 'TcLinkTo' := '.nErrorID:=TIID^Device 1 (OPC UA Virtual)^Client #1^PLC1.GVL_methods.Mathematics.Add2^Inputs^nErrorID;
                             .bDone:=TIID^Device 1 (OPC UA Virtual)^Client #1^PLC1.GVL_methods.Mathematics.Add2^Inputs^bDone;
                             .bBusy:=TIID^Device 1 (OPC UA Virtual)^Client #1^PLC1.GVL_methods.Mathematics.Add2^Inputs^bBusy;
                             .bError:=TIID^Device 1 (OPC UA Virtual)^Client #1^PLC1.GVL_methods.Mathematics.Add2^Inputs^bError;
                             .ReturnValue:=TIID^Device 1 (OPC UA Virtual)^Client #1^PLC1.GVL_methods.Mathematics.Add2^Inputs^ReturnValue;
                             .bExecute:=TIID^Device 1 (OPC UA Virtual)^Client #1^PLC1.GVL_methods.Mathematics.Add2^Outputs^bExecute;
                             .a:=TIID^Device 1 (OPC UA Virtual)^Client #1^PLC1.GVL_methods.Mathematics.Add2^Outputs^a;
                             .b:=TIID^Device 1 (OPC UA Virtual)^Client #1^PLC1.GVL_methods.Mathematics.Add2^Outputs^b'}
    PLC1_GVL_methods_Mathematics_Add2_Client_1_Device_1 : FB_PLC1_GVL_methods_Mathematics_Add2_Client_1_Device_1;

    {attribute 'TcLinkTo' := '.nErrorID:=TIID^Device 1 (OPC UA Virtual)^Client #1^PLC1.GVL_methods.Mathematics.Sub2^Inputs^nErrorID;
                             .bDone:=TIID^Device 1 (OPC UA Virtual)^Client #1^PLC1.GVL_methods.Mathematics.Sub2^Inputs^bDone;
                             .bBusy:=TIID^Device 1 (OPC UA Virtual)^Client #1^PLC1.GVL_methods.Mathematics.Sub2^Inputs^bBusy;
                             .bError:=TIID^Device 1 (OPC UA Virtual)^Client #1^PLC1.GVL_methods.Mathematics.Sub2^Inputs^bError;
                             .ReturnValue:=TIID^Device 1 (OPC UA Virtual)^Client #1^PLC1.GVL_methods.Mathematics.Sub2^Inputs^ReturnValue;
                             .bExecute:=TIID^Device 1 (OPC UA Virtual)^Client #1^PLC1.GVL_methods.Mathematics.Sub2^Outputs^bExecute;
                             .a:=TIID^Device 1 (OPC UA Virtual)^Client #1^PLC1.GVL_methods.Mathematics.Sub2^Outputs^a;
                             .b:=TIID^Device 1 (OPC UA Virtual)^Client #1^PLC1.GVL_methods.Mathematics.Sub2^Outputs^b'}
    PLC1_GVL_methods_Mathematics_Sub2_Client_1_Device_1 : FB_PLC1_GVL_methods_Mathematics_Sub2_Client_1_Device_1;

END_VAR
```

```
FUNCTION_BLOCK FB_PLC1_GVL_methods_Mathematics_Add2_Client_1_Device_1
VAR_INPUT
    bExecute AT %Q* : BOOL;
    a AT %Q* : LREAL;
    b AT %Q* : LREAL;
END_VAR
VAR_OUTPUT
    nErrorID AT %I* : DINT;
    bDone AT %I* : BOOL;
    bBusy AT %I* : BOOL;
    bError AT %I* : BOOL;
    ReturnValue AT %I* : LREAL;
END_VAR

FUNCTION_BLOCK FB_PLC1_GVL_methods_Mathematics_Sub2_Client_1_Device_1
VAR_INPUT
    bExecute AT %Q* : BOOL;
    a AT %Q* : LREAL;
    b AT %Q* : LREAL;
END_VAR
VAR_OUTPUT
    nErrorID AT %I* : DINT;
    bDone AT %I* : BOOL;
    bBusy AT %I* : BOOL;
    bError AT %I* : BOOL;
    ReturnValue AT %I* : LREAL;
END_VAR
```

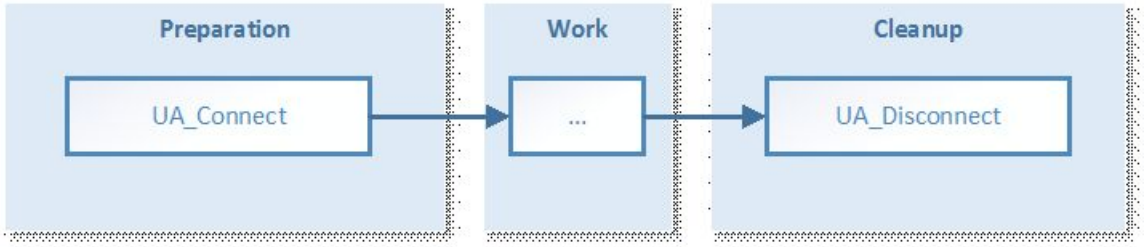
4.13 PLCopen 功能块

TwinCAT OPC UA 客户端提供多种选择，可从控制逻辑直接与一个或多个 OPC UA 服务器通信。一方面是 TwinCAT I/O 设备，它提供了一个基于映射的简单界面。另一方面，PLCopen 提供标准化功能块，可用于直接从 PLC 逻辑启动与 OPC UA 服务器的连接。下文将详细介绍如何处理这些功能块。本文由以下章节组成：

- 工作流程
- 确定通信参数
- 建立连接
- 读取变量
- 写入变量
- 调用方法

工作流程

使用 PLCopen 功能块的一般工作流程示意图如下：



在准备阶段，设置通信参数并建立与服务器的连接。然后执行所需功能（读取、写入、方法调用），接着断开通信连接。

确定通信参数

一般来说，图形 OPC UA 客户端用于确定必须与 PLC 功能块一起使用的节点属性或方法，例如：

- NodeID
- NamespaceIndex 和相应的 NamespaceURI
- DataType
- MethodNodeID 和 ObjectNodeID

以下文档以通用 OPC UA 客户端 UA 专家为例。该客户端可通过统一自动化网页购买：www.unified-automation.com。

节点具有以下三个属性，它们构成了所谓的 NodeID：

- NamespaceIndex：节点所在的命名空间，如 PLC 运行时
- 标识符：节点在其命名空间中的唯一标识符
- IdentifierType：节点类型：字符串、全局唯一标识符和数字

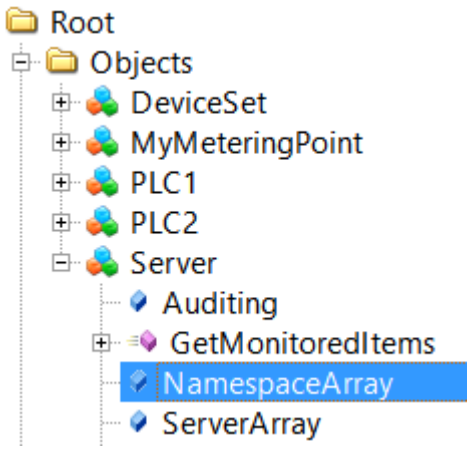
这些属性代表所谓的 NodeID（OPC UA 服务器上的节点表示），许多后续功能块都需要这些属性。

借助 UA 专家软件，您只需与 OPC UA 服务器建立连接，并浏览到所需节点，即可确定节点的属性。然后，属性会在“属性”面板中显示出来，例如

NodeId	NodeId
NamespaceIndex	5
IdentifierType	String
Identifier	MAIN.nCounter

根据 OPC UA 规范，NamespaceIndex 可以是动态生成的值。因此，在检测到节点句柄之前，OPC UA 客户端务必使用相应的命名空间 URI 来解析 NamespaceIndex。

使用功能块 UA_GetNamespaceIndex [▶ 81] 获取 NamespaceURI 的 NamespaceIndex。通过建立与 OPC UA 服务器的连接并浏览 NamespaceArray 节点，可以借助 UA Expert 确定所需的 NamespaceURI。



该节点包含在 OPC UA 服务器上注册的所有命名空间相关信息。
例如，在“属性”面板中可以看到相应的命名空间 URI：

Value	
SourceTimestamp	16.02.2015 08:56:06.350
ServerTimestamp	16.02.2015 09:31:01.945
SourcePicoseconds	0
ServerPicoseconds	0
Value	String Array[9]
[0]	http://opcfoundation.org/UA/
[1]	urn:SvenG-NB04:BeckhoffAutomation:TcOpcUaServer:1
[2]	http://opcfoundation.org/UA/DI/
[3]	http://PLCopen.org/OpcUa/IEC61131-3/
[4]	urn://SVENG-NB04/BeckhoffAutomation/Ua/Typesystem
[5]	urn://SVENG-NB04/BeckhoffAutomation/Ua/PLC1
[6]	urn://SVENG-NB04/BeckhoffAutomation/Ua/PLC2
[7]	http://www.opcfoundation.org/Energy/DataAcquisition/
[8]	http://Beckhoff.com/TwinCAT/TF6100/Server/Configuration

上节显示的是 NodeID 的示例，其中的命名空间索引为 5。根据图中所示的 NamespaceArray，相应的 NamespaceURI 为 urn://SVENG-NB04/BeckhoffAutomation/Ua/PLC1。该 URI 现在可用于功能块 UA_GetNamespaceIndex。OPC UA 服务器可确保 URI 始终保持不变，即使在重启后也是如此。

注意正确的 NamespaceIndex

由于显示的 NamespaceIndex 可能会发生变化，因此应始终将 NamespaceURI 与功能块 UA_GetNamespaceIndex 结合使用，以便日后与其他功能块（如 UA_Read [▶ 92]、UA_Write [▶ 94]）一起使用，以解析正确的 NamespaceIndex。

DataType

需要节点的数据类型，才能知道需要使用哪种 PLC 数据类型能为节点分配读取值或写入值。借助 UA Expert，您只需与 OPC UA 服务器建立连接并浏览到所需节点，即可确定节点的数据类型。
例如，会在“属性”面板中看到数据类型：

DataType	Int16
NamespaceIndex	0
IdentifierType	Numeric
Identifier	4

在这种情况下，数据类型 (DataType) 为“Int16”。必须将其分配给 PLC 中的等效数据类型，如“INT”。

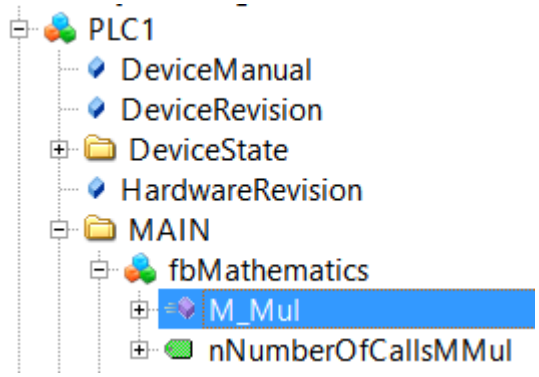
MethodNodeID 和 ObjectNodeID

从 OPC UA 命名空间调用方法时，如果使用功能块 UA_MethodGetHandle [▶ 86] 获取方法句柄，则需要两个标识符：

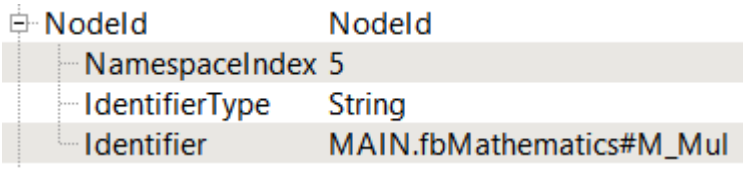
- ObjectNodeID：标识包含方法的 UA 对象。
- MethodNodeID：标识方法本身。

借助 UA Expert，您只需与 OPC UA 服务器建立连接，并浏览到所需方法或包含该方法的所需 UA 对象，即可确定这两个 NodeID。

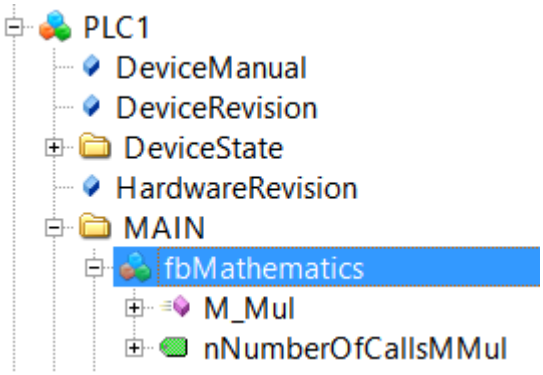
示例方法 M_Mul：



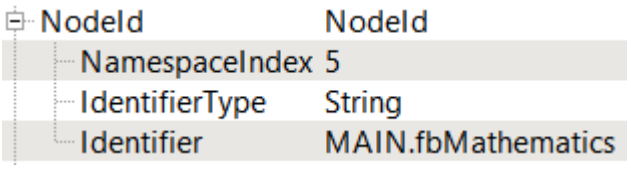
然后，会在“属性”面板中看到方法标识符。



示例对象 fbMathematics：



然后，会在“属性”面板中看到对象标识符。



建立连接

下一节将介绍如何使用功能块 TcX_PLCopen_OpcUa 建立与本地或远程 OPC UA 服务器的连接。这种连接随后可用于调用其他功能，如读取或写入节点，或调用方法。

要建立与 OPC UA 服务器的连接并随后中断会话，需要使用以下功能块：UA_Connect [▶ 78]、UA_Disconnect [▶ 80]。

首先请阅读如何确定通信参数一节，以便更好地了解某些 UA 功能（例如，如何确定 NodeIdentifier）。

要与本地或远程 OPC UA 服务器建立连接，功能块 UA_Connect 需要以下信息：

- 服务器 URL
- 会话连接信息

服务器 URL 基本由前缀、主机名和端口组成。前缀描述的是连接应使用的 OPC UA 传输协议，例如“opc.tcp://”表示二进制 TCP 连接（默认）。主机名或 IP 地址部分描述的是 OPC UA 目标服务器的地址信息，如“192.168.1.1”或“CX-12345”。端口号是 OPC UA 服务器的目标端口，如“4840”。服务器 URL 可以如下所示：opc.tcp://CX-12345:4840。

声明：

```
(* Declarations for UA_Connect *)
fbUA_Connect : UA_Connect;
SessionConnectInfo : ST_UASessionConnectInfo;
nConnectionHdl : DWORD;

(* Declarations for UA_Disconnect *)
fbUA_Disconnect : UA_Disconnect;

(* Declarations for state machine and output handling *)
iState : INT;
bDone : BOOL;
bBusy : BOOL;
bError : BOOL;
nErrorID : DWORD;
```

实现：

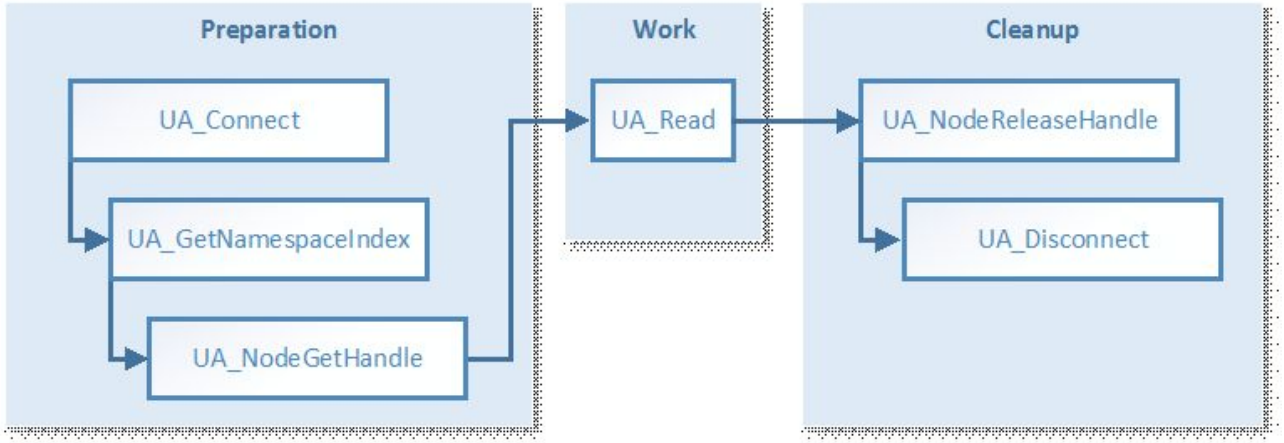
```
CASE iState OF

  0:
      bError := FALSE;
      nErrorID := 0;
      SessionConnectInfo.tConnectTimeout := T#1M;
      SessionConnectInfo.tSessionTimeout := T#1M;
      SessionConnectInfo.sApplicationName := "";
      SessionConnectInfo.sApplicationUri := "";
      SessionConnectInfo.eSecurityMode := eUASecurityMsgMode_None;
      SessionConnectInfo.eSecurityPolicyUri := eUASecurityPolicy_None;
      SessionConnectInfo.eTransportProfileUri := eUATransportProfileUri_UATcp;
      stNodeAddInfo.nIndexRangeCount := nIndexRangeCount;
      stNodeAddInfo.stIndexRange := stIndexRange;
      iState := iState + 1;

  1:
    fbUA_Connect(
      Execute := TRUE,
      ServerURL := "opc.tcp://192.168.1.1:4840",
      SessionConnectInfo := SessionConnectInfo,
      Timeout := T#5S,
      ConnectionHdl => nConnectionHdl);
    IF NOT fbUA_Connect.Busy THEN
      fbUA_Connect(Execute := FALSE);
      IF NOT fbUA_Connect.Error THEN
        iState := iState + 1;
      ELSE
        bError := TRUE;
        nErrorID := fbUA_Connect.ErrorID;
        nConnectionHdl := 0;
        iState := 0;
      END_IF
    END_IF

  2:
    fbUA_Disconnect(
      Execute := TRUE,
      ConnectionHdl := nConnectionHdl);

    IF NOT fbUA_Disconnect.Busy THEN
      fbUA_Disconnect(Execute := FALSE);
      IF NOT fbUA_Disconnect.Error THEN
        iState := 0;
      ELSE
        bError := TRUE;
        nErrorID := fbUA_Disconnect.ErrorID;
        iState := 0;
        nConnectionHdl := 0;
      END_IF
    END_IF

END_CASE
```

读取变量

下一节将介绍如何使用功能块 TcX_PLCopen_OpcUa，从本地或远程 OPC UA 服务器读取 OPC UA 节点。要建立与 OPC UA 服务器的连接、读取 UA 节点以及随后中断会话，需要使用以下功能块：UA_Connect [▶ 78]、UA_GetNamespaceIndex [▶ 81]、UA_NodeGetHandle [▶ 88]、UA_Read [▶ 92]、UA_NodeReleaseHandle [▶ 90]、UA_Disconnect [▶ 80]。

每个 TwinCAT OPC UA 客户端的示意性工作流程可分为三个不同阶段：“准备”、“工作”和“清理”。

本节描述的用例可以直观地显示如下：



- 要建立与本地或远程 OPC UA 服务器的连接，功能块 UA_Connect 需要以下信息（另请参见如何建立连接）：
  - 服务器 URL
  - 会话连接信息
- 功能块 UA_GetNamespaceIndex 需要一个连接句柄（来自 UA_Connect）和一个 NamespaceURI，以便解析为一个 NamespaceIndex，随后由 UA_NodeGetHandle 用来捕获节点句柄（另请参见 ）。
- 功能块 UA_NodeGetHandle 需要一个连接句柄（来自 UA_Connect）和 NodeID（来自 ST_UANodeID）来捕获节点句柄（另请参见 ）。
- 功能块 UA_Read 需要一个连接句柄（来自 UA_Connect）、一个节点句柄（来自 UA_NodeGetHandle）和一个目标变量指针（保存读取值的位置）。确保目标变量的数据类型正确（另请参见 ）。
- 功能块 UA_NodeReleaseHandle 需要一个连接句柄（来自 UA_Connect）和一个节点句柄（来自 UA_NodeGetHandle）。

声明：

```
(* Declarations for UA_GetNamespaceIndex *)
fbUA_GetNamespaceIndex : UA_GetNamespaceIndex;
nNamespaceIndex : UINT;

(* Declarations for UA_NodeGetHandle *)
fbUA_NodeGetHandle : UA_NodeGetHandle;
NodeID : ST_UANodeID;
nNodeHdl : DWORD;

(* Declarations for UA_Read *)
fbUA_Read : UA_Read;
stIndexRange : ARRAY [1..nMaxIndexRange] OF ST_UAIndexRange;
nIndexRangeCount : UINT;
stNodeAddInfo : ST_UANodeAdditionalInfo;
sNodeIdentifier : STRING(MAX_STRING_LENGTH) := 'MAIN.nCounter';
nReadData : INT;
cbDataRead : UDINT;

(* Declarations for UA_NodeReleaseHandle *)
fbUA_NodeReleaseHandle : UA_NodeReleaseHandle;
```

实现：

```
CASE iState OF
  0:
    [...]

  2: (* GetNS Index *)
    fbUA_GetNamespaceIndex(
```

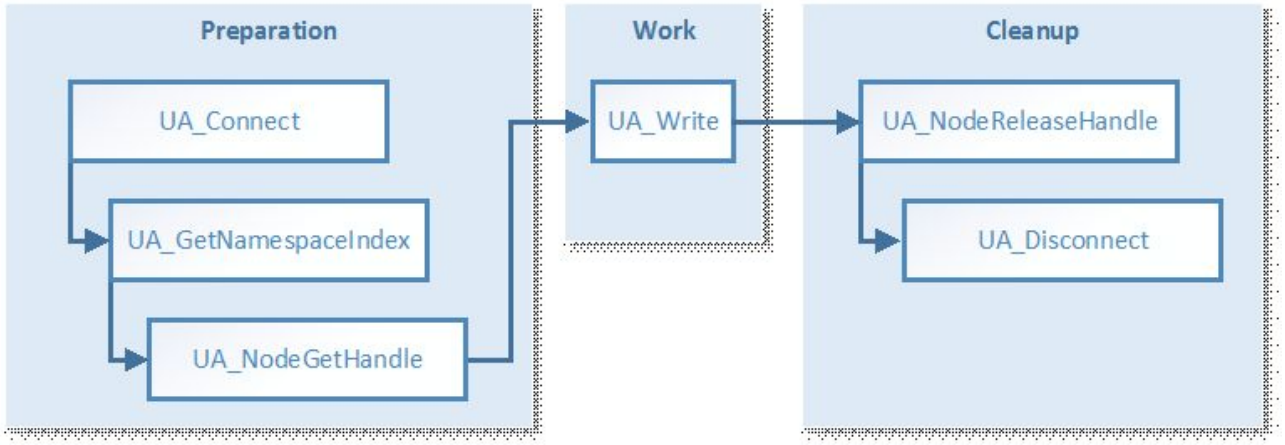
```
      Execute := TRUE,
      ConnectionHdl := nConnectionHdl,
      NamespaceUri := sNamespaceUri,
      NamespaceIndex => nNamespaceIndex
      );
    IF NOT fbUA_GetNamespaceIndex.Busy THEN
      fbUA_GetNamespaceIndex(Execute := FALSE);
      IF NOT fbUA_GetNamespaceIndex.Error THEN
        iState := iState + 1;
      ELSE
        bError := TRUE;
        nErrorID := fbUA_GetNamespaceIndex.ErrorID;
        iState := 6;
      END_IF
    END_IF

  3: (* UA_NodeGetHandle *)
    NodeID.eIdentifierType := eUAIdentifierType_String;
    NodeID.nNamespaceIndex := nNamespaceIndex;
    NodeID.sIdentifier := sNodeIdentifier;
    fbUA_NodeGetHandle(
      Execute := TRUE,
      ConnectionHdl := nConnectionHdl,
      NodeID := NodeID,
      NodeHdl => nNodeHdl);
    IF NOT fbUA_NodeGetHandle.Busy THEN
      fbUA_NodeGetHandle(Execute := FALSE);
      IF NOT fbUA_NodeGetHandle.Error THEN
        iState := iState + 1;
      ELSE
        bError := TRUE;
        nErrorID := fbUA_NodeGetHandle.ErrorID;
        iState := 6;
      END_IF
    END_IF

  4: (* UA_Read *)
    fbUA_Read(
      Execute := TRUE,
      ConnectionHdl := nConnectionHdl,
      NodeHdl := nNodeHdl,
      cbData := SIZEOF(nReadData),
      stNodeAddInfo := stNodeAddInfo,
      pVariable := ADR(nReadData));
    IF NOT fbUA_Read.Busy THEN
      fbUA_Read( Execute := FALSE, cbData_R => cbDataRead);
      IF NOT fbUA_Read.Error THEN
        iState := iState + 1;
      ELSE
        bError := TRUE;
        nErrorID := fbUA_Read.ErrorID;
        iState := 6;
      END_IF
    END_IF

  5: (* Release Node Handle *)
    fbUA_NodeReleaseHandle(
      Execute := TRUE,
      ConnectionHdl := nConnectionHdl,
      NodeHdl := nNodeHdl);
    IF NOT fbUA_NodeReleaseHandle.Busy THEN
      fbUA_NodeReleaseHandle(Execute := FALSE);
      IF NOT fbUA_NodeReleaseHandle.Error THEN
        iState := iState + 1;
      ELSE
        bError := TRUE;
        nErrorID := fbUA_NodeReleaseHandle.ErrorID;
        iState := 6;
      END_IF
    END_IF

  6:
    [...]

END_CASE
```

写入变量

下一节将介绍如何使用功能块 TcX_PLCopen_OpcUa，从本地或远程 OPC UA 服务器向 OPC UA 节点写入值。要建立与 OPC UA 服务器的连接、写入 UA 节点并随后中断会话，需要使用以下功能块：UA_Connect [▶ 78],UA_GetNamespaceIndex [▶ 81],UA_NodeGetHandle [▶ 88],UA_Write [▶ 94],UA_NodeReleaseHandle [▶ 90],UA_Disconnect [▶ 80]。

每个 TwinCAT OPC UA 客户端的示意性工作流程可分为三个不同阶段：“准备”、“工作”和“清理”。

本节描述的用例可以直观地显示如下：



- 要与本地或远程 OPC UA 服务器建立连接，功能块 UA_Connect 需要以下信息（另请参见）：
  - 服务器 URL
  - 会话连接信息
- 功能块 UA_GetNamespaceIndex 需要一个连接句柄（来自 UA_Connect）和一个 NamespaceURI，以便解析为一个 NamespaceIndex，随后由 UA_NodeGetHandle 用来捕获节点句柄（另请参见）。
- 功能块 UA_NodeGetHandle 需要一个连接句柄（来自 UA_Connect）和 NodeID（来自 ST_UANodeID）来捕获节点句柄（另请参见）。
- 功能块 UA_Write 需要一个连接句柄（来自 UA_Connect）、一个节点句柄（来自 UA_NodeGetHandle）和一个指向包含要写入值的变量的指针。确保目标变量的数据类型正确（另请参见）。
- 功能块 UA_NodeReleaseHandle 需要一个连接句柄（来自 UA_Connect）和一个节点句柄（来自 UA_NodeGetHandle）。

声明：

```
(* Declarations for UA_GetNamespaceIndex *)
fbUA_GetNamespaceIndex : UA_GetNamespaceIndex;
nNamespaceIndex : UINT;

(* Declarations for UA_NodeGetHandle *)
fbUA_NodeGetHandle : UA_NodeGetHandle;
NodeID : ST_UANodeID;
nNodeHdl : DWORD;

(* Declarations for UA_Write *)
fbUA_Write : UA_Write;
stIndexRange : ARRAY [1..nMaxIndexRange] OF ST_UAIndexRange;
nIndexRangeCount : UINT;
stNodeAddInfo : ST_UANodeAdditionalInfo;
sNodeIdentifier: STRING(MAX_STRING_LENGTH) := 'MAIN.nNumber';
nWriteData: INT := 42;

(* Declarations for UA_NodeReleaseHandle *)
fbUA_NodeReleaseHandle : UA_NodeReleaseHandle;
```

实现：

```
CASE iState OF
  0:
    [...]

  2: (* GetNS Index *)
    fbUA_GetNamespaceIndex(
      Execute := TRUE,
      ConnectionHdl := nConnectionHdl,
      NamespaceUri := sNamespaceUri,
      NamespaceIndex => nNamespaceIndex
      );
    IF NOT fbUA_GetNamespaceIndex.Busy THEN
      fbUA_GetNamespaceIndex(Execute := FALSE);
      IF NOT fbUA_GetNamespaceIndex.Error THEN
        iState := iState + 1;
      ELSE
        bError := TRUE;
        nErrorID := fbUA_GetNamespaceIndex.ErrorID;
        iState := 6;
      END_IF
    END_IF

  3: (* UA_NodeGetHandle *)
```

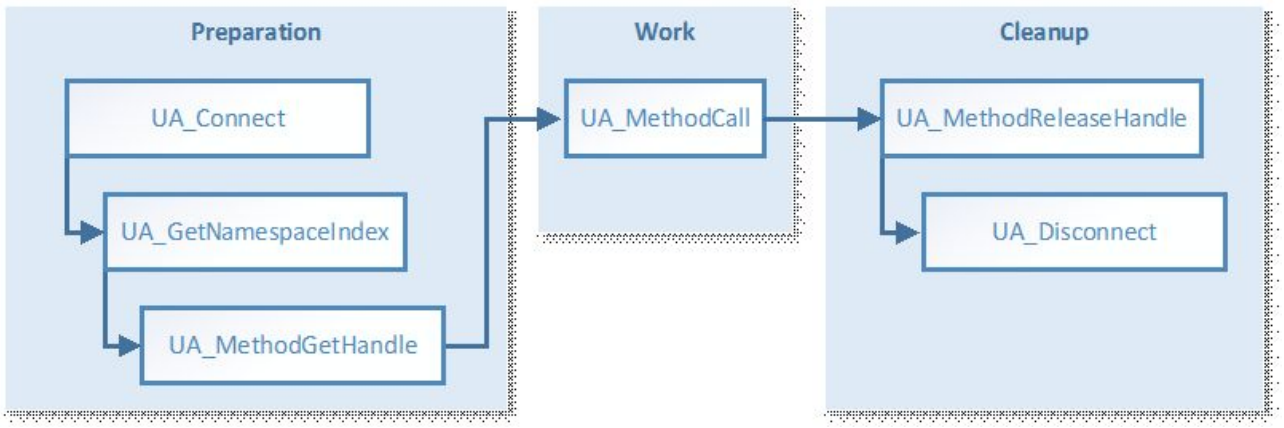
```
    NodeID.eIdentifierType := eUAIdentifierType_String;
    NodeID.nNamespaceIndex := nNamespaceIndex;
    NodeID.sIdentifier := sNodeIdentifier;
    fbUA_NodeGetHandle(
      Execute := TRUE,
      ConnectionHdl := nConnectionHdl,
      NodeID := NodeID,
      NodeHdl => nNodeHdl);
    IF NOT fbUA_NodeGetHandle.Busy THEN
      fbUA_NodeGetHandle(Execute := FALSE);
      IF NOT fbUA_NodeGetHandle.Error THEN
        iState := iState + 1;
      ELSE
        bError := TRUE;
        nErrorID := fbUA_NodeGetHandle.ErrorID;
        iState := 6;
      END_IF
    END_IF

  4: (* UA_Write *)
    fbUA_Write(
      Execute := TRUE,
      ConnectionHdl := nConnectionHdl,
      NodeHdl := nNodeHdl,
      stNodeAddInfo := stNodeAddInfo,
      cbData := SIZEOF(nWriteData),
      pVariable := ADR(nWriteData));
    IF NOT fbUA_Write.Busy THEN
      fbUA_Write(
        Execute := FALSE,
        pVariable := ADR(nWriteData));
      IF NOT fbUA_Write.Error THEN
        iState := iState + 1;
      ELSE
        bError := TRUE;
        nErrorID := fbUA_Write.ErrorID;
        iState := 6;
      END_IF
    END_IF

  5: (* Release Node Handle *)
    fbUA_NodeReleaseHandle(
      Execute := TRUE,
      ConnectionHdl := nConnectionHdl,
      NodeHdl := nNodeHdl);
    IF NOT fbUA_NodeReleaseHandle.Busy THEN
      fbUA_NodeReleaseHandle(Execute := FALSE);
      IF NOT fbUA_NodeReleaseHandle.Error THEN
        iState := iState + 1;
      ELSE
        bError := TRUE;
        nErrorID := fbUA_NodeReleaseHandle.ErrorID;
        iState := 6;
      END_IF
    END_IF

  6:
    [...]

END_CASE
```

调用方法

下一节将介绍如何使用功能块 TcX_PLCopen_OpcUa 在本地或远程 OPC UA 服务器上调用方法。连接到 OPC UA 服务器、调用 UA 方法以及随后中断会话需要使用以下功能块：UA_Connect [▶ 78]、UA_GetNamespaceIndex [▶ 81]、UA_MethodGetHandle [▶ 86]、UA_MethodCall [▶ 84]、UA_MethodReleaseHandle [▶ 87]、UA_Disconnect [▶ 80]。

每个 TwinCAT OPC UA 客户端的示意性工作流程可分为三个不同阶段：“准备”、“工作”和“清理”。

本节描述的用例可以直观地显示如下：



- 要与本地或远程 OPC UA 服务器建立连接，功能块 UA_Connect 需要以下信息（另请参见）：
  - 服务器 URL
  - 会话连接信息
- 功能块 UA_GetNamespaceIndex 需要一个连接句柄（来自 UA_Connect）和一个 NamespaceURI，以便解析为一个 NamespaceIndex，随后由 UA_NodeGetHandle 用来捕获节点句柄（另请参见）。
- 功能块 UA_MethodGetHandle 需要一个连接句柄（来自 UA_Connect）、一个 ObjectNodeID 和一个 MethodNodeID，以捕获一个方法句柄（另见）。
- 功能块 UA_MethodCall 需要一个连接句柄（来自 UA_Connect）、一个方法句柄（来自 UA_MethodGetHandle）以及要调用的方法的输入和输出参数信息。输入参数相关信息由 UA_MethodCall 的输入参数 pInputArgInfo 和 pInputArgData 表示。输出参数相关信息由 UA_MethodCall 的输入参数 pOutputArgInfo 和 pOutputArgData 表示。然后，输入参数 pOutputArgInfoAndData 表示指向一个结构的指针，该结构包含方法调用的结果，包括所有输出参数。以下代码片段将在 M_Init 方法中计算和创建 pInputArgInfo 和 pInputArgData 参数。
- 功能块 UA_NodeReleaseHandle 需要一个连接句柄（来自 UA_Connect）和一个方法句柄（来自 UA_MethodGetHandle）。

包含 UA 方法调用的功能块的 M_Init 初始化方法：

```
MEMSET(ADR(nInputData),0,SIZEOF(nInputData));
nArg := 1;

(********** Input parameter 1 **********)
InputArguments[nArg].DataType := eUAType_Int16;
InputArguments[nArg].ValueRank := -1; (* Scalar = -1 or Array *)
InputArguments[nArg].ArrayDimensions[1] := 0; (* Number of Dimension in case its an array *)
InputArguments[nArg].nLenData := SIZEOF(numberIn1); (* Length if its a STRING *)
IF nOffset + SIZEOF(numberIn1) > nInputArgSize THEN
  bInputDataError := TRUE;
  RETURN;
ELSE
  MEMCPY(ADR(nInputData)+nOffset,ADR(numberIn1),SIZEOF(numberIn1)); (* VALUE in BYTES FORM *)
  nOffset := nOffset + SIZEOF(numberIn1);
END_IF
nArg := nArg + 1;

(********** Input parameter 2 **********)
InputArguments[nArg].DataType := eUAType_Int16;
InputArguments[nArg].ValueRank := -1; (* Scalar = -1 or Array *)
InputArguments[nArg].ArrayDimensions[1] := 0; (* Number of Dimension in case its an array *)
InputArguments[nArg].nLenData := SIZEOF(numberIn2); (* Length if its a STRING *)
IF nOffset + SIZEOF(numberIn2) > nInputArgSize THEN
  bInputDataError := TRUE;
  RETURN;
ELSE
  MEMCPY(ADR(nInputData)+nOffset,ADR(numberIn2),SIZEOF(numberIn2));(* VALUE in BYTES FORM *)
  nOffset := nOffset + SIZEOF(numberIn2);
END_IF

cbWriteData := nOffset;
```

声明：

```
(* Declarations for UA_GetNamespaceIndex *)
fbUA_GetNamespaceIndex : UA_GetNamespaceIndex;
nNamespaceIndex : UINT;

(* Declarations for UA_MethodGetHandle *)
fbUA_MethodGetHandle: UA_MethodGetHandle;
ObjectNodeID: ST_UANodeID;
```

```
MethodNodeID: ST_UANodeID;
nMethodHdl: DWORD;

(* Declarations for UA_MethodCall *)
fbUA_MethodCall: UA_MethodCall;
sObjectNodeIdIdentifier : STRING(MAX_STRING_LENGTH) := 'MAIN.fbMathematics';
sMethodNodeIdIdentifier : STRING(MAX_STRING_LENGTH) := 'MAIN.fbMathematics#M_Mul';
nAdrWriteData: PVOID;
numberIn1: INT := 42; // change according to input value and data type
numberIn2: INT := 42; // change according to input value and data type
numberOutPro: DINT; // result (output parameter of M_Mul())
cbWriteData: UDINT; // calculated automatically by M_Init()
InputArguments: ARRAY[1..2] OF ST_UAMethodArgInfo; // change according to input parameters
stOutputArgInfo: ARRAY[1..1] OF ST_UAMethodArgInfo; // change according to output parameters
stOutputArgInfoAndData: ST_OutputArgInfoAndData;
nInputData: ARRAY[1..4] OF BYTE; // numberIn1(INT16)(2) + numberIn2(INT16)(2)
nOffset: UDINT; // calculated by M_Init()
nArg: INT; // used by M_Init()

(* Declarations for UA_MethodReleaseHandle *)
fbUA_MethodReleaseHandle: UA_MethodReleaseHandle;
```

实现：

```
CASE iState OF
  0:
    [...]

  2: (* GetNS Index *)
    fbUA_GetNamespaceIndex(
      Execute := TRUE,
      ConnectionHdl := nConnectionHdl,
      NamespaceUri := sNamespaceUri,
      NamespaceIndex => nNamespaceIndex);
    IF NOT fbUA_GetNamespaceIndex.Busy THEN
      fbUA_GetNamespaceIndex(Execute := FALSE);
      IF NOT fbUA_GetNamespaceIndex.Error THEN
        iState := iState + 1;
      ELSE
        bError := TRUE;
        nErrorID := fbUA_GetNamespaceIndex.ErrorID;
        iState := 7;
      END_IF
    END_IF

  3: (* Get Method Handle *)
    ObjectNodeID.eIdentifierType := eUAIdentifierType_String;
    ObjectNodeID.nNamespaceIndex := nNamespaceIndex;
    ObjectNodeID.sIdentifier := sObjectNodeIdIdentifier;
    MethodNodeID.eIdentifierType := eUAIdentifierType_String;
    MethodNodeID.nNamespaceIndex := nNamespaceIndex;
    MethodNodeID.sIdentifier := sMethodNodeIdIdentifier;

    M_Init();

    IF bInputDataError = FALSE THEN
      iState := iState + 1;
    ELSE
      bBusy := FALSE;
      bError := TRUE;
      nErrorID := 16#70A; //out of memory
    END_IF

  4: (* Method Get Handle *)
    fbUA_MethodGetHandle(
      Execute := TRUE,
      ConnectionHdl := nConnectionHdl,
      ObjectNodeID := ObjectNodeID,
      MethodNodeID := MethodNodeID,
      MethodHdl => nMethodHdl);
    IF NOT fbUA_MethodGetHandle.Busy THEN
      fbUA_MethodGetHandle(Execute := FALSE);
      IF NOT fbUA_MethodGetHandle.Error THEN
        iState := iState + 1;
      ELSE
        bError := TRUE;
        nErrorID := fbUA_MethodGetHandle.ErrorID;
        iState := 6;
      END_IF
    END_IF

  5: (* Method Call *)
    stOutputArgInfo[1].nLenData := SIZEOF(stOutputArgInfoAndData.pro);
    fbUA_MethodCall(
      Execute := TRUE,
      ConnectionHdl := nConnectionHdl,
      MethodHdl := nMethodHdl,
      nNumberOfInputArguments := nNumberOfInputArguments,
      pInputArgInfo := ADR(InputArguments),
      cbInputArgInfo := SIZEOF(InputArguments),
      pInputArgData := ADR(nInputData),
      cbInputArgData := cbWriteData,
      pInputWriteData := 0,
      cbInputWriteData := 0,
```

```
      nNumberOfOutputArguments := nNumberOfOutputArguments,
      pOutputArgInfo := ADR(stOutputArgInfo),
      cbOutputArgInfo := SIZEOF(stOutputArgInfo),
      pOutputArgInfoAndData := ADR(stOutputArgInfoAndData),
      cbOutputArgInfoAndData := SIZEOF(stOutputArgInfoAndData));
    IF NOT fbUA_MethodCall.Busy THEN
      fbUA_MethodCall(Execute := FALSE);
      IF NOT fbUA_MethodCall.Error THEN
        iState := iState + 1;
        numberOutPro := stOutputArgInfoAndData.pro;
      ELSE
        bError := TRUE;
        nErrorID := fbUA_MethodCall.ErrorID;
        iState := 6;
      END_IF
    END_IF

  6: (* Release Method Handle *)
    fbUA_MethodReleaseHandle(
      Execute := TRUE,
      ConnectionHdl := nConnectionHdl,
      MethodHdl := nMethodHdl);
    IF NOT fbUA_MethodReleaseHandle.Busy THEN
      fbUA_MethodReleaseHandle(Execute := FALSE);
      bBusy := FALSE;
      IF NOT fbUA_MethodReleaseHandle.Error THEN
        iState := 7;
      ELSE
        bError := TRUE;
        nErrorID := fbUA_MethodReleaseHandle.ErrorID;
        iState := 7;
      END_IF
    END_IF

  7:
    [...]

END_CASE
```

4.14 安全

4.14.1 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 的数据通信可以在两个层面上确保安全：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，该端点指定了要使用的安全功能。

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“信息安全”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 [▶ 52]。

应用层

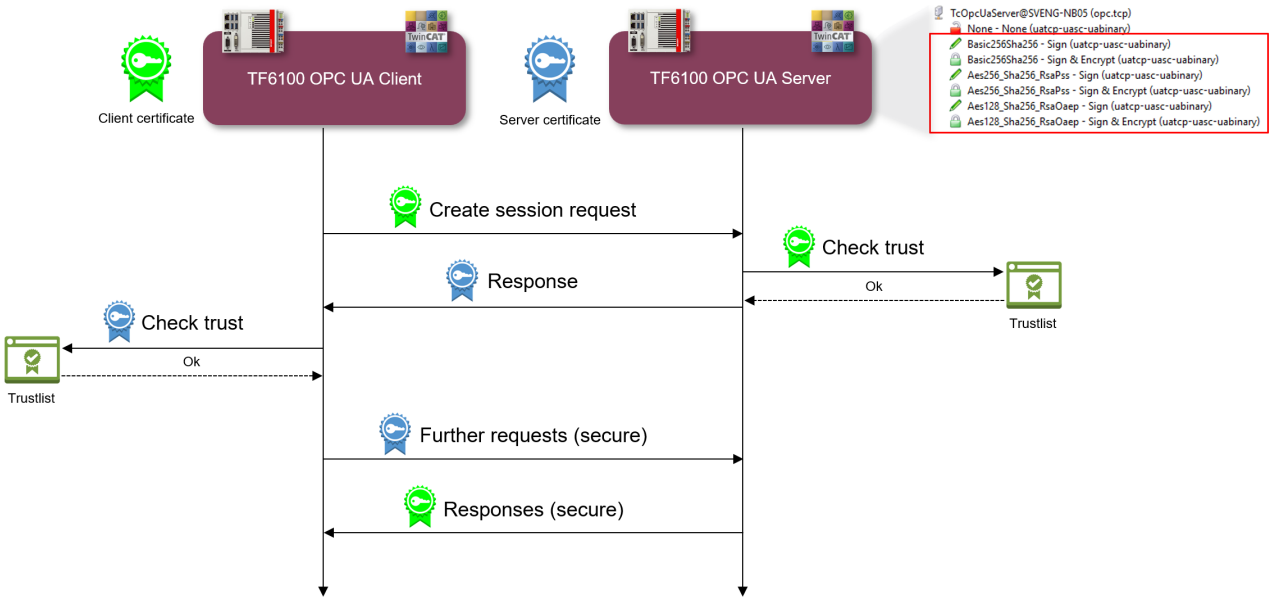
除了传输层，还可以在应用层确保连接的安全性。为此，服务器端点提供的各种认证机制 [▶ 53]可供使用。

4.14.2 证书交换

要通过安全端点确保传输层通信连接的安全，就必须在客户端和服务器之间建立连接。默认情况下，TwinCAT OPC UA 服务器和 TwinCAT OPC UA 客户端在首次启动时都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。然而，您也可以使用任意证书颁发机构或技术，例如 Active Directory 或 OpenSSL，将其集成到您的 IT 基础设施中。为了实现简单管理和安全访问证书，设置全局发现服务器是很有意义的。

要在 OPC UA 服务器和 TwinCAT OPC UA 客户端之间建立信任关系，需要服务器证书的公钥。客户端必须相应地对此表示信任。客户端在应用程序目录 [▶ 28]的一个子目录中管理服务器证书的信任设置。

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信息进行解密。

信息是反向签名的：信息是用发件人的私钥签名的。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/拒绝目录之间移动服务器证书，可以相应地调整信任设置。服务器证书的公钥会在服务器首次尝试与安全端点互联互通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目录，服务器在下一次尝试互联互通时就会受到信任，可以进行连接。

4.14.3 身份验证

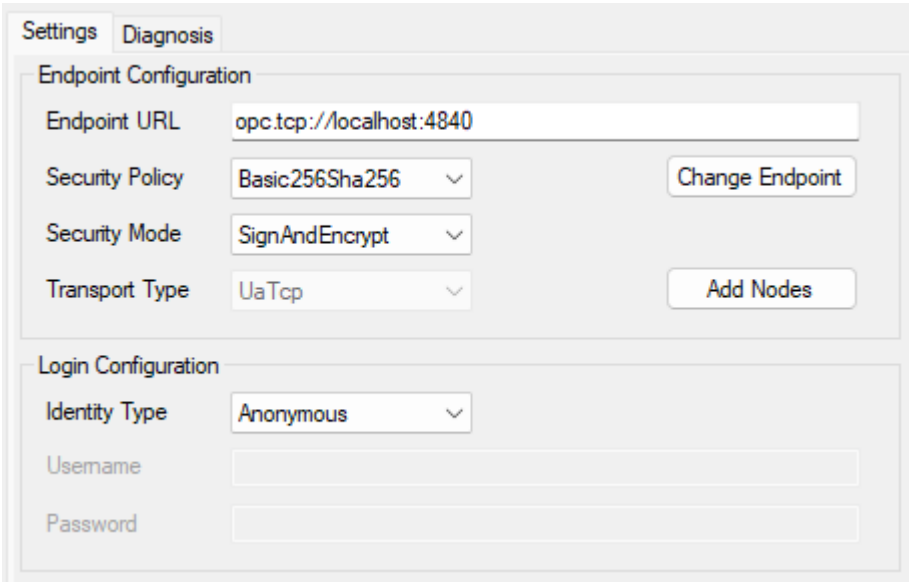
TwinCAT OPC UA 客户端支持使用不同的 IdentityToken 对 OPC UA 服务器进行身份验证。

- 匿名
- 用户名/密码
- 用户证书

以下截图是使用 TwinCAT OPC UA I/O 客户端截取的。使用 PLCOpen 功能块 [▶ 41]时，同样支持所述的所有功能。

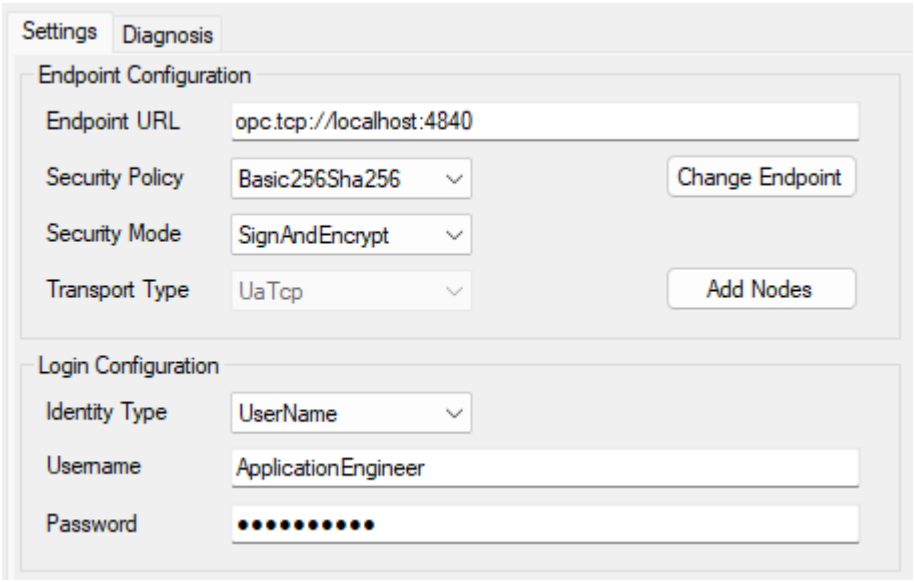
匿名

这种类型的身份验证使 TwinCAT OPC UA 客户端能够在不指定用户身份的情况下与服务器应用程序建立连接。以下是 TwinCAT OPC UA I/O 客户端的示例截图。



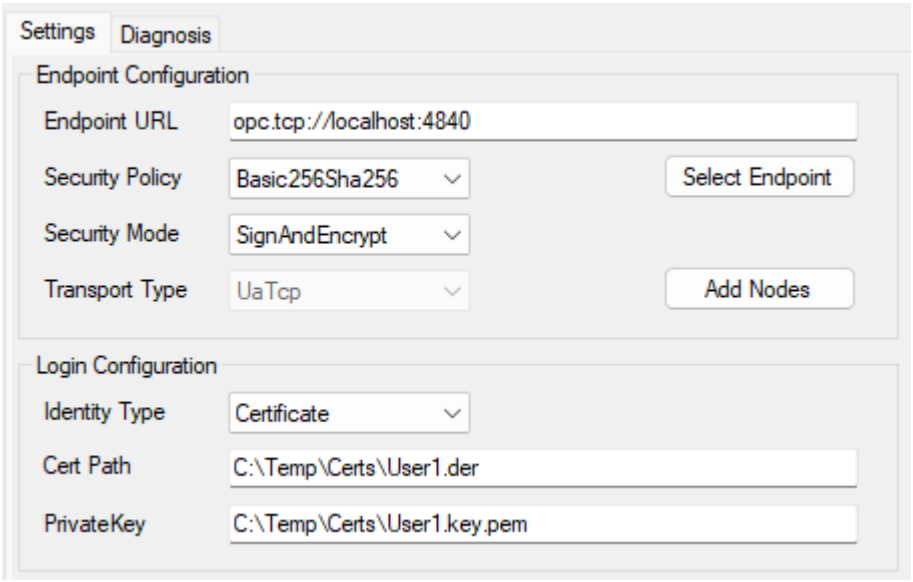
用户名/密码

这种类型的身份验证使用用户名/密码组合来验证 TwinCAT OPC UA 客户端到服务器应用程序的身份。以下是 TwinCAT OPC UA I/O 客户端的示例截图。



用户证书

在这种类型的身份验证中，TwinCAT OPC UA 客户端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。以下是 TwinCAT OPC UA I/O 客户端的示例截图。



注意

身份验证和服务器证书

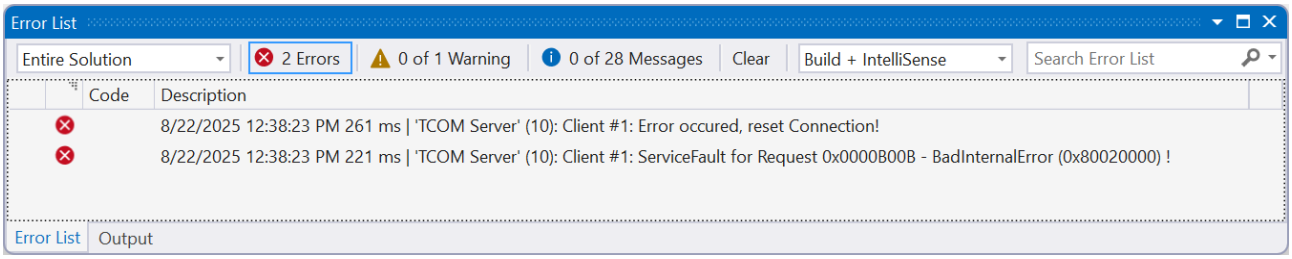
在结合身份验证使用未加密端点时，TwinCAT OPC UA 客户端仍然需要 OPC UA 服务器证书中的公钥，以便在传输过程中对密码进行加密。为此，证书必须在 TwinCAT OPC UA 客户端中得到信任（见证书交换 [▶ 52]）。

4.15 日志记录

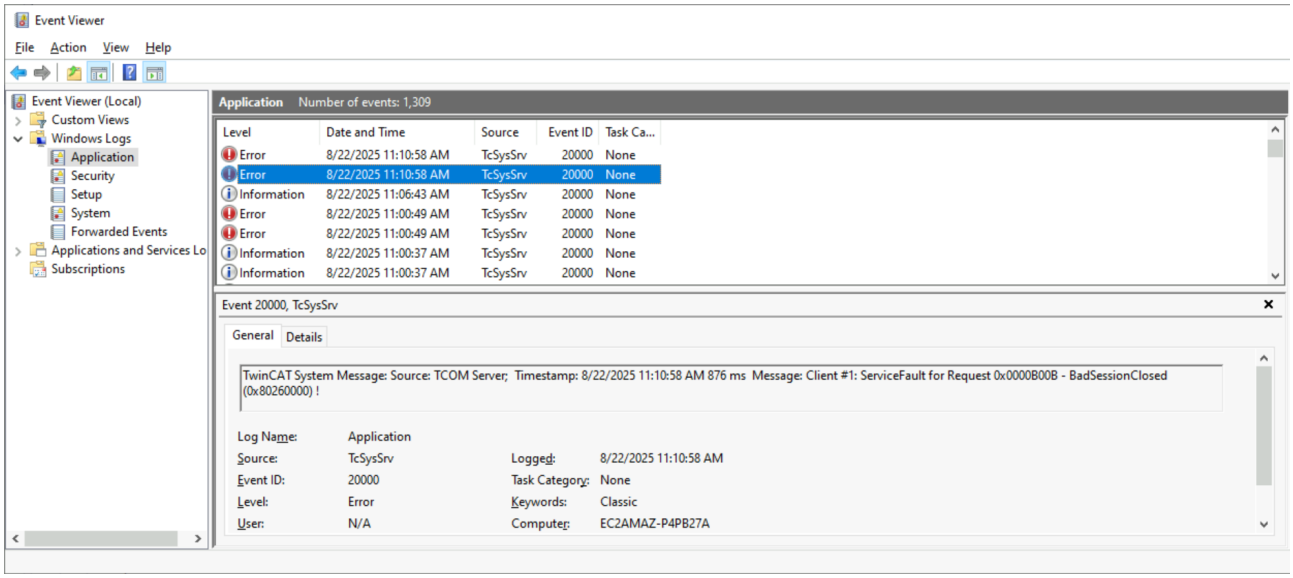
TwinCAT OPC UA 客户端可提供多种错误诊断选项。

客户端版本 2.x.x 以上

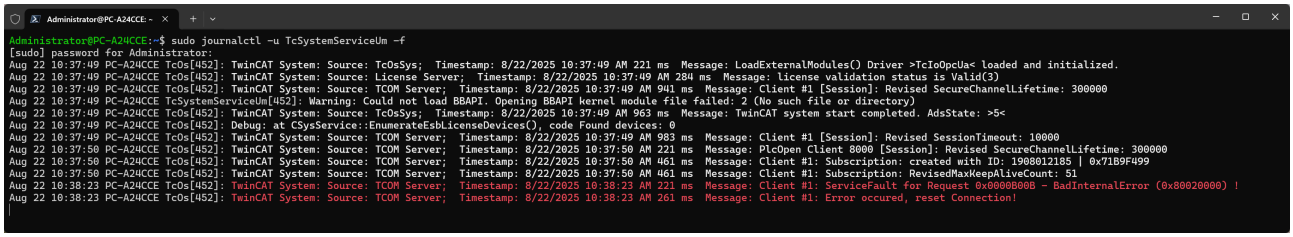
TwinCAT XAE ErrorLog 默认已启用，可在 TwinCAT XAE 工程设计环境中查看，用于记录错误状态：



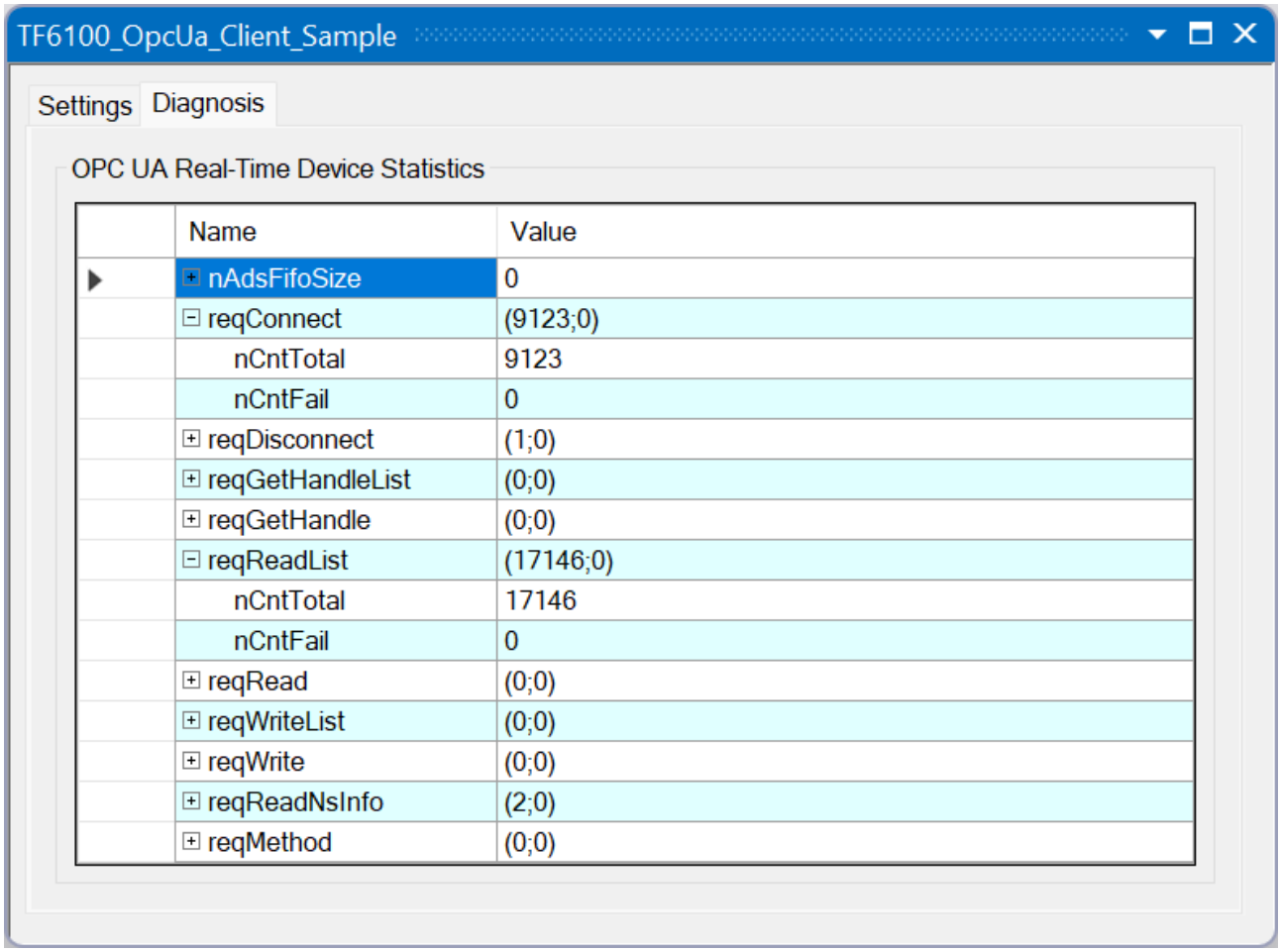
根据所使用的操作系统，可以通过其他机构读出 ErrorLog。在 Microsoft Windows 下，Windows 事件查看器可用于此目的，其中的消息保存在“应用程序日志”中。



在 Linux® 下，可以使用“journalctl”命令查看这些消息。



除了 TwinCAT XAE ErrorLog 的这些功能外，I/O 设备上还提供扩展诊断选项。



先前客户端版本

可以启用所谓的“StackTrace”和“AppTrace”。StackTrace 会提供 OPC UA 堆栈相关诊断信息，而 AppTrace 则限于纯客户端应用程序，在大多数情况下也足够了。下表列出的所有设置均在以下注册表键值下创建：

4026: HKLM\SOFTWARE\WOW6432Node\Beckhoff\TwinCAT3 Functions\TF6100 OPC UA Client

4024: HKLM\SOFTWARE\WOW6432Node\Beckhoff\TwinCAT3 Functions\TF6100 OPC UA\Configuration

类型	名称	数据类型	数据
AppTrace	日志文件	REG_SZ	日志文件名称
AppTrace	日志模式	REG_DWORD	0x00 (0) = 禁用 0x02 (2) = 启用 0x12 (18) = 启用日志记录，并提供额外调试输出
StackTrace	tracelogfile	REG_SZ	日志文件名称
StackTrace	tracelogmode	REG_DWORD	0x00 (0) = 禁用 0x01 (1) = 启用

更改后，必须至少更改一次 TwinCAT 上下文（Windows CE：完全重启设备）。这样可确保启用对日志记录的更改。

日志保存在以下文件夹中：C:\ProgramData\Beckhoff\TF6100-OPC-UA\TcOpcUaClient\logs。文件名可通过注册表更改。

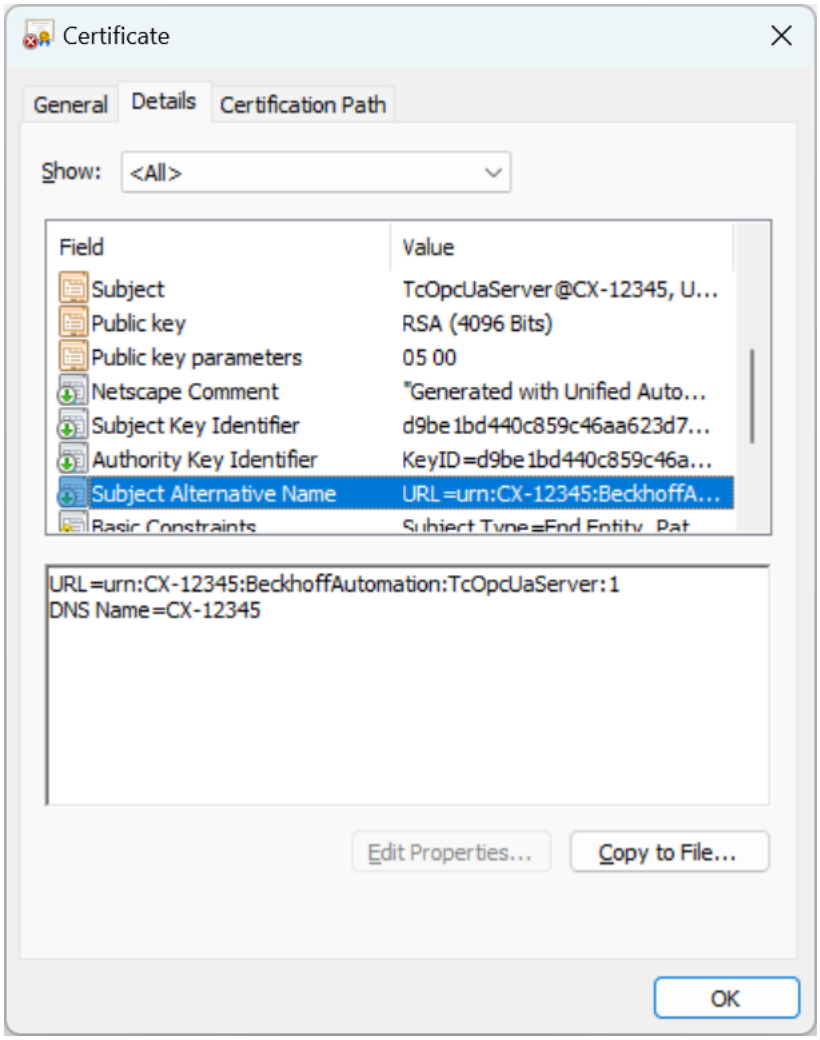
4.16 ApplicationURI

在 OPC UA（开放平台通信统一架构）中，ApplicationURI 是全局唯一标识符字符串，代表一个特定的 OPC UA 应用程序实例（客户端、服务器或二者皆可）。您可以将其视为 OPC UA 生态系统内部应用程序的数字身份。

ApplicationURI 有几个关键作用：安全与证书、应用程序识别和发现过程。其格式通常为 URI，通常使用“urn:”方案，例如：

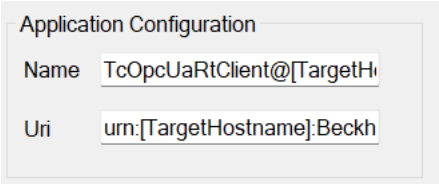
urn:CX-12345:BeckhoffAutomation:TcOpcUaServer:1

ApplicationURI 已添加到 OPC UA 应用程序证书的 SubjectAlternativeName 中，例如：



为 TwinCAT OPC UA 客户端设置 ApplicationURI

在配置 TwinCAT OPC UA 客户端时，可在客户端 I/O 设备的配置页面上设置其 ApplicationURI。



我们建议使用 ApplicationURI 的默认设置，但如果特殊用例需要更改，可在此配置页面执行操作。更改设置并激活配置后，驱动程序会决定能否在目标设备上找到与 ApplicationURI 匹配的现有应用程序证书。如果找到匹配的应用证书，则会使用该证书。如果未找到匹配的应用程序证书，则驱动程序会创建新的（自签名）应用程序证书，并使用该证书与服务器建立通信。这也意味着，您可以针对项目中的多台 TwinCAT OPC UA 客户端 I/O 设备使用不同的应用程序证书。默认配置为所有设备使用一个应用程序证书。

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 [▶ 59] 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

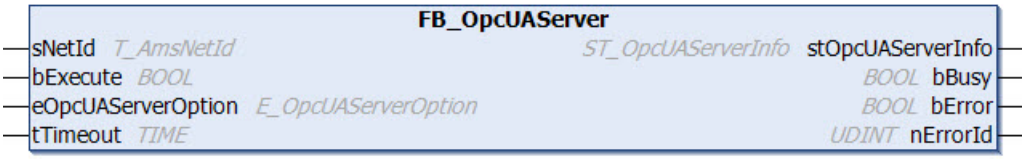
```
TYPE E_OpcUAServerStatus
(
    eOPCUAServerStatus_None,
    eOPCUAServerStatus_Alive,
    eOPCUAServerStatus_NotResponding
);
END_TYPE
```

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
```

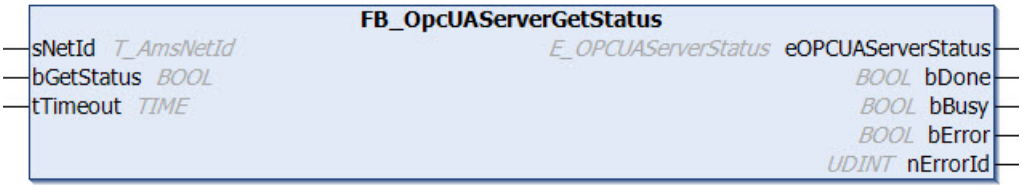
输入

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 [▶ 59]	指定要执行的操作。
tTimeout	TIME	ADS 超时

输出

名称	类型	描述
stOpcUAServerInfo	ST_OpcUAServerInfo [▶ 59]	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

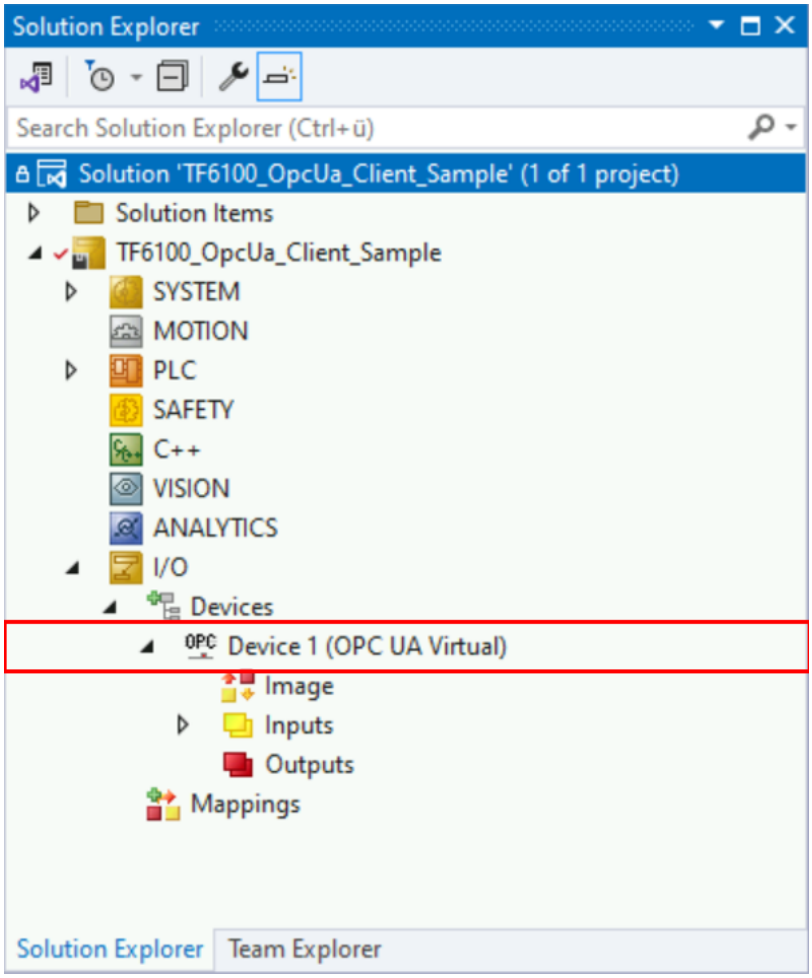
输出

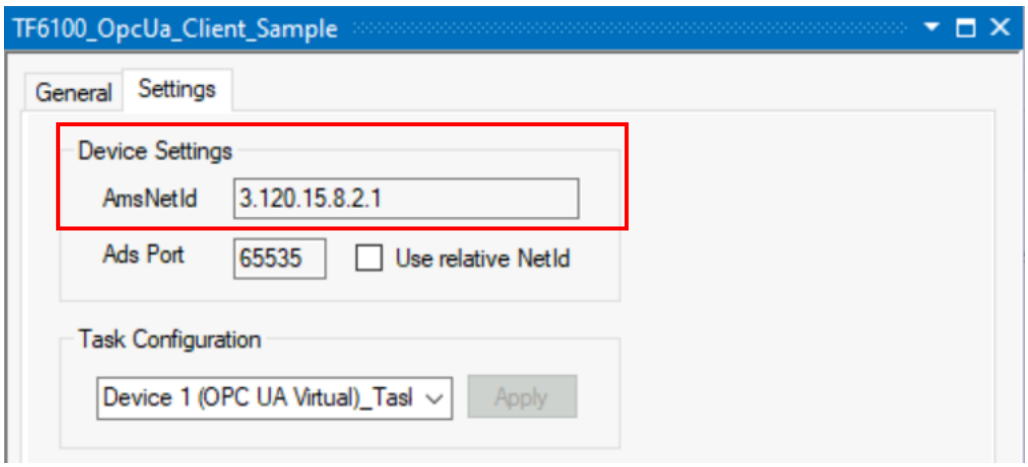
名称	类型	描述
eOPCUAServerStatus	E_OpcUAServerStatus [▶ 60]	包含服务器的状态信息。
bDone	BOOL	功能块处理完成时为 TRUE。
bBusy	BOOL	只要功能块的处理还在进行，就为 TRUE。
bError	BOOL	一旦出现情况，则变为 TRUE。
nErrorId	UDINT	包含发生错误 (bError) 时的错误代码。

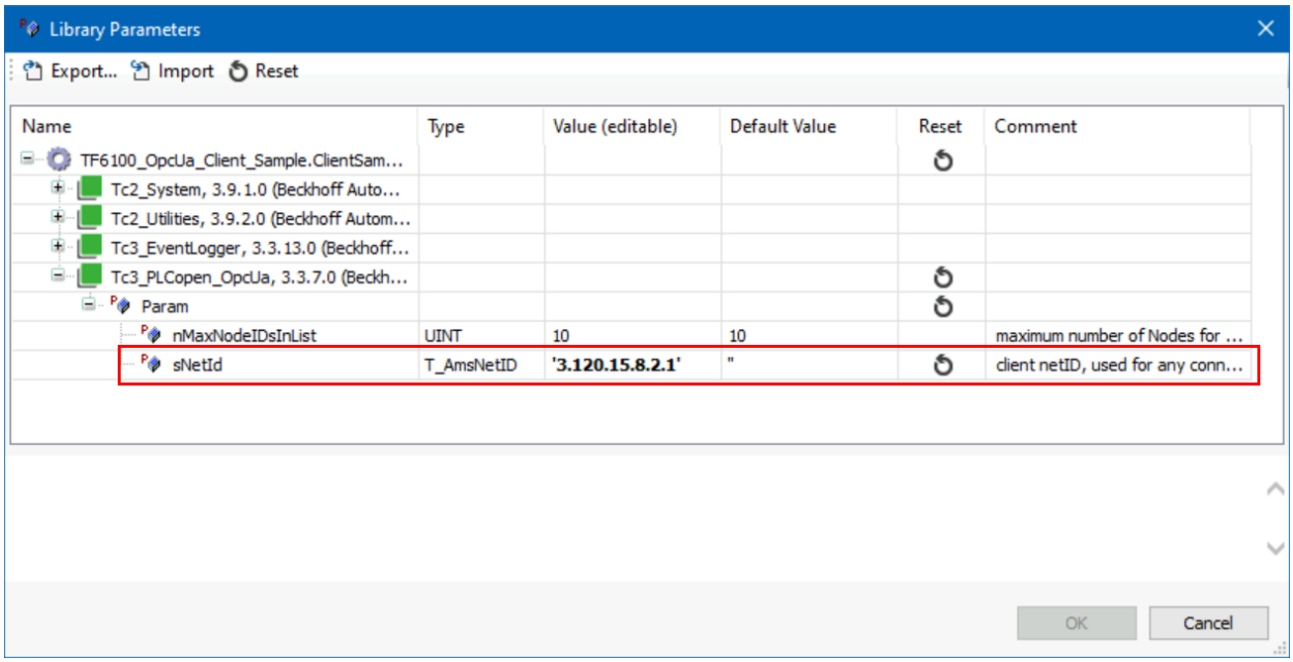
5.2 Tc3_PLCopen_OpcUa

5.2.1 使用说明

对于 TwinCAT OPC UA 客户端版本 2.x.x 以上，必须在 TwinCAT 的 I/O 部分创建“OPC UA 虚拟设备”，才能使用 Tc3_PLCopen_OpcUa 库，并且设备的 AMS Net ID 必须存储在该库参数列表中的 sAmsNetId 参数中。只需要执行一次操作。







5.2.2 数据类型

5.2.2.1 E_UAAttributeID

语法

```
TYPE E_UAAttributeID:
(
    eUAAI_NodeID          := 1,
    eUAAI_NodeClass       := 2,
    eUAAI_BrowseName      := 3,
    eUAAI_DisplayName     := 4,
    eUAAI_Description     := 5,
    eUAAI_WriteMask       := 6,
    eUAAI_UserWriteMask   := 7,
    eUAAI_IsAbstract      := 8,
    eUAAI_Symmetric       := 9,
    eUAAI_InverseName     := 10,
    eUAAI_ContainsNoLoops := 11,
    eUAAI_EventNotifier   := 12,
    eUAAI_Value           := 13,
    eUAAI_DataType        := 14,
    eUAAI_ValueRank       := 15,
    eUAAI_ArrayDimensions := 16
)DINT;
END_TYPE
```

值

名称	描述
NodeID	OPC UA NodeID
NodeClass	OPC UA NodeClass
BrowseName	OPC UA BrowseName
DisplayName	OPC UA DisplayName
描述	OPC UA 描述
WriteMask	OPC UA WriteMask
UserWriteMask	OPC UA UserWriteMask
IsAbstract	OPC UA IsAbstract
对称	OPC UA 对称
InverseName	OPC UA InverseName
ContainsNoLoops	OPC UA ContainsNoLoops
EventNotifier	OPC UA EventNotifier
值	OPC UA 值
DataType	OPC UA DataType
ValueRank	OPC UA ValueRank
ArrayDimensions	OPC UA ArrayDimensions

要求

开发环境	目标平台	要包括的 PLC 库
TwinCAT 3.1	Win32、Win64、CE-X86、CE-Arm®	Tc3_PLCopen_OpcUa

5.2.2.2 E_UABrowseDirection

语法

```
TYPE E_UABrowseDirection:
(
    eUABD_Forward   := 0,
    eUABD_Inverse   := 1,
    eUABD_Both      := 2
)DINT;
END_TYPE
```

值

名称	描述
eUABD_Forward	正向参考案例
eUABD_Inverse	反向参考案例
eUABD_Both	正向和反向参考案例

要求

开发环境	目标平台	要包括的 PLC 库
TwinCAT 3.1	Win32、Win64、CE-X86、CE-Arm®	Tc3_PLCopen_OpcUa

5.2.2.3 E_UABrowseResultMask

语法

```
TYPE E_UABrowseResultMask:
(
    eUABRM_ReferenceTypeId   := 1,
    eUABRM_IsForward         := 2,
    eUABRM_ReferenceTypeInfo := 3,
    eUABRM_NodeClass         := 4,
    eUABRM_BrowseName        := 8,
    eUABRM_DisplayName       := 16,
    eUABRM_TypeDefinition    := 32,
    eUABRM_TargetInfo        := 60,
    eUABRM_All               := 63
)DINT;
END_TYPE
```

值

名称	描述
eUABRM_ReferenceTypeId	ReferenceTypeId
eUABRM_IsForward	IsForward
eUABRM_ReferenceTypeInfo	ReferenceTypeInfo
eUABRM_NodeClass	NodeClass
eUABRM_BrowseName	BrowseName
eUABRM_DisplayName	DisplayName
eUABRM_TypeDefinition	TypeDefinition
eUABRM_TargetInfo	TargetInfo
eUABRM_All	全部

要求

开发环境	目标平台	要包括的 PLC 库
TwinCAT 3.1	Win32、Win64、CE-X86、CE-Arm®	Tc3_PLCopen_OpcUa

5.2.2.4 E_UAConnectionStatus

语法

```
TYPE E_UAConnectionStatus:
(
    Connected       := 0
    ConnectionError := 1,
    Shutdown        := 2
)DINT;
END_TYPE
```

值

名称	描述
已连接	连接已建立。
连接错误	建立连接时发生错误。
关闭	连接已断开。

要求

开发环境	目标平台	要包括的 PLC 库
TwinCAT 3.1	Win32、Win64、CE-X86、CE-Arm®	Tc3_PLCopen_OpcUa

PLC 库	所需版本
Tc3_PLCopen_OpcUa	>= 3.2.11.0

5.2.2.5 E_UADataType

语法

```
TYPE E_UADataType:
(
    eUAType_Undefinied      := -1,
    eUAType_Null            := 0,
    eUAType_Boolean         := 1,
    eUAType_SByte           := 2,
    eUAType_Byte            := 3,
    eUAType_Int16           := 4,
    eUAType_UInt16          := 5,
    eUAType_Int32           := 6,
    eUAType_UInt32          := 7,
    eUAType_Int64           := 8,
    eUAType_UInt64          := 9,
    eUAType_Float           := 10,
    eUAType_Double          := 11,
    eUAType_String          := 12,
    eUAType_DateTime        := 13,
    eUAType_Guid            := 14,
    eUAType_ByteString      := 15,
    eUAType_XmlElement      := 16,
    eUAType_NodeId          := 17,
    eUAType_ExpandedNodeId  := 18,
    eUAType_StatusCode      := 19,
    eUAType_QualifiedName   := 20,
    eUAType_LocalizedText   := 21,
    eUAType_ExtensionObject := 22,
    eUAType_DataValue       := 23,
    eUAType_Variant         := 24,
    eUAType_DiagnosticInfo  := 25
)DINT;
END_TYPE
```

值

名称	描述
eUAType_Undefinied	未确定
eUAType_Null	零
eUAType_Boolean	布尔类型
eUAType_SByte	SByte
eUAType_Byte	字节
eUAType_Int16	Int16
eUAType_UInt16	UInt16
eUAType_Int32	Int32
eUAType_UInt32	UInt32
eUAType_Int64	Int64
eUAType_UInt64	UInt64
eUAType_Float	浮动
eUAType_Double	双倍
eUAType_String	字符串
eUAType_DateTime	DateTime
eUAType_Guid	Guid
eUAType_ByteString	ByteString
eUAType_XmlElement	XmlElement
eUAType_NodeId	NodeId
eUAType_ExpandedNodeId	ExpandedNodeId
eUAType_StatusCode	StatusCode
eUAType_QualifiedName	QualifiedName

名称	描述
eUAType_LocalizedText	LocalizedText
eUAType_ExtensionObject	ExtensionObject
eUAType_DataValue	DataValue
eUAType_Variant	变体
eUAType_DiagnosticInfo	DiagnosticInfo

要求

开发环境	目标平台	要包括的 PLC 库
TwinCAT 3.1	Win32、Win64、CE-X86、CE-Arm®	Tc3_PLCopen_OpcUa

5.2.2.6 E_UAIdentifierType

语法

```
TYPE E_UAIdentifierType:
(
    eUAIdentifierType_String  := 1,
    eUAIdentifierType_Numeric := 2,
    eUAIdentifierType_GUID    := 3,
    eUAIdentifierType_Opaque  := 4
)DINT;
END_TYPE
```

值

名称	描述
eUAIdentifierType_String	字符串
eUAIdentifierType_Numeric	数字
eUAIdentifierType_GUID	GUID
eUAIdentifierType_Opaque	不透明

要求

开发环境	目标平台	要包括的 PLC 库
TwinCAT 3.1	Win32、Win64、CE-X86、CE-Arm®	Tc3_PLCopen_OpcUa

5.2.2.7 E_UANodeClassMask

语法

```
TYPE E_UANodeClassMask:
(
    eUANCM_Unspecified   := 0,
    eUANCM_Object        := 1,
    eUANCM_Variable      := 2,
    eUANCM_Method        := 4,
    eUANCM_ObjectType    := 8,
    eUANCM_VariableType  := 16,
    eUANCM_ReferenceType := 32,
    eUANCM_DataType      := 64,
    eUANCM_View          := 128,
    eUANCM_All           := 255
)DINT;
END_TYPE
```

值

名称	描述
eUANCM_Unspecified	未指定

名称	描述
eUANCM_Object	对象
eUANCM_Variable	变量
eUANCM_Method	方法
eUANCM_ObjectType	ObjectType
eUANCM_VariableType	VariableType
eUANCM_ReferenceType	ReferenceType
eUANCM_DataType	DataType
eUANCM_View	视图
eUANCM_All	全部

要求

开发环境	目标平台	要包括的 PLC 库
TwinCAT 3.1	Win32、Win64、CE-X86、CE-Arm®	Tc3_PLCopen_OpcUa

5.2.2.8 E_UASecurityMsgMode

语法

```
TYPE E_UASecurityMsgMode:
(
    eUASecurityMsgMode_BestAvailable := 0,
    eUASecurityMsgMode_None          := 1,
    eUASecurityMsgMode_Sign          := 2,
    eUASecurityMsgMode_Sign_Encrypt  := 3
)DINT;
END_TYPE
```

值

名称	描述
eUASecurityMsgMode_BestAvailable	现有最高安全性
eUASecurityMsgMode_None	非安全
eUASecurityMsgMode_Sign	签名
eUASecurityMsgMode_Sign_Encrypt	签名和加密

要求

开发环境	目标平台	要包括的 PLC 库
TwinCAT 3.1	Win32、Win64、CE-X86、CE-Arm®	Tc3_PLCopen_OpcUa

5.2.2.9 E_UASecurityPolicy

语法

```
TYPE E_UASecurityPolicy:
(
    eUASecurityPolicy_BestAvailable := 0
    eUASecurityPolicy_None          := 1,
    eUASecurityPolicy_Basic128      := 2,
    eUASecurityPolicy_Basic128Rsa15 := 3,
    eUASecurityPolicy_Basic256      := 4
)DINT;
END_TYPE
```

值

名称	描述
BestAvailable	现有最高安全性。
无	最低安全要求配置指南。
Basic128	低度至中度安全要求的配置指南。
Basic128Rsa15	为具有中度到高度安全要求的配置定义安全准则。
Basic256	为具有高度安全要求的配置定义安全策略。

要求

开发环境	目标平台	要包括的 PLC 库
TwinCAT 3.1	Win32、Win64、CE-X86、CE-Arm®	Tc3_PLCopen_OpcUa

5.2.2.10 E_UAServerState

语法

```
TYPE E_UAServerState:
(
    Running             := 0
    Failed              := 1,
    NoConfiguration     := 2,
    Suspended           := 3,
    Shutdown            := 4,
    Test                := 5,
    CommunicationFault  := 6,
    Unknown             := 7
)DINT;
END_TYPE
```

值

名称	描述
运行	运行
失败	失败
NoConfiguration	NoConfiguration
暂停	暂停
关闭	关闭
测试	测试
CommunicationFault	CommunicationFault
未知	未知

要求

开发环境	目标平台	要包括的 PLC 库
TwinCAT 3.1	Win32、Win64、CE-X86、CE-Arm®	Tc3_PLCopen_OpcUa

PLC 库	所需版本
Tc3_PLCopen_OpcUa	>= 3.2.11.0

5.2.2.11 E_UATransportProfile

语法

```
TYPE E_UATransportProfile:
(
    eUATransportProfileUri_UATcp            := 1,
```

```
    eUATransportProfileUri_WSHttpBinary      := 2,
    eUATransportProfileUri_WSHttpXmlOrBinary := 3,
    eUATransportProfileUri_WSHttpXml         := 4
)DINT;
END_TYPE
```

值

名称	描述
eUATransportProfileUri_UATcp	UATcp
eUATransportProfileUri_WSHttpBinary	WSHttpBinary
eUATransportProfileUri_WSHttpXmlOrBinary	WSHttpXmlOrBinary
eUATransportProfileUri_WSHttpXml	WSHttpXml

要求

开发环境	目标平台	要包括的 PLC 库
TwinCAT 3.1	Win32、Win64、CE-X86、CE-Arm®	Tc3_PLCopen_OpcUa

5.2.2.12 E_UAUserIdentityTokenType

语法

```
TYPE E_UAUserIdentityTokenType:
(
    eUAUITT_Anonymous      := 0,
    eUAUITT_Username       := 1,
    eUAUITT_x509           := 2,
    eUAUITT_IssuedToeken   := 3
)DINT;
END_TYPE
```

值

名称	描述
eUAUITT_Anonymous	匿名用户。
eUAUITT_Username	使用用户名登录。
eUAUITT_x509	用于登录的证书文件。
eUAUITT_IssuedToeken	通过令牌登录。

要求

开发环境	目标平台	要包括的 PLC 库
TwinCAT 3.1	Win32、Win64、CE-X86、CE-Arm®	Tc3_PLCopen_OpcUa

5.2.2.13 ST_UABrowseDescription

语法

```
TYPE ST_UABrowseDescription:
STRUCT
    stStartingNodeId  : ST_UANodeId;
    eDirection        : E_UABrowseDirection;
    stReferenceTypeId : ST_UANodeId;
    bIncludeSubtypes  : BOOL;
    eNodeClass        : E_UANodeClassMask;
    eResultMask       : E_UABrowseResultMask;
END_STRUCT
END_TYPE
```

值

名称	描述
stStartingNodeId	默认启动节点：ObjectRoot
eDirection	默认浏览方向：正向
stReferenceTypeId	默认 ReferenceType：层级
bIncludeSubtype	默认 IncludeSubtype：TRUE
eNodeClass	默认 NodeClassMask：全部
eResultMask	默认 BrowseResultMask：全部

要求

开发环境	目标平台	要包括的 PLC 库
TwinCAT 3.1	Win32、Win64、CE-X86、CE-Arm®	Tc3_PLCopen_OpcUa

5.2.2.14 ST_UAExpandedNodeID

语法

```
TYPE ST_UAExpandedNodeID:
STRUCT
    nServerIndex   : UDINT;
    sNamespaceURI  : STRING(MAX_STRING_LENGTH);
    stNodeID       : ST_UANodeID;
END_STRUCT
END_TYPE
```

值

名称	描述
nServerIndex	ServerIndex
sNamespaceURI	NamespaceName
stNodeID	NodeID (ST_UANodeID [▶ 73])

要求

开发环境	目标平台	要包括的 PLC 库
TwinCAT 3.1	Win32、Win64、CE-X86、CE-Arm®	Tc3_PLCopen_OpcUa

5.2.2.15 ST_UASessionConnectInfo

语法

```
TYPE ST_UASessionConnectInfo:
STRUCT
    sApplicationName     : STRING(MAX_STRING_LENGTH);

    eSecurityMode        : E_UASecurityMsgMode;
    eSecurityPolicyUri   : E_UASecurityPolicy;
    eTransportProfileUri : E_UATransportProfile;
    stUserIdentTokenType : ST_UAUserIdentityTokenType;

    tSessionTimeout      : TIME;
    tConnectTimeout      : TIME;
END_STRUCT
END_TYPE
```

值

名称	描述
sApplicationUri（已废弃）	应用程序 Uri 最大字符串长度 255。 针对 TcUAClient 2.0.0.14 以上版本，根据 PLCOpen 规范的定义，证书会自动指定这一项。因此，在当前版本的库中已不再使用。
sApplicationName	应用程序名称，最大字符串长度为 255。
eSecurityMode	安全信息模式。有关可用模式，请参见 E_UASecurityMsgMode [▶ 68]。
eSecurityPolicyUri	安全策略 Uri。有关可用的安全策略 Uri，请参见 E_UASecurityPolicy [▶ 68]。
eTransportProfileUri	传输配置文件 Uri。有关可用的传输配置文件 Uri，请参见 E_UATransportProfile [▶ 69]；
stUserIdentTokenType	包含身份验证数据的结构，用于登录 OPC UA 服务器。 ST_UAUserIdentityTokenType [▶ 75] 下的完整描述。
tSessionTimeout	会话超时值。
tConnectTimeout	连接超时值。必须在 UA_Connect 功能块中进行设置，使其与 ADS 超时相匹配。 经验法则是：ADS 超时 > 2 * ConnectionTimeout。

要求

开发环境	目标平台	要包括的 PLC 库
TwinCAT 3.1	Win32、Win64、CE-X86、CE-Arm®	Tc3_PLCopen_OpcUa

5.2.2.16 ST_UAIndexRange

语法

```
TYPE ST_UAIndexRange:
STRUCT
    nStartIndex : UDINT;
    nEndIndex   : UDINT;
END_STRUCT
END_TYPE
```

值

名称	描述
nStartIndex	数据的起始索引。
nEndIndex	数据的结束索引。

适用于所有维度：

- 必须指定 StartIndex 和 EndIndex。
- StartIndex 必须小于 EndIndex。
- 要访问维度中的所有元素，必须根据元素总数在维度中指定 StartIndex 和 EndIndex。
- 通过指定相同的 StartIndex 和 EndIndex，可以选择维度的各个元素。

要求

开发环境	目标平台	要包括的 PLC 库
TwinCAT 3.1	Win32、Win64、CE-X86、CE-Arm®	Tc3_PLCopen_OpcUa

5.2.2.17 ST_UALocalizedText

语法

```
TYPE ST_UALocalizedText:
STRUCT
    sLocale : STRING(6);
    sText   : STRING(MAX_STRING_LENGTH);
END_STRUCT
END_TYPE
```

值

名称	描述
sLocale	LocalizedText 的语言标识符
sText	Text

要求

开发环境	目标平台	要包括的 PLC 库
TwinCAT 3.1	Win32、Win64、CE-X86、CE-Arm®	Tc3_PLCopen_OpcUa

5.2.2.18 ST_UAMethodArgInfo

语法

```
TYPE ST_UAMethodArgInfo:
STRUCT
    DataType        : E_UADataType := -1;
    ValueRank       : DINT := 2147483647;
    ArrayDimensions : ARRAY[1..3] OF UDINT := [0,0,0];
    nLenData        : DINT;
END_STRUCT
END_TYPE
```

值

名称	描述
DataType	定义方法参数的 UA 数据类型。（类型：E_UADataType [▶ 66]）
ValueRank	确定参数是标量 (-1) 还是数组。
ArrayDimensions	如果参数是数组，则指定数组的维数。每个元素决定每个维度的长度。
nLenData	指定参数的长度。对于输出信息，STRUCT 只要求该元素。

要求

开发环境	目标平台	要包括的 PLC 库
TwinCAT 3.1	Win32、Win64、CE-X86、CE-Arm®	Tc3_PLCopen_OpcUa

5.2.2.19 ST_UANodeID

语法

```
TYPE ST_UANodeID:
STRUCT
    nNamespaceIndex  : UINT;
    nReserved        : ARRAY [1..2] OF BYTE; //fill bytes
    sIdentifier      : STRING(MAX_STRING_LENGTH);
    eIdentifierType  : E_UAIdentifierType;
END_STRUCT
END_TYPE
```

值

名称	描述
nNamespaceIndex	节点所在的命名空间索引。可通过功能块 UA_GetNamespaceIndex [▶ 81] 确定。
nReserved	占位符
sIdentifier	UA 名称空间（属性“标识符”）中显示的标识符。
eIdentifierType	变量类型，由 E_UAIdentifierType [▶ 67] 描述。

要求

开发环境	目标平台	要包括的 PLC 库
TwinCAT 3.1	Win32、Win64、CE-X86、CE-Arm®	Tc3_PLCopen_OpcUa

5.2.2.20 ST_UANodeAdditionalInfo

语法

```
TYPE ST_UANodeAdditionalInfo:
STRUCT
    eAttributeID     : E_UAAttributeID;
    nIndexRangeCount : UINT;
    nReserved        : ARRAY[1..2] OF BYTE; // fill bytes
    stIndexRange     : ARRAY[1..nMaxIndexRange] OF ST_UAIndexRange;
END_STRUCT
END_TYPE
```

值

名称	描述
eAttributeID	指定 OPC UA 属性的 ID。默认使用 eUAAI_Value。（类型：E_UAAttributeID [▶ 63]）。
nIndexRangeCount	确定 stIndexRange 中使用的索引范围数量。
nReserved	占位符
stIndexRange	指定从数组中读取值的索引范围。（类型：ST_UAIndexRange [▶ 72]）。

要求

开发环境	目标平台	要包括的 PLC 库
TwinCAT 3.1	Win32、Win64、CE-X86、CE-Arm®	Tc3_PLCopen_OpcUa

5.2.2.21 ST_UAReferenceDescription

语法

```
TYPE ST_UAReferenceDescription:
STRUCT
    stReferenceTypeId : ST_UANodeId;
    bIsForward        : BOOL;
    stNodeId          : ST_UAExpandedNodeId;
    stBrowseName      : STRING(MAX_STRING_LENGTH);
    stDisplayName     : ST_UALocalizedText;
    eNodeClass        : E_UANodeClassMask;
    stTypeDefinition  : ST_UAExpandedNodeId;
END_STRUCT
END_TYPE
```

值

名称	描述
stReferenceTypeId	作为数据类型 ST_UANodeId [▶ 73] 的参考案例类型（如 Organizes、HasChild、HasTypeDefinition...）的 NodeId。
bIsForward	表示参考案例是正向还是反向。
stNodeId	NodeId 作为 ST_UAExpandedNodeId [▶ 71] 数据类型。
stBrowseName	参考案例的 BrowseName。
stDisplayName	参考案例的 DisplayName (ST_UALocalizedText [▶ 73])。
eNodeClass	参考案例的 NodeClass (E_UANodeClassMask [▶ 67])。
stTypeDefinition	类型定义 (HasTypeDefinition) (ST_UAExpandedNodeId [▶ 71])。

要求

开发环境	目标平台	要包括的 PLC 库
TwinCAT 3.1	Win32、Win64、CE-X86、CE-Arm®	Tc3_PLCopen_OpcUa

5.2.2.22 ST_UAUserIdentityTokenType

语法

```
TYPE ST_UAUserIdentityTokenType:
STRUCT
    eUserIdentTokenType   : E_UAUserIdentityTokenType;
    sTokenParam1          : STRING(MAX_STRING_LENGTH);
    sTokenParam2          : STRING(MAX_STRING_LENGTH);
END_STRUCT
END_TYPE
```

值

名称	描述
eUserIdentTokenType	用户类型，使用 E_UAUserIdentityTokenType [▶ 70]... 描述。
sTokenParam1	用于登录 OPC UA 服务器的用户名。
sTokenParam2	用于登录 OPC UA 服务器的密码。

要求

开发环境	目标平台	要包括的 PLC 库
TwinCAT 3.1	Win32、Win64、CE-X86、CE-Arm®	Tc3_PLCopen_OpcUa

5.2.2.23 UAHADataValue

该功能块充当数据对象。一个实例代表 OPC UA 历史访问功能的一个值。在调用时，这些值的整个字段会被传输到 UA_HistoryUpdate [▶ 82] 功能块。

语法

```
aDataValues : ARRAY [1..50] OF UAHADataValue(ValueSize:=SIZEOF(LREAL));
```

每个数据对象的初始化值为预期大小（字节）。

属性

名称	类型	访问	初始值	描述
值	PVOID	设置	-	指定包含所需值的变量地址。通常使用运算符 ADR() 进行分配。值本身也会同时赋值并复制到数据对象中。
StatusCode	UAHAUpdateStatusCode [▶ 76]	获取，设置	UAHAUpdateStatusCode.HistorianRaw	表示值的状态代码。
SourceTimeStamp	ULINT	获取，设置	0	以 UTC 格式表示源的时间戳。可借助功能 F_GetSystemTime（Tc2_System PLC 库）进行确定。
ServerTimeStamp	ULINT	获取，设置	0	以 UTC 格式表示 OPC UA 服务器的时间戳。目前不支持此功能。

值的数据类型大小

所使用数据类型的大小已在数据对象的声明中说明和定义。稍后赋值时，将以这一大小为基础。

因此，STRING 类型的值也以固定的初始化大小保存和传输。无法显示当前文本长度。

示例

```
{attribute 'OPC.UA.DA' := '1'}
fMyValue    : LREAL;       // Variable for HistorcalAccess
aDataValues : ARRAY [1..50] OF UAHADataValue(ValueSize:=SIZEOF(LREAL));
fMyValue := 27.75;
aDataValues[1].Value           := ADR(fMyValue);
aDataValues[1].StatusCode      := UAHAUpdateStatusCode.HistorianRaw;
aDataValues[1].SourceTimeStamp := F_GetSystemTime();
```

在本示例中，定义了一个包含 50 个值的字段，每个值由一个数据对象表示。将变量 fMyValue (= 27.75) 的当前内容分配给第一个值。

现在，可以在随后的 PLC 循环中通过进一步分配来填补该字段。

要求

开发环境	目标平台	要包括的 PLC 库
TwinCAT 3.1 >= 4024.1	Win32、Win64、WinCE-x86	Tc3_PLCopen_OpcUa >= v3.1.9.0

5.2.2.24 UAHAUpdateStatusCode

已为使用 OPC UA 历史访问功能传输的每个数据值分配一个状态代码。这是 UAHADataValue [▶ 75] 对象的一个属性。

语法

```
{attribute 'qualified_only'}
TYPE UAHAUpdateStatusCode :
(
    HistorianRaw          := 0,          // A raw data value.
    HistorianCalculated   := 1,   // A data value which was calculated.
    HistorianInterpolated := 2, // A data value which was interpolated.
    Reserved              := 3,             // Undefined.
    HistorianPartial      := 4,      // A data value which was calculated with an incomplete interva
l.
    HistorianExtraData    := 8,    // A raw data value that hides other data at the same timestamp.
    HistorianMultiValue   := 16   // Multiple values match the Aggregate criteria (i.e. multiple min
```

```
imum values at different timestamps within the same interval).
) UDINT;
END_TYPE
```

值

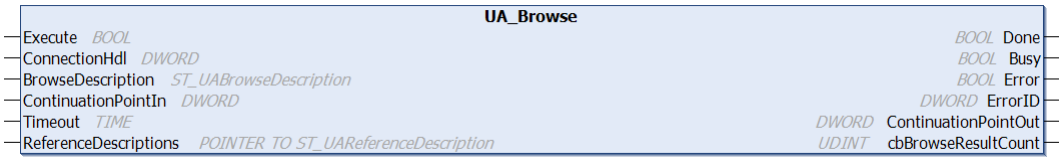
名称	描述
HistorianRaw	HistorianRaw
HistorianCalculated	HistorianCalculated
HistorianInterpolated	HistorianInterpolated
保留	保留
HistorianPartial	HistorianPartial
HistorianExtraData	HistorianExtraData
HistorianMultiValue	HistorianMultiValue

要求

开发环境	目标平台	要包括的 PLC 库
TwinCAT 3.1 >= 4024.1	Win32、Win64、WinCE-x86	Tc3_PLCopen_OpcUa >= v3.1.9.0

5.2.3 功能块

5.2.3.1 UA_Browse



该功能块允许浏览服务器的命名空间。从起始节点开始，读取并相应返回其参考案例。

输入

```
VAR_INPUT
    Execute             : BOOL;
    ConnectionHdl       : DWORD;
    BrowseDescription   : ST_UABrowseDescription;
    ContinuationPointIn : DWORD;
    Timeout             : TIME;
END_VAR
```

名称	类型	描述
Execute	BOOL	命令由该输入端的上升沿触发。
ConnectionHdl	DWORD	之前由功能块 UA_Connect 输出的连接句柄。
BrowseDescription	ST_UABrowseDescription [▶ 70]	此处指定要读取的节点地址信息。
ContinuationPointIn	DWORD	如果之前调用的功能块返回了一个 ContinuationPointOut 值，则可以在此处创建该值，以便从服务器获取更多数据。
超时	TIME	功能中止前的时间。

输入/输出

```
VAR_IN_OUT
    ReferenceDescriptions : POINTER TO ST_UAReferenceDescriptions;
END_VAR
```

名称	类型	描述
ReferenceDescription	指向 ST_UAReferenceDescription 的指针	包含服务器返回的 ReferenceDescription 列表，即 UA_Browse 调用的结果。然后，所包含的 ReferenceDescription 可用于在 BrowseDescription 中进一步调用 UA_Browse，例如深入命名空间。

输出

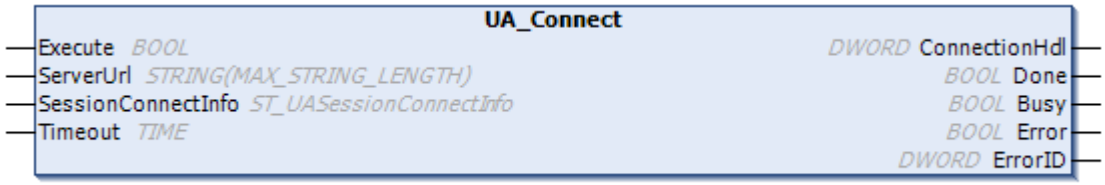
```
VAR_OUTPUT
    Done                : BOOL;
    Busy                : BOOL;
    Error               : BOOL;
    ErrorID             : DWORD;
    ContinuationPointOut : DWORD;
    cbBrowseResultCnt   : UDINT;
END_VAR
```

名称	类型	描述
已完成	BOOL	如果功能块执行成功，则切换为 TRUE。
Busy	BOOL	在功能块执行命令之前为 TRUE，最多在输入的“超时”期间。只要“busy” = TRUE，输入就不接受新命令。监控的不是连接时间，而是接收时间。
Error	BOOL	如果在执行命令时发生错误，则切换为 TRUE。特定命令的错误代码包含在 ErrorID 中。
ErrorID	DWORD	包含最近执行命令的特定命令错误代码。
ContinuationPointOut	DWORD	如果服务器批量返回数据 (ContinuationPointOut != 0)，则在下一次调用功能块时，可将 ContinuationPointOut 的值用作 ContinuationPointIn，以获取更多数据。
cbBrowseResultCnt	UDINT	ReferenceDescription 的数量。

要求

开发环境	目标平台	要包括的 PLC 库
TwinCAT 3.1	Win32、Win64、CE-X86、CE-Arm®	Tc3_PLCopen_OpcUa

5.2.3.2 UA_Connect



该功能块建立与另一个 OPC UA 服务器的 OPC UA 远程连接连接，该连接通过 ServerUrl 和 SessionConnectInfo 指定。该功能块返回一个连接句柄，可用于其他功能块，如 UA_Read。

输入

```
VAR_INPUT
    Execute            : BOOL;
    ServerUrl          : STRING(MAX_STRING_LENGTH);
    SessionConnectInfo : ST_UASessionConnectInfo;
    Timeout            : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

名称	类型	描述
Execute	BOOL	命令由该输入端的上升沿触发。
ServerUrl	STRING(MAX_STRING_LENGTH)	OPC UA 服务器 URL，即“opc.tcp://172.16.3.207:4840”或“opc.tcp://CX_0193BF:4840”。
SessionConnectInfo	ST_UASessionConnectInfo	连接信息（见 ST_UASessionConnectInfo [▶ 71]）
超时	TIME	功能中止前的时间。DEFAULT_ADS_TIMEOUT 是一个全局常量，设置为 5 秒。必须将该值设置为与 ST_UASessionConnectInfo.tConnectionTimeout 匹配。经验法则是：ADS 超时 > 2 * ConnectionTimeout。

输出

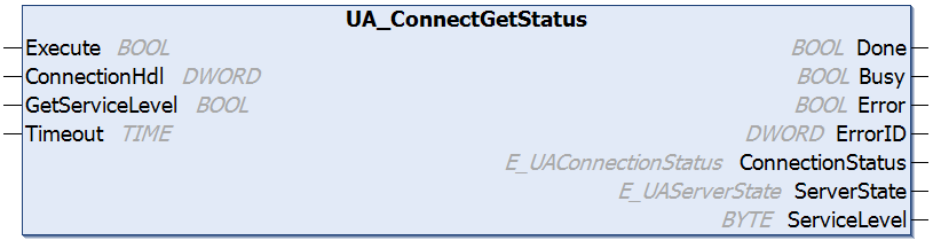
```
VAR_OUTPUT
    ConnectionHdl : DWORD;
    Done          : BOOL;
    Busy          : BOOL;
    Error         : BOOL;
    ErrorID       : DWORD;
END_VAR
```

名称	类型	描述
ConnectionHdl	DWORD	OPC UA 连接句柄。
已完成	BOOL	如果功能块执行成功，则切换为 TRUE。
Busy	BOOL	在功能块执行命令之前为 TRUE，最多在输入的“超时”期间。只要“busy” = TRUE，输入就不接受新命令。监控的不是连接时间，而是接收时间。
Error	BOOL	如果在执行命令时发生错误，则切换为 TRUE。特定命令的错误代码包含在 ErrorID 中。
ErrorID	DWORD	包含最近执行命令的特定命令错误代码。

要求

开发环境	目标平台	要包括的 PLC 库
TwinCAT 3.1	Win32、Win64、CE-X86、CE-Arm®	Tc3_PLCopen_OpcUa

5.2.3.3 UA_ConnectGetStatus



该功能块检查与另一个 OPC UA 服务器的现有连接的连接状态。连接通过各自的连接句柄进行引用。然后以 E_UAConnectionStatus [▶ 65] 返回状态。根据内部会话信息或 OPC UA 心跳报文确定连接状态，不执行额外通信（读取或类似操作）。

OPC UA 服务器的服务级别可通过附加输入参数 GetServiceLevel 读取。为此，会在后台向服务器发送读取命令，以确定这些信息。

输入

```
VAR_INPUT
    Execute             : BOOL;
    ConnectionHdl       : DWORD;
    GetServiceLevel     : BOOL;
    Timeout             : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

名称	类型	描述
Execute	BOOL	命令由该输入端的上升沿触发。
ConnectionHdl	DWORD	现有链接的连接句柄。
GetServiceLevel	BOOL	读出 OPC UA 服务器的 ServiceLevel。
超时	TIME	功能中止前的时间。DEFAULT_ADS_TIMEOUT 是一个全局常量，设置为 5 秒。必须将该值设置为与 ST_UASessionConnectInfo.tConnectionTimeout 匹配。经验法则是：ADS 超时 > 2 * ConnectionTimeout。

输出

```
VAR_OUTPUT
    Done              : BOOL;
    Busy              : BOOL;
    Error             : BOOL;
    ErrorID           : DWORD;
    ConnectionStatus  : E_UAConnectionStatus;
    ServerState       : E_UAServerState;
    ServiceLevel      : BYTE;
END_VAR
```

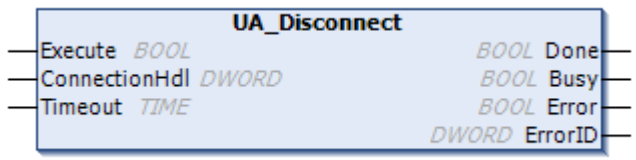
名称	类型	描述
已完成	BOOL	如果功能块执行成功，则切换为 TRUE。
Busy	BOOL	在功能块执行命令之前为 TRUE，最多在输入的“超时”期间。只要“busy” = TRUE，输入就不接受新命令。监控的不是连接时间，而是接收时间。
Error	BOOL	如果在执行命令时发生错误，则切换为 TRUE。特定命令的错误代码包含在 ErrorID 中。
ErrorID	DWORD	包含最近执行命令的特定命令错误代码。
ConnectionStatus	E_UAConnectionStatus	连接状态（见 E_UAConnectionStatus [▶ 65]）。
ServerState	E_UAServerState	服务器状态（见 E_UAServerState [▶ 69]）。
ServerState	BYTE	OPC UA 服务器的 ServiceLevel。

要求

开发环境	目标平台	要包括的 PLC 库
TwinCAT 3.1	Win32、Win64、CE-X86、CE-Arm®	Tc3_PLCopen_OpcUa

PLC 库	所需版本
Tc3_PLCopen_OpcUa	>= 3.2.11.0

5.2.3.4 UA_Disconnect



该功能块会关闭与另一个 OPC UA 服务器的 OPC UA 远程连接。连接通过其连接句柄指定。

断开所有连接

如果调用 UA-Disconnect 方法并传递 0 的连接句柄，那么 OPC UA 客户端便会断开所有现有连接。这也适用于通过 OPC UA I/O 客户端配置建立的连接。

输入

```
VAR_INPUT
    Execute           : BOOL;
    ConnectionHdl     : DWORD;
    Timeout           : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

名称	类型	描述
Execute	BOOL	命令由该输入端的上升沿触发。
ConnectionHdl	DWORD	之前由功能块 UA_Connect 输出的连接句柄。
超时	TIME	功能中止前的时间。DEFAULT_ADS_TIMEOUT 是一个全局常量，设置为 5 秒。

输出

```
VAR_OUTPUT
    Done         : BOOL;
    Busy         : BOOL;
    Error        : BOOL;
    ErrorID      : DWORD;
END_VAR
```

名称	类型	描述
已完成	BOOL	如果功能块执行成功，则切换为 TRUE。
Busy	BOOL	在功能块执行命令之前为 TRUE，最多在输入的“超时”期间。只要“busy” = TRUE，输入就不接受新命令。监控的不是连接时间，而是接收时间。
Error	BOOL	如果在执行命令时发生错误，则切换为 TRUE。特定命令的错误代码包含在 nErrorID 中。
ErrorID	DWORD	包含最近执行命令的特定命令错误代码。

要求

开发环境	目标平台	要包括的 PLC 库
TwinCAT 3.1	Win32、Win64、CE-X86、CE-Arm®	Tc3_PLCopen_OpcUa

5.2.3.5 UA_GetNamespaceIndex

该功能块会收集命名空间 URI 的命名空间索引。例如，如果使用 UA_Read [▶ 92] 或 UA_Write [▶ 94] 功能块，则需要使用命名空间索引来识别符号。

输入

```
VAR_INPUT
    Execute       : BOOL;
    ConnectionHdl : DWORD;
    NamespaceUri  : STRING(MAX_STRING_LENGTH);
    Timeout       : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

名称	类型	描述
Execute	BOOL	命令由该输入端的上升沿触发。
ConnectionHdl	DWORD	之前由功能块 UA_Connect 输出的连接句柄。
NamespaceUri	STRING	要解析的命名空间 URI。对于 TwinCAT OPC UA 服务器，这是第一个 PLC 运行时的“urn:BeckhoffAutomation:Ua:PLC1”。
超时	TIME	功能中止前的时间。DEFAULT_ADS_TIMEOUT 是一个全局常量，设置为 5 秒。

输出

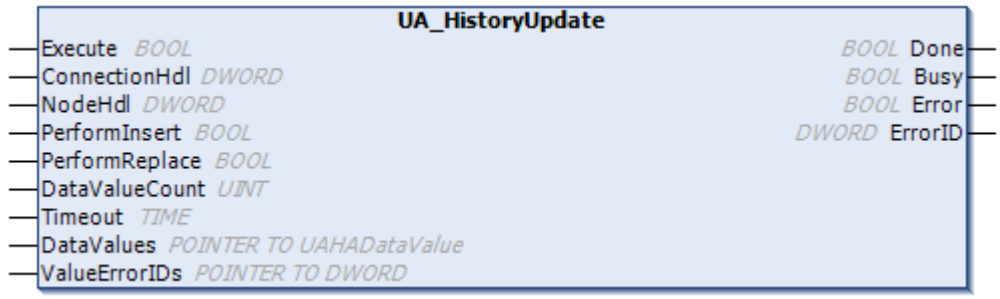
```
VAR_OUTPUT
    NamespaceIndex : UINT;
    Done           : BOOL;
    Busy           : BOOL;
    Error          : BOOL;
    ErrorID        : DWORD;
END_VAR
```

名称	类型	描述
NamespaceIndex	UINT	给定命名空间 URI 的命名空间索引。这可用于其他功能块，如 UA_NodeGetHandle 或 UA_MethodGetHandle。
已完成	BOOL	如果功能块执行成功，则切换为 TRUE。
Busy	BOOL	在功能块执行命令之前为 TRUE，最多在输入的“超时”期间。只要“busy” = TRUE，输入就不接受新命令。监控的不是连接时间，而是接收时间。
Error	BOOL	如果在执行命令时发生错误，则切换为 TRUE。特定命令的错误代码包含在 ErrorID 中。
ErrorID	DWORD	包含最近执行命令的特定命令错误代码。

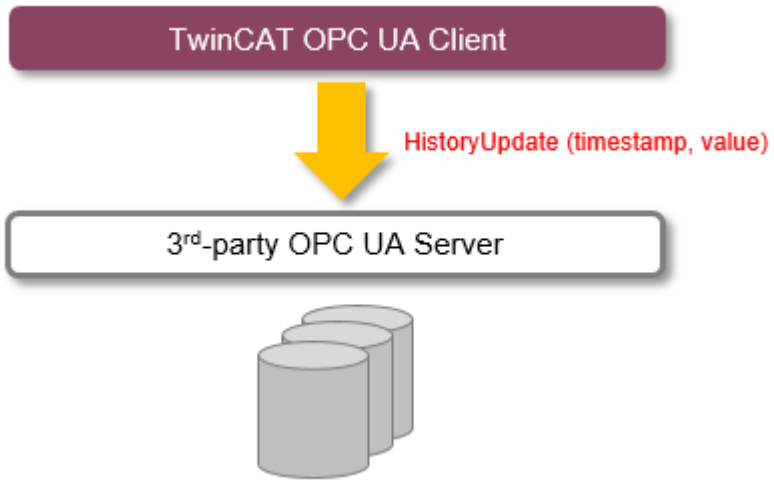
要求

开发环境	目标平台	要包括的 PLC 库
TwinCAT 3.1	Win32、Win64、CE-X86、CE-Arm®	Tc3_PLCopen_OpcUa

5.2.3.6 UA_HistoryUpdate



该功能块通过 OPC UA 向支持 OPC UA HistoryUpdate 功能的服务器（如 TwinCAT OPC UA 服务器）发送历史数据。只需一次调用，就能为节点句柄向服务器传输包括时间戳在内的大量数值。服务器可确保传输的数值保存在数据存储器中，并可通过“历史访问”进行访问。



如果要传输多个节点句柄（不同变量）的值，则可以对该功能块执行多次实例化。

与 TwinCAT OPC UA 服务器一起运行

如果您在 TwinCAT OPC UA 服务器中使用“历史访问”，并希望提供某个时间段内的数据，例如，在该时间段内某个特殊的机器状态，那么该功能块就非常适合。可以有目的地传输所需的周期值。

另一方面，如果数值是周期性发送的，并且要通过“历史访问”在服务器中提供，那么服务器端的“历史访问”功能会更合适，因为在这种情况下，只需在配置器中配置记录节点并设置所需的采样率即可。

另请参见：程序示例 TF6100_OPCUA_HASample

输入

```
VAR_INPUT
    Execute         : BOOL;
    ConnectionHdl   : DWORD;
    NodeHdl         : DWORD;
    PerformInsert   : BOOL;
    PerformReplace  : BOOL;
    DataValueCount  : UINT;
    Timeout         : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

名称	类型	描述
Execute	BOOL	命令由该输入端的上升沿触发。
ConnectionHdl	DWORD	之前由功能块 UA_Connect 输出的连接句柄。
NodeHdl	DWORD	之前由功能块 UA_NodeGetHandle 输出的节点句柄。
PerformInsert	BOOL	默认为 TRUE。
PerformReplace	BOOL	默认为 FALSE。如果给定时间戳的值已经存在于历史记录中，那么在已设置 PerformReplace 选项的情况下 (= TRUE)，则应替换该值。目前，只有 SQL 适配器可以选择该选项。其他适配器不支持该选项。
DataValueCount	UINT	定义传输值的数量。最多支持 1000 个值。
超时	TIME	功能中止前的时间。DEFAULT_ADS_TIMEOUT 是一个全局常量，设置为 5 秒。

输入/输出

```
VAR_IN_OUT
    DataValues      : ARRAY[*] OF UAHADataValue;
    ValueErrorIDs   : ARRAY[*] OF DWORD;
END_VAR
```

名称	类型	描述
DataValue（只读）	ARRAY	所有收集到的值均以 UAHADataValue 类型字段的形式传输。字段的长度没有规定，但必须至少与 DataValueCount 的规格一致。在内部，这些值仅供读取。
ValueErrorID（只写）	ARRAY	执行命令后，该字段包含每个值的错误代码。字段的长度必须至少与 DataValueCount 的规格一致。如果一个或多个值报错，也会通过功能块的输出 Error 和 ErrorID 发出信号。借助该字段，您可以确定哪个值发生了错误。例如，错误代码 16#80000000 表示操作失败，即无法写入数值。

输出

```
VAR_OUTPUT
    Done       : BOOL;
    Busy       : BOOL;
    Error      : BOOL;
    ErrorID    : DWORD;
END_VAR
```

名称	类型	描述
已完成	BOOL	如果功能块执行成功，则切换为 TRUE。
Busy	BOOL	在功能块执行命令之前为 TRUE，最多在输入的“超时”期间。只要“busy” = TRUE，输入就不接受新命令。监控的不是连接时间，而是接收时间。
Error	BOOL	如果在执行命令时发生错误，则切换为 TRUE。特定命令的错误代码包含在 ErrorID 中。
ErrorID	DWORD	包含最近执行命令的特定命令 ADS 错误代码。

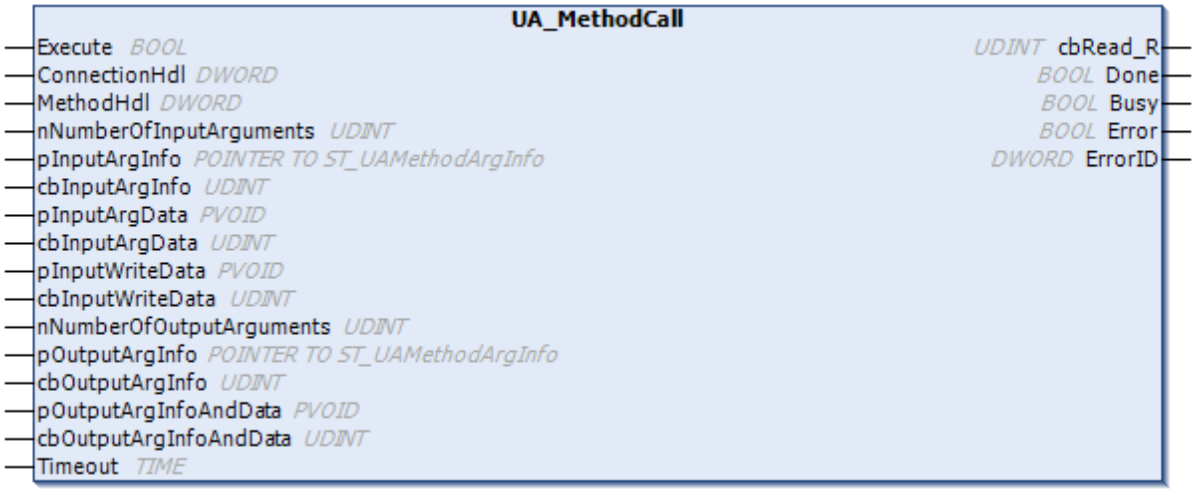
传输的数值数量

数量越大，所需的计算工作量就越大，因此执行命令时 PLC 的执行时间就越长。

要求

开发环境	目标平台	要包括的 PLC 库
TwinCAT 3.1 >= 4024.1	Win32、Win64、WinCE-x86	Tc3_PLCopen_OpcUa >= v3.1.9.0

5.2.3.7 UA_MethodCall



该功能块会调用远程 UA 服务器上的一个方法。方法由连接和方法句柄决定。前者可通过 UA_Connect [▶ 78] 查询，后者可通过 UA_MethodGetHandle [▶ 86] 查询。

输入

```
VAR_INPUT
    Execute                 : BOOL;
    ConnectionHdl           : DWORD;
    MethodHdl               : DWORD;
    nNumberOfInputArguments : UDINT;
    pInputArgInfo           : POINTER TO ST_UAMethodArgInfo;
    cbInputArgInfo          : UDINT;
    pInputArgData           : PVOID;
    cbInputArgData          : UDINT;
    pInputWriteData         : PVOID;
    cbInputWriteData        : UDINT;
    nNumberOfOutputArguments : UDINT;
    pOutputArgInfo          : POINTER TO ST_UAMethodArgInfo;
    cbOutputArgInfo         : UDINT;
    pOutputArgInfoAndData   : PVOID;
    cbOutputArgInfoAndData  : UDINT;
    Timeout                 : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

名称	类型	描述
Execute	BOOL	命令由该输入端的上升沿触发。
ConnectionHdl	DWORD	之前由功能块 UA_Connect 输出的连接句柄。
MethodHdl	DWORD	之前由功能块 UA_MethodGetHandle 输出的方法句柄。
nNumberOfInputArguments	UDINT	输入参数数量。
pInputArgInfo	指向 ST_UAMethodArgInfo 的指针	指向以数组 ST_UAMethodArgInfo 形式存储输入参数信息的缓冲区地址的指针。
cbInputArgInfo	UDINT	存储输入参数信息的缓冲区大小。
pInputArgData	PVOID	指向存储输入参数（恒定长度）的缓冲区地址的指针。
cbInputArgData	UDINT	存储输入参数（恒定长度）的输入缓冲区的大小。
pInputWriteData	PVOID	指向存储输入参数（动态长度）的缓冲区地址的指针。
cbInputWriteData	UDINT	存储输入参数（动态长度）的输入缓冲区大小。
nNumberOfOutputArguments	UDINT	输出参数数量。
pOutputArgInfo	指向 ST_UAMethodArgInfo 的指针	指向以数组 ST_UAMethodArgInfo 形式存储输出参数信息的缓冲区地址的指针。 nLenData 用于确定各个输出参数的目标内存。其他元素的设置方式可以是对返回参数进行类型检查或保持未定义。
cbOutputArgInfo	UDINT	存储输出参数信息的缓冲区大小。
pOutputArgInfoAndData	PVOID	指向将输出参数保存为 BYTE 数组的缓冲区地址的指针。BYTE 数组包含作为 DINT 的输出参数数量、四个预留字节和作为 ST_UAMethodArgInfo [▶ 73] 的 ARRAY 的参数信息（输出参数的长度），之后是纯数据。请注意，数据是以 1 字节对齐方式打包的。
cbOutputArgInfoAndData	UDINT	将输出参数保存为 BYTE 数组的缓冲区大小。

名称	类型	描述
超时	TIME	功能中止前的时间。DEFAULT_ADS_TIMEOUT 是一个全局常量，设置为 5 秒。

输出

```
VAR_OUTPUT
    cbRead_R      : UDINT;
    Done          : BOOL;
    Busy          : BOOL;
    Error         : BOOL;
    ErrorID       : UDINT;
END_VAR
```

名称	类型	描述
cbRead_R	UDINT	对收到的所有字节进行计数。
已完成	BOOL	如果功能块执行成功，则切换为 TRUE。
Busy	BOOL	在功能块执行命令之前为 TRUE，最多在输入的“超时”期间。只要“busy” = TRUE，输入就不接受新命令。监控的不是连接时间，而是接收时间。
Error	BOOL	如果在执行命令时发生错误，则切换为 TRUE。特定命令的错误代码包含在 nErrorID 中。
ErrorID	UDINT	包含最近执行命令的特定命令错误代码。

要求

开发环境	目标平台	要包括的 PLC 库
TwinCAT 3.1	Win32、Win64、CE-X86、CE-Arm®	Tc3_PLCopen_OpcUa

5.2.3.8 UA_MethodGetHandle

此功能块可收集 UA 方法的句柄，然后使用 UA_MethodCall [▶ 84] 调用方法。

输入

```
VAR_INPUT
    Execute           : BOOL;
    ConnectionHdl     : DWORD;
    ObjectNodeID      : ST_UANodeID;
    MethodNodeID      : ST_UANodeID;
    Timeout           : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

名称	类型	描述
Execute	BOOL	命令由该输入端的上升沿触发。
ConnectionHdl	DWORD	之前由功能块 UA_Connect 输出的连接句柄。
ObjectNodeID	ST_UANodeID	要调用的方法的对象节点 ID。（类型：ST_UANodeID [▶ 73]）。
MethodNodeID	ST_UANodeID	要调用方法的方法节点 ID。与 UA 命名空间中的 ID 属性相对应。（类型：UA_Connect [▶ 78]）。
超时	TIME	功能中止前的时间。DEFAULT_ADS_TIMEOUT 是一个全局常量，设置为 5 秒。

输出

```
VAR_OUTPUT
    MethodHdl    : DWORD;
    Done         : BOOL;
    Busy         : BOOL;
    Error        : BOOL;
    ErrorID      : UDINT;
END_VAR
```

名称	类型	描述
MethodHdl	DWORD	返回可用于通过 UA_MethodCall [▶ 84] 调用方法的方法句柄。
已完成	BOOL	如果功能块执行成功，则切换为 TRUE。
Busy	BOOL	在功能块执行命令之前为 TRUE，最多在输入的“超时”期间。只要“busy” = TRUE，输入就不接受新命令。监控的不是连接时间，而是接收时间。
Error	BOOL	如果在执行命令时发生错误，则切换为 TRUE。特定命令的错误代码包含在 nErrorID 中。
ErrorID	UDINT	包含最近执行命令的特定命令错误代码。

要求

开发环境	目标平台	要包括的 PLC 库
TwinCAT 3.1	Win32、Win64、CE-X86、CE-Arm®	Tc3_PLCopen_OpcUa

5.2.3.9 UA_MethodReleaseHandle

此功能块会释放指定的方法句柄。

输入

```
VAR_INPUT
    Execute           : BOOL;
    ConnectionHdl     : DWORD;
    MethodHdl         : DWORD;
    Timeout           : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

名称	类型	描述
Execute	BOOL	命令由该输入端的上升沿触发。
ConnectionHdl	DWORD	之前由功能块 UA_Connect 输出的连接句柄。
MethodHdl	DWORD	之前由功能块 UA_MethodGetHandle 输出的方法句柄。
超时	TIME	功能中止前的时间。DEFAULT_ADS_TIMEOUT 是一个全局常量，设置为 5 秒。

输出

```
VAR_OUTPUT
    Done      : BOOL;
    Busy      : BOOL;
    Error     : BOOL;
    ErrorID   : UDINT;
END_VAR
```

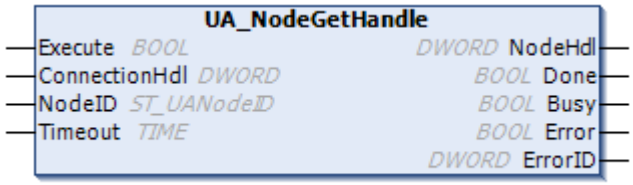
名称	类型	描述
已完成	BOOL	如果功能块执行成功，则切换为 TRUE。

名称	类型	描述
Busy	BOOL	在功能块执行命令之前为 TRUE，最多在输入的“超时”期间。只要“busy” = TRUE，输入就不接受新命令。监控的不是连接时间，而是接收时间。
Error	BOOL	如果在执行命令时发生错误，则切换为 TRUE。特定命令的错误代码包含在 nErrorID 中。
ErrorID	UDINT	包含最近执行命令的特定命令 ADS 错误代码。

要求

开发环境	目标平台	要包括的 PLC 库
TwinCAT 3.1	Win32、Win64、CE-X86、CE-Arm®	Tc3_PLCopen_OpcUa

5.2.3.10 UA_NodeGetHandle



该功能块用于查询 UA 命名空间中给定符号的节点句柄。符号由连接句柄及其节点 ID 指定。

输入

```
VAR_INPUT
    Execute       : BOOL;
    ConnectionHdl : DWORD;
    NodeID        : ST_UANodeID;
    Timeout       : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

名称	类型	描述
Execute	BOOL	命令由该输入端的上升沿触发。
ConnectionHdl	DWORD	之前由功能块 UA_Connect 输出的连接句柄。
节点 ID	ST_UANodeID	UA 节点的唯一寻址，由 Identifier、IdentifierType 和 NamespaceIndex 组成，可通过 NamespaceName（例如 UA_GetNamespaceIndex [▶ 81] 方法）进行解析。
超时	TIME	功能中止前的时间。DEFAULT_ADS_TIMEOUT 是一个全局常量，设置为 5 秒。

输出

```
VAR_OUTPUT
    NodeHdl     : DWORD;
    Done        : BOOL;
    Busy        : BOOL;
    Error       : BOOL;
    ErrorID     : DWORD;
END_VAR
```

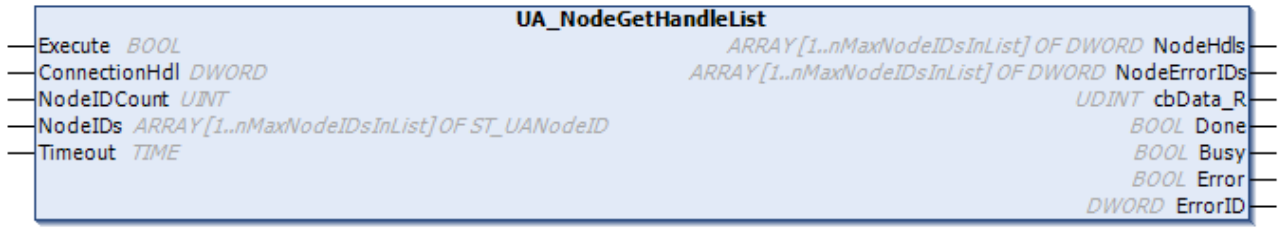
名称	类型	描述
NodeHdl	DWORD	节点句柄，可用于其他功能块，如 UA_Read 或 UA_Write。
已完成	BOOL	如果功能块执行成功，则切换为 TRUE。
Busy	BOOL	在功能块执行命令之前为 TRUE，最多在输入的“超时”期间。只要usy为 TRUE，输入端就不接受新命令。监控的不是连接时间，而是接收时间。
Error	BOOL	如果在执行命令时发生错误，则切换为 TRUE。特定命令的错误代码包含在 ErrorID 中。

名称	类型	描述
ErrorID	DWORD	包含最近执行命令的特定命令 ADS 错误代码。

要求

开发环境	目标平台	要包括的 PLC 库
TwinCAT 3.1	Win32、Win64、CE-X86、CE-Arm®	Tc3_PLCopen_OpcUa

5.2.3.11 UA_NodeGetHandleList



该功能块用于查询 UA 命名空间中节点的节点句柄。

输入

```
VAR_INPUT
    Execute          : BOOL;
    ConnectionHdl    : DWORD;
    NodeIDCount      : UINT;
    NodeIDs          : ARRAY[1..nMaxNodeIDsInList] OF ST_UANodeID;
    Timeout          : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

名称	类型	描述
Execute	BOOL	命令由该输入端的上升沿触发。
ConnectionHdl	DWORD	之前由功能块 UA_Connect 输出的连接句柄。
NodeIDCount	UINT	需要节点句柄的节点数。
NodeID	ARRAY	使用结构 ST_UANodeID [▶ 73] 创建的 NodeID 数组。
超时	TIME	功能中止前的时间。DEFAULT_ADS_TIMEOUT 是一个全局常量，设置为 5 秒。

输出

```
VAR_OUTPUT
    NodeHdls     : ARRAY[1..nMaxNodeIDsInList] OF DWORD;
    NodeErrorIDs : ARRAY[1..nMaxNodeIDsInList] OF DWORD;
    cbData_R     : UDINT;
    Done         : BOOL;
    Busy         : BOOL;
    Error        : BOOL;
    ErrorID      : DWORD;
END_VAR
```

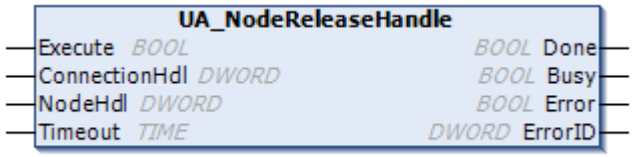
名称	类型	描述
NodeHdl	ARRAY	请求的节点句柄数组。
NodeErrorID	ARRAY	如果没有节点句柄，错误 ID 的数组。
cbData_R	UDINT	读取数据的大小。
已完成	BOOL	如果功能块执行成功，则切换为 TRUE。
Busy	BOOL	在功能块执行命令之前为 TRUE，最多在输入的“超时”期间。只要“busy” = TRUE，输入就不接受新命令。监控的不是连接时间，而是接收时间。
Error	BOOL	如果在执行命令时发生错误，则切换为 TRUE。特定命令的错误代码在 nErrID 中。

名称	类型	描述
ErrorID	DWORD	如果发生错误，则包含错误 ID。

要求

开发环境	目标平台	要包括的 PLC 库
TwinCAT 3.1	Win32、Win64、CE-X86、CE-Arm®	Tc3_PLCopen_OpcUa

5.2.3.12 UA_NodeReleaseHandle



此功能块会释放节点句柄。

输入

```
VAR_INPUT
    Execute       : BOOL;
    ConnectionHdl : DWORD;
    NodeHdl       : DWORD;
    Timeout       : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

名称	类型	描述
Execute	BOOL	命令由该输入端的上升沿触发。
ConnectionHdl	DWORD	之前由功能块 UA_Connect 输出的连接句柄。
NodeHdl	DWORD	要释放的节点句柄。
超时	TIME	功能中止前的时间。DEFAULT_ADS_TIMEOUT 是一个全局常量，设置为 5 秒。

输出

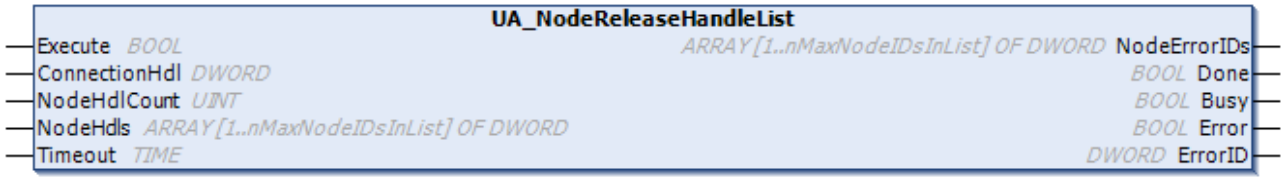
```
VAR_OUTPUT
    Done       : BOOL;
    Busy       : BOOL;
    Error      : BOOL;
    ErrorID    : DWORD;
END_VAR
```

名称	类型	描述
已完成	BOOL	如果功能块执行成功，则切换为 TRUE。
Busy	BOOL	在功能块执行命令之前为 TRUE，最多在输入的“超时”期间。只要“busy” = TRUE，输入就不接受新命令。监控的不是连接时间，而是接收时间。
Error	BOOL	如果在执行命令时发生错误，则切换为 TRUE。特定命令的错误代码包含在 ErrorID 中。
ErrorID	DWORD	包含最近执行命令的特定命令 ADS 错误代码。

要求

开发环境	目标平台	要包括的 PLC 库
TwinCAT 3.1	Win32、Win64、CE-X86、CE-Arm®	Tc3_PLCopen_OpcUa

5.2.3.13 UA_NodeReleaseHandleList



该功能块会释放多个节点句柄。

输入

```
VAR_INPUT
    Execute          : BOOL;
    ConnectionHdl    : DWORD;
    NodeHdlCount     : UINT;
    NodeHdls         : ARRAY[1..nMaxNodeIDsInList] OF DWORD;
    Timeout          : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

名称	类型	描述
Execute	BOOL	命令由该输入端的上升沿触发。
ConnectionHdl	DWORD	之前由功能块 UA_Connect 输出的连接句柄。
NodeHdlCount	UINT	节点句柄数量。
NodeHdl	ARRAY	要释放的节点句柄数组。
超时	TIME	功能中止前的时间。DEFAULT_ADS_TIMEOUT 是一个全局常量，设置为 5 秒。

输出

```
VAR_OUTPUT
    NodeErrorIDs : ARRAY[1..nMaxNodeIDsInList] OF DWORD;
    Done         : BOOL;
    Busy         : BOOL;
    Error        : BOOL;
    ErrorID      : DWORD;
END_VAR
```

名称	类型	描述
NodeErrorID	ARRAY	无法释放节点句柄时的错误 ID 数组。
已完成	BOOL	如果功能块执行成功，则切换为 TRUE。
Busy	BOOL	在功能块执行命令之前为 TRUE，最多在输入的“超时”期间。只要“busy” = TRUE，输入就不接受新命令。监控的不是连接时间，而是接收时间。
Error	BOOL	如果在执行命令时发生错误，则切换为 TRUE。特定命令的错误代码在 nErrID 中。
ErrorID	DWORD	如果发生错误，则包含错误 ID。

要求

开发环境	目标平台	要包括的 PLC 库
TwinCAT 3.1	Win32、Win64、CE-X86、CE-Arm®	Tc3_PLCopen_OpcUa

5.2.3.14 UA_Read

该功能块会从给定的节点和连接句柄读取数值。

输入

```
VAR_INPUT
    Execute           : BOOL;
    ConnectionHdl     : DWORD;
    NodeHdl           : DWORD;
    stNodeAddInfo     : ST_UANodeAdditionalInfo;
    pVariable         : PVOID;
    cbData            : UDINT;
    Timeout           : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

名称	类型	描述
Execute	BOOL	命令由该输入端的上升沿触发。
ConnectionHdl	DWORD	之前由功能块 UA_Connect 输出的连接句柄。
NodeHdl	DWORD	之前由功能块 UA_NodeGetHandle 输出的节点句柄。
stNodeAddInfo	ST_UANodeAdditionalInfo	定义附加信息，例如从 UA 命名空间读取哪个属性（默认：“值”属性）或使用哪个 IndexRange。由结构 ST_UANodeAdditionalInfo [▶ 74] 指定。
pVariable	PVOID	指向要存储读取数据的数据存储器的指针。
cbData	UDINT	确定要读取数据的大小。
超时	TIME	功能中止前的时间。DEFAULT_ADS_TIMEOUT 是一个全局常量，设置为 5 秒。

输出

```
VAR_OUTPUT
    Done          : BOOL;
    Busy          : BOOL;
    Error         : BOOL;
    ErrorID       : UDINT;
    cbData_R      : UDINT;
END_VAR
```

名称	类型	描述
已完成	BOOL	如果功能块执行成功，则切换为 TRUE。
Busy	BOOL	在功能块执行命令之前为 TRUE，最多在输入的“超时”期间。只要“busy” = TRUE，输入就不接受新命令。监控的不是连接时间，而是接收时间。
Error	BOOL	如果在执行命令时发生错误，则切换为 TRUE。特定命令的错误代码包含在 nErrorID 中。
ErrorID	UDINT	包含最近执行命令的特定命令 ADS 错误代码。
cbData_R	UDINT	要读取的字节数。

要求

开发环境	目标平台	要包括的 PLC 库
TwinCAT 3.1	Win32、Win64、CE-X86、CE-Arm®	Tc3_PLCopen_OpcUa

5.2.3.15 UA_ReadList

该功能块会从给定的节点和连接句柄中读取数值。

输入

```
VAR_INPUT
    Execute          : BOOL;
    ConnectionHdl    : DWORD;
    NodeHdlCount     : UINT;
    NodeHdls         : ARRAY[1..nMaxNodeIDsInList] OF DWORD;
    stNodeAddInfo    : ARRAY[1..nMaxNodeIDsInList] OF ST_UANodeAdditionalInfo;
    pVariable        : PVOID;
    cbData           : ARRAY[1..nMaxNodeIDsInList] UDINT;
    cbDataTotal      : UDINT;
    Timeout          : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

名称	类型	描述
Execute	BOOL	命令由该输入端的上升沿触发。
ConnectionHdl	DWORD	之前由功能块 UA_Connect [▶ 78] 输出的连接句柄。
NodeHdlCount	UINT	输入变量 NodeHdl 中存储的节点句柄数量。
NodeHdl	ARRAY	之前由功能块 UA_NodeGetHandle [▶ 88] 或 UA_NodeGetHandleList [▶ 89] 接收的节点句柄数组。
stNodeAddInfo	ARRAY	定义附加信息，例如从 UA 命名空间读取哪个属性（默认：“值”属性）或使用哪个 IndexRange。由结构 ST_UANodeAdditionalInfo [▶ 74] 指定。
pVariable	PVOID	指向要存储读取数据的数据存储器的指针。
cbData	ARRAY	确定要读取数据的大小。
cbDataTotal	UDINT	接收数据的总大小。
超时	TIME	功能中止前的时间。DEFAULT_ADS_TIMEOUT 是一个全局常量，设置为 5 秒。

输出

```
VAR_OUTPUT
    Done       : BOOL;
    Busy       : BOOL;
    Error      : BOOL;
    ErrorID    : UDINT;
    cbData_R   : UDINT;
END_VAR
```

名称	类型	描述
已完成	BOOL	如果功能块执行成功，则切换为 TRUE。
Busy	BOOL	在功能块执行命令之前为 TRUE，最多在输入的“超时”期间。只要“busy” = TRUE，输入就不接受新命令。监控的不是连接时间，而是接收时间。
Error	BOOL	如果在执行命令时发生错误，则切换为 TRUE。特定命令的错误代码在 nErrID 中。

名称	类型	描述
ErrorID	UDINT	包含最近执行命令的特定命令 ADS 错误代码。
cbData_R	UDINT	读取的字节数。

要求

开发环境	目标平台	要包括的 PLC 库
TwinCAT 3.1	Win32、Win64、CE-X86、CE-Arm®	Tc3_PLCopen_OpcUa

5.2.3.16 UA_Write

该功能块会将数值写入给定的节点和连接句柄。

输入

```
VAR_INPUT
    Execute         : BOOL;
    ConnectionHdl   : DWORD;
    NodeHdl         : DWORD;
    stNodeAddInfo   : ST_UANodeAdditionalInfo;
    pVariable       : PVOID;
    cbData          : UDINT;
    Timeout         : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

名称	类型	描述
Execute	BOOL	命令由该输入端的上升沿触发。
ConnectionHdl	DWORD	之前由功能块 UA_Connect [▶ 78] 输出的连接句柄。
NodeHdl	DWORD	之前由功能块 UA_NodeGetHandle [▶ 88] 输出的节点句柄。
stNodeAddInfo	ST_UANodeAdditionalInfo	定义附加信息，例如要写入哪个 IndexRange 或哪个属性（默认使用“值”属性）。由结构 ST_UANodeAdditionalInfo [▶ 74] 指定。
pVariable	PVOID	指向要写入的数据的指针。
cbData	UDINT	设置要写入的值的大小。
超时	TIME	功能中止前的时间。DEFAULT_ADS_TIMEOUT 是一个全局常量，设置为 5 秒。

输出

```
VAR_OUTPUT
    Done        : BOOL;
    Busy        : BOOL;
    Error       : BOOL;
    ErrorID     : DWORD;
END_VAR
```

名称	类型	描述
已完成	BOOL	如果功能块执行成功，则切换为 TRUE。

名称	类型	描述
Busy	BOOL	在功能块执行命令之前为 TRUE，最多在输入的“超时”期间。只要“busy” = TRUE，输入就不接受新命令。监控的不是连接时间，而是接收时间。
Error	BOOL	如果在执行命令时发生错误，则切换为 TRUE。特定命令的错误代码包含在 ErrorID 中。
ErrorID	DWORD	包含最近执行命令的特定命令 ADS 错误代码。

要求

开发环境	目标平台	要包括的 PLC 库
TwinCAT 3.1	Win32、Win64、CE-X86、CE-Arm®	Tc3_PLCopen_OpcUa

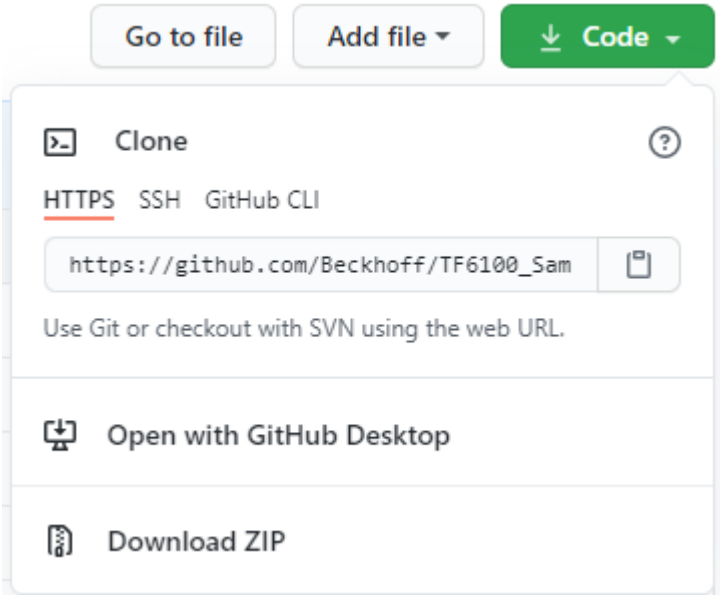
5.2.4 参数列表

Tc3_PLCopen_OpcUa 库中的下列参数会影响通信的工作方式。

名称	类型	默认值	描述
nMaxNodeIDsInList	UINT	10	指定列表中节点的最大数量，例如用于功能块 UA_NodeGetHandleList [▶ 89]、UA_ReadList [▶ 93] 和 UA_NodeReleaseHandleList [▶ 91]。
sNetId	T_AmsNetID	127.0.0.1.1.1	指定运行 TwinCAT OPC UA 客户端的设备的 AMS Net ID。通常，PLC 和客户端在同一系统上运行。

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。

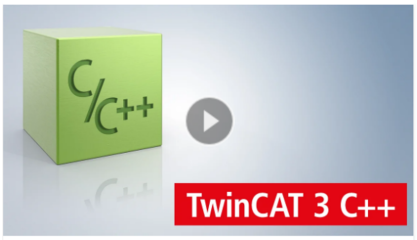


有以下示例：

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 错误诊断

在以下章节中，将以表格的形式列出 OPC UA 设置的所有组件可能出现的错误。此外，还会针对相应的错误提供有用的故障排除提示。

行为	解决措施
使用 PLCopen 功能块从服务器读取或写入 StructuredDataType 的尝试失败。	基于 PLCopen 的客户端不支持结构化数据类型。为此，请使用 I/O 客户端。
执行 PLCopen 功能块时，“忙碌”输出保持为 TRUE，但不会输出错误，也不会触发超时。	这表明 ADS 路由器内存不足。请在 TwinCAT ADS 路由器的相应设置中增加路由器内存。
在使用服务器主机名指定服务器 URL 以创建新的 I/O 客户端时，随后无法通过 AddNode 对话框建立连接。	请检查网络中的名称解析是否正常运行。或者通过服务器的 IP 地址再试一次。
I/O 客户端的某些配置项不存在，尽管根据文档应该存在。	在这种情况下，系统管理器描述文件 (TMC) 可能在 TF6100 更新后未更新。请从 I/O 客户端的上下文菜单中执行“重新加载 TMC”命令，重新加载描述文件。
写入变量的命令未被 I/O 客户端执行或未到达服务器。	请检查 I/O 客户端是否已启用“写入启用”输出。
从产品版本 1.x 升级到 2.x 后，PLCopen 功能块报告 ADS 错误“6”。	要在产品版本 2.x 中使用 PLCopen 功能块，必须在 TwinCAT 项目的 I/O 区域创建虚拟 OPC UA 设备，并将其 AMS Net ID（可在**设置**选项卡中找到）存储 [▶ 62]在 Tc3_PLCopen_OpcUa 库参数列表的“sNetId”变量中。

8.2 状态代码

8.2.1 ADS 返回代码

错误代码分组：
全局错误代码：0x0000 [▶ 98]... （0x9811_0000 ...）
路由器错误代码：0x0500 [▶ 99]... （0x9811_0500 ...）
一般 ADS 错误：0x0700 [▶ 99]... （0x9811_0700 ...）
RTime 错误代码：0x1000 [▶ 101]... （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。

Hex	Dec	HRESULT	名称	描述
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。

Hex	Dec	HRESULT	名称	描述
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。
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。

Hex	Dec	HRESULT	名称	描述
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。
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。

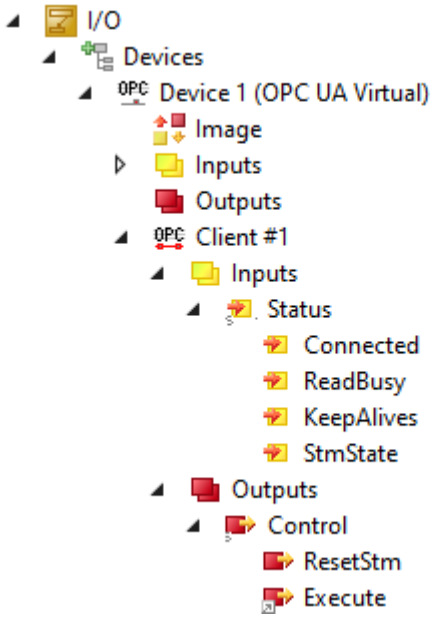
Hex	Dec	名称	描述
更多 Winsock 错误代码：Win32 错误代码			

8.2.2 客户端 I/O

属于虚拟 OPC UA 设备的 OPC UA 客户端模块可提供不同的状态变量以及控制变量。下文将对这些变量进行解释。

读取状态代码

请注意，此处列出的状态机状态代码为十六进制。如果代码在 TwinCAT 中显示为十进制数，则必须进行转换才能进行解释。



附图 1: OPCUAClientModulesStatusCodes

变量	模式	0	1- 状态机状态	2- 如果使用订阅，则保持存活计数	3- 连接状态（和读取繁忙）
状态	0x0123	-	0 = 初始化 (init)		0 = false（和关闭）
			1 = 连接		1 = true（和关闭）
			2 = 解析命名空间		2 = false（和打开）
			3 = 获取节点句柄		3 = true（和打开）
			4 = 连续读取/写入		
			5 = 触发读取/写入		
			6 = 等待数据更改通知（订阅）		
			7 = 断开		
			8 = 重置		
Control	0x0123	-	-	-	0 = 标准（默认） 1 = 重置状态机 2 = 执行（在触发读取模式下）

变量	数据类型	描述
Connected	BIT	1 TRUE | 0= FALSE。

变量	模式	0	1- 状态机状态	2- 如果使用订阅，则保持存活计数	3- 连接状态（和读取繁忙）
ReadBusy		BIT			1 TRUE | 0= FALSE。该功能仅在通过触发变量读取和写入时有效。
KeepAlives		BIT4			显示已计算的 KeepAlive 信息数量。仅在使用订阅进行读取和写入时有效。
StmState		BYTE			可以在上表中读取。
ResetStm		BIT			该位设置为 1 时，客户端复位。
Execute		BIT			1 TRUE | 0= FALSE。如果在通过触发变量进行读取和写入时将该位设置为 1，则会发生读取/写入操作。 **如果该位保持设置状态，则循环读取/写入没有区别。**

8.2.3 客户端 PLCopen

TwinCAT OPC UA 客户端的功能块都有自己的无效代码，这些代码表示发生错误，并使用 ErrorID 显示有关已出现问题的更多信息。可能会出现 HighWord 为 0x0000 的 TwinCAT ADS 错误信息 (ADS 返回代码 [▶ 98]) 和来自客户端或 PLC 库的自定义错误信息（HighWord 为 0xE4DD）。

可能出现的 TwinCAT ADS 错误包括以下几种：

十六进制	名称	描述
0x 0000 0705	DEVICE_INVALIDSIZE	参数大小不正确
0x 0000 0706	DEVICE_INVALIDDATA	参数值无效
0x 0000 070A	DEVICE_NOMEMORY	内存不足

该错误代码列表显示了可能的自定义错误值：

十六进制	名称	描述
0x E4DD 0001	UAC_E_FAIL	UA 服务调用失败
0x E4DD 0100	UAC_E_CONNECTED	服务器已连接
0x E4DD 0101	UAC_E_CONNECT	建立连接时出现一般错误
0x E4DD 0102	UAC_E_UASECURITY	无法设置 UA 安全系统
0x E4DD 0103	UAC_E_ITEMEXISTS	元素 ID 已存在
0x E4DD 0104	UAC_E_ITEMNOTFOUND	元素不存在
0x E4DD 0105	UAC_E_ITEMTYPE	无效或不支持的项目类型
0x E4DD 0106	UAC_E_CONVERSION	无法转换变量类型
0x E4DD 0107	UAC_E_SUSPENDED	设备挂起。请稍后再试.....
0x E4DD 0108	UAC_E_TYPE_NOT_SUPPORTED	不支持转换变量类型。
0x E4DD 0109	UAC_E_NSNAME_NOTFOUND	未找到指定名称的命名空间。
0x E4DD 0110	UAC_E_CONNECT_NOTFOUND	连接失败：无法找到目标主机。
0x E4DD 0111	UAC_E_TIMEOUT	超时：即目标主机没有响应
0x E4DD 0112	UAC_E_INVALIDHDL	会话句柄无效
0x E4DD 0113	UAC_E_INVALIDNODEID	UA 节点 ID 未知
0x E4DD 0114	UAC_E_INVAL_IDENTIFIER_TYPE	UaNodeId 的标识符类型无效
0x E4DD 0115	UAC_E_IDENTIFIER_NOTSUPP	不支持标识符类型 UaNodeId
0x E4DD 0116	UAC_E_INVAL_NODE_HDL	无效的节点句柄

十六进制	名称	描述
0x E4DD 0117	UAC_E_UAREADFAILED	UA 读取失败，原因不明
0x E4DD 0118	UAC_E_UAWRITEFAILED	UA 写入失败，原因不明
0x E4DD 0119	UAC_E_INVAL_NODEMETHOD_HDL	无效的方法句柄
0x E4DD 011A	UAC_E_CALL_FAILED	调用失败，原因不明
0x E4DD 011B	UAC_E_CALLDECODE_FAILED	调用成功，解码返回值失败
0x E4DD 011C	UAC_E_NOTMAPPEDTYPE	返回值中未分配数据类型
0x E4DD 011D	UAC_E_CALL_FAILED_BADINTERNAL	UA_BadInternal 调用失败
0x E4DD 011E	UAC_E_METHODIDINVALID	未知 MethodID（调用时返回，即使 GetMethodHdl 提供了该 ID）
0x E4DD 011F	UAC_E_TOOMUCHDIM	方法调用返回的参数超过 3 个维度；不支持。
0x E4DD 0120	UAC_E_CALL_FAILED_INVALIDARG	调用失败，OpcUa_BadInvalidArgument
0x E4DD 0121	UAC_E_CALL_FAILED_TYPEMISMATCH	调用失败，UAC_E_CALL_FAILED_TYPEMISMATCH
0x E4DD 0122	UAC_E_CALL_FAILED_OUTOFRANGE	调用失败，UAC_E_CALL_FAILED_OUTOFRANGE
0x E4DD 0123	UAC_E_CALL_FAILED_BADSTRUCTURE	调用失败，OpcUa_BadStructureMissing
0x E4DD 0124	UAC_E_CALL_TYPEMISMATCH_OUTPARAM	调用成功，但提供的输出信息类型不匹配
0x E4DD 0125	UAC_E_NONVALIDTYPEINFO	节点类型信息不足
0x E4DD 0126	UAC_E_INVALIDATTRIBID	访问无效节点属性
0x E4DD 0128	UAC_E_NOTSUPPORTED	相连 UaServer 不支持该命令，例如在调用 UA_HistoryUpdate 时。
0x E4DE 0100	UAC_E_INVALID_ARRAY_LENGTH	分配给 UA_HistoryUpdate 的数组长度无效，与 DataValueCount 不匹配。
0x E4DE 0101	UAC_E_INVALID_DATASIZE	为 UA_HistoryUpdate 分配了一个数据类型大小无效的数据值。所有分配的 DataValue 必须是相同的数据类型。
0x E4DE 0102	UAC_E_SUBERROR	至少有一个传输数据值输出了低级错误。请参见 UA_HistoryUpdate 中的 ValueErrorIDs。

8.3 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.

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

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.

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