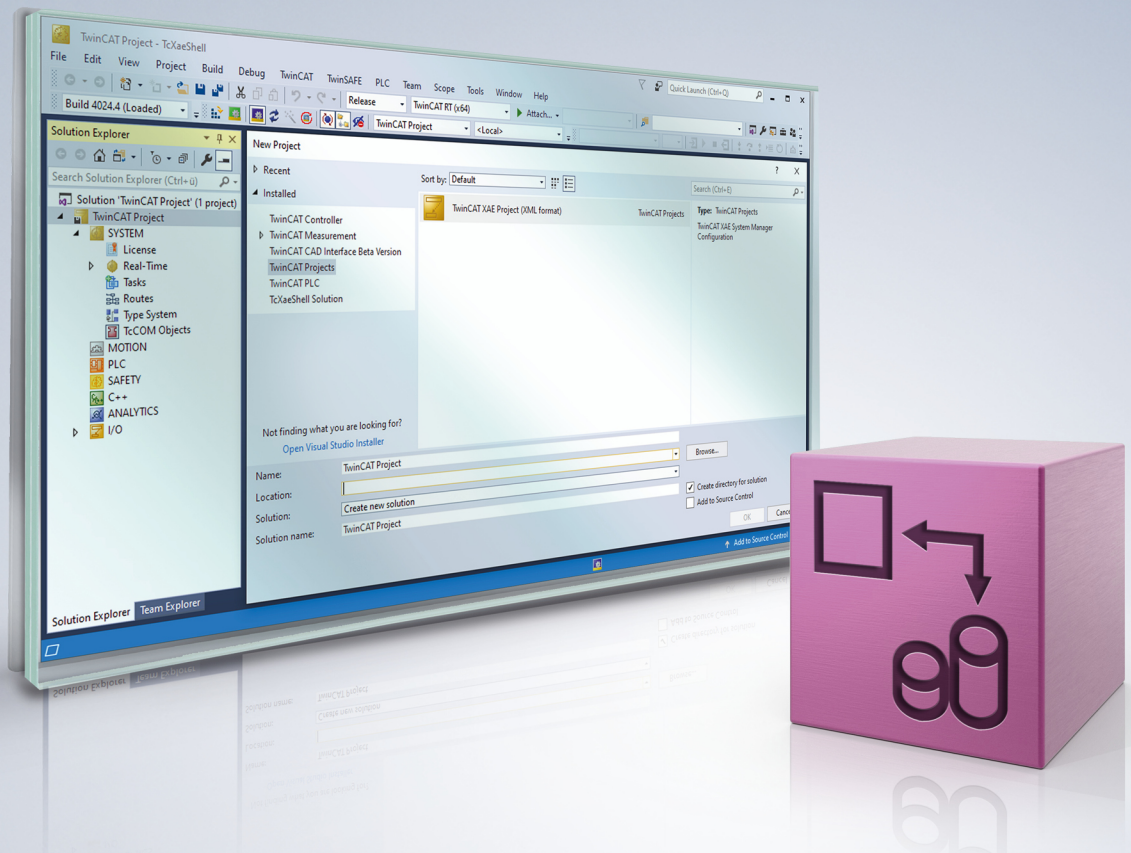
BECKHOFF New Automation Technology

手册 | ZH

TF6420

TwinCAT 3 | Database Server



2025-06-02 | 版本： 1.13.5

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

尽管本文档经过精心编制， 然而，所述产品正在不断开发中。
我们保留随时修订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®、TwinCAT®、TwinCAT/BSD®、TC/BSD®、EtherCAT®、EtherCAT G®、EtherCAT G10®、EtherCAT P®、Safety over EtherCAT®、TwinSAFE®、XFC®、XTS® 和 XPlanar® 是德国倍福自动化有限公司的注册商标并已获得授权。
本文档中所使用的其它名称可能是商标名称，任何第三方为其自身目的而引用，都可能触犯商标所有者的权利。

正在申请的专利

涵盖 EtherCAT 技术，包括但不限于以下专利申请和专利：
EP1590927、EP1789857、EP1456722、EP2137893、DE102015105702
并在多个其他国家进行了相应的专利申请或注册。

EtherCAT® 是注册商标和专利技术，由德国倍福自动化有限公司授权使用。

版权所有

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。

2 概述

TwinCAT 数据库服务器能够实现 TwinCAT 系统与各个数据库系统之间的数据交换。对于小型应用程序，通过配置器可以使用它，而无需对现有的程序代码进行干预。对于复杂的任务，数据库服务器提供了一个大型 PLC 功能块库，以实现高度的灵活性。例如，从 PLC 可以直接使用诸如插入或选择等 SQL 命令。如果需要的话，为了减轻 PLC 的负载，可以存储过程（存储过程），然后从数据库中调用。在这种情况下，数据库将与存储过程联合使用相应的 PLC 功能块所传输的参数，并将结果返回给控制器。

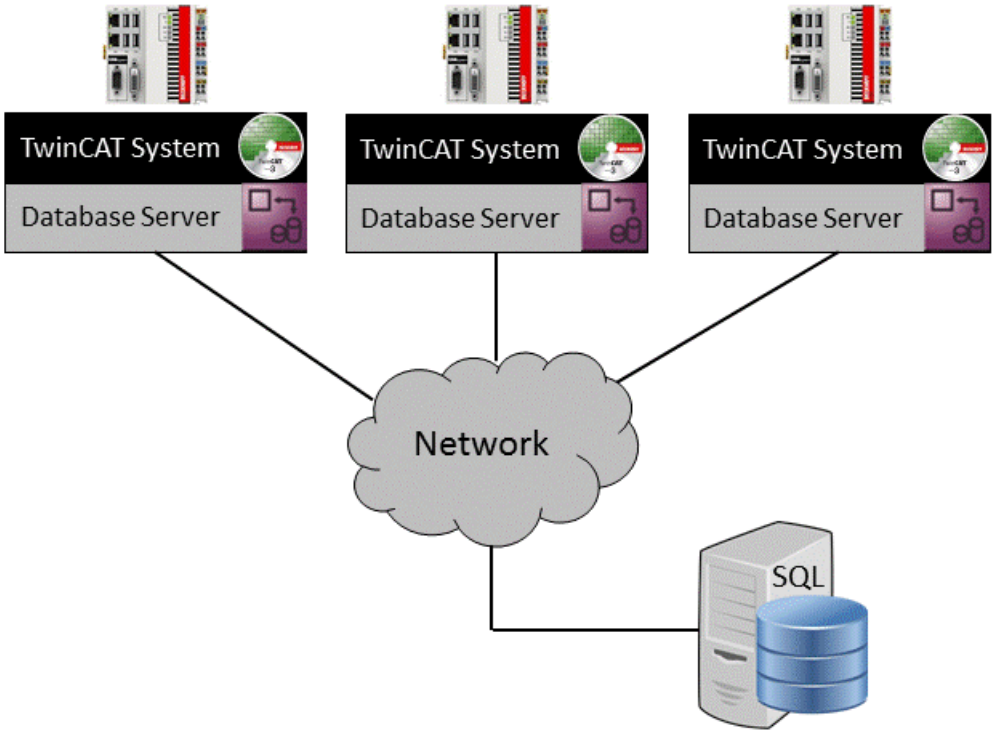
TwinCAT 数据库服务器支持多种不同的数据库系统，MS SQL、MS SQL Compact、MS Access、MySQL、PostgreSQL、DB2、Oracle、Interbase、Firebird、ASCII（例如 .txt 或 .csv）和 XML 文件，现在还包括基于 MongoDB 支持的 NoSQL 数据库。（另请参见：）

组件

- TwinCAT 数据库服务器 [▶ 17]：该服务与 TwinCAT 一起启动和停止。它构成了 TwinCAT 系统和数据库之间的纽带。
- 配置器 [▶ 99]：TwinCAT 数据库服务器配置器有助于直观地设置与相应的数据库进行基本通信所需的数据库参数。
- PLC 库 [▶ 278]：PLC 库包括各种功能块。它们能够建立数据库连接、创建新表、使用插入命令将数据写入任何表结构，以及通过选择命令进行读取。此外，还可以更新或删除数据库条目，并触发存储过程。NoSQL 数据库有自己经过优化的功能块，可以处理 PLC 中的灵活 JSON 文档等。工作原理都是相同的。

工作原理

在 TwinCAT 系统中，数据库服务器通过 ADS 进行通信。在外部，它可以链接到相应的配置数据库。如要查看可能的网络拓扑结构，请参见“应用领域和网络技术 [▶ 18]“部分。



3 安装

3.1 系统要求

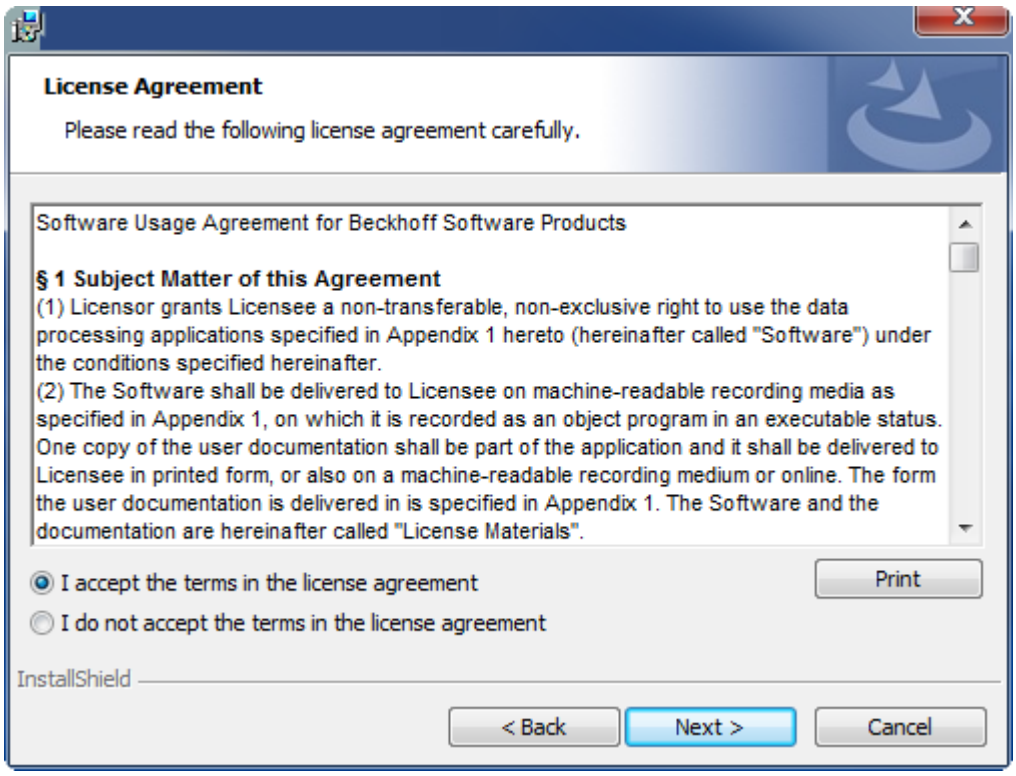
技术数据	TF6420 TwinCAT 3 数据库服务器
目标系统	Windows 10、WinCE、TwinCAT/BSD x86、x64 和 ARM
.NET 框架	.Net 4.5.1 或更高版本 WinCE：.NET 3.5
最低 TwinCAT 版本	3.1.4018
最低 TwinCAT 等级	TC1200 | TwinCAT 3 PLC

3.2 安装

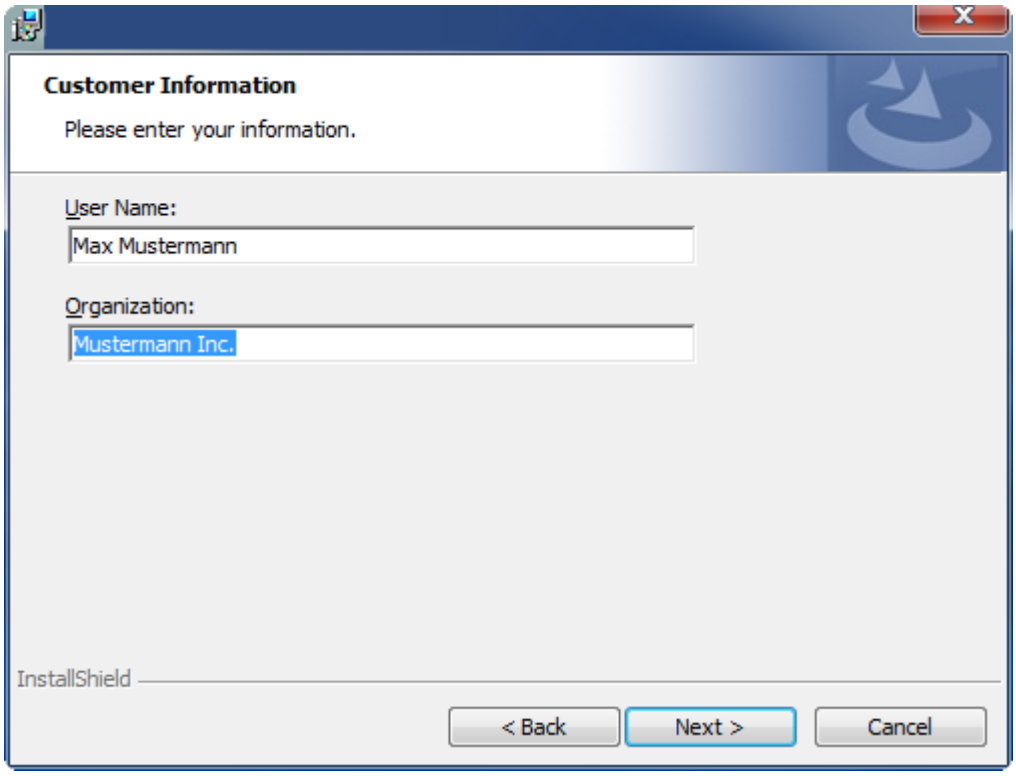
下文将介绍如何在基于 Windows 的操作系统上安装 TwinCAT 3 功能组件。

✓ 从倍福网站下载 TwinCAT 3 功能组件的安装文件。

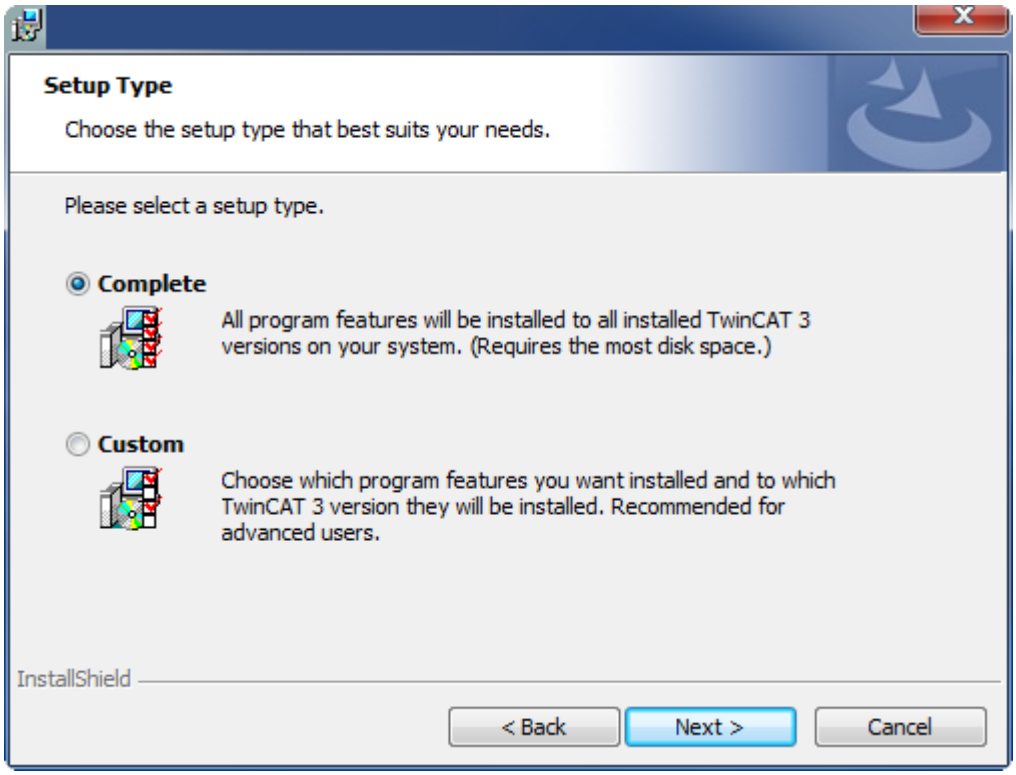
1. 以管理员身份运行安装文件。为此，请右击文件并在菜单中选择“**以管理员身份运行**”命令。
   ⇨ 安装对话框打开。
2. 接受最终用户许可协议，然后点击“**Next**”（下一步）。



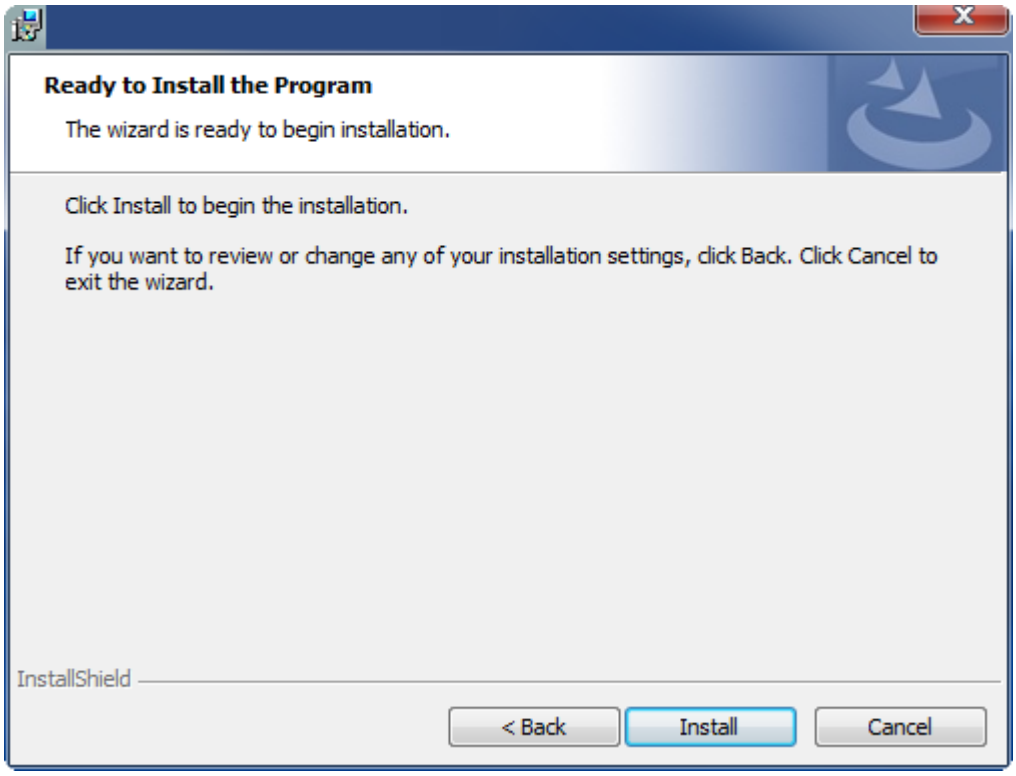
3. 输入用户数据。



4. 如果要安装完整版 TwinCAT 3 功能组件，请选择“**Complete**”（完整版）作为安装类型。如果要单独安装 TwinCAT 3 功能组件，请选择“**Custom**”（自定义）。

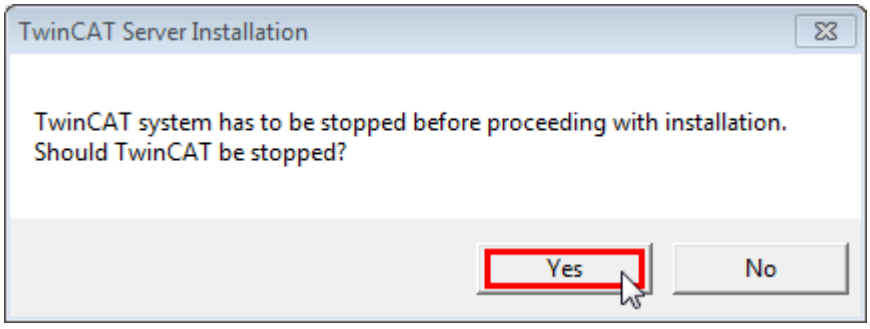


5. 选择“**Next**”（下一步），然后选择“**Install**”（安装）开始安装。

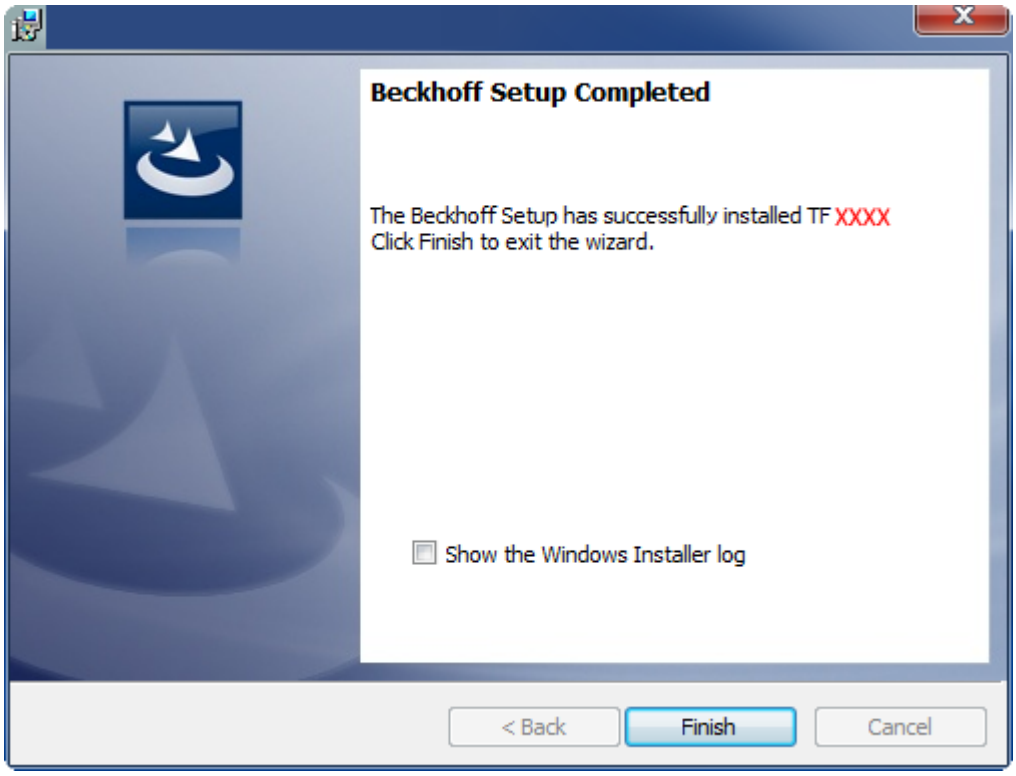


⇨ 将显示一个对话框，提示您必须关闭 TwinCAT 系统才能继续安装。

6. 选择“**Yes**”（是）。



7. 选择“**Finish**”（完成）退出安装。



⇨ TwinCAT 3 功能组件已成功安装。

3.3 授权

TwinCAT 3 功能可以作为完整版或 7 天测试版激活。这两种许可证类型都可以通过 TwinCAT 3 开发环境（XAE）激活。

授权使用 TwinCAT 3 功能的完整版本

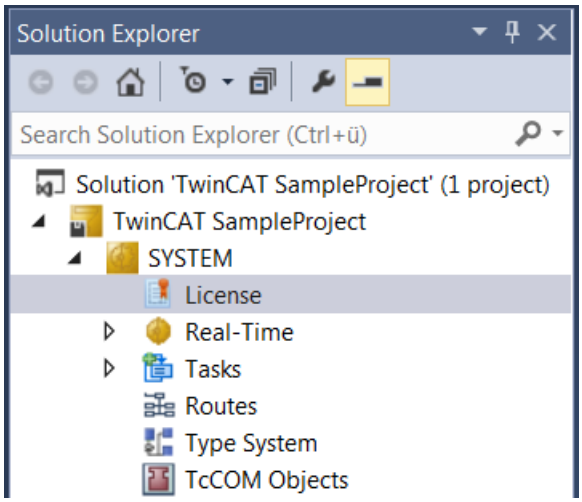
关于许可证完整版本的程序描述，可参见倍福信息系统的文档“TwinCAT 3 授权”。

授权一个 TwinCAT 3 功能的 7 天测试版本

对于TwinCAT 3 许可证加密狗无法启用 7 天测试版本。

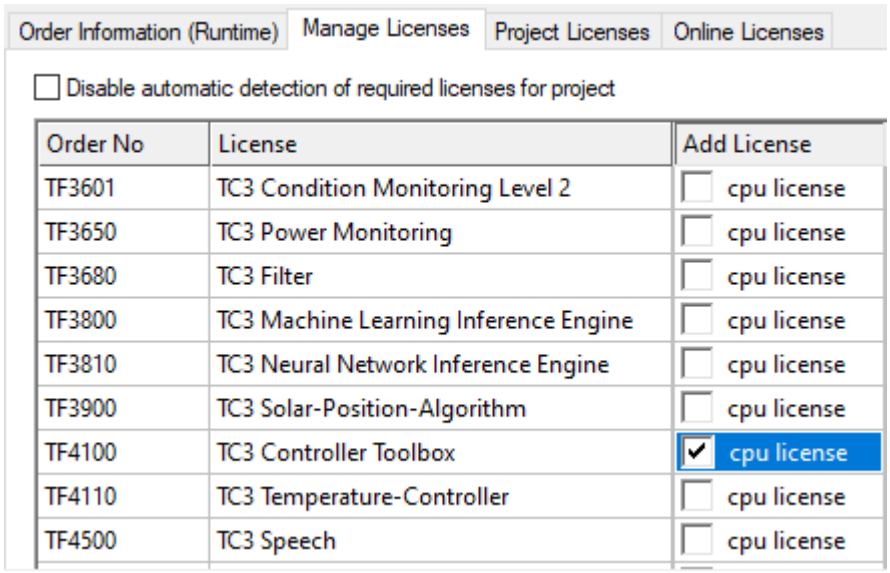
1. 启动 TwinCAT 3 开发环境（XAE）。
2. 打开一个现有的 TwinCAT 3 项目或创建一个新项目。
3. 如果想为一个远程设备激活授权，请设置所需的目标系统。为此，从工具栏的**选择目标系统**下拉列表中选择目标系统。
   ⇨ 授权设置总是指选定的目标系统。在目标系统上激活项目时，自动将相应的 TwinCAT 3 许可证复制到该系统中。

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



⇨ TwinCAT 3 许可证管理器打开。

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

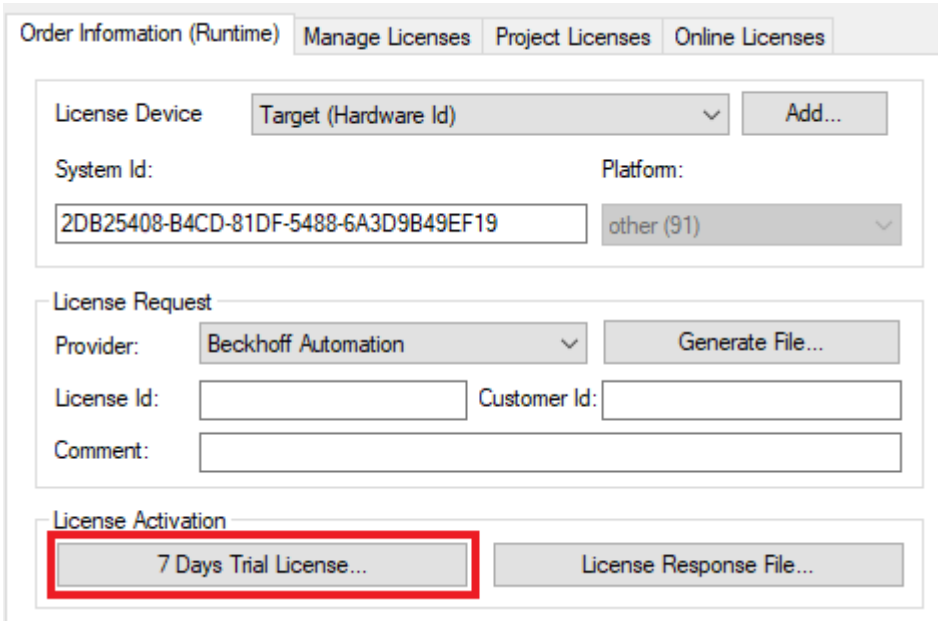


Order No	License	Add License
TF3601	TC3 Condition Monitoring Level 2	cpu license
TF3650	TC3 Power Monitoring	cpu license
TF3680	TC3 Filter	cpu license
TF3800	TC3 Machine Learning Inference Engine	cpu license
TF3810	TC3 Neural Network Inference Engine	cpu license
TF3900	TC3 Solar-Position-Algorithm	cpu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

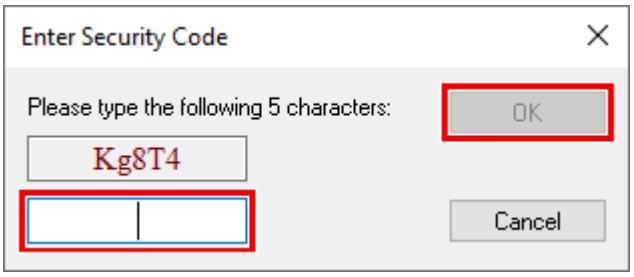
6. 打开**订单信息（运行时间）**选项卡。

⇨ 在许可证的表格概览中，以前选择的许可证显示为“缺失”状态。

7. 点击**7 天试用许可证...**，以激活 7 天试用许可证。



⇨ 一个对话框打开，提示输入对话框中显示的安全代码。



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

⇨ 7 天试用版启用。

3.4 Windows CE 安装

下文将介绍如何在装有 Windows CE 的倍福嵌入式控制器上安装 TwinCAT 3 功能组件 (TFxxx)

1. 下载并安装设置文件 [▶ 14]
2. 将 CAB 文件传输到 Windows CE 设备上 [▶ 15]
3. 在 Windows CE 设备上运行 CAB 文件 [▶ 15]

如果 Windows CE 设备上已经安装了旧版本的 TFxxx，则可以对其进行更新：

- 软件升级 [▶ 15]

下载并安装设置文件

Windows CE 的 CAB 安装文件是 TFxxx 设置的一部分。此文件可在倍福网站 www.beckhoff.com 上获取，并自动包含适用于 Windows XP、Windows 7 和 Windows CE（x86 和 ARM）的所有版本。

下载 TFxxx 设置文件，并按照安装 [▶ 9]部分的说明安装 TwinCAT 3 功能组件。

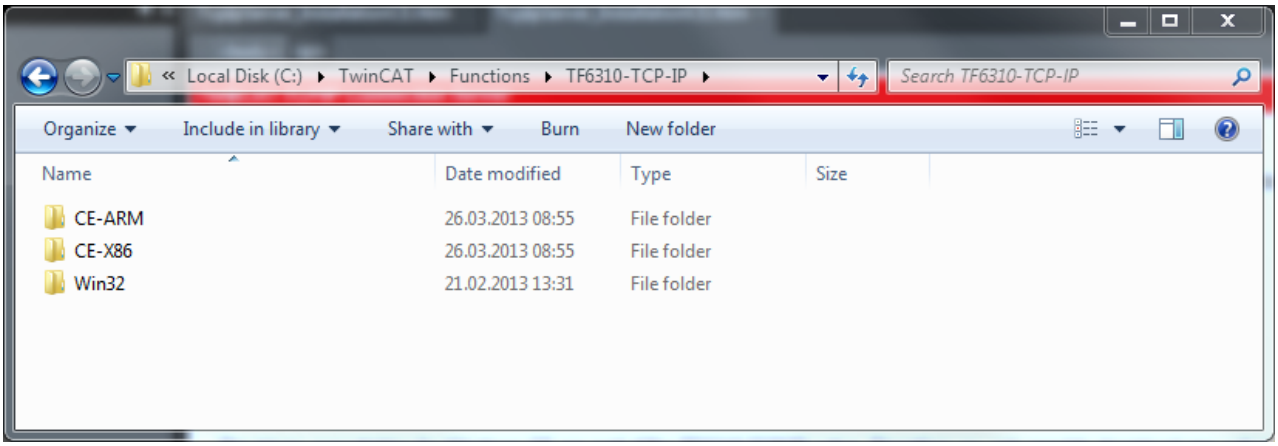
安装完成后，安装文件夹包含三个目录（每个硬件平台一个目录）：

- **CE-ARM**：基于 ARM 的嵌入式控制器，运行 Windows CE，例如 CX8090、CX9020

- **CE-X86**：基于 X86 的嵌入式控制器，运行 Windows CE，例如 CX50xx、CX20x0
- **Win32**：嵌入式控制器，运行 Windows XP、Windows 7 或 Windows Embedded Standard

CE-ARM 和 CE-X86 目录包含与 Windows CE 设备各自硬件平台相关的 Windows CE TwinCAT 3 功能组件的 CAB 文件。

例如：安装文件夹“TF6310”



将 CAB 文件传输到 Windows CE 设备上

将相应的 CAB 文件传输到 Windows CE 设备上。

传输可执行文件有多种选择：

- 通过网络共享
- 通过集成的 FTP 服务器
- 通过 ActiveSync
- 通过 CF/SD 卡

更多信息请参见倍福信息系统的“操作系统”文档（嵌入式控制器 > 操作系统 > CE）。

在 Windows CE 设备上运行 CAB 文件

将 CAB 文件传输到 Windows CE 设备后，双击该文件。选择 **OK**（确定），确认安装对话框。然后重启 Windows CE 设备。

重启设备后，TwinCAT 3 功能组件 (TFxxxx) 的文件将在后台自动加载，然后可以使用。

软件安装在 Windows CE 设备的以下目录中：
|Hard Disk|TwinCAT|Functions|TFxxxx

软件升级

如果 Windows CE 设备上已经安装了旧版本的 TwinCAT 3 功能组件，请在 Windows CE 设备上执行以下步骤升级至新版本：

1. 单击**开始 > 运行**，输入"Explorer"，打开 CE 资源管理器。
2. 导航至 *|Hard Disk|TwinCAT|Functions|TFxxx|xxxx*。
3. 将文件 *Tc*.exe* 重命名为 *Tc*.old*。
4. 重启 Windows CE 设备。
5. 将新的 CAB 文件传输至 Windows CE 设备。
6. 在 Windows CE 设备上运行 CAB 文件并安装新版本。
7. 删除 *Tc*.old* 文件。
8. 重启 Windows CE 设备。

⇨ 重启后，新版本生效。

3.5 安装 TwinCAT/BSD

TwinCAT 3 数据库服务器可作为软件包，在软件包库中用于 TwinCAT/BSD。可以通过命令安装“TF6420-Database-Server”软件包

```
doas pkg install TF6420-Database-Server
```

有关软件包服务器的更多信息，请参见 TwinCAT/BSD 手册的嵌入式控制器部分。

在系统重启或 TwinCAT 重启后，TwinCAT 3 数据库服务器也会启动，并且通过 ADS 可以从客户端系统进行配置。

一些数据库（例如 SQLite）只有在已安装相应的软件包之后才能使用。

4 技术简介

4.1 基本概念

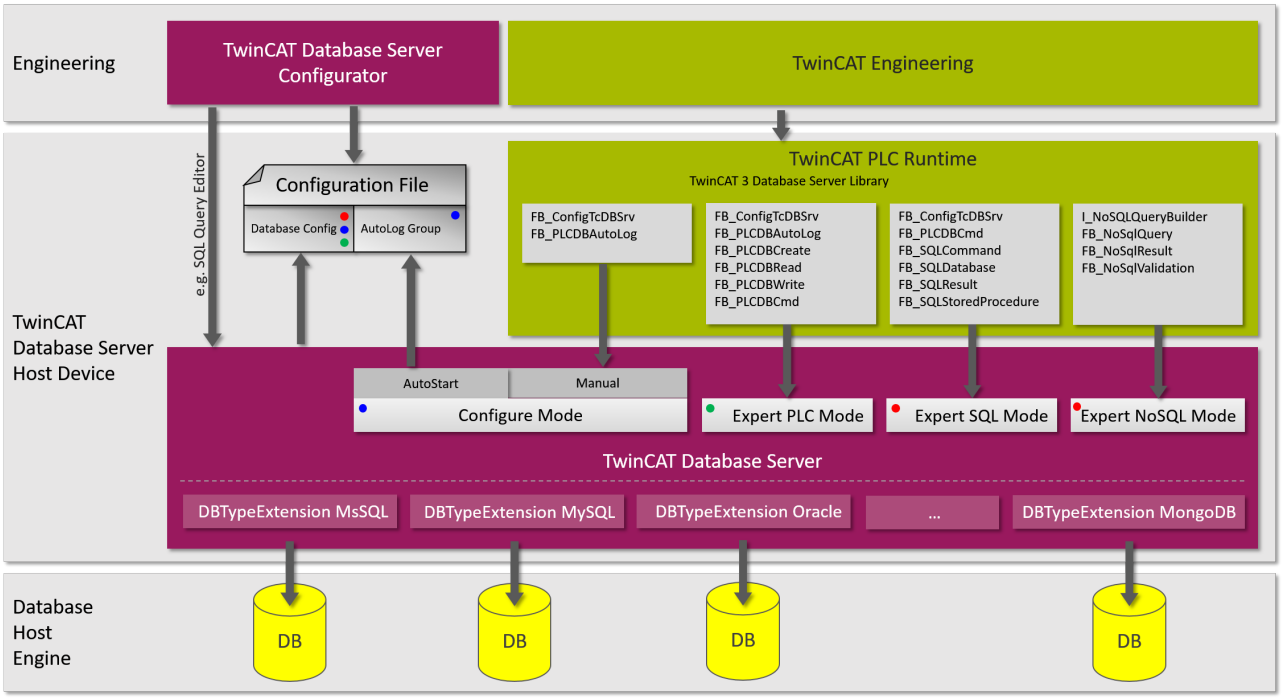
TwinCAT 数据库服务器旨在使所有 TwinCAT 用户能够尽可能方便地与控制器建立数据库连接。尽管要求简单性，但仍需保留充分的灵活性，这就是 TwinCAT 数据库服务器提供 4 个基本类别的原因：

- **配置模式：纯配置解决方案**
  基于图形配置的简单应用程序的数据库连接，无需代码实现。
- **PLC 专家模式：传统 PLC 程序员的编程解决方案**
  基于 PLC 功能块的简单或复杂应用程序的数据库连接，其中的数据库命令大多由数据库服务器自动生成。
- **SQL 专家模式：传统 PLC 程序员和数据库专家的编程解决方案**
  基于 PLC 功能块和 C++ 接口的简单或复杂应用程序的数据库连接，其中的数据库命令在程序执行期间自动构建。可实现高度的灵活性。
- **NoSQL 专家模式：PLC 程序员和 NoSQL 数据库专家的编程解决方案**
  通过 PLC 功能块实现的简单到复杂应用程序的数据库连接，其中的 NoSQL 命令可以在程序序列中进行创建和发送。

当然，在一个应用程序中可以将所有 4 个类别组合在一起。

通过图形配置器可以设置 TwinCAT 数据库服务器。可以将配置写入 XML 文件，然后将其下载到目标系统。在非 Windows CE 设备上，配置文件位于 *C:|TwinCAT|3.1|Boot* 文件夹中，在 Windows CE 设备上，配置文件位于 *|Hard Disk|TwinCAT| 3.1|Boot* 文件夹中。
不同的数据库系统都具有读取和写入访问权限。为此，可以使用可取消选择的数据库扩展。在“数据库 [▶ 120]”部分介绍了支持的数据库。

TwinCAT 数据库服务器服务与相应的控制计算机上的 TwinCAT 系统一起启动。它是 PLC 和数据库之间的纽带。



配置模式

在配置模式下，大部分工作在配置器中完成。务必为所需的数据库和 AutoLog 组设置配置。Target Browser 可用于配置 AutoLog 组、在线访问目标系统以及选择要传达的变量。如果使用 **AutoStart**（自动启动）选项，则在 TwinCAT 系统启动时会直接建立与配置的数据库的通信。如果选择 **Manual**（手动）选项，则必须通过功能块 FB_PLCDBAutoLog [▶ 156] 或 AutoLog 视图启用通信。

PLC 专家模式

在 PLC 专家模式下，在配置器中只能设置数据库配置。在应用程序的 PLC 代码中可以实现其他功能。使用功能块 FB_PLCDBCreate [▶ 166]，可以省去配置器，甚至可以从 PLC 配置数据库。如有需要，可提供用于读取和写入的功能块。功能块 FB_PLCDBCmd [▶ 177] 可以在 PLC 专家模式和 SQL 专家模式之间进行转换。在这里，可以轻松地将表结构映射为 PLC 结构，并且可以将带有当前结构值占位符的 SQL 命令传输到 TwinCAT 数据库服务器。然后，TwinCAT 数据库服务器会自动插入所有值，并将命令发送到数据库。

SQL 专家模式

在 SQL 专家模式下，用户可以在 PLC 中对插入、选择或更新等 SQL 命令进行构建，并通过 TwinCAT 数据库服务器将它们发送到数据库。这是一个非常灵活且功能强大的选项。此外，也可以从 PLC 调用数据库中的存储过程 [▶ 194]。

结构记录

在记录结构时，应注意相应的字节对齐方式。

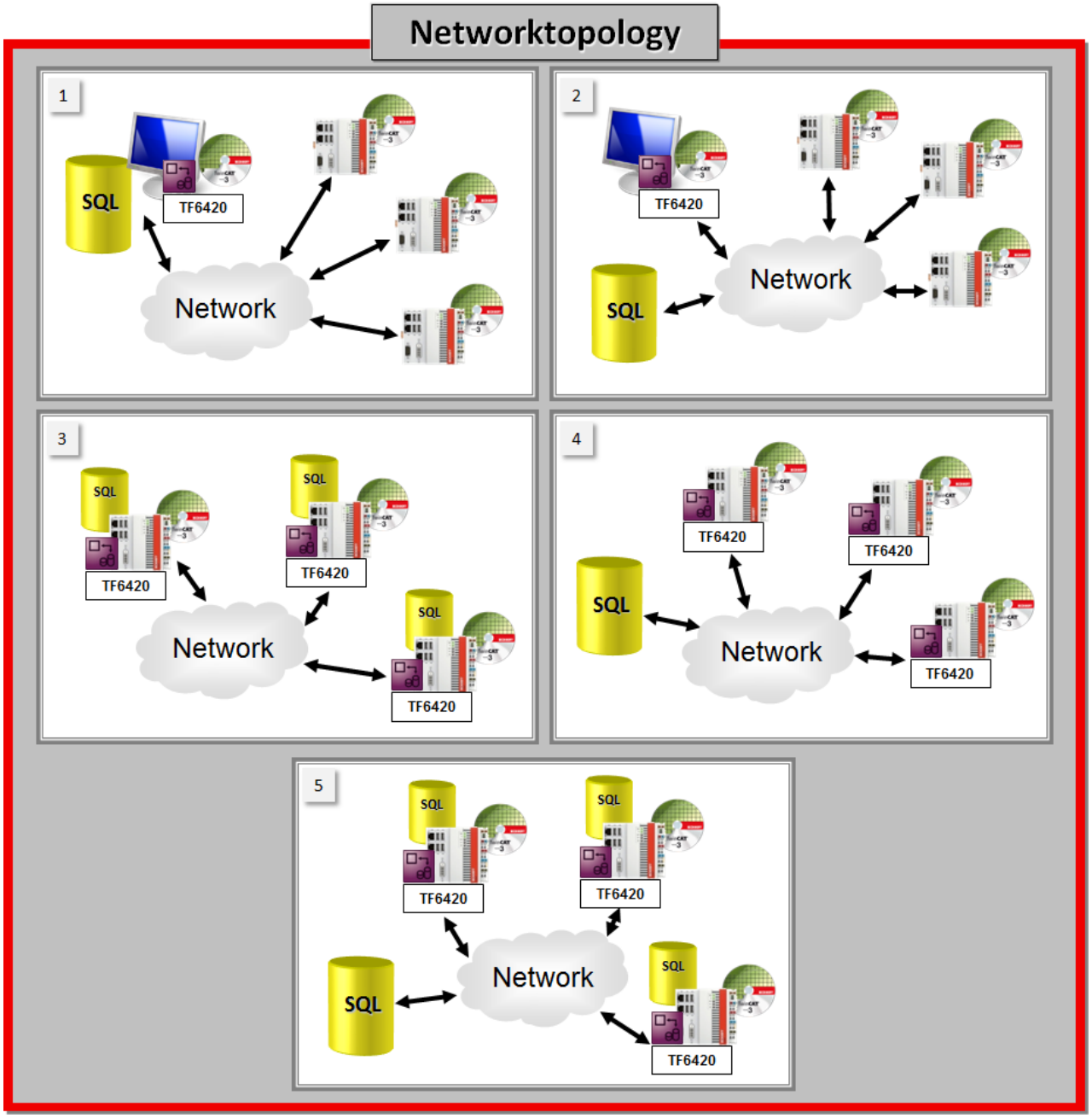
NoSql 专家模式

在 NoSQL 专家模式下，用户可以编译诸如插入或查找等 NoSQL 查询以及许多其他的数据库特定查询，并通过 TwinCAT 数据库服务器将它们发送到数据库。支持新的、更灵活的数据模式，例如，层次结构和数组。

4.2 应用领域和网络拓扑结构

TwinCAT 数据库服务器可用于任何控制应用：读取生产设备中的配方数据、为产品贴上生产数据标签、状态监测或设备控制、记录风力发电机组运行状态或楼宇自动化。数据库服务器可以集成在现有的网络架构中。

网络拓扑结构主要受数据库类型、本地条件和应用领域的影响。下图显示了可以使用 TwinCAT 数据库服务器的各种网络拓扑结构。



1. TwinCAT 和 TwinCAT 数据库服务器与数据库位于同一台计算机上。数据库服务器可以通过 ADS 充当许多控制器的网关。务必考虑性能。
2. TwinCAT 和 TwinCAT 数据库服务器位于同一台计算机上，而数据库则位于外部设备上。在这里，数据库服务器也可以通过 ADS 充当许多控制器的网关。务必考虑性能。
3. TwinCAT 数据库服务器位于每个已安装数据库的控制设备的本地。并非所有数据库都适合此类应用。
4. 这是最常见的用例。在每个控制设备上都会安装 TwinCAT 数据库服务器，数据库位于网络中的外部服务器上。
5. 案例 3 和案例 4 的组合。主数据库位于网络中的服务器上，并且现场的每个控制器都有一个本地数据库，当遇到断开连接等情况时，该本地数据库就会启动，并且第一时间在本地存储数据。在每个控制设备上都会安装数据库服务器。

TwinCAT 数据库服务器对数据库的远程访问

为了实现 TwinCAT 数据库服务器对数据库的远程访问，必须在数据库端考虑各个方面：

- 一般而言，是否允许远程访问?
- 同时允许进行多少个连接？（如果 TwinCAT 数据库服务器需要打开多个连接的话）
- 想要使用数据库服务器登录数据库的用户是否具有足够的权限?
- 远程系统的防火墙是否已进行适当配置?

有关数据库服务器配置的更多详细信息，请参见相应的数据库文档 [▶ 120]。

4.3 兼容性

TwinCAT 数据库服务器是一款经过试验和测试的 TwinCAT 产品，已经上市多年。人们对于该产品的要求越来越高。TwinCAT 数据库服务器的最新发展成果旨在迎合这些日益增长的需求。

之前已在 3.**0**.x、3.**1**.x 和 3.**2**.x 版本中提供 TwinCAT 数据库连接。当前版本为 3.**3**.x。与之前一样，数据库服务器由以下组件组成：配置器、ADS 服务器和 PLC 库。3.0.x 版本包含的 PLC 库为 Tc2_Database.compiled 库。3.1.x 及以上版本中的 PLC 库被称为 Tc3_Database.compiled-library。

已发布的数据库服务器版本概述

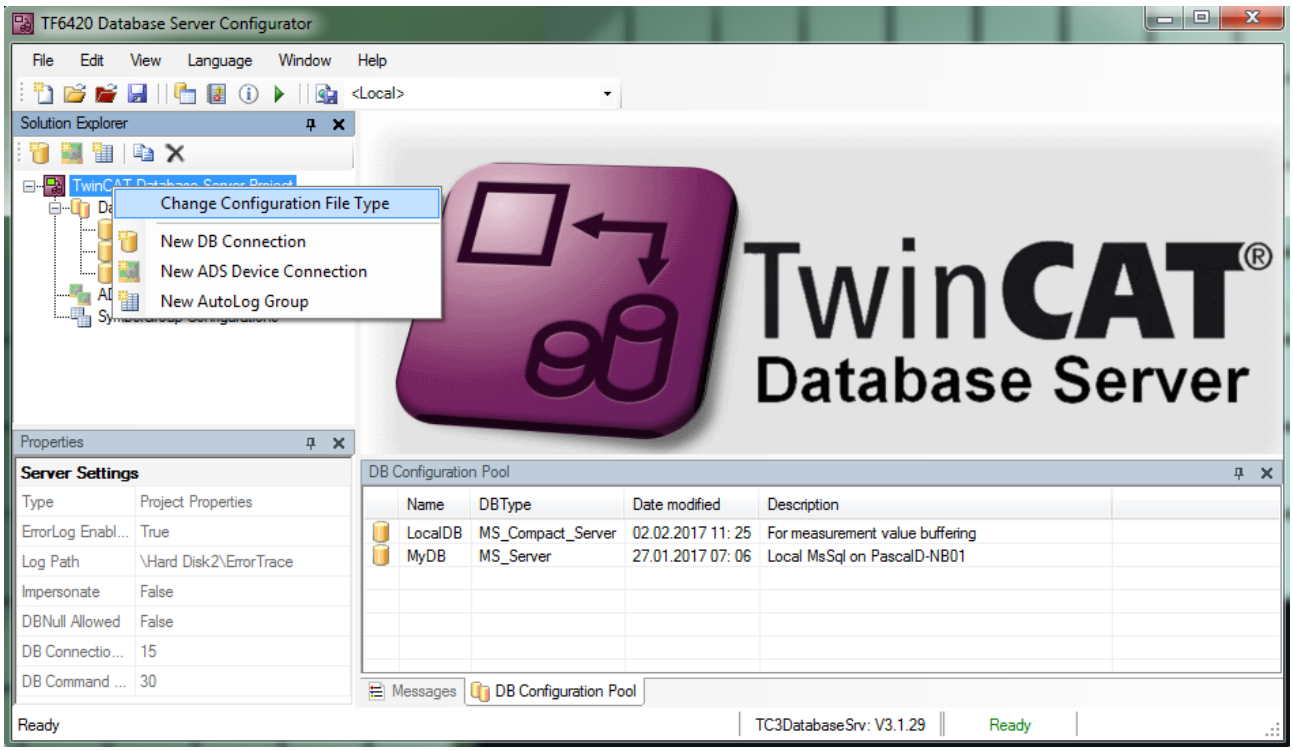
数据库服务器 3.0.x	3.0.23	3.0.26	3.0.27	3.0.28					
数据库服务器 3.1.x					3.1.29	3.1.30	3.1.31		
数据库服务器 3.2.x								3.2.32	
数据库服务器 3.3.x									3.3.33

关于从 3.0.x 过渡到 3.1.x 的说明

除了全新的更高性能的功能之外，一个关键方面是 3.0.x 和 3.1.x 版本之间的兼容性。例如，使用 Tc2_Database.compiled 库的旧 PLC 代码也可与新的 3.1.x 版本 ADS 服务器一起使用。在 3.1.x 版本中，在设置期间会继续安装旧的 Tc2_Database.compiled 库。配置器为服务器创建的 XML 文件在 3.0.x 和 3.1.x 版本之间有所不同。可以使用新的配置器（独立）读取旧的配置文件，甚至可以根据需要将其转换为新格式。

备份旧的 XML 配置

在从 TwinCAT 数据库服务器 3.0.x 版本升级到新的 3.1.x 版本期间会保存旧的 XML 配置。它会被重命名为“CurrentConfigDataBase_**OLD**.xml”，并保留在 TwinCAT 启动目录中。



尽管存在上文提到的通用兼容性，但是旧的配置器和旧的 ADS 服务器（3.0.x 版本）不能与新的 Tc3_Database.compiled 库一起使用。下图提供了兼容性概述。

	Konfigurator 3.0.x	Server 3.0.x	Tc2_Database.compiled-library	Konfigurator 3.1.x Standalone	Konfigurator 3.1.x Visual Studio	Server 3.1.x	Tc3_Database.compiled-library
Konfigurator 3.1.x Standalone		✓	✓			✓	✓
Konfigurator 3.1.x Visual Studio		✗	✓			✓	✓
Server 3.1.x	✓		✓		✓		✓
Tc3_Database.compiled-library	✗	✗			✓	✓	

关于从 3.1.x 过渡到 3.2.x 的说明

配置的文件格式保持不变。ADS 服务器只是扩展了新的功能。所有其他功能仍然可用。在 3.2.x 版本中，在设置期间会并行安装旧的 Tc2_Database.compiled-library 与 Tc3_Database.compiled-library。关于从 3.0.x 过渡到 3.1.x 的说明适用。

在 Tc3_Database.compiled-library 中，所有先前的功能块均已从 3.2.x 版本更新。更新涉及 I_TcMessage EventLogger 接口。为了确保旧版应用程序继续运行，在新功能块的名称后会附加“Evt”。虽然所有旧功能块仍然包含在库中，但是它们现在处于过时文件夹中，编译器会对它们进行相应地标记。

示例：

在 3.1.x 版本中：FB_SQLCommand

在 3.2.x 版本中：FB_SQLCommandEvt

对于新项目，我们建议使用以“Evt”结尾的功能块。需要注意的是，仅从 TwinCAT 3.1 版本 4022.20 开始可以提供 EventLogger，因此仅从 4022.20 开始可以使用功能块。

关于从 3.2.x 过渡到 3.3.x 的说明

在 3.3.x 版本中，TwinCAT 数据库服务器的模块化在驱动程序端得到了进一步提升。确保完全兼容性。不过，在设置中不再自动包含一些驱动程序，用户必须单独提供它们。例如：Windows CE 下的 MySQL。有关详细信息，请参见配置部分中相应数据库的设置。

5 配置

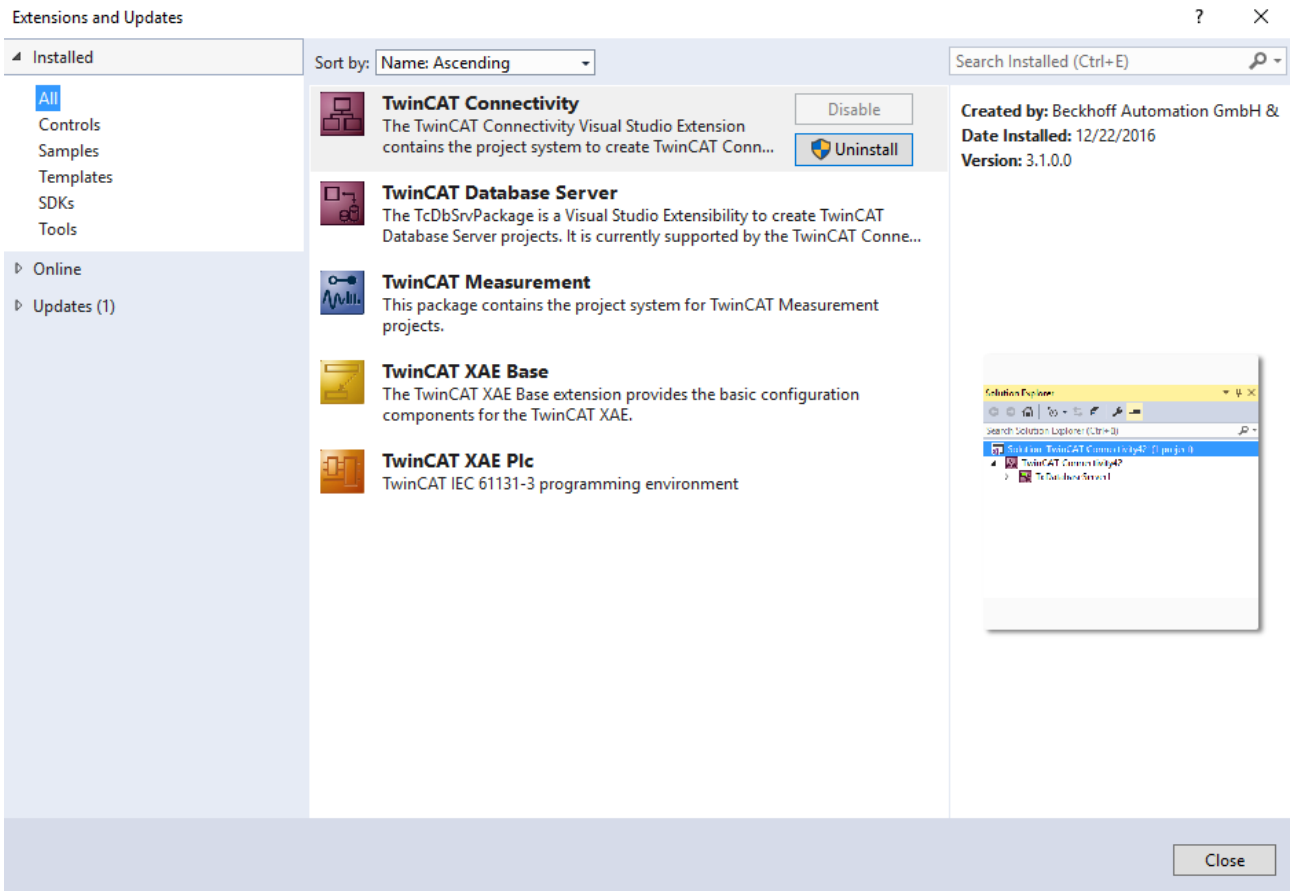
5.1 配置器

通过配置器可以设置和控制 TwinCAT 数据库服务器。该工具还提供一系列开发工具，以加快 PLC 中应用程序的开发。

配置器集成在 Visual Studio 中，以便尽可能以简便的方式进行开发。TwinCAT 项目和 TwinCAT 数据库服务器项目可以放在一个共同的解决方案中。或者，客户可以继续使用独立的配置器，而无需依赖 Visual Studio 。

5.1.1 集成在 Visual Studio 中

TwinCAT 数据库服务器集成在 Visual Studio 2013 和 Visual Studio 2015 中。借助于 2 个扩展功能可以实现这种集成。在安装期间会在 Visual Studio 扩展窗口中添加 2 个必需的扩展功能，其中包含项目系统的功能和其他功能。

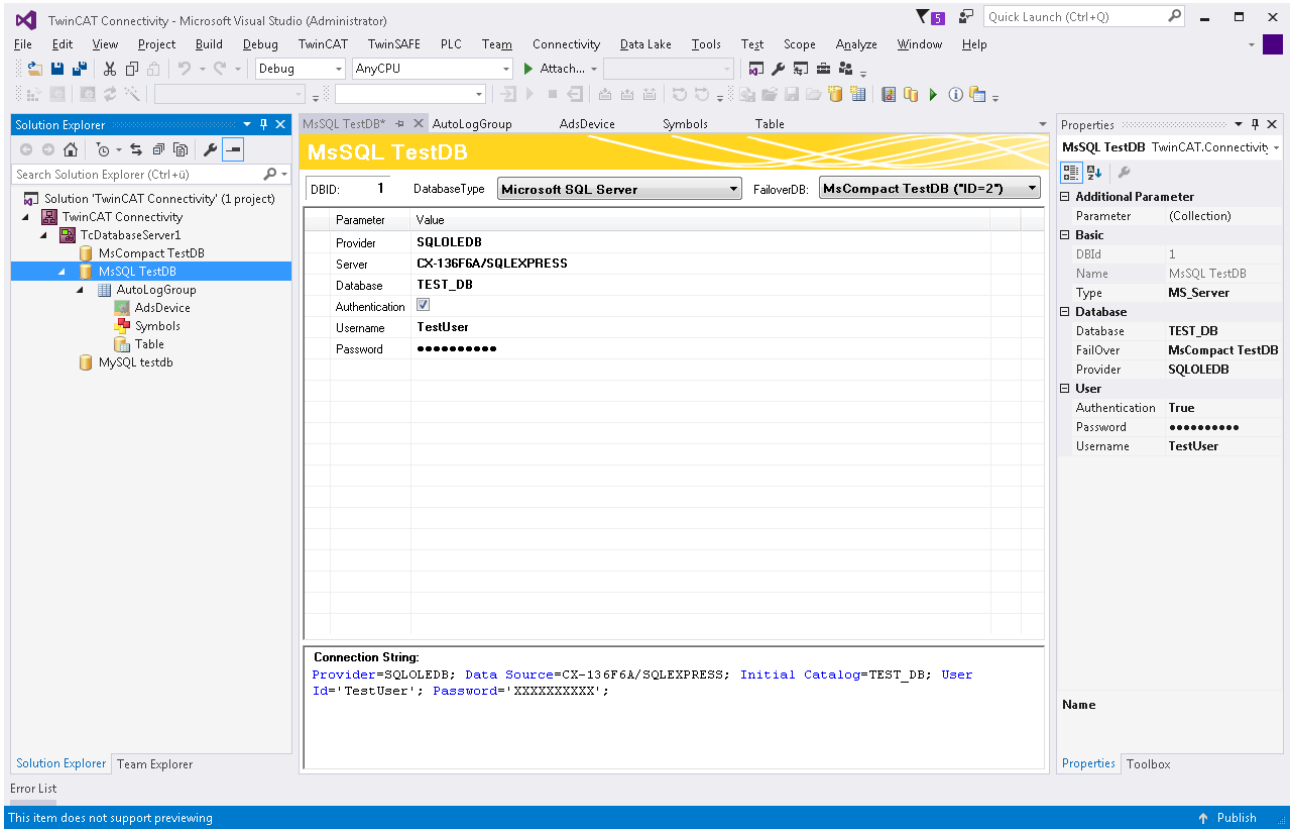


5.1.1.1 基本信息

本章介绍了 TwinCAT 数据库服务器的 Visual Studio 集成的各种功能和组件。

5.1.1.1.1 用户界面组件

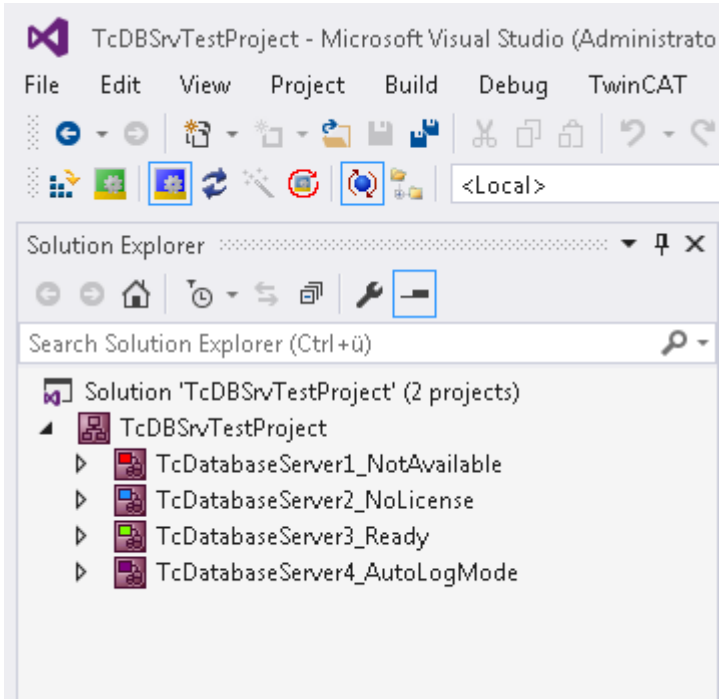
TwinCAT 数据库服务器集成在 Visual Studio 2013 和 Visual Studio 2015 中。Visual Studio 的 TwinCAT Connectivity 扩展提供了一个新的项目系统。例如，这可以用于创建基于文件的 TwinCAT 数据库服务器项目。支持各种典型组件，例如，属性窗口、解决方案资源管理器以及错误输出。此外，还提供了用于编辑配置文件的各种编辑器。



任意数量的 TwinCAT 数据库服务器项目都可以集成到一个 TwinCAT Connectivity 项目中。

项目图标会显示设定的目标系统的状态：

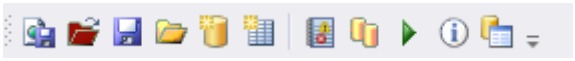
- 红色：无法访问 TwinCAT 数据库服务器。
- 蓝色：TwinCAT 数据库服务器没有有效的授权。（请参见 授权 [▶ 12]）
- 绿色：可以访问 TwinCAT 数据库服务器，并且它已准备就绪。
- 紫色：TwinCAT 数据库服务器处于 AutoLog 模式。在 PLC 和数据库之间交换数据。



TwinCAT 数据库项目映射一个基于文件的项目系统。在解决方案资源管理器中可以管理各个配置文档。编辑器中任何待处理的修改都会在文档中以 * 标识。如果在未打开文档的情况下进行更改（通过属性窗口），则仍会记录更改。有关项目系统的更多信息，请参见“TwinCAT 数据库服务器项目 [▶ 26]”部分。

工具栏和命令

工具栏具有以下元素：

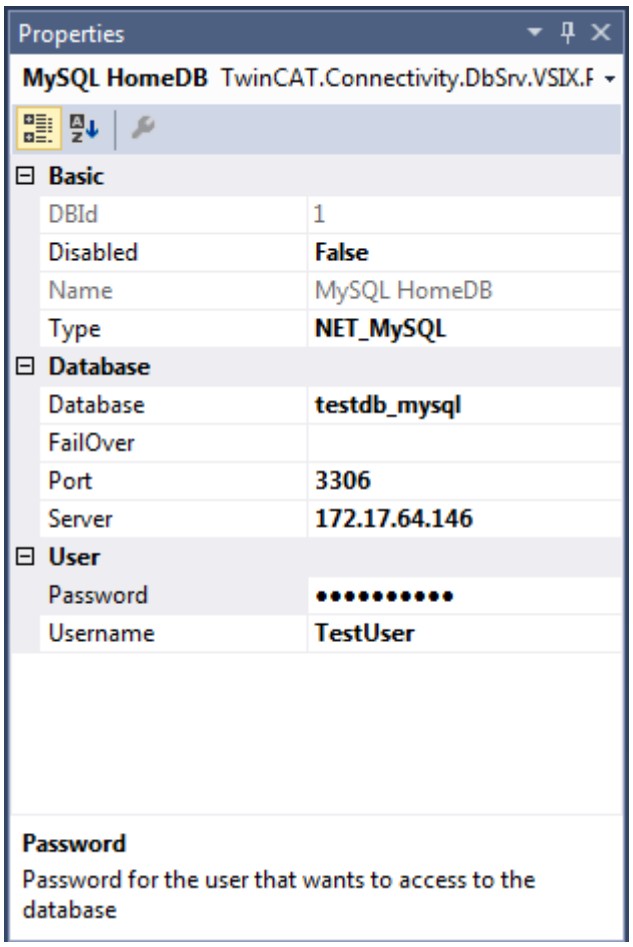


工具栏按钮	描述
	激活配置
	读取目标设备的配置
	在 XML 文件中保存配置
	从 XML 文件读取配置
	添加新的数据库配置
	添加新的 AutoLog 组
	事件显示
	数据库池
	AutoLog Viewer
	InformationLog View
	SQL 查询编辑器

属性窗口

通过专用编辑器或属性窗口可以配置不同项目文档的设置。在此过程中可以修改文件内容，但是不能修改 TwinCAT Connectivity 项目的项目文件中的元数据。

在属性窗口的下部，对各个属性进行了更加详细的描述。请注意，某些列表只能在编辑器中编辑。



输出和错误列表

Visual Studio 具有集成的控制台输出。TwinCAT 数据库服务器使用该功能来发布通知、警告或错误消息。为此，在输出中可以选择“TwinCAT 数据库服务器”类别。如果之前没有来自 TwinCAT 数据库服务器的消息，则该类别可能尚不存在。

此外，Visual Studio 错误列表也可用于传达主要信息。

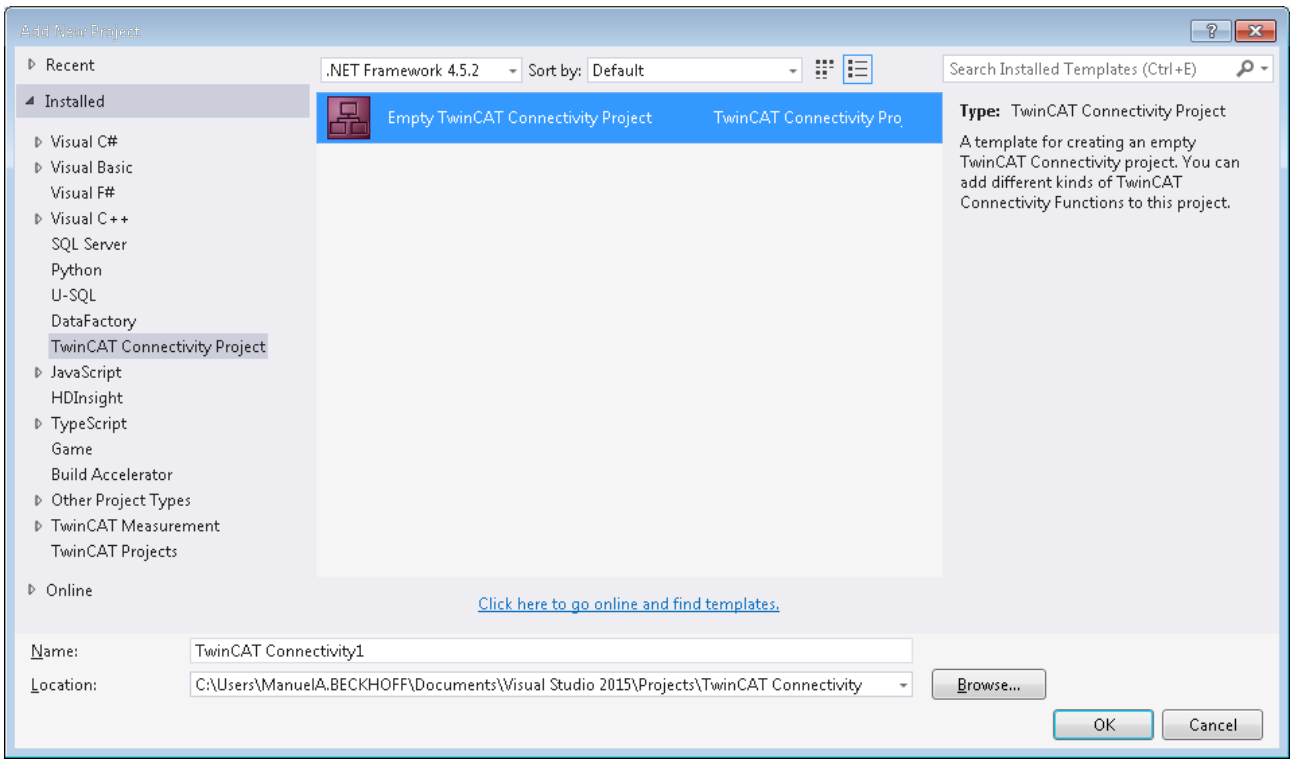
在 Visual Studio 中，通过 **View > Error list/Output**（视图 > 错误列表/输出）可以打开这两个窗口。

5.1.1.1.2 TwinCAT 数据库服务器项目

构建项目

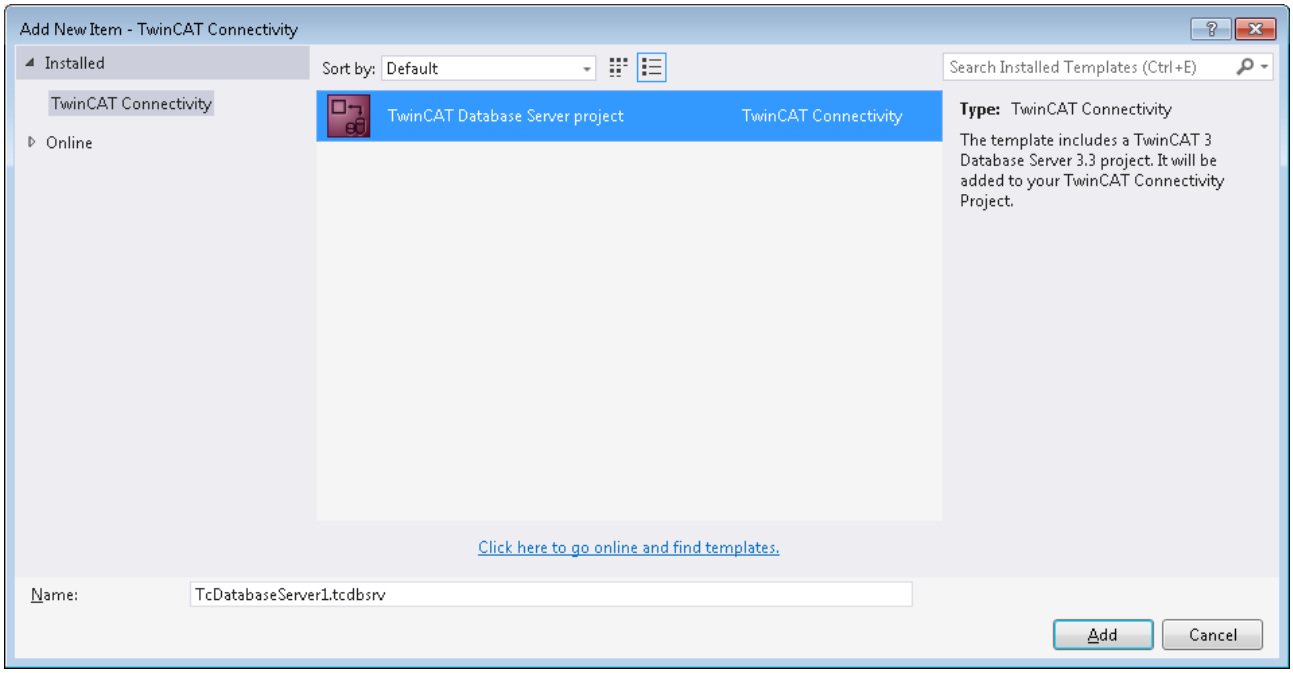
Visual Studio 的 TwinCAT Connectivity 扩展提供了一个新的项目模板。在创建新项目时，**TwinCAT Connectivity 项目**类别将作为一个选项出现。

要创建一个新的 TwinCAT Connectivity 项目，请选择 **Empty TwinCAT Connectivity Project**（空的 TwinCAT Connectivity 项目），指定项目名称和存储位置，然后点击 **OK**（确定）将其添加到解决方案中。这样可以方便地与 TwinCAT 或其他 Visual Studio 项目并行创建 TwinCAT Connectivity 项目或 TwinCAT 数据库服务器项目。

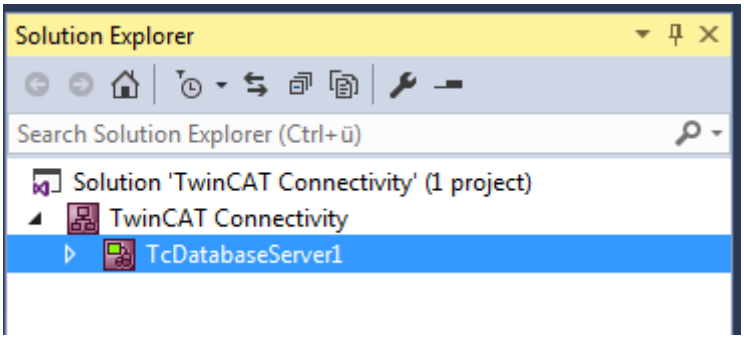


在解决方案中会出现一个新的项目节点。在 Connectivity 项目节点下方，您可以为支持的连接功能添加子项目。

使用 **Add**（添加）将一个新的 TwinCAT 数据库服务器项目添加到 TwinCAT Connectivity 项目中。在现有项目模板列表中可以找到 TwinCAT 数据库服务器项目。



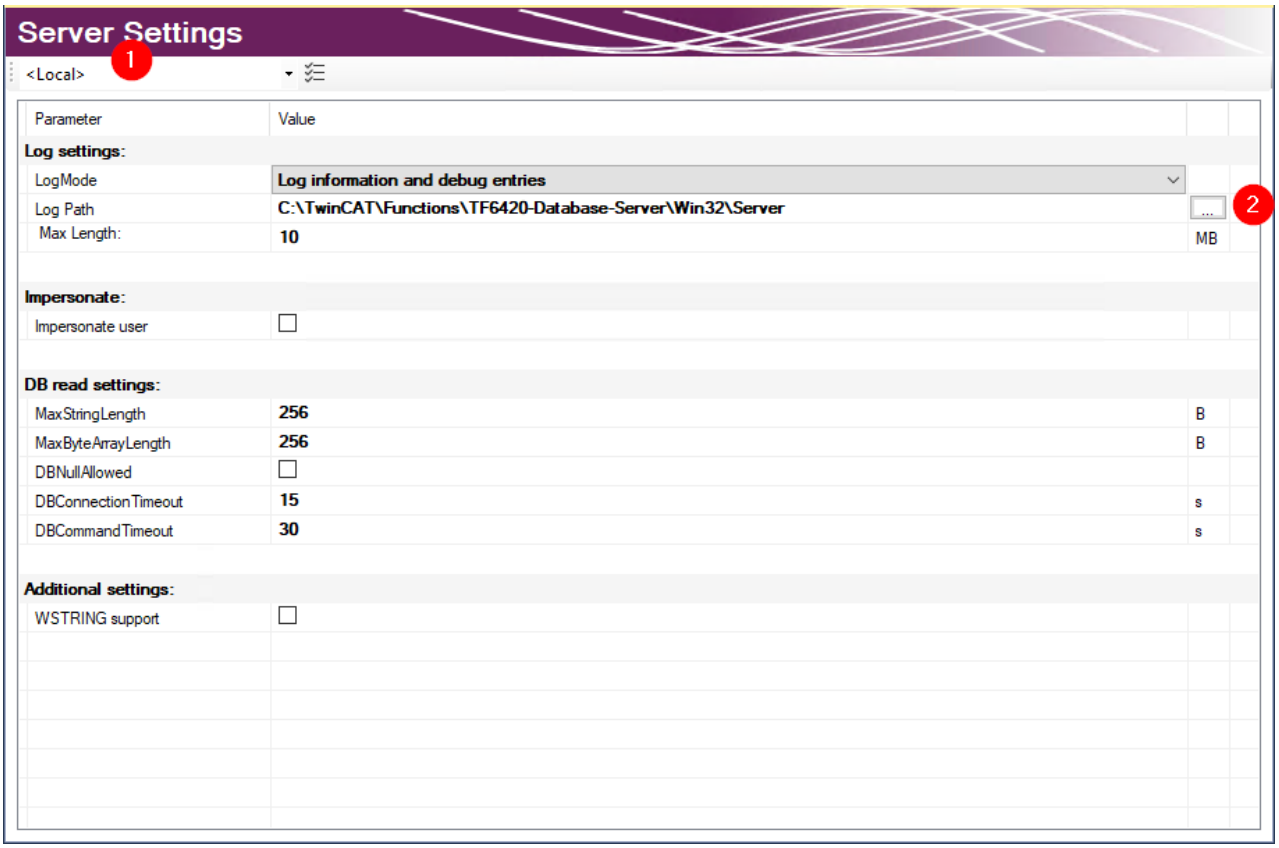
在 TwinCAT Connectivity 节点下，创建一个新的 TwinCAT 数据库服务器项目。



现在可以将其用作 TwinCAT 数据库服务器待配置的基础。通过属性窗口或编辑器可以编辑文档。

一个 Connectivity 项目可以与任意数量的 TwinCAT 数据库服务器项目或其他项目相关联，因此它可能包含多项配置。

服务器设置的编辑器



服务器设置编辑器可以用于编辑 TwinCAT 数据库服务器的设置。这些是与相应的服务器相关的常规设置。在下拉菜单（1）中，您可以通过 Ams NetID 选择目标系统。为此，您必须通过 TwinCAT 创建一条通往目标系统的路由。在传输已完成的配置时，设置将存储在该目标系统的 TwinCAT 数据库服务器中。

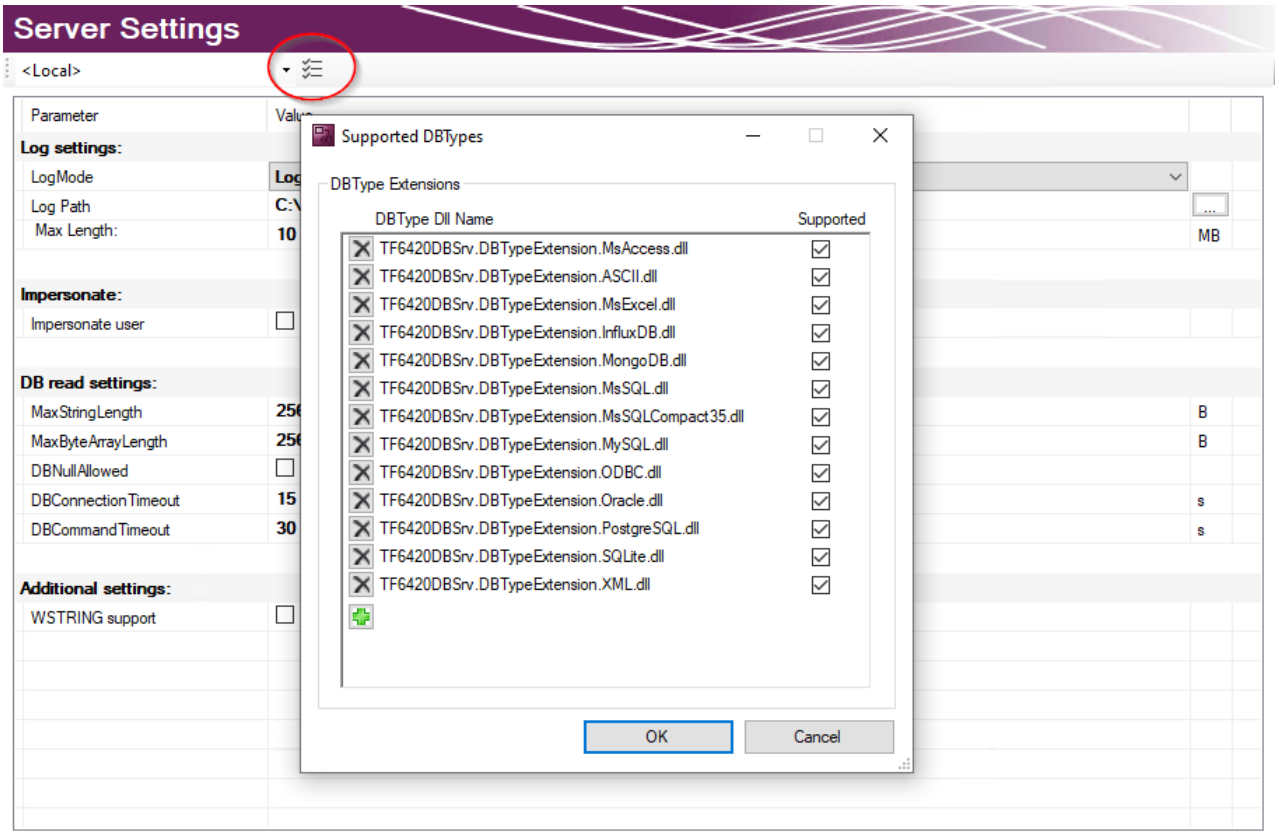
在 **Log settings**（日志设置）下可以配置关于记录故障或错误的设置。在发生故障或错误时，数据库服务器会在日志文件中生成一个详细的条目。通过信息日志查看器 [▶ 48]可以读取日志文件。在 **Log Settings**（日志设置）下，您可以指定文件位置的路径和最大文件大小。您还可以影响日志的准确性。出于性能方面的考虑，我们建议在不再需要的情况下，在错误分析之后再次停用日志记录。

对于基于文件的数据库（例如 Access 或 SQL Compact）的网络访问，必须设置 **Impersonate**（模拟）选项，以便 TwinCAT 数据库服务器可以与该网络驱动器连接。**Windows CE 目前不支持该功能。**

可以使用进一步的配置设置来控制从数据库读取数据的过程。这些设置指的是目标系统上的 TwinCAT 数据库服务器：

MaxStringLength	PLC 中的变量的最大字符串长度
MaxByteArrayLength	PLC 中的变量的最大字节数组长度
DBNullAllowed	表示 TwinCAT 数据库服务器是否接受零值。
DBConnectionTimeout	表示在尝试建立连接时，TwinCAT 数据库服务器出现连接错误的时间。
DBCommandTimeout	表示在发送命令时，TwinCAT 数据库服务器出现连接故障的时间。如果涉及大量数据，则处理一条命令可能需要相当长的时间，具体取决于数据库和基础设施。

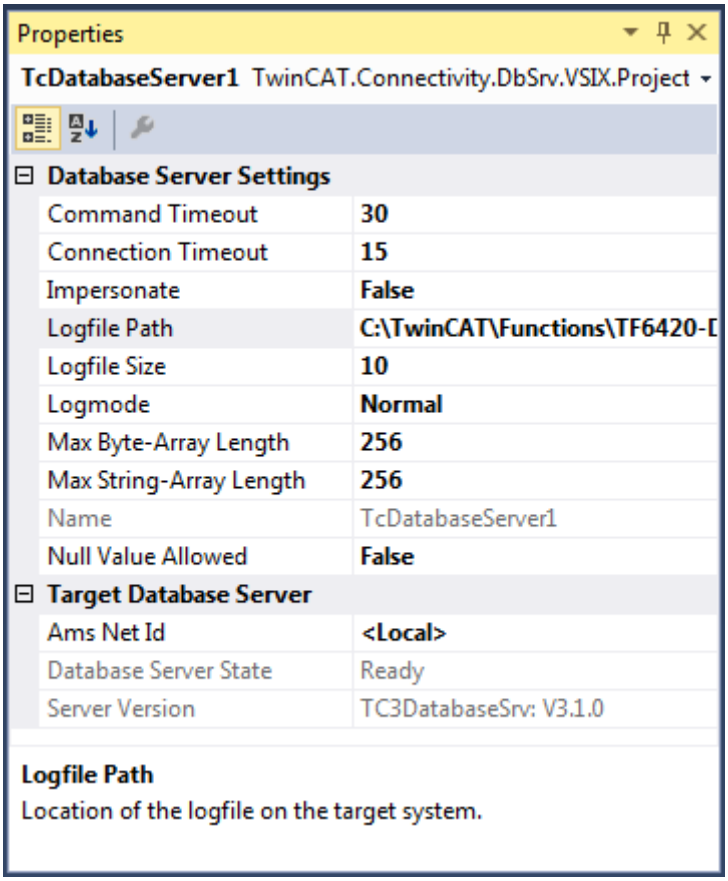
支持的数据库类型



在服务器设置中可以选择已安装的数据库类型。默认情况下会选择所有已安装的数据库。TwinCAT 3 数据库服务器将加载相应的数据库接口。这样可以取消选择目标系统上未使用的数据库。

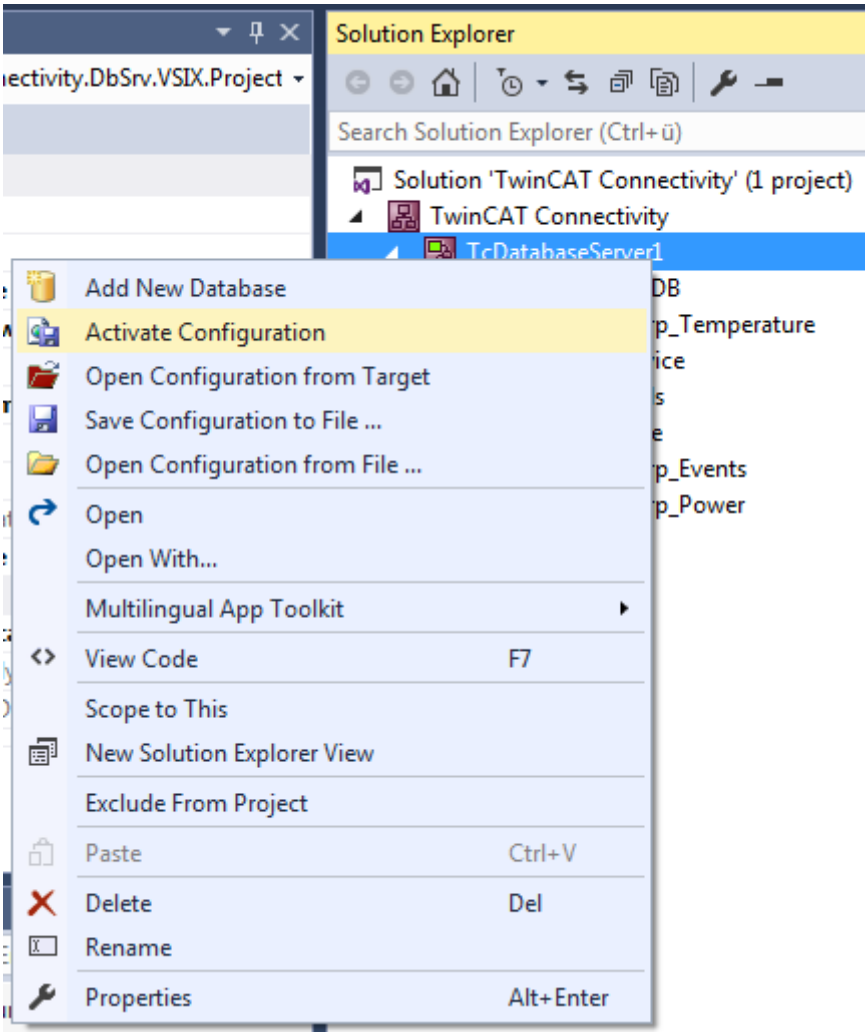
属性窗口中的服务器设置

在数据库服务器的编辑器窗口或属性窗口中可以调整 TwinCAT 数据库服务器的设置。这些属性也会直接影响配置文件。



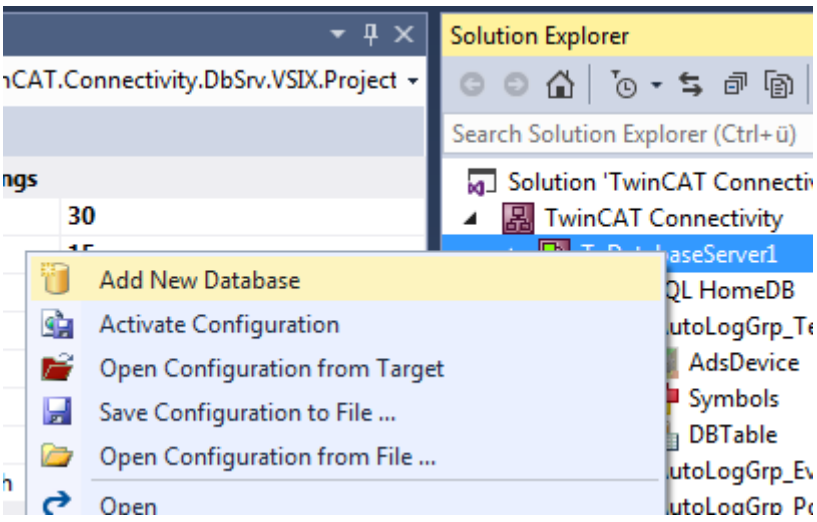
激活项目

要在 TwinCAT 数据库服务器上激活一个已配置的项目，可使用 TwinCAT 数据库服务器项目的上下文菜单中的 **Activate Configuration**（激活配置）命令。



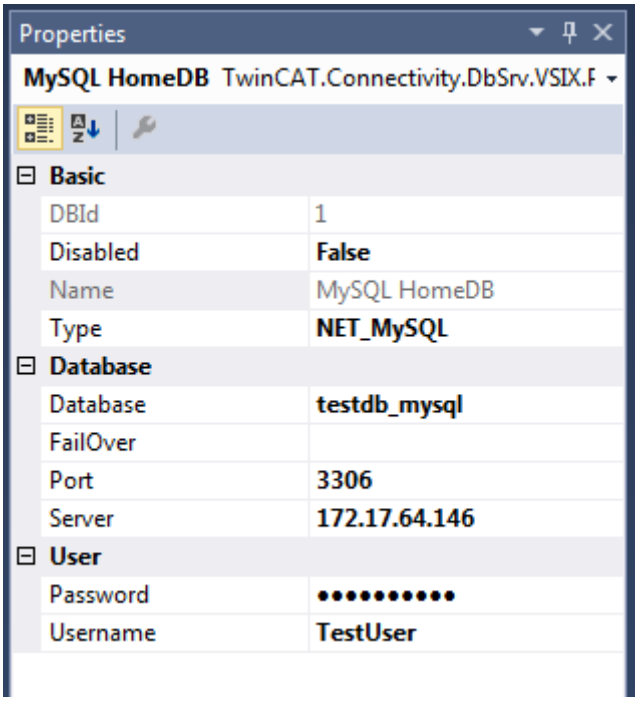
5.1.1.1.3 配置数据库

通过数据库服务器项目的上下文菜单中的 **Add New Database**（添加新数据库）命令或工具栏中的相应命令可以添加新的数据库配置。



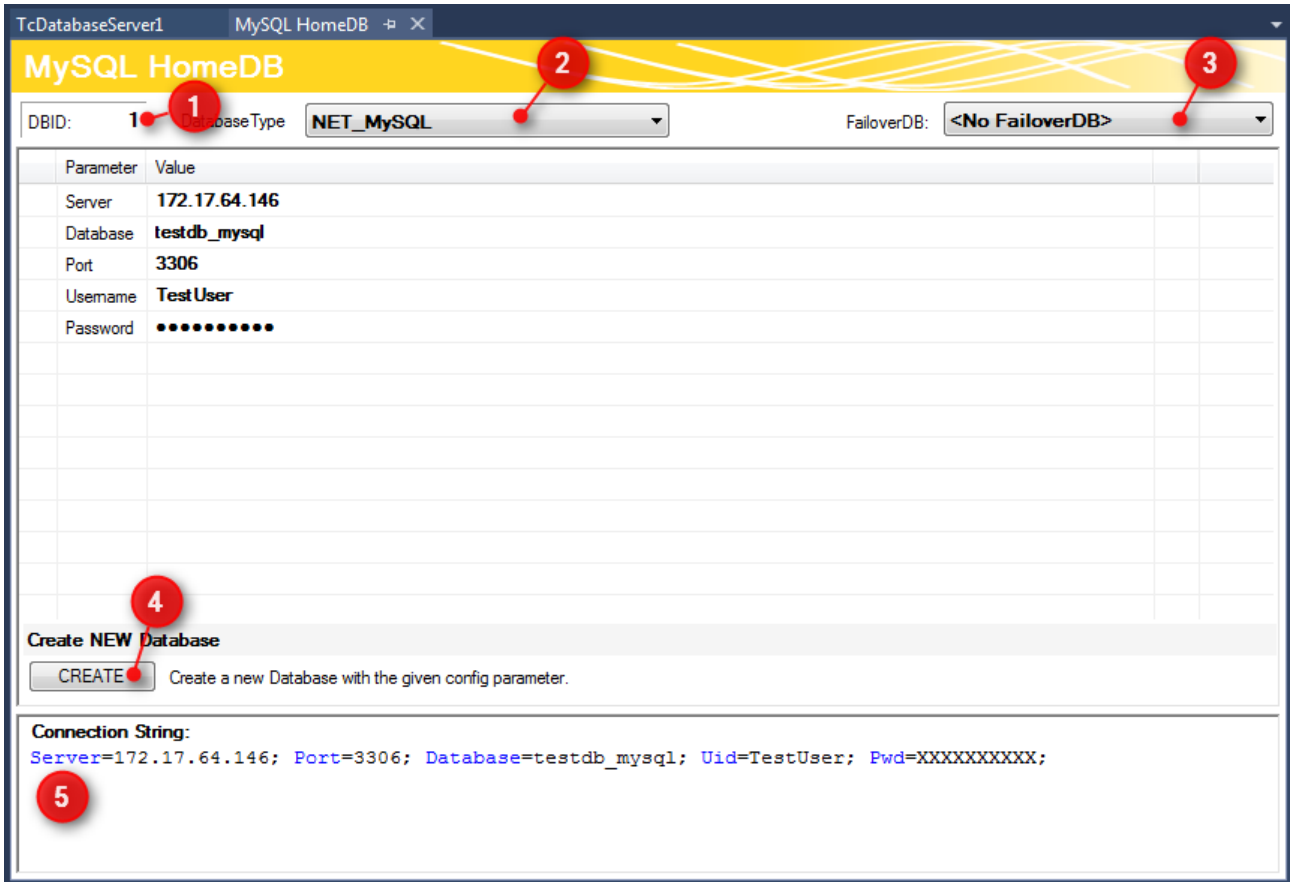
新的数据库配置以文件的形式被添加到项目文件夹中，并集成到项目中。与所有 Visual Studio 项目一样，新文件的信息存储在 Connectivity 项目中。

通过属性窗口或特殊的编辑器可以编辑 TwinCAT 数据库服务器项目中的新数据库配置：



由于数据库具有不同的参数，因此属性会动态适应选定的数据库类型。这些设置与文件内容有关，而与文件属性无关。

数据库配置编辑器

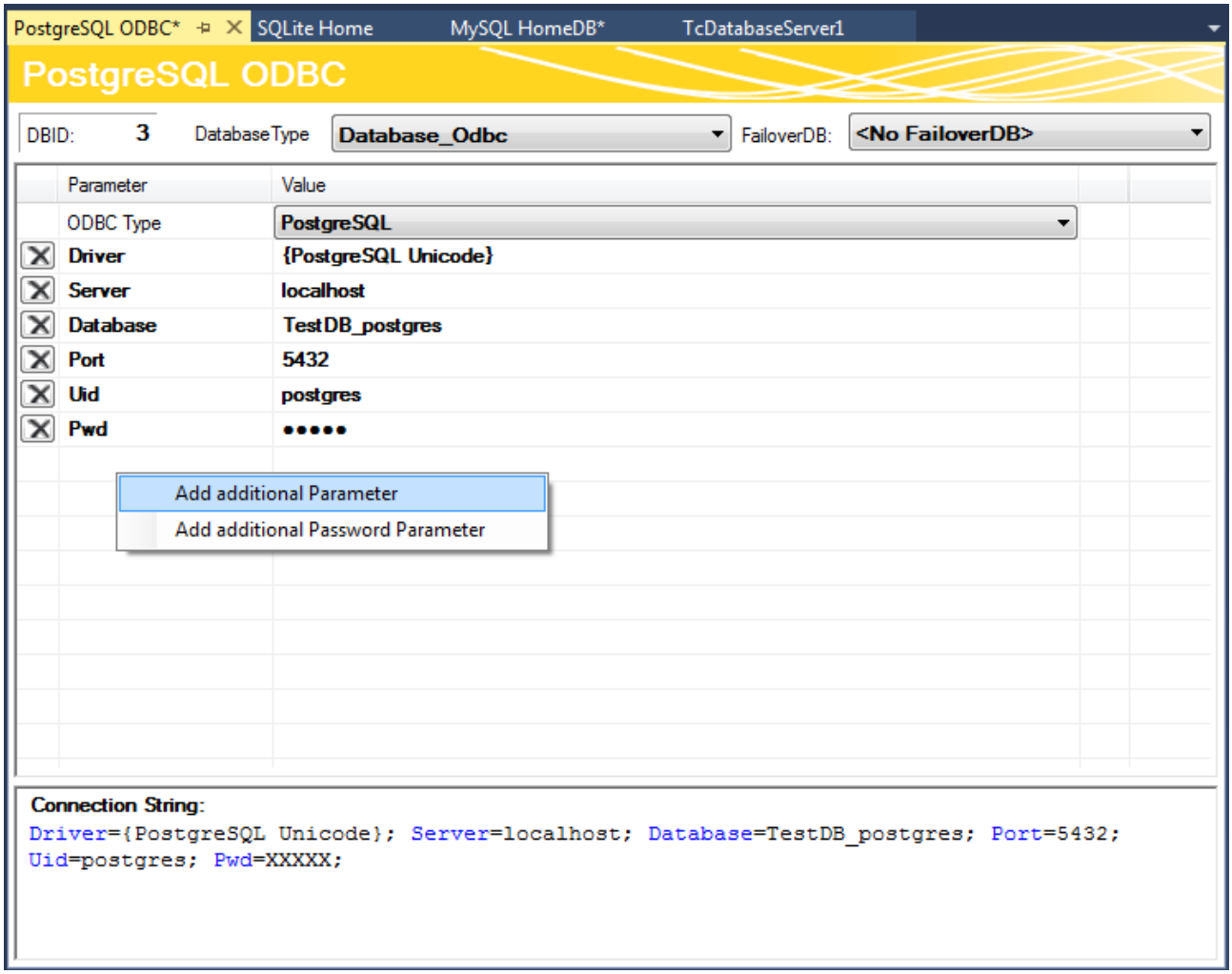


PLC 中的一些功能块所需的数据库 ID 会在编辑器的上部（1）显示。从下拉菜单（2）中可以选择目标数据库的数据库类型。另一个选项是数据库的 ODBC 接口，不过目前还不支持。请注意，根据数据库的不同，并非 TwinCAT 数据库服务器的所有功能都能得到保证。

您还可以选择所谓的故障转移数据库（3），在配置模式下遇到错误时就会触发故障转移数据库。在网络断开的情况下，该功能可以自动确保数据存储在其他地方，而不会丢失。

对于每个数据库 [▶ 120]，都有附加的可调整参数。根据数据库会创建一个连接字符串（5），用于描述与数据库的连接。这样旨在使您设置的参数更加透明。

CREATE（创建）（4）按钮可以用于创建新的数据库。只有在相关数据库支持该功能的情况下，才会显示它。

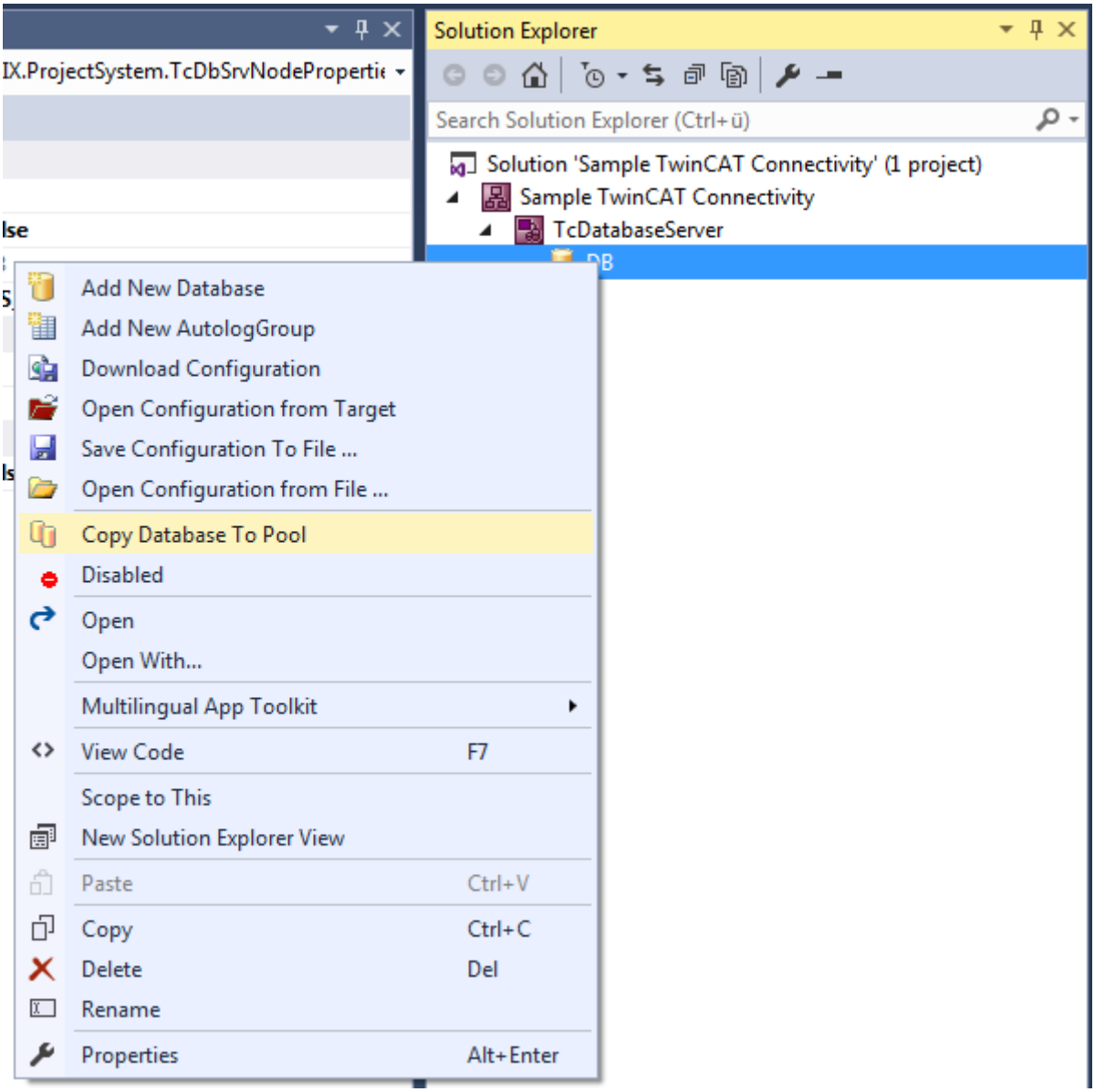


通过 ODBC 接口可以配置未知数据库。在 **ODBC Type**（ODBC 类型）下拉列表中，选择“Unknown Database”（未知数据库），然后通过上下文菜单中的命令添加参数。它们可能包含以加密形式存储的密码。利用这些参数可以组合成所需的连接字符串。请注意，只能使用 TwinCAT 数据库服务器的有限功能。仅支持 SQL 专家模式的显式功能块。

只能通过编辑器应用附加参数，而不能通过属性窗口。

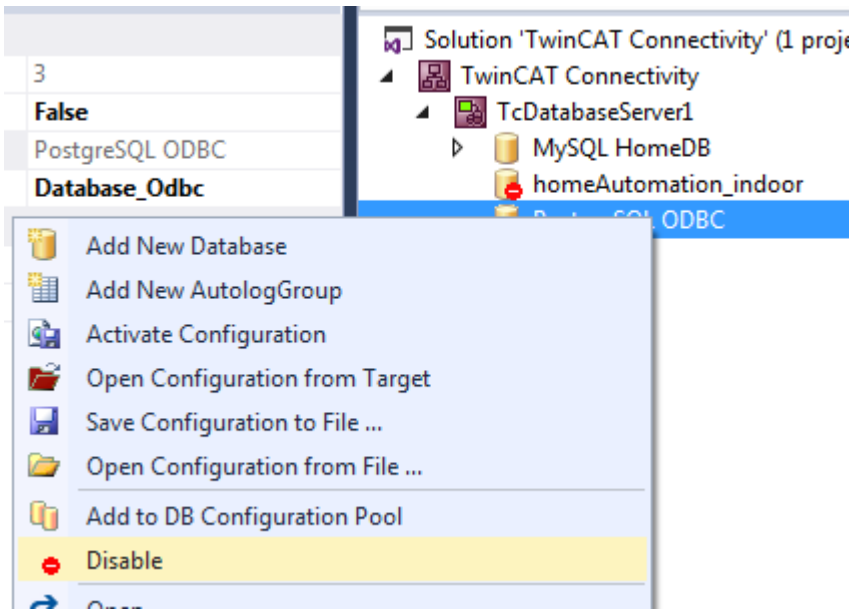
将数据库配置复制到数据库池

在上下文菜单中有一个相应的命令，用于将数据库配置复制到数据库池 [▶ 47]中。此外，还可以使用拖放功能，在项目和数据库池之间移动数据库配置。



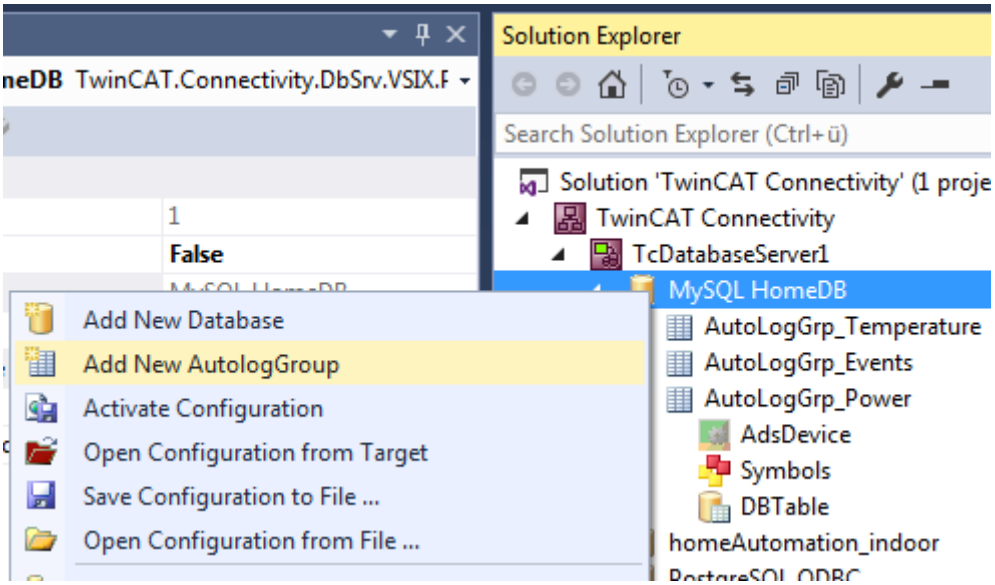
停用数据库配置

在项目中可以禁用各个数据库配置。在激活项目时，这些配置将被标记为红色并被忽略。

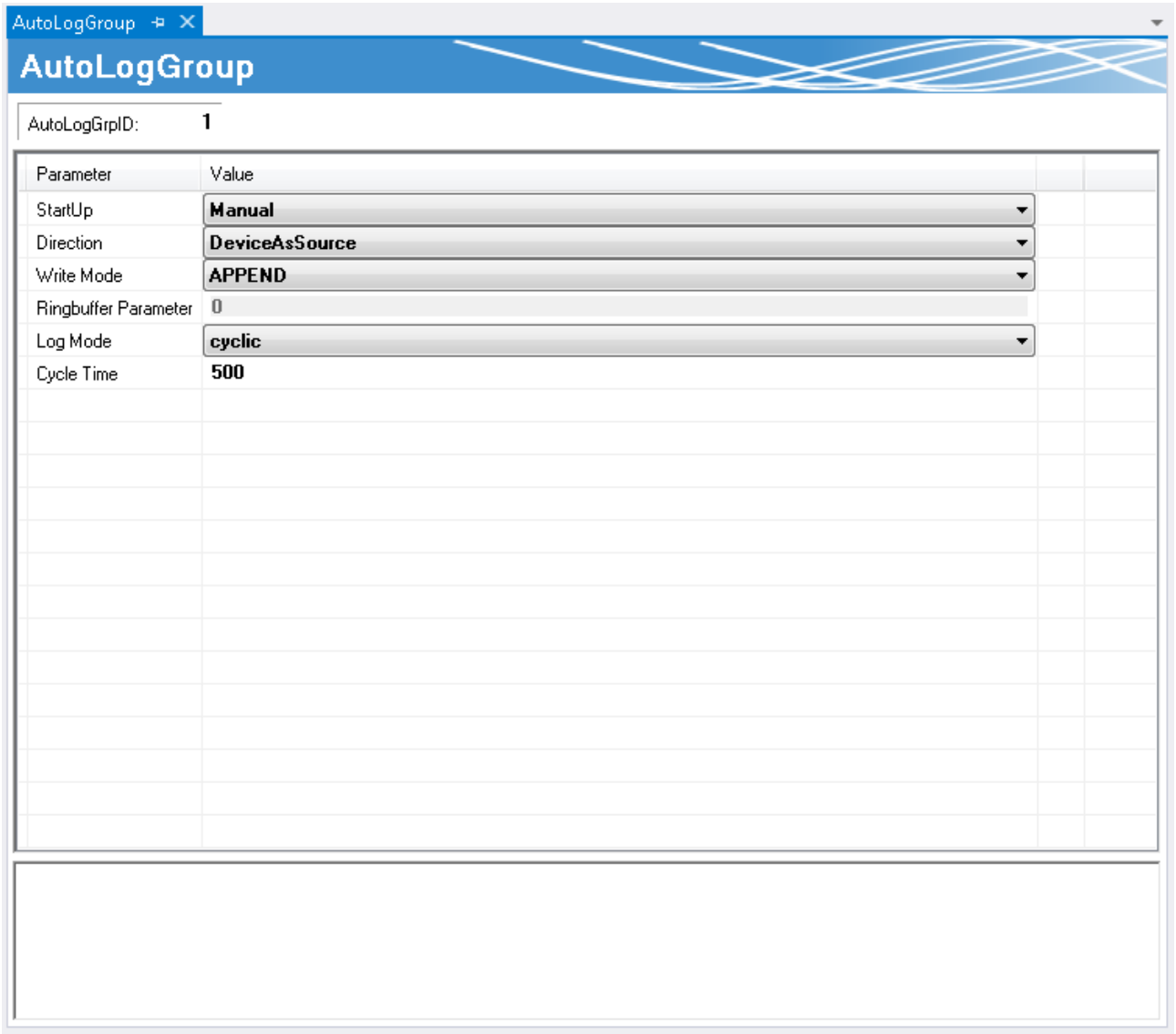


5.1.1.1.4 配置 AutoLog 组

通过数据库配置的上下文菜单中的 **Add New AutologGroup**（添加新的 Autolog 组）命令或工具栏可以为数据库配置添加新的 AutoLog 组。这些 AutoLog 组指的是父级数据库。



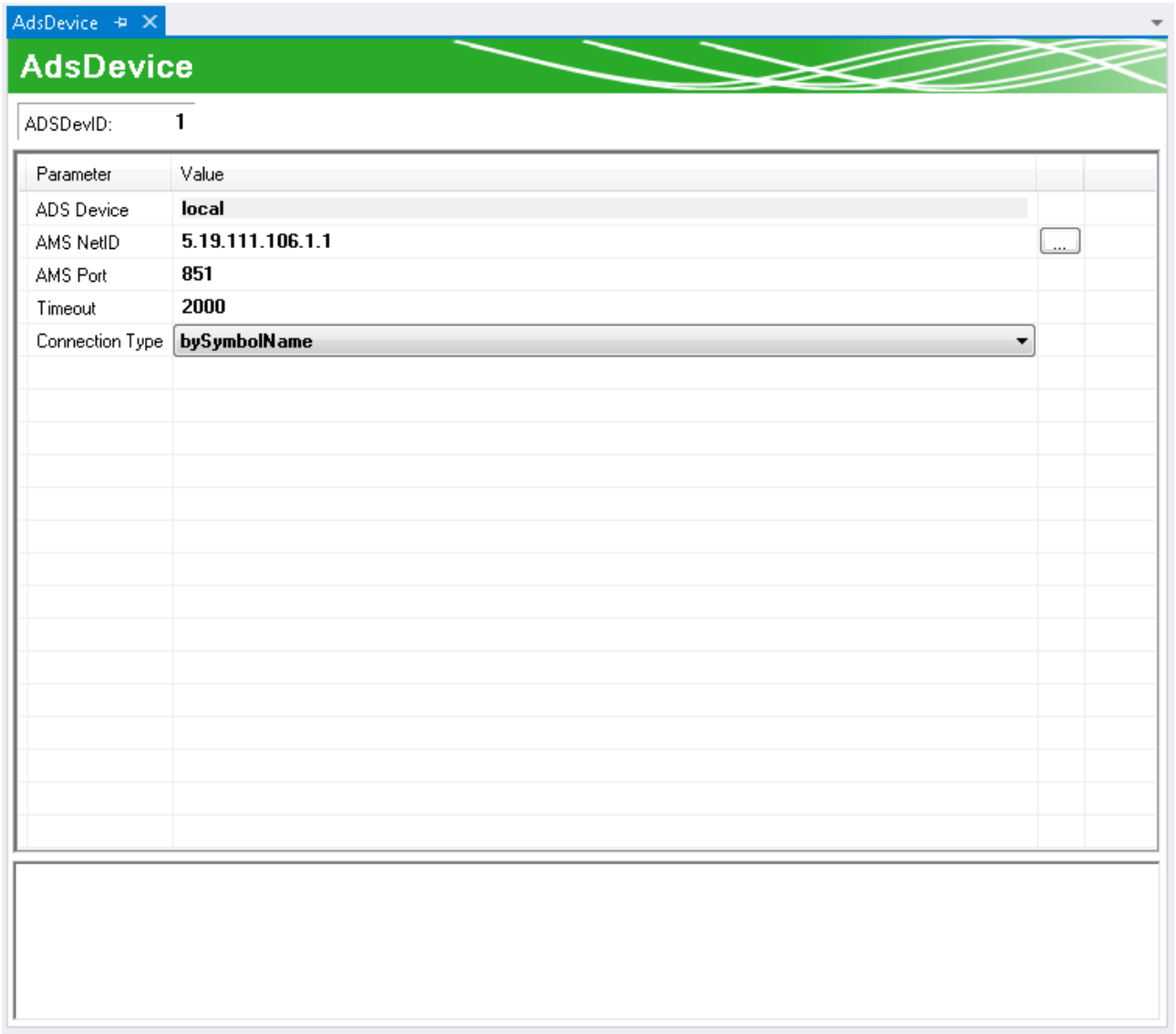
新的 AutoLog 组以及相应的组件作为文件被添加到项目文件夹中，并集成到项目中。它们包括 ADS 设备、符号组和表格设置。为了在项目中保存这些文件，您应保存 TwinCAT Connectivity 项目文件。然后，在编辑器或属性窗口中可以编辑这些文件。



启动	您可以手动启用 AutoLog 模式（通过 PLC 或配置器中的命令），或者在系统启动时自动启用该模式。
方向	设定的 ADS 设备可用作数据目标或数据源。
写入模式	这些数据可以逐行添加到数据库中，也可以按时间或数量保存在环形缓冲区中，或者只是在相应的位置进行更新。
环形缓冲区参数	根据设置的不同，该参数可表示环形缓冲区更新的时间或周期。
日志模式	在经过特定的循环时间后或在发生变化时，就会写入变量。
周期时间	写入变量后的周期时间。

配置 ADS 设备

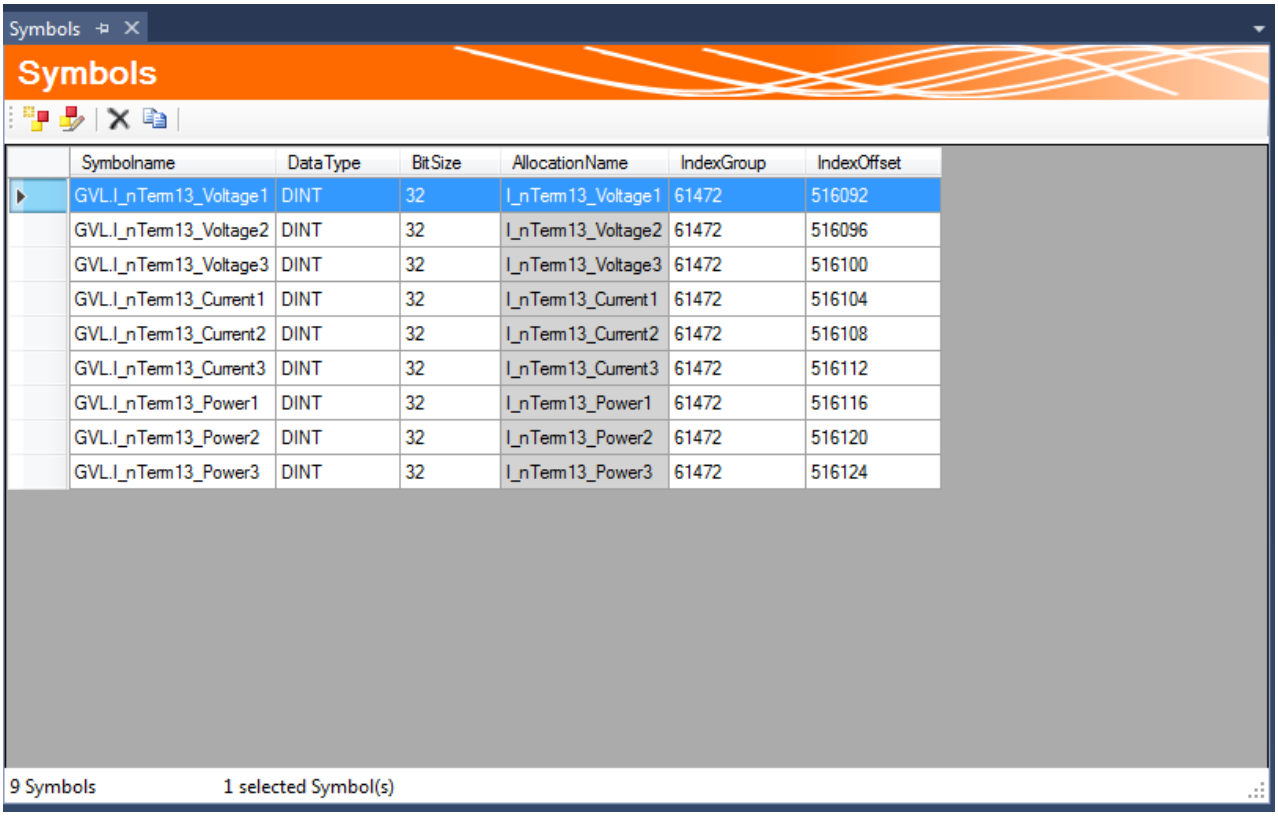
在 AutoLog 组下会自动创建 ADS 设备。在最常见的用例中，ADS 设备为 PLC Runtime。在编辑器中可以设置以下参数：



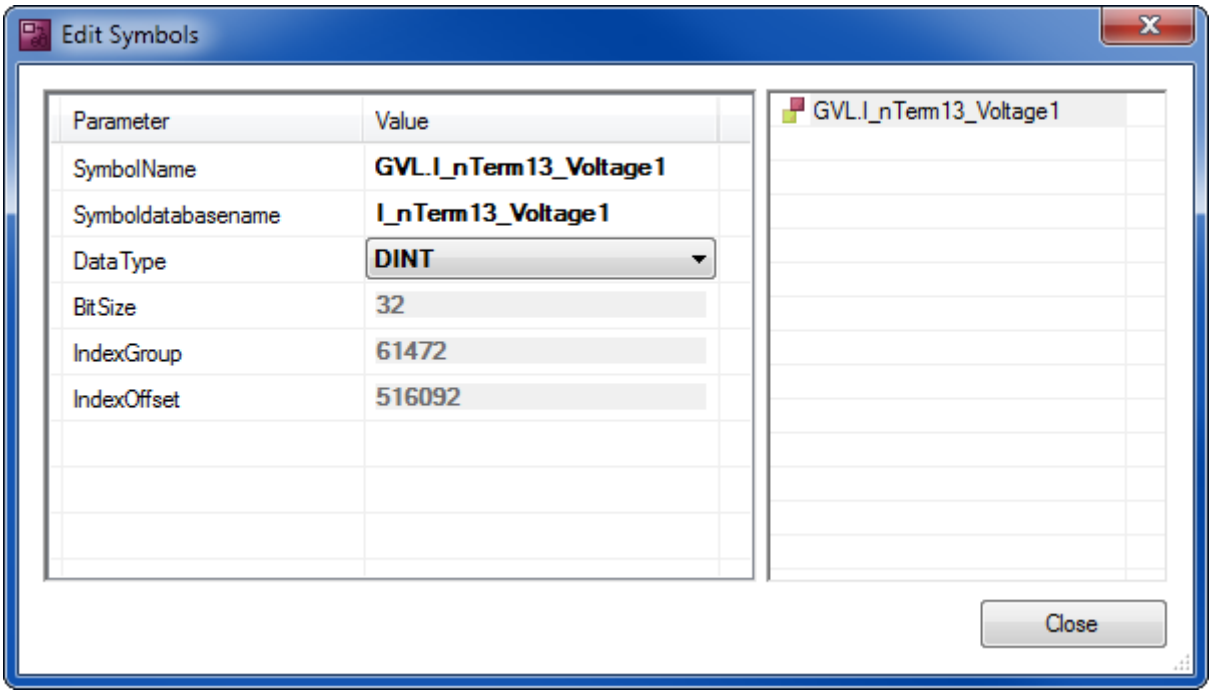
ADS 设备	ADS 目标设备的名称。
AMS NetID	TwinCAT 网络中的目标设备的地址。
AMS 端口	TwinCAT 网络中的目标设备的端口。
超时	与目标设备失去连接的时间。
连接类型	bySymbolName：根据符号名称建立连接。 byIndexGroup：根据内存索引建立连接。

配置符号

您在这里设置的符号会被写入数据库或从数据库读取，具体取决于 ADS 设备是数据目标还是数据源。TwinCAT Target Browser [▶ 49]可以用于便捷访问。在这里，您可以搜索目标系统上的符号，并通过拖放操作在 2 个工具之间进行通信。



您还可以将符号手动添加到符号组或进行编辑。所需的信息各不相同，具体取决于在 ADS 设备中是通过符号名称还是通过索引组来选择连接类型。起始点始终是 ADS 设备。



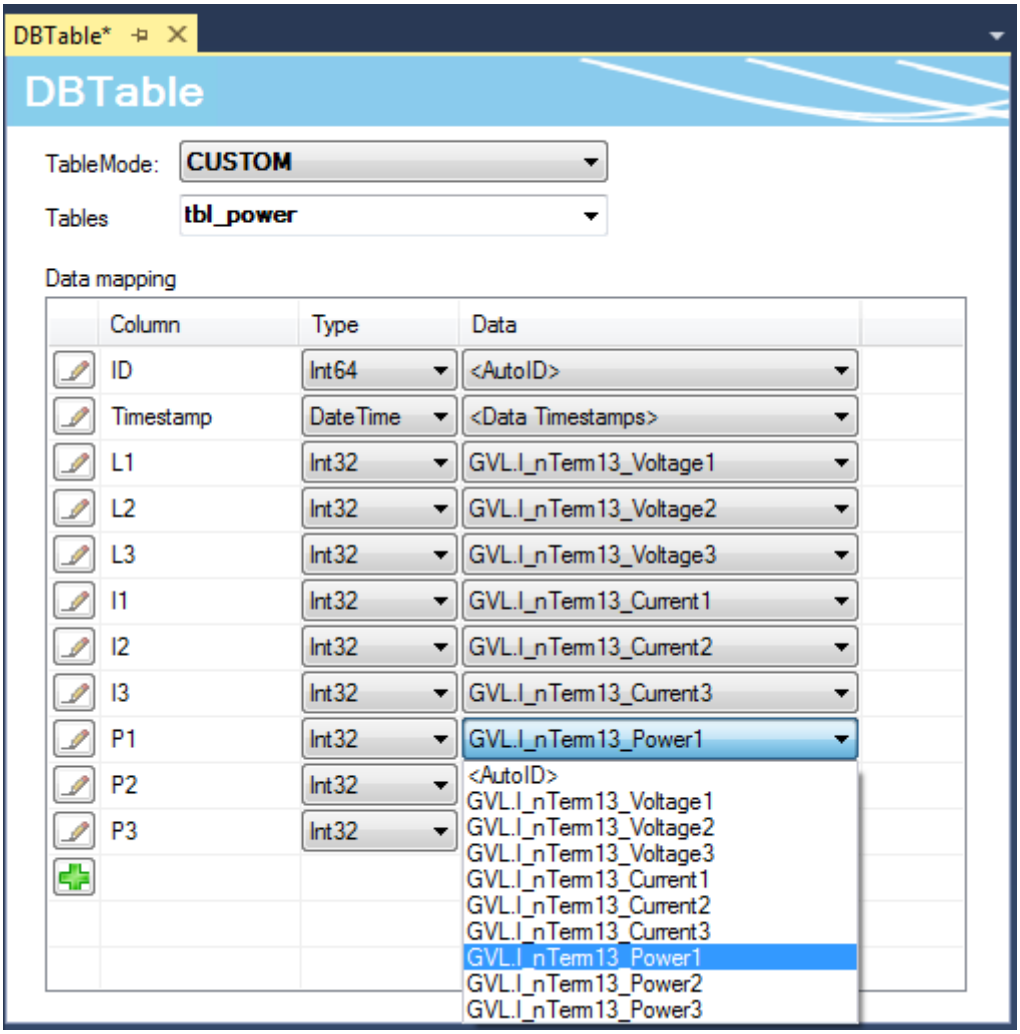
SymbolName	根据设置的 ADS 设备，对符号进行寻址
符号数据库名称	数据库表中的变量的名称
DataType	符号的 PLC 数据类型
BitSize	符号的位大小（为数据类型自动设置）
IndexGroup	TwinCAT 系统中的索引组
IndexOffset	TwinCAT 系统中的索引偏移

配置表格

数据库中的表格可以基于标准表结构，也可以基于个性化结构。

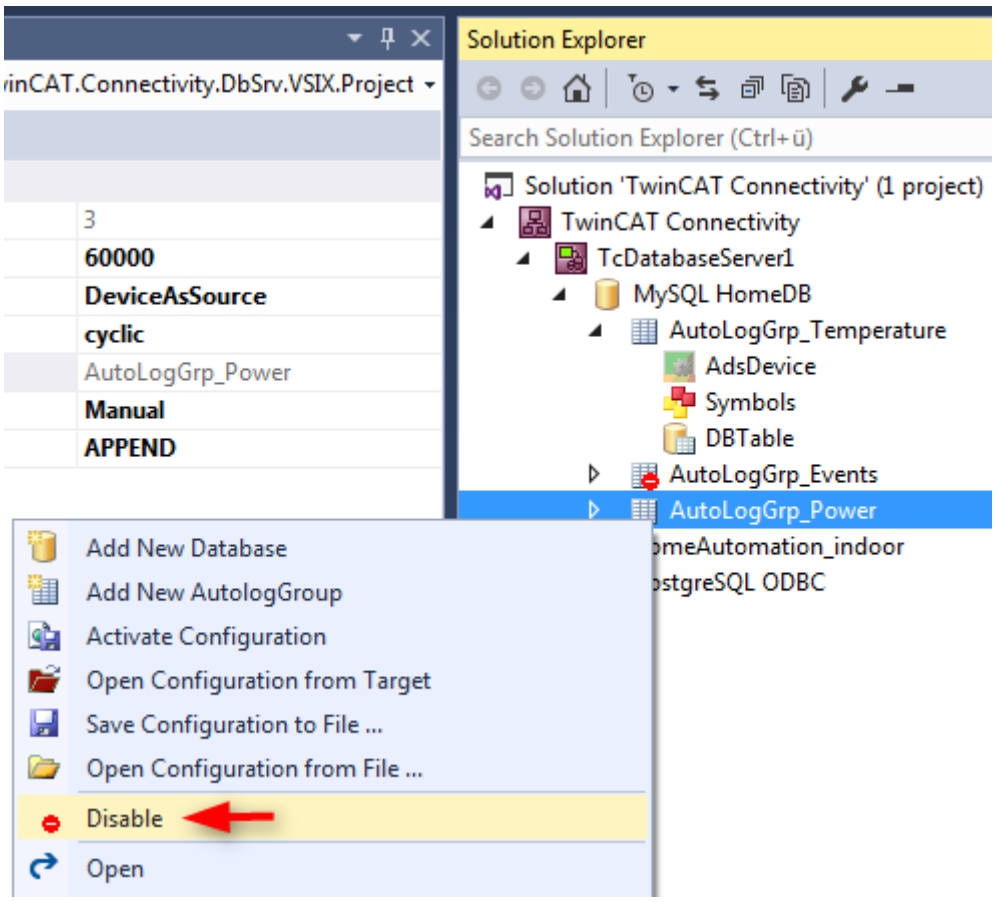
从可能的表格列表中可以选择相应的表格。如果表格尚不存在，您可以通过 SQL 查询编辑器创建它。如果您选择标准表结构，则蓝色勾号表示选定的表格是否与该结构相对应。

特定的表格类型提供了根据需要将在符号组中设置的各个符号分配到数据库中的表格列的选项。在 AutoLog 模式下将数据集写入数据库时，采样时符号组的当前值将保存在相应的表格列中。



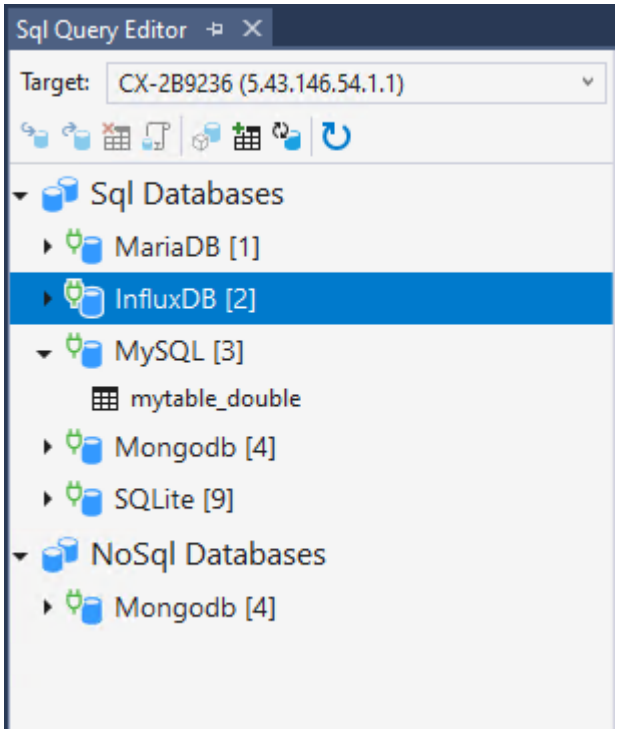
禁用 AutoLog 组

与各个数据库配置一样，在项目中也可以禁用各个 AutoLog 组。在目标系统上启用项目时，这些组将被忽略。停用的 AutoLog 组以红色标记表示。使用相同的命令可以将其重新激活。



5.1.1.1.5 SQL 查询编辑器

SQL 查询编辑器是一种数据库服务器工具，为您的应用程序开发提供支持。该工具可以用于测试连接状态和 SQL 命令，并检查 PLC 和数据库之间的兼容性。



在选择目标系统的 TwinCAT 数据库服务器之后，SQL 查询编辑器将加载当前的数据库配置以及已成功连接的数据库的表格。根据数据库是否支持 SQL 和 NoSQL 接口（来自 TwinCAT 数据库服务器），它将在相应的类别下列出。

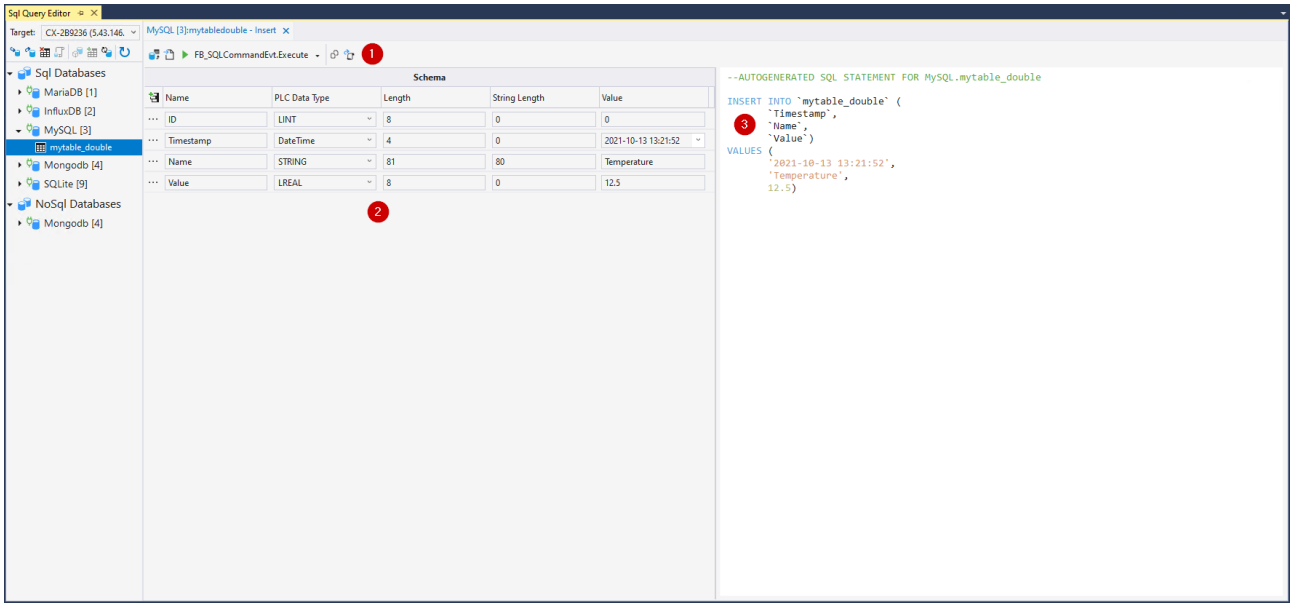
在选择目标系统下方，有一个带有可用命令的状态栏：

表格层	
插入工作区	打开插入工作区，使用 SQL 将数据集写入选定的表格。
选择工作区	打开选择工作区，使用 SQL 从选定的表格读取数据集。
删除/丢弃工作区	打开删除/丢弃工作区，使用 SQL 从选定的表格中删除数据集或删除整个表格。
NoSQL 工作区	打开 NoSQL 工作区，执行 NoSQL 特定的查询。
数据库层	
存储过程工作区	打开存储过程工作区，执行数据库的存储过程。
表格工作区	打开表格工作区，在选定的数据库中创建新的表格。
更新表格	更新选定数据库的可用表格。
一般信息	
更新数据库	更新整个数据库树。

在相应选项卡下的树形结构的右侧会打开工作区。此外，在同一个表格中还可以同时打开多个选项卡。

插入工作区

插入工作区允许通过 SQL 功能块的 TwinCAT 数据库服务器接口将数据写入选定的表格中。



在下部区域（2）有一个表格，其中列出了要写入的数据集中的各个数据符号。在这里可以确定名称、PLC 数据类型、字节长度以及值。然后，通过命令可以使用所输入的值生成 SQL 语句。

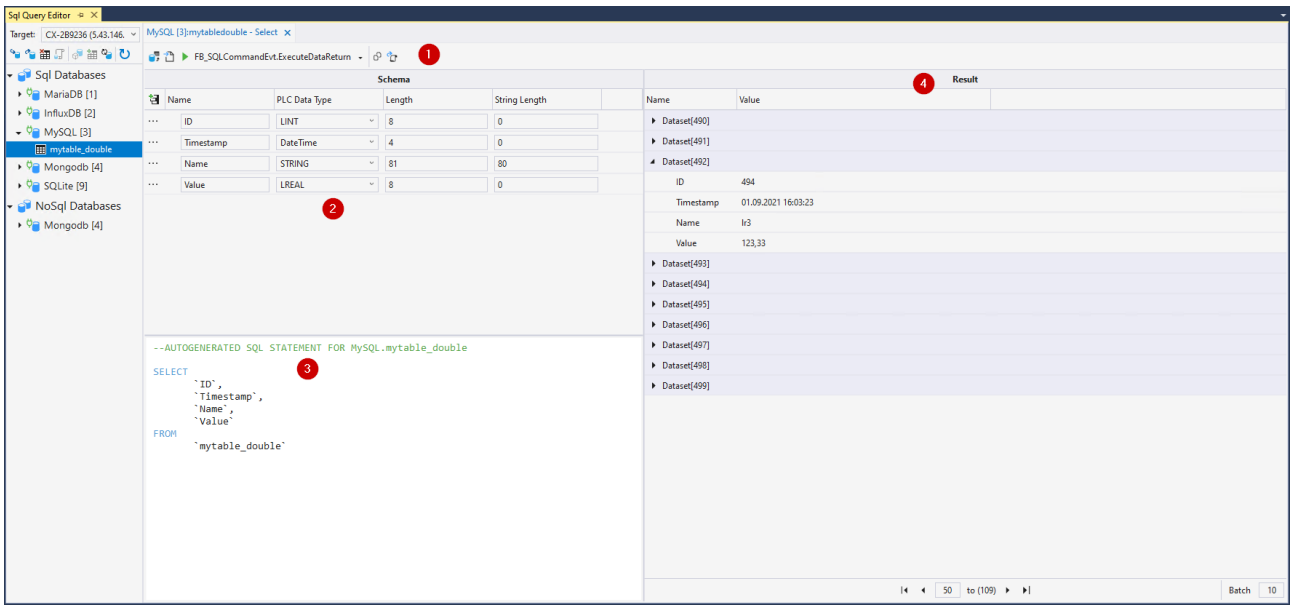
然后，该 SQL 语句就会在文本字段（3）中出现。根据数据库语法的不同，内容可能会有所差异。

上部状态栏包含与 TwinCAT 数据库服务器交互的命令（1）。

命令	**描述**
读取表格结构	读出工作区的表格的表格结构。
生成 SQL 语句	根据数据库语法的不同，从当前表格生成 SQL 语句。
执行	通过 TwinCAT 数据库服务器的相应接口执行文本字段（3）中的语句。
复制语句	将文本字段（3）中的语句复制为与 TwinCAT 兼容的语法。
导出为结构	将输入值表格的结构导出到与 TwinCAT 3 兼容的 DUT 中。

选择工作区

选择工作区允许通过 TwinCAT 数据库服务器接口提供的 SQL 功能块将数据读取到选定的表中。



在下部区域（2）有一个表格，其中列出了要写入读取的数据集中的各个数据符号。在这里可以确定名称、PLC 数据类型以及字节长度。然后，需要这些信息来解释数据。

然后，该 SQL 语句就会在文本字段（3）中出现。根据数据库语法的不同，内容可能会有所差异。

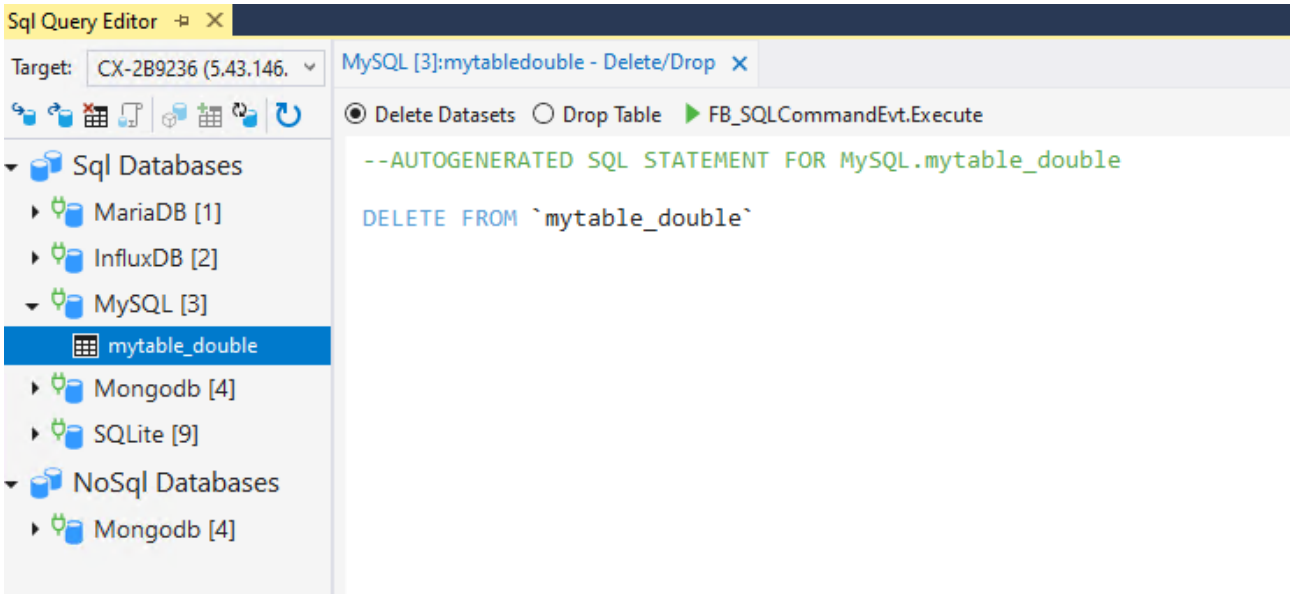
在执行语句后，结果字段（4）会显示数据。如果返回多个结果，则可以在页面中进行切换。

上部状态栏包含与 TwinCAT 数据库服务器交互的命令（1）。

命令	描述
读取表格结构	读出工作区的表格的表格结构。
生成 SQL 语句	根据数据库语法的不同，从当前表格生成 SQL 语句。
执行	通过 TwinCAT 数据库服务器的相应接口执行文本字段（3）中的语句。
复制语句	将文本字段（3）中的语句复制为与 TwinCAT 兼容的语法。
导出为结构	将输入值表格的结构导出到与 TwinCAT 3 兼容的 DUT 中。

删除/丢弃工作区

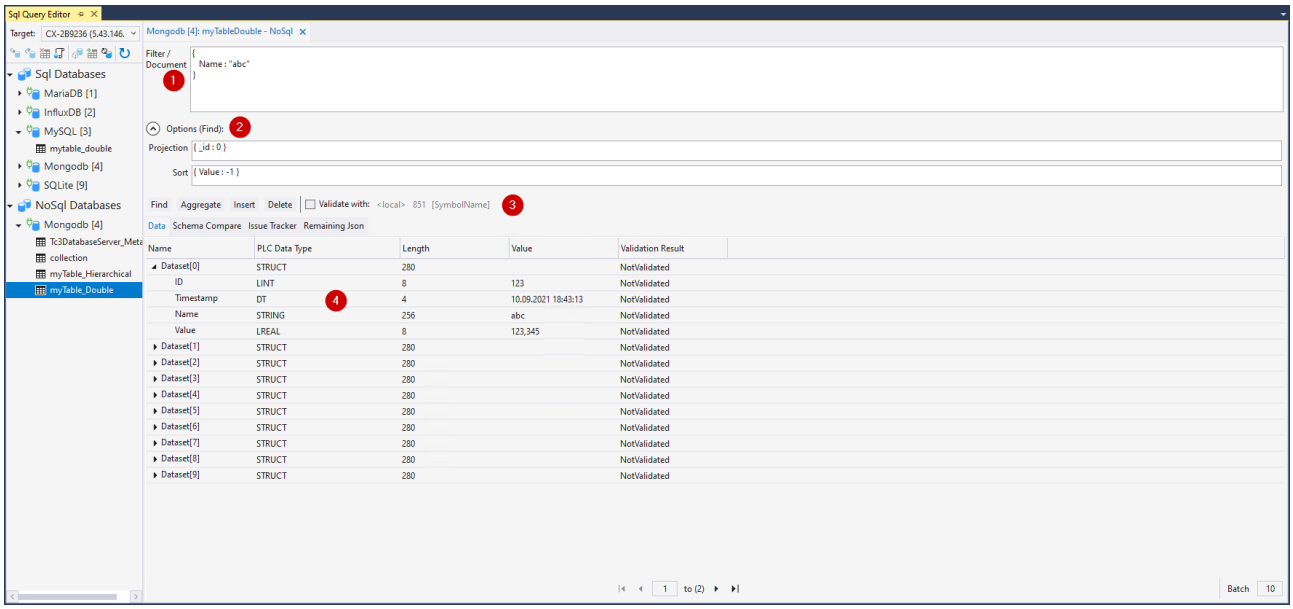
删除/丢弃工作区提供了发布 SQL 语句的选项，可从表格中删除数据或从数据库中删除整个表格。



为此，您可以在状态栏中选择这两个选项。然后，在语句字段中会生成与数据库相对应的语法。要使用 TwinCAT 数据库服务器接口执行此操作，可使用开关 **FB_SQLCommandEvt.Execute**。

NoSql 工作区

NoSql 工作区支持 NoSql 数据库或 TwinCAT 数据库服务器接口为 NoSQL 提供的特殊功能。



ID	Name	功能
1	筛选器/文档	根据所使用的功能的不同，该输入字段可充当文档或 JSON 格式的筛选器。如果您既要执行查找操作，又要执行投影或排序操作，您可以在下面的选项（查找）中填写这些字段。
2	选项（查找）	描述查找功能的其他参数，例如，投影或排序。
3	控制元素	用于与 NoSQL 的 TwinCAT 数据库服务器接口交互的控制元素。
4	数据显示	返回数据的列表。使用导航可以逐页浏览。

查找：使用在文本字段（1）中输入的筛选器执行搜索查询。或者，通过选项（查找）字段也可以执行投影或排序操作。数据将返回并在数据显示（4）中列出。筛选器的语法是数据库特定的。

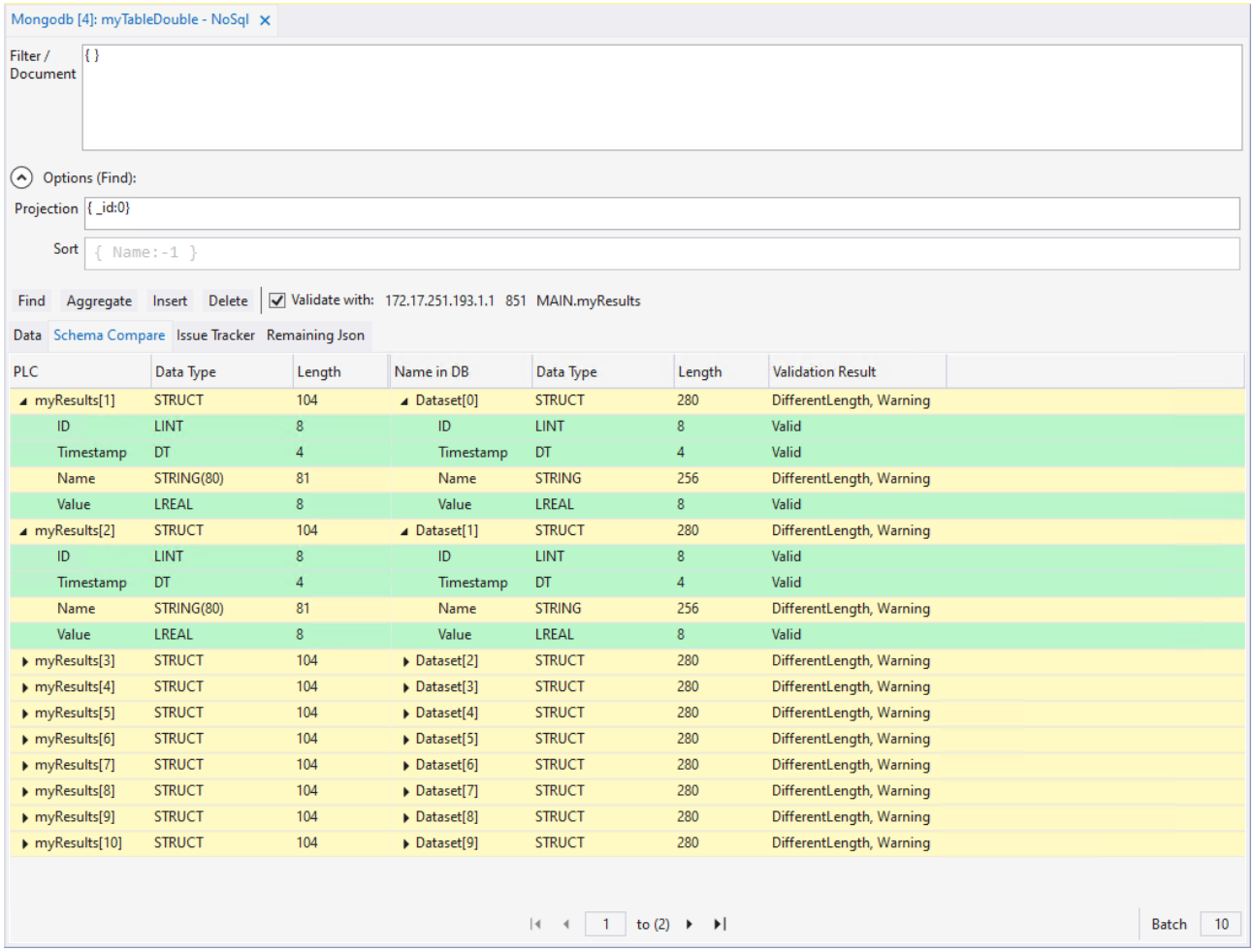
聚合：使用在文本字段（1）中输入的参数执行聚合。数据将返回并在数据显示（4）中列出。筛选器的语法是数据库特定的。

插入：执行在文本字段（1）中输入的（JSON）文档或文档数组的插入查询。然后将它们写入数据集。

删除：对使用文本字段（1）中的筛选器找到的数据执行删除查询。从数据集中删除任何找到的数据。

验证：如果选择该选项，则数据查询不会根据自己的结构自动进行解析，而会尝试将这些数据映射到通过这些参数指定的 PLC 中的符号结构。

使用后一种功能时，查找查询可能会导致冲突。与 PLC 过程映像中的结构相比，NoSQL 数据库中的数据集不必遵循固定的结构。查询的文档可能没有 PLC 结构中的特定元素的数据。或者数据集包含的数据没有在 PLC 结构中出现。在 PLC 中，这些数据通过名称或“ElementName”属性进行分配。



通过 **Schema Compare**（模式比较）选项卡可以检查数据中的差异。从上述示例中可以看出，在 PLC 结构中返回的文档中，变量“名称”的数据类型长度与数据库的数据类型长度不同。相应的颜色可显示冲突的权重：

红色：可用的数据过多或过少。

黄色：数据集的字节长度不匹配，或底层数据集剩余或缺失。

绿色：无冲突

这些冲突也在 **Issue Tracker**（问题跟踪）选项卡下列出。如果需要，也可以将它作为字符串数组读入 PLC。

Remaining Json（剩余 Json）选项卡以 JSON 形式返回任何剩余数据集。也可以将该信息作为字符串读入 PLC。

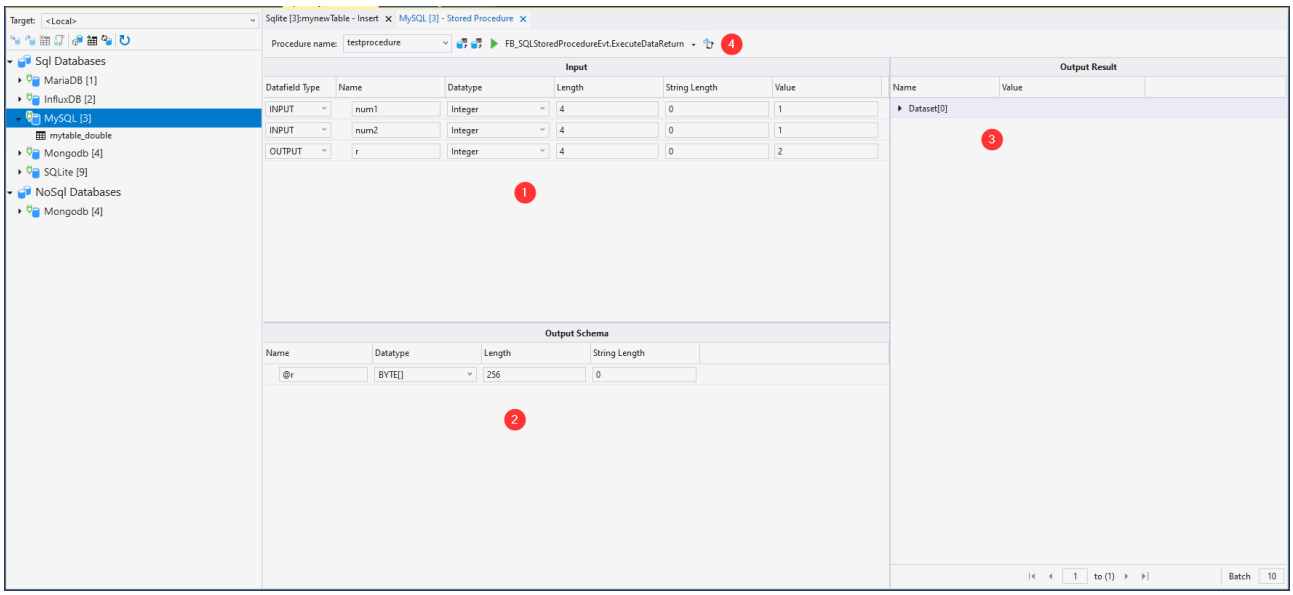
根据其他显示数据可知，状态栏中的控制元素可以用于遍历数据。可以指定同时显示的数据集的数量。

存储过程工作区

TwinCAT 数据库服务器支持“存储过程”，它支持大量的数据库，用于在数据库层处理更加复杂的查询或提供简化的接口。

如果数据库中存在**存储过程**，则它们将在状态栏（4）的下拉列表中列出。

下面是输入参数（1）和输出结构（2）的表格。此外，还有一个输出结果（3）的视图。如果成功执行**存储过程**，则会在这里显示结果。

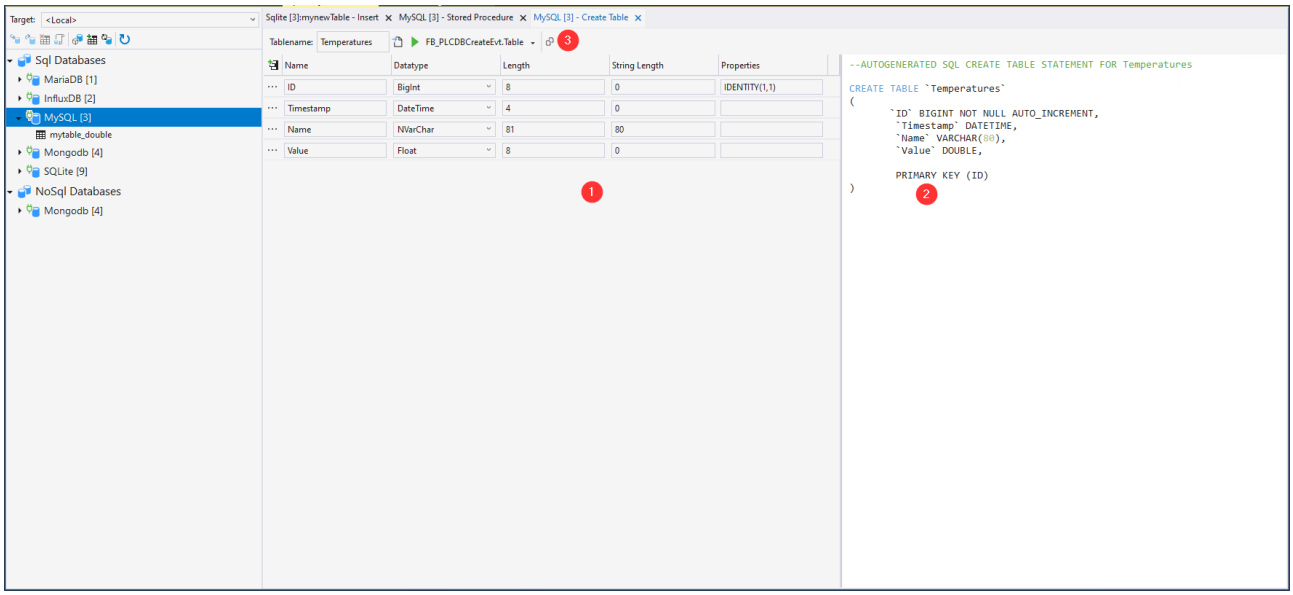


状态栏有以下命令：

命令	描述
读取存储过程的输入结构	读出输入参数结构。结果在表 1 中显示。
读取存储过程输出结构	读出输出参数结构。结果在表 2 中显示。**信息：** 这需要执行存储过程。根据编程的不同，在这里可以更改数据。
设计	通过 TwinCAT 数据库服务器的相应接口执行存储过程。
导出为结构	将表格的结构导出到与 TwinCAT 3 兼容的 DUT 中。

表格工作区

表格工作区可用于创建新的表格。

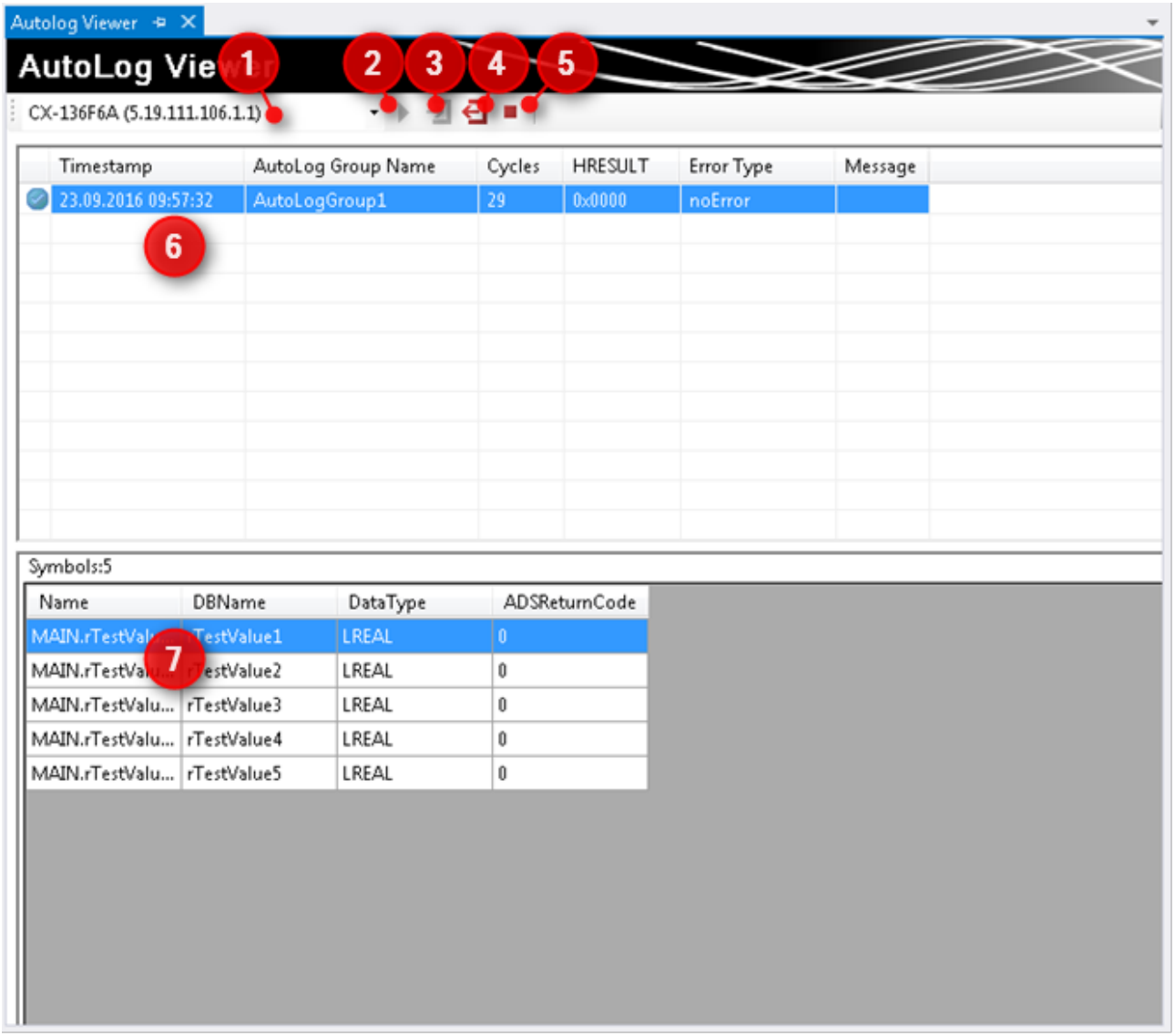


在这里可以创建表结构（1），并在相应的字段（2）中从其中生成 SQL 语句。为此，可以使用带有以下命令的状态栏（3）：

命令	描述
表格名称	指定新表格的表格名称。
生成 SQL 语句	根据数据库语法的不同，从当前表格生成 SQL 语句。
执行	通过 TwinCAT 数据库服务器的相应接口执行存储过程。
复制语句	将文本字段（2）中的语句复制为与 TwinCAT 兼容的语法。

5.1.1.1.6 AutoLog 视图

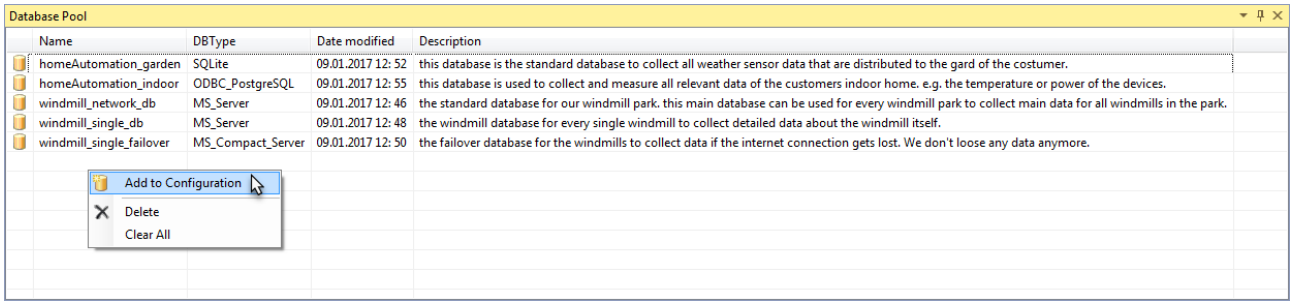
TwinCAT 数据库服务器的 AutoLog 查看器是一种用于控制和监控 AutoLog 模式的工具。您可以登录目标系统，这类似于 TwinCAT PLC。在登录状态下，可以启动或停止 AutoLog 模式。有关日志记录的当前状态的信息在窗口的下部显示。在选择 AutoLog 组时，通过记录的符号可显示更多信息。



ID	Name	功能
1	目标系统	选择已安装 TwinCAT 数据库服务器的目标系统
2	启动	手动启动 AutoLog 模式
3	登录	登陆进入活动状态的 AutoLog 进程
4	退出	退出活动状态的 AutoLog 进程
5	停止	手动停止 AutoLog 模式
6	AutoLog 组	目标系统上已配置的 AutoLog 组的列表
7	符号	选定 AutoLog 组的已配置符号列表

5.1.1.1.7 数据库配置池

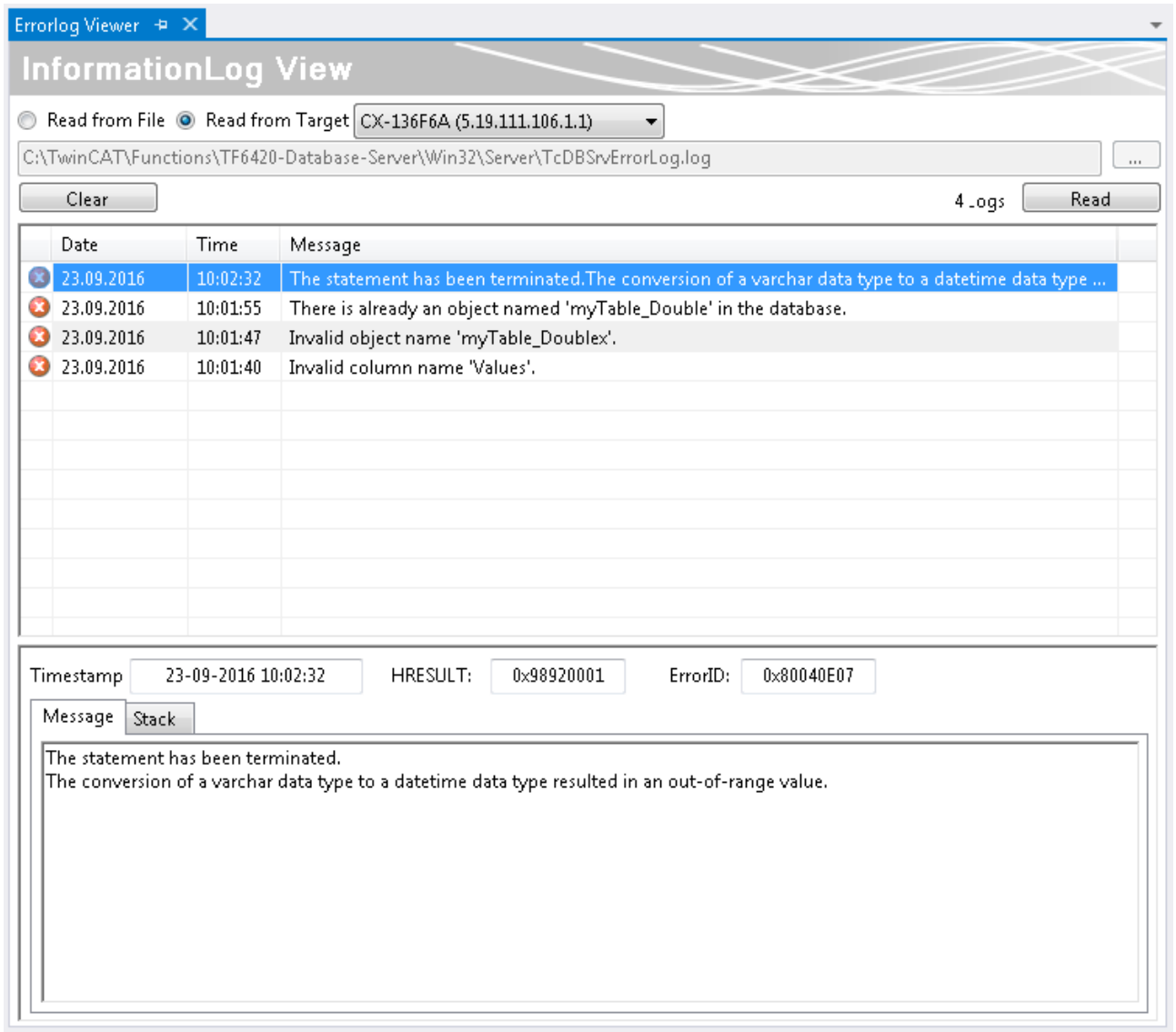
数据库配置池是开发系统上的数据库配置的全局存储库。开发人员可以将其用作独立于项目的数据库配置的存储位置，或者是可重复使用配置的模板。对于 Visual Studio 集成和独立配置器，该池使用相同的用户特定存储位置。在卸载 TwinCAT 数据库服务器时，将会保留这些文件。



5.1.1.1.8 InformationLog View

InformationLog View是一种从 TwinCAT 数据库服务器读取日志文件的工具。记录的信息会以纯文本的形式显示，包含时间戳、ID 和错误消息。

不仅可以直接通过文件访问功能来查看或清空日志文件，还可以直接通过目标系统实现这些操作。这在同一网络下有分布式数据库服务器时特别有优势，可以借此快速、方便地访问日志文件。要进行这种访问，必须有一条连接目标设备的路由。

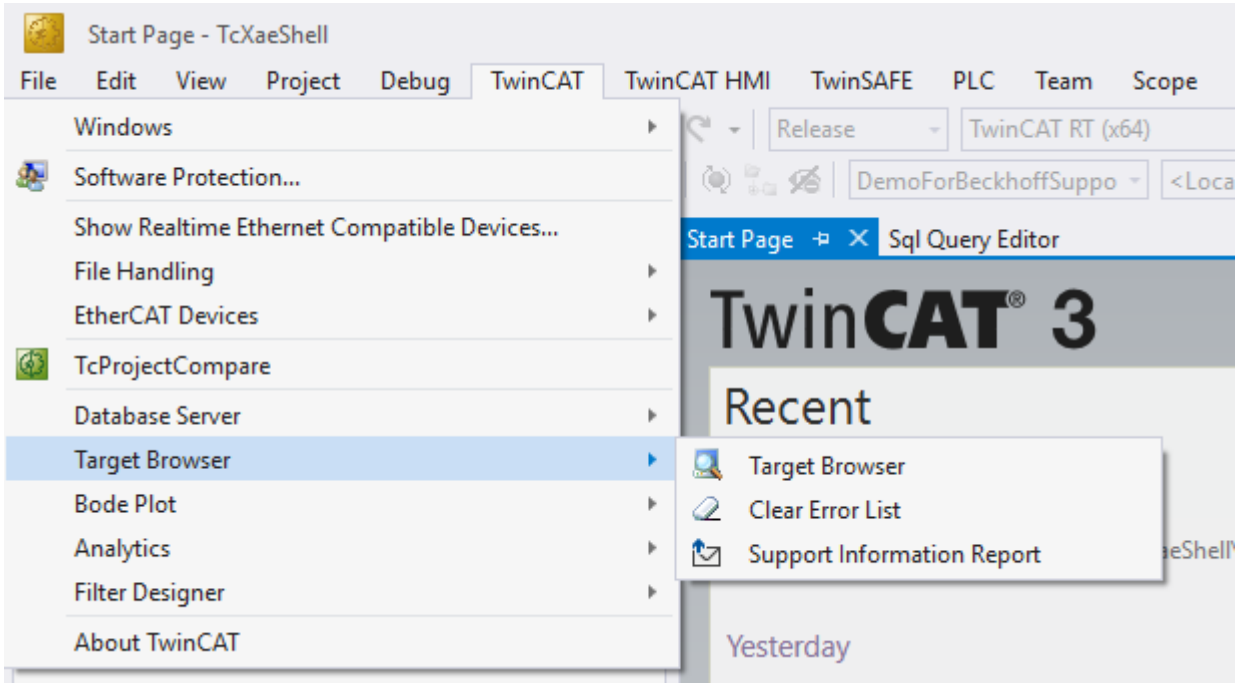


5.1.1.1.9 TwinCAT Target Browser

TwinCAT Target Browser 是 TwinCAT 3 开发环境的中央数据管理界面。它可以通过 ADS 实时提供来自各 TwinCAT 目标系统的数据，或者访问数据库中的历史数据。凭借开发环境系统中安装的 TwinCAT 3 功能组件，TwinCAT Target Browser 的 ADS 标准扩展可通过其他扩展程序进行扩展（参见”扩展 [▶ 49]“）。

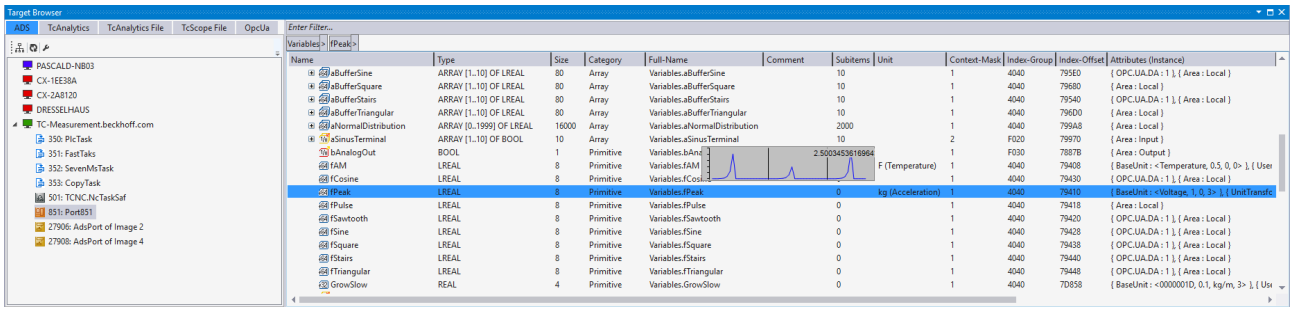
调用选项

可通过 Visual Studio® 中的 Scope 菜单来调用 Target Browser。由于其在 TwinCAT 系统中的应用日益广泛，因此也可以通过TwinCAT 菜单下的 “**Target Browser**”项 打开 Target Browser。



此外，使用 Target Browser 的各工具还可提供其他调用方式。例如，有些产品会通过各自项目节点的上下文菜单来实现这一功能。

架构



在 Microsoft Visual Studio® 中，TwinCAT Target Browser 是一个工具窗口，分为 2 个部分。目标系统显示在左侧（“特定目标区”）。您可以通过选项卡在各项扩展之间进行切换。右侧（“通用符号区”）分别显示了各目标系统或所选对象的详细信息。

许多扩展程序都支持变量的 “Value Preview”（值预览）功能。这表示，如果您选择一个变量并按住空格键，就会出现一个小图表。这样便可以确定数据是否已经到达数据库或存在于数据库中。右上方的搜索栏可以筛选/缩减可见符号。必须用 [Enter] 键确认输入内容。

面包屑导航栏会显示当前定位。

扩展

下表显示了对当前扩展以及安装这些扩展的产品的概述。有关扩展的更多信息，请参阅相关文件章节。

扩展名	产品
ADS [▶ 50]	TwinCAT XAE、TwinCAT Scope、TwinCAT IoT Data Agent、TwinCAT Database Server、TwinCAT Analytics Service Tool
OPC UA [▶ 52]	TwinCAT Scope、TwinCAT IoT Data Agent
TcAnalytics [▶ 54]	TwinCAT Analytics Workbench 和 Service Tool、TwinCAT Scope
TcAnalytics File [▶ 56]	TwinCAT Analytics Workbench 和 Service Tool、TwinCAT Scope
TcDBSrv [▶ 57]	TwinCAT Scope

5.1.扩展 — ADS

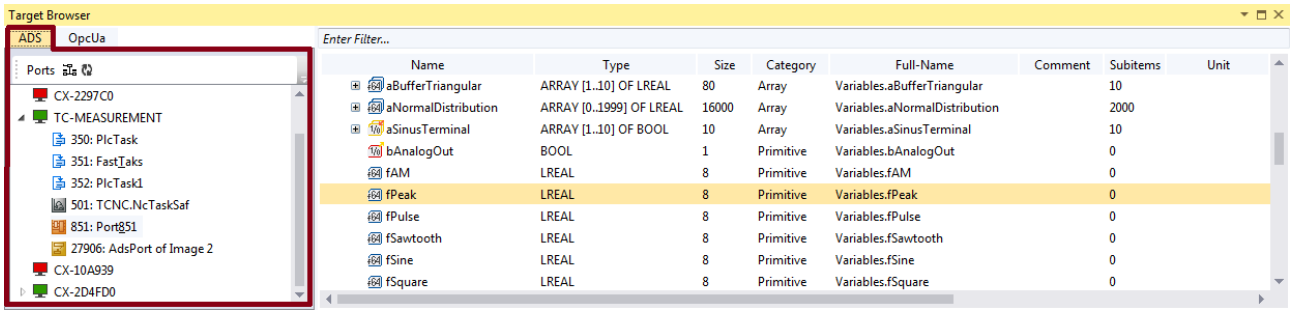
1.1.

9.1

TwinCAT Target Browser ADS 扩展最常用在 TwinCAT 系统中。

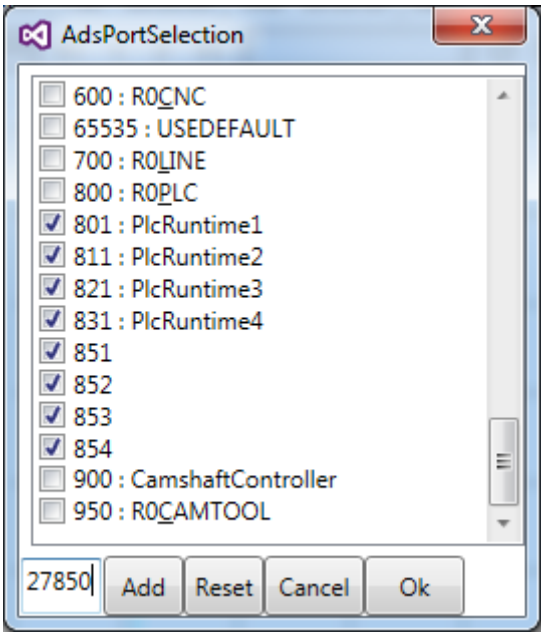
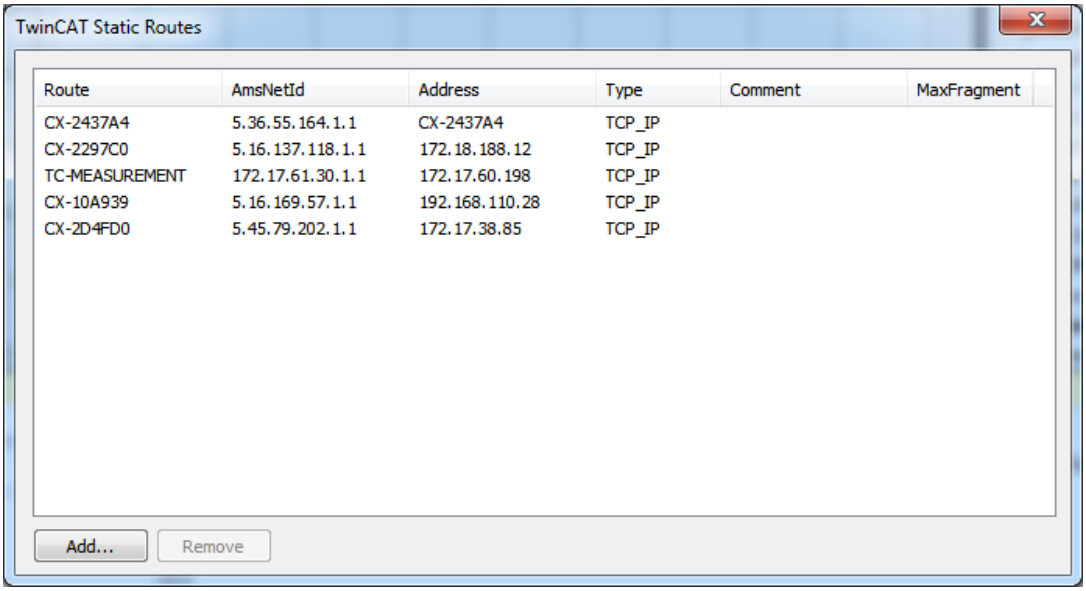
特定目标区

在本地 TwinCAT 3 开发环境中注册的所有目标系统都以树形结构显示在 TwinCAT Target Browser (ADS) 的左侧区域。首先是本地系统，然后是目标系统，如工业 PC 或嵌入式控制器，按注册顺序进行排列。带前缀的屏幕符号表示系统的状态（绿色：运行模式，蓝色：配置模式，红色：停止模式或无法连接）。可用的默认 ADS 端口列于目标系统下方。如果选择了一个端口，会在 TwinCAT Target Browser 右侧的通用符号区显示可用符号/变量。



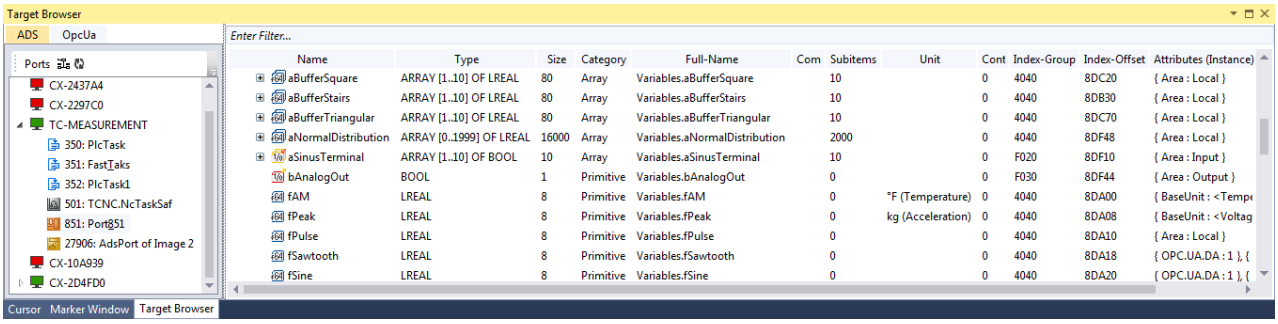
工具栏

ADS 扩展的工具栏提供以下功能：

Ports（端口） Ports	如果默认不显示 ADS 端口，可通过该命令添加更多端口以供选择。 
Edit Routes（编辑路由）	如果缺少通往目标系统的 ADS 路由，可通过该按钮添加更多目标系统。 
Refresh（刷新）	可通过该按钮手动更新目标系统状态的显示。

通用符号区

所选端口的适用 ADS 符号显示在 TwinCAT Target Browser 的右侧区域。除了名称、数据类型、大小和符号名称外，还会显示地址和属性。特殊属性，如单位的属性，会在各自的列中进行解释和输出。

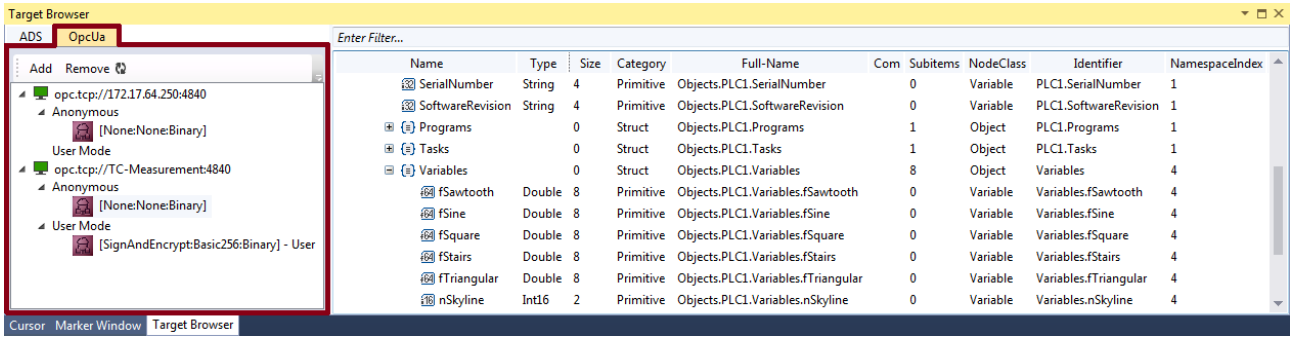


5.1.扩展 — OPC UA
1.1.
9.2

TwinCAT Target Browser OPC UA 扩展为 TwinCAT 3 开发环境提供了一种标准化方式。

特定目标区

使用工具栏中的 “**Add**”（添加）命令添加的所有 OPC UA 服务器均以树形结构显示在 TwinCAT Target Browser (OpcUa) 的左侧区域。树形结构第一级的服务器名称前的屏幕符号表示连接状态。在服务器下方创建的端点又细分为 “Anonym”（匿名）和 “Authenticated”（用户模式）。会在括号中显示各端点的加密方法。如果选择了一个端点，可用的 OPC UA 节点将显示在 TwinCAT Target Browser 的右侧（另请参见：显示 OPC UA 节点） [▶ 53]



工具栏

OPC UA 扩展的工具栏提供以下功能：

Add（添加） Add	可通过该命令建立与现有 OPC UA 服务器的新连接（另请参见：添加 OPC UA 服务器） [▶ 52]
Remove（删除） Remove	可通过该命令删除已注册的服务器。
Refresh（刷新）	可通过此按钮手动更新 Target Browser 树形结构中的显示内容。

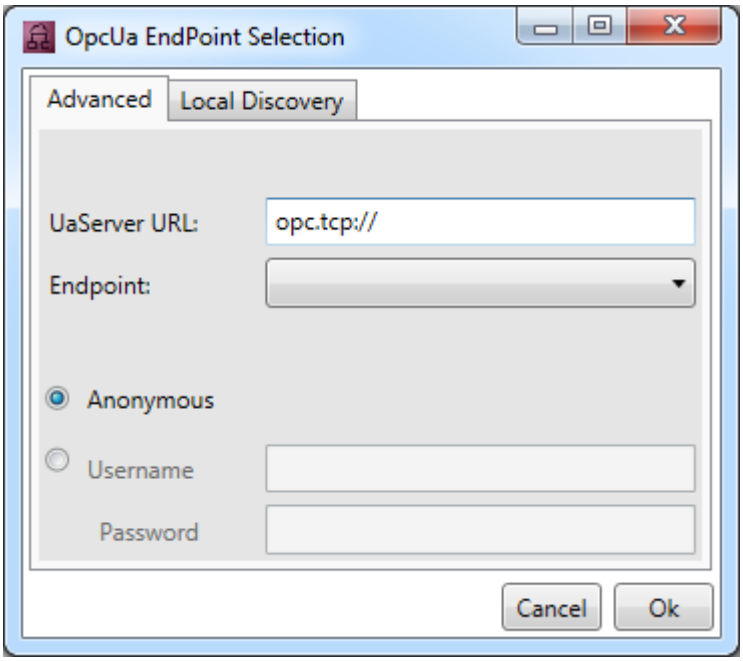
通用符号区

可用的 OPC UA 节点显示在 TwinCAT Target Browser 的右侧。这些节点体现了 PLC 项目的层次结构。除了名称、数据类型、大小和完整的对象名称外，还会显示节点类别和标识符。

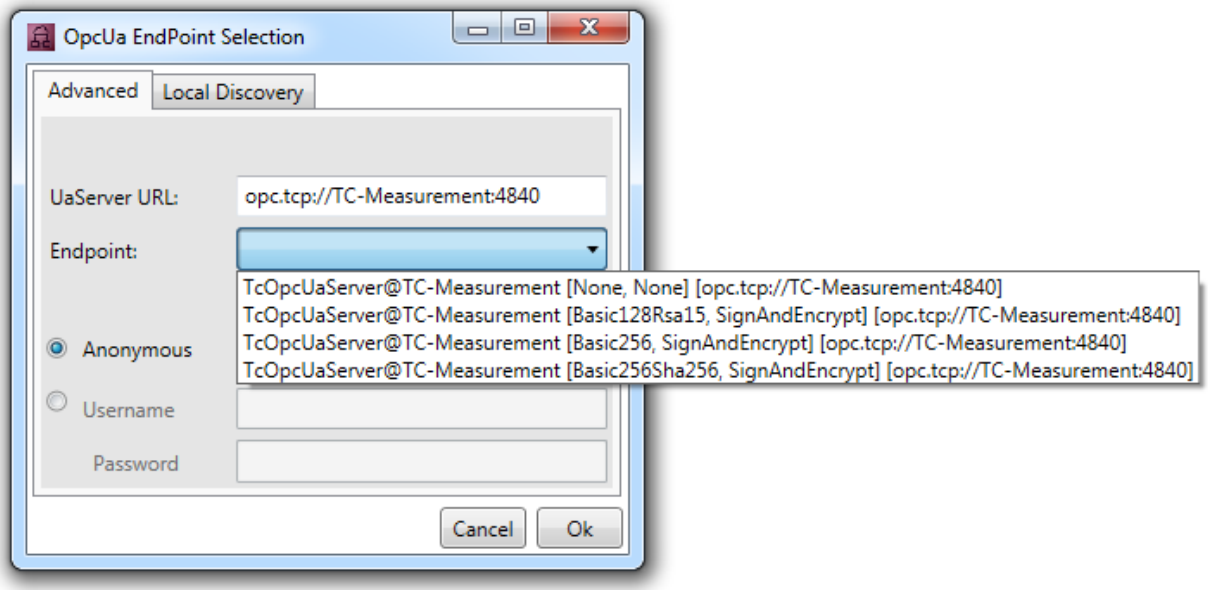
添加 OPC UA 服务器

1. 点击 OPC UA 工具栏中的“**Add**”（添加）。

⇨ “**OpcUa EndPoint Selection**”（OpcUa 端点选择）对话框打开。



2. 输入服务器的 URL。
3. 从下拉列表中选择端点。使用 OPC UA 时，可以通过相应的端点确定是否使用加密方式以及使用哪种加密方式。也可以在服务器中添加多个端点。为此，请再次执行“**Add**”（添加）命令。

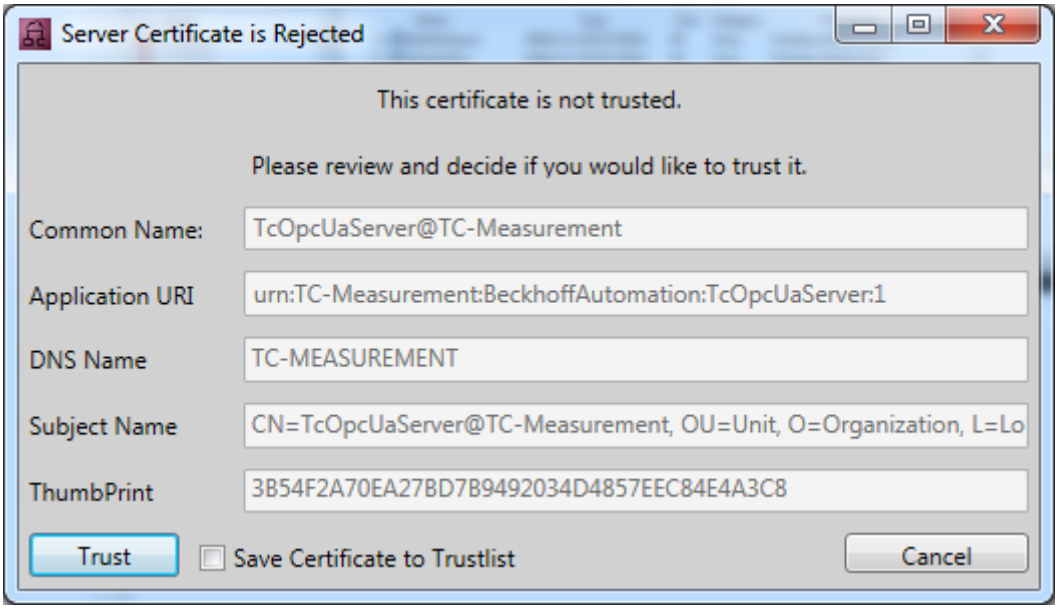


4. 选择是匿名访问还是身份验证访问。如果是身份验证访问，请输入用户名和密码。如果为 OPC UA 服务器设置了受密码保护的用户管理（例如，不同的用户帐户具有不同的权限），可能需要进行身份验证访问。
5. 确认对话框。

⇨ 已通过所选端点将 OPC UA 服务器添加到 Target Browser 的树形结构中。

显示 OPC UA 节点

若要显示可用的 OPC UA 节点，请在左侧的树形结构中选择相应的端点。如果选择的是未经认证访问的端点，将会直接显示节点。如果所选端点已通过认证，必须首先在相应对话框中信任服务器证书。



您可以单次信任该证书（在 Visual Studio 实例结束前），也可以勾选 “**Save Certificate to Trustlist**”（将证书保存至信任列表）复选框，将证书添加到信任证书列表中。

首次尝试连接 OPC UA 服务器时，还需要在服务器端信任客户端 (Target Browser) 的证书。为此，请将 OPC UA 服务器证书目录中的相应证书从 “rejected” 文件夹复制到 “trusted” 文件夹。

5.1.扩展 — TcAnalytics

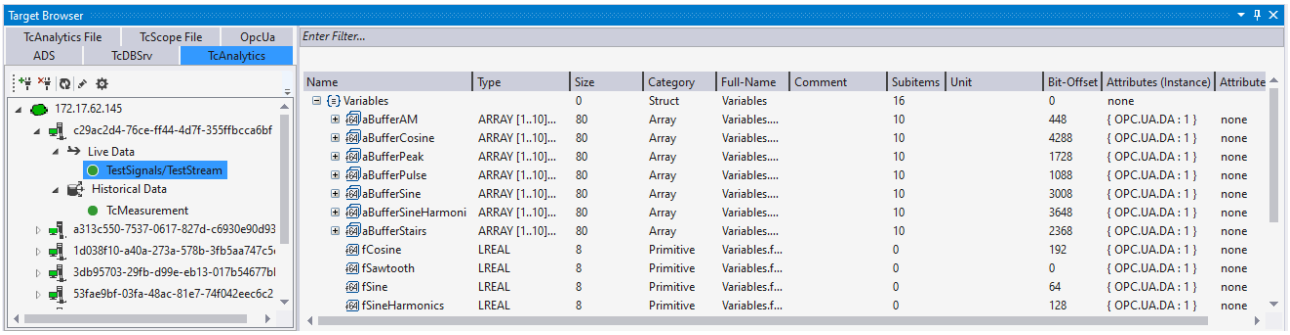
1.1.

9.3

借助 TwinCAT Target Browser 的 TcAnalytics 扩展，可以显示不同代理、不同主题的 MQTT 数据流，并且可以将其用于不同的测量产品。只需将所需的流符号拖放到相应的开发环境工具中即可。

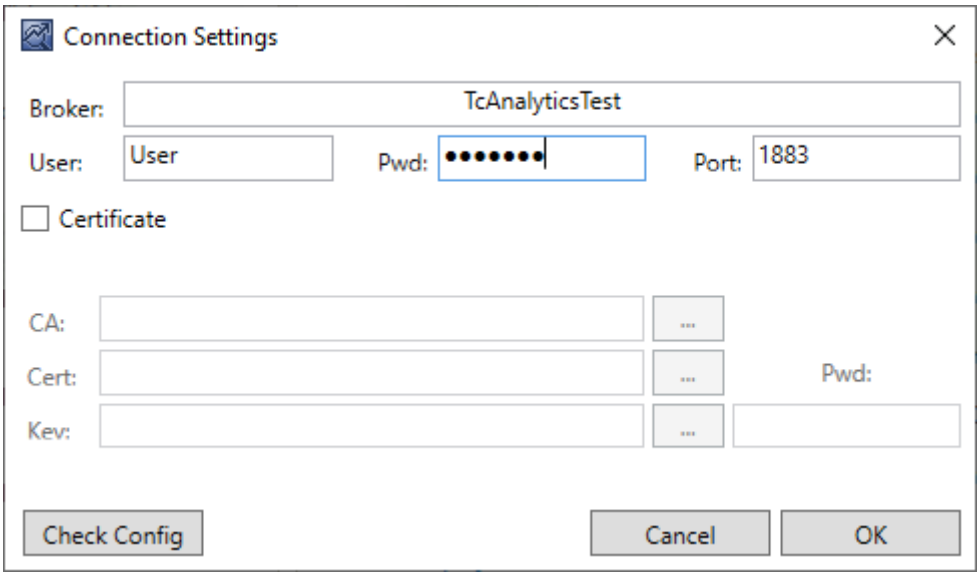
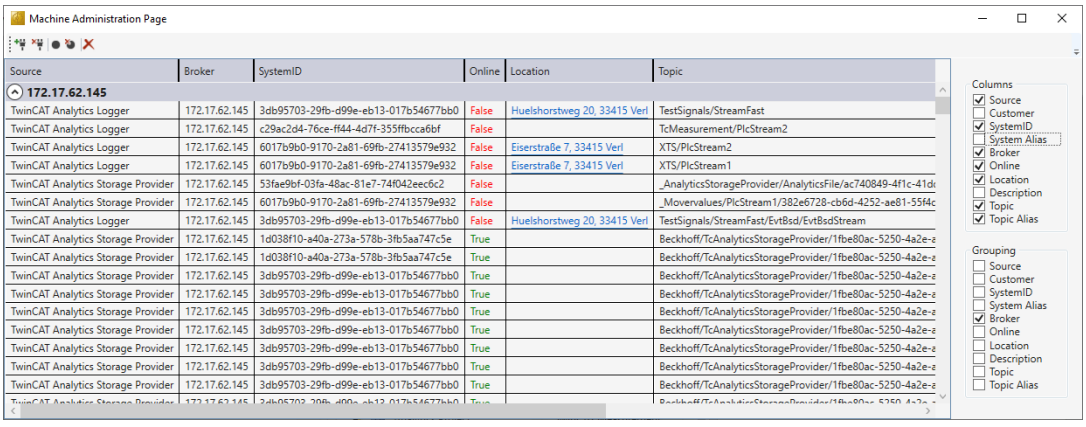
特定目标区

在 TwinCAT Target Browser (TcAnalytics) 的左侧区域，所有代理及其数据流均以树形结构显示。此外，还会显示历史数据流。前面的符号表示系统或数据流的当前状态（绿色：可用，红色：无法连接，灰色：状态未知）。



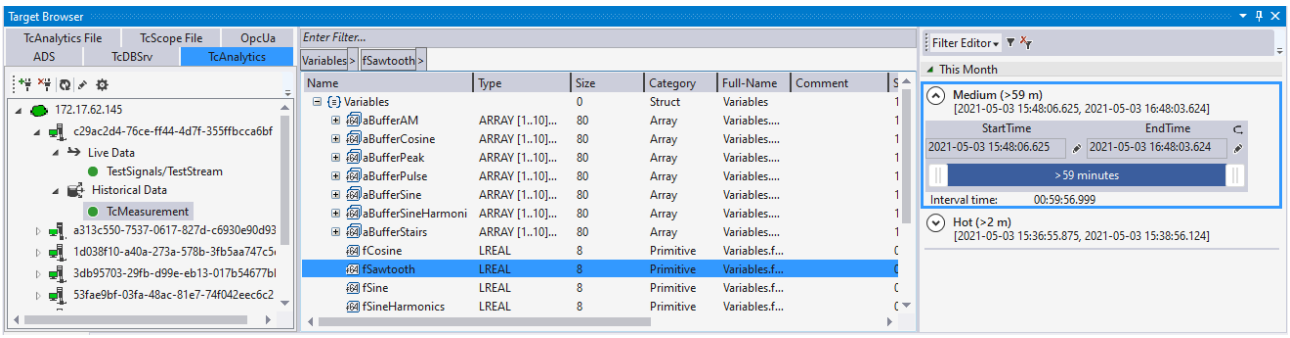
工具栏

TcAnalytics 扩展的工具栏提供以下功能：

Add broker connection（添加代理连接）	该按钮可用于添加代理连接。 
Delete broker connection（删除代理连接）	该按钮可用于删除树形结构中与代理的现有连接。
Refresh（刷新）	可通过该按钮手动更新目标系统状态的显示。
Edit connection（编辑连接）	该按钮可用于后续更改连接参数。
Machine Administration Page（机器管理页面）	该按钮可打开机器管理页面。可通过该页面管理各“机器”的输入数据流。 

通用符号区

不同数据流的符号显示在 TwinCAT Target Browser (TcAnalytics) 的右侧区域。例如，除了名称、数据类型、大小和符号名称外，还会显示属性。特殊属性，如单位的属性，会在各自的列中进行解释和输出。如果选择了历史数据流，还会显示各个记录及其时间范围。



5.1.扩展 — TcAnalytics File

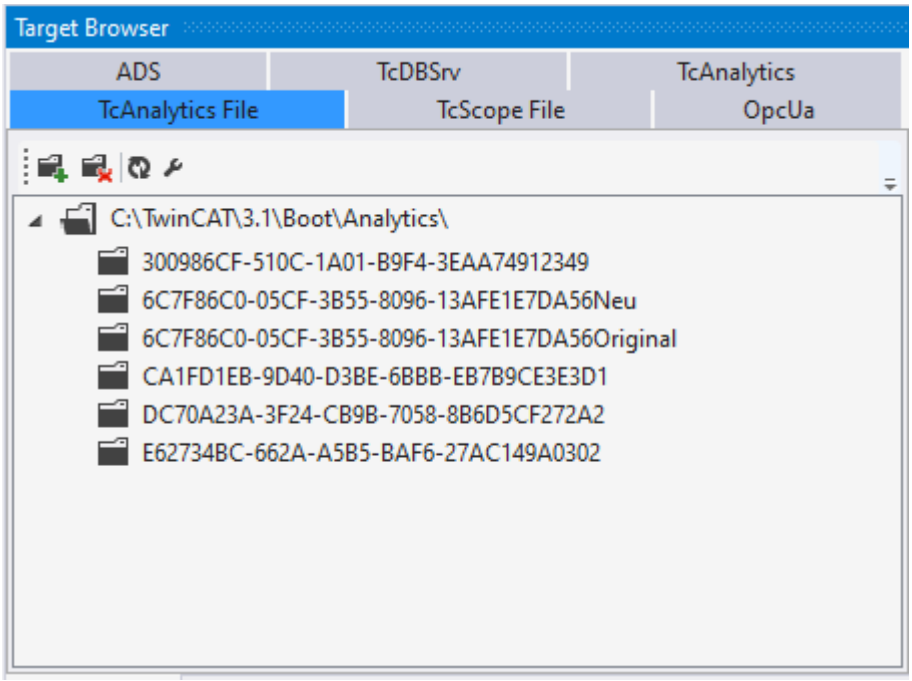
1.1.

9.4

借助 TwinCAT Target Browser 的 TcAnalytics 扩展，可以显示不同代理、不同主题的 MQTT 数据流，并且可以将其用于不同的测量产品。只需将所需的流符号拖放到相应的开发环境工具中即可。

特定目标区

TwinCAT Target Browser (TcAnalyticsFile) 的左侧区域将显示所有要搜索 AnalyticsFile 文件夹的文件夹。找到的任何 AnalyticsFile 文件夹随后均会以树形结构显示。



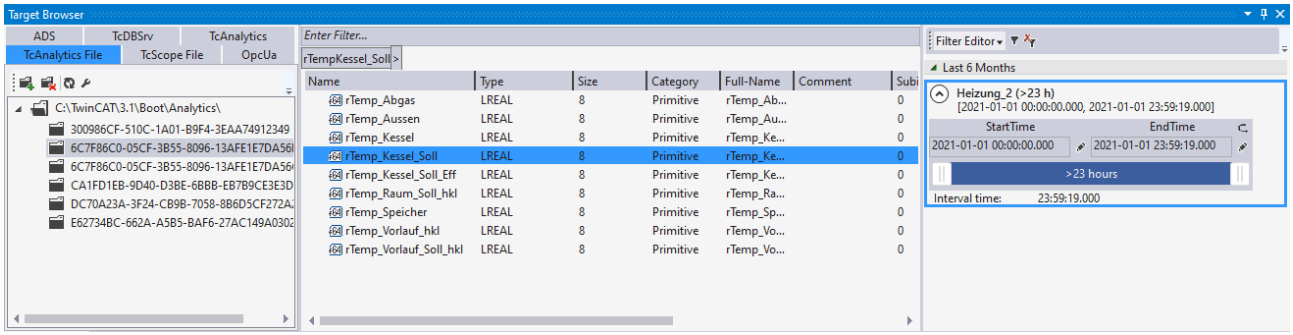
工具栏

TcAnalyticsFile 扩展的工具栏提供以下功能：

New folder（新建文件夹）	该按钮可用于添加要搜索 AnalyticsFile 文件夹的文件夹路径。
Delete folder（删除文件夹）	该按钮可用于从树形结构中删除所选文件夹。
Refresh（刷新）	该按钮可用于手动更新显示内容。
Properties（属性）	该按钮可用于自定义 TcAnalyticsFile 扩展的各种属性。

通用符号区

不同 AnalyticsFile 的符号显示在 TwinCAT Target Browser (TcAnalyticsFile) 的右侧。例如，除了名称、数据类型、大小和符号名称外，还会显示属性。特殊属性，如单位的属性，会在各自的列中进行解释和输出。此外，还会显示各个记录及其时间范围。



5.1.扩展 — TcDBSrv

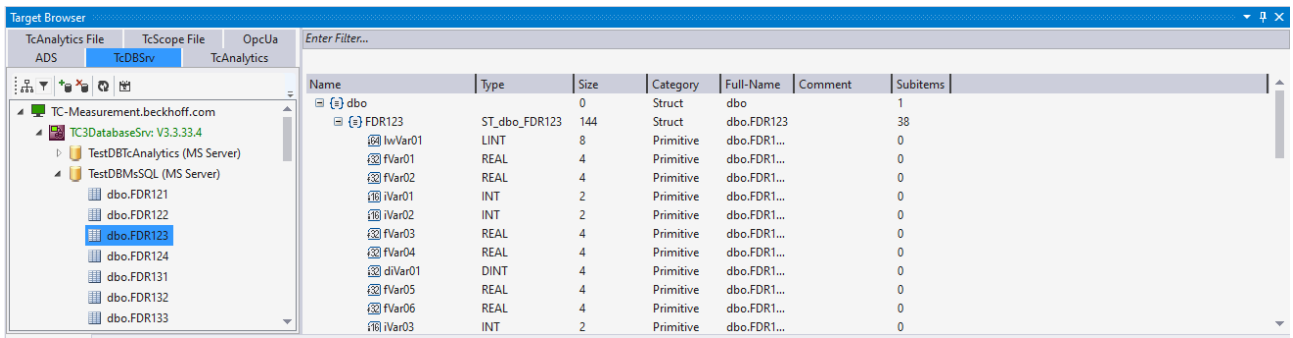
1.1.

9.5

TwinCAT Target Browser 的 TcDBSrv 扩展可用于通过 TwinCAT Database Server 显示 TwinCAT Scope 数据库中的数据集。只需将所需的表格列拖入 TwinCAT Scope 即可。相应的 SQL 命令会自动生成，当然也可以手动生成。

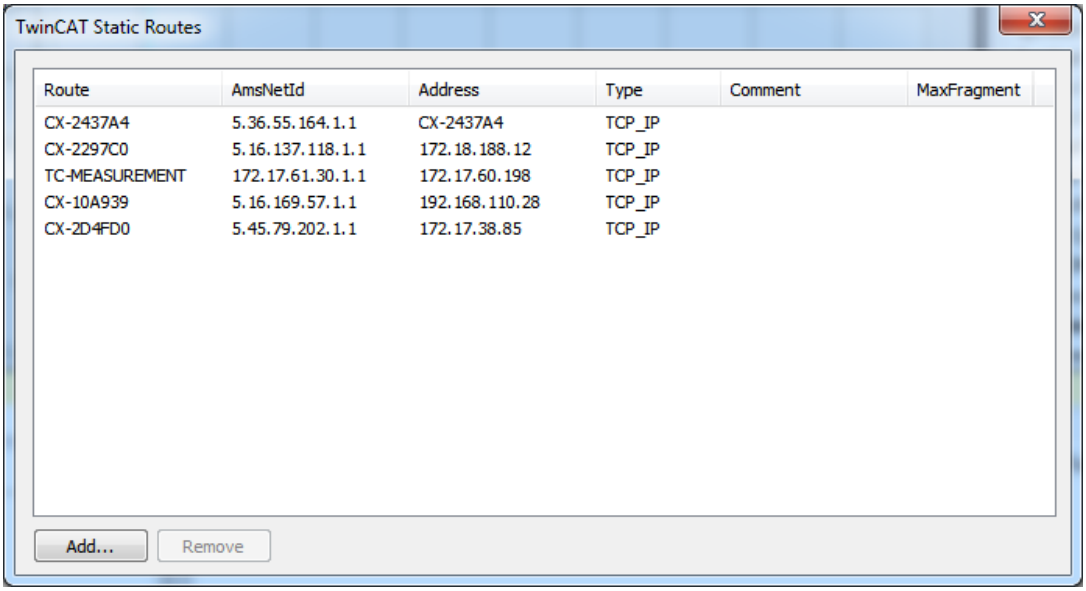
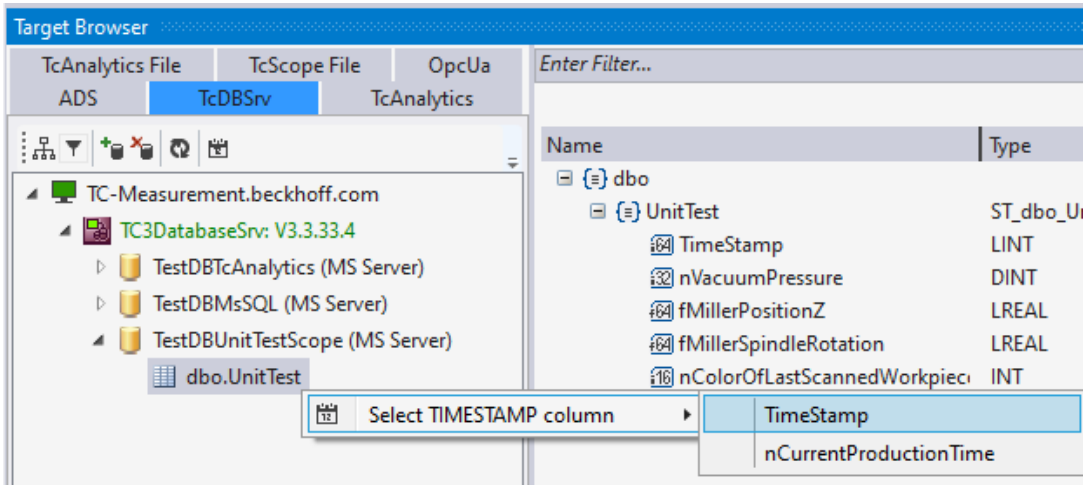
特定目标区

在本地 TwinCAT 3 开发环境中注册的所有目标系统都以树形结构显示在 TwinCAT Target Browser (TcDBSrv) 的左侧区域。首先是本地系统，然后是目标系统，如工业 PC 或嵌入式控制器，按注册顺序进行排列。带前缀的屏幕符号表示系统的状态（绿色：运行模式，蓝色：配置模式，红色：停止模式或无法连接）。可用的 TwinCAT Database Server 实例列于目标系统下方。下面是所有已配置的数据库及其可访问的表格。如果选择了表格节点，“通用符号区” 中会显示各个表格列。



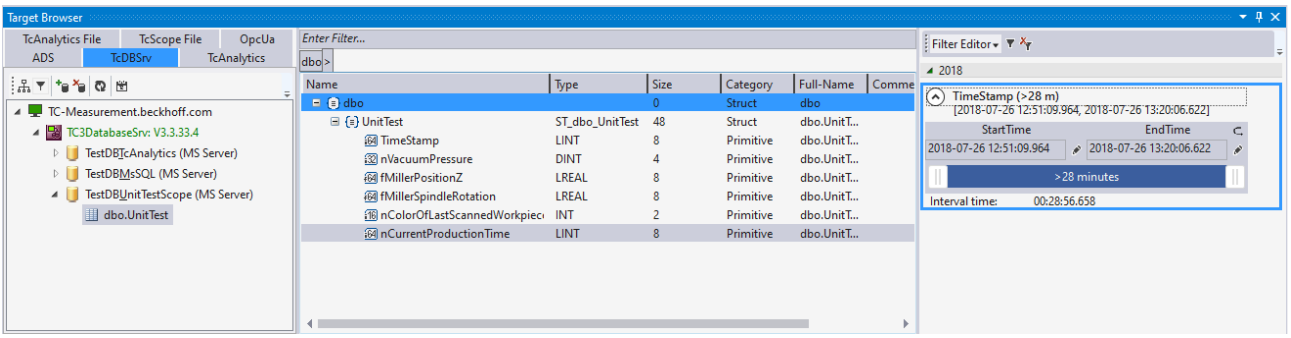
工具栏

TcDBSrv 扩展的工具栏提供以下功能：

Edit Routes（编辑路由）	如果缺少通往目标系统的 ADS 路由，可通过该按钮添加更多目标系统。 
Filter（筛选）	如果筛选器处于激活状态，仅会显示已安装并可访问 TwinCAT Database Server 的路由。
Add database（添加数据库）	该按钮可用于创建数据库配置。可以在 TwinCAT Database Server 上创建数据库池中的数据库或新配置。将打开熟悉的配置编辑器。
Delete database（删除数据库）	该按钮可用于从 TwinCAT Database Server 中删除选定的数据库配置。
Refresh（刷新）	可通过该按钮手动更新目标系统状态的显示。
Set timestamp column（设置时间戳列）	可使用该按钮或表格节点中相应的上下文菜单项指定一列作为时间戳；将根据该列读出表格的时间范围。 

通用符号区

表格及其列显示在 TwinCAT Target Browser (TcDBSrv) 的右侧区域。除名称和数据类型外，还会显示列的大小。如果将某一列定义为时间戳列，会显示数据的时间范围



5.1.1.1.10 Support Information Report

Support Information Report 是一种收集产品信息并提交给倍福技术支持部门的工具。收集产品相关数据，如 TwinCAT 版本/构建、产品版本、图像版本和设备类型等，可大大减少电子邮件流量，提高咨询效率。

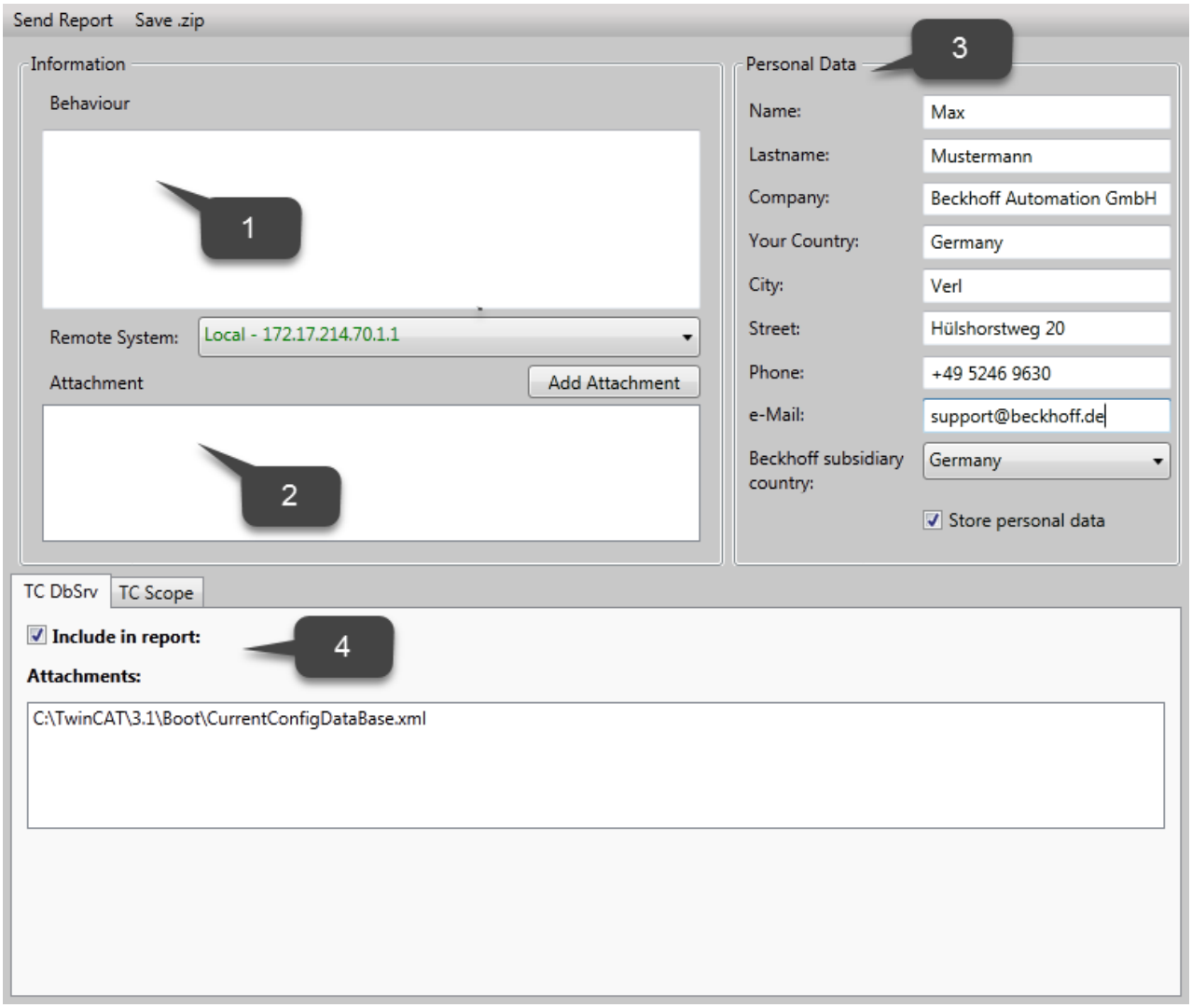
插拔式装置

各倍福产品通过插拔式装置与 Support Information Report 连接。这些产品，如 TwinCAT Database Server，在相应的产品菜单中均有 Support Information Report 菜单项。

创建并提交 Support Information Report

✓ Support Information Report 已打开。

1. 使用 “**Behaviour**” （行为）文本域尽可能详细地描述发生的行为。
2. 在 “**Attachment**” （附件）区域，如有需要，可以通过 “**Add Attachment**” （添加附件）按钮将文件（截图等）添加到报告中。可选择通过远程访问的方式选择文件。为此，请从 “**Remote System**” （远程系统）下拉列表中选择一个目标。根据所选目标，可浏览 Windows CE 设备。
3. 输入您的联系方式，并选择您所在国家/地区的倍福子公司。
   在提交 Support Information Report 时必须提供此信息。
4. 可以选择存储您的联系方式，以备将来提交 Support Information Report 时使用。为此，请勾选 “**Store personal data**” （存储个人数据）复选框。
5. 可在 Support Information Report 的下部区域找到产品专用的插件。勾选 “**Include in report**” （包含在报告中）复选框。将会自动添加产品所需的信息（若有）。例如，该截图以 XML 文件的形式显示了 TwinCAT Database Server 的当前配置。
6. 提交 Support Information Report：
   - 如果设备具有电子邮件连接功能，可以通过 “**Send Report**” （发送报告）按钮直接向您所在国家/地区的倍福子公司提交 Support Information Report。
   - 如果设备没有电子邮件连接功能，可以通过 “**Save .zip**” （保存压缩文件）按钮以 .zip 文件的形式将 Support Information Report 保存在本地，然后通过 FTP、USB 等方式进行提交。



5.1.1.2 配置模式

本章汇集了使用 TwinCAT 数据库服务器的配置模式所需的所有内容。它包含以下主题：

- 创建项目
- 创建和设置数据库配置
- 创建和设置 AutoLog 组
- 激活数据库服务器项目
- 监控和控制自动日志记录

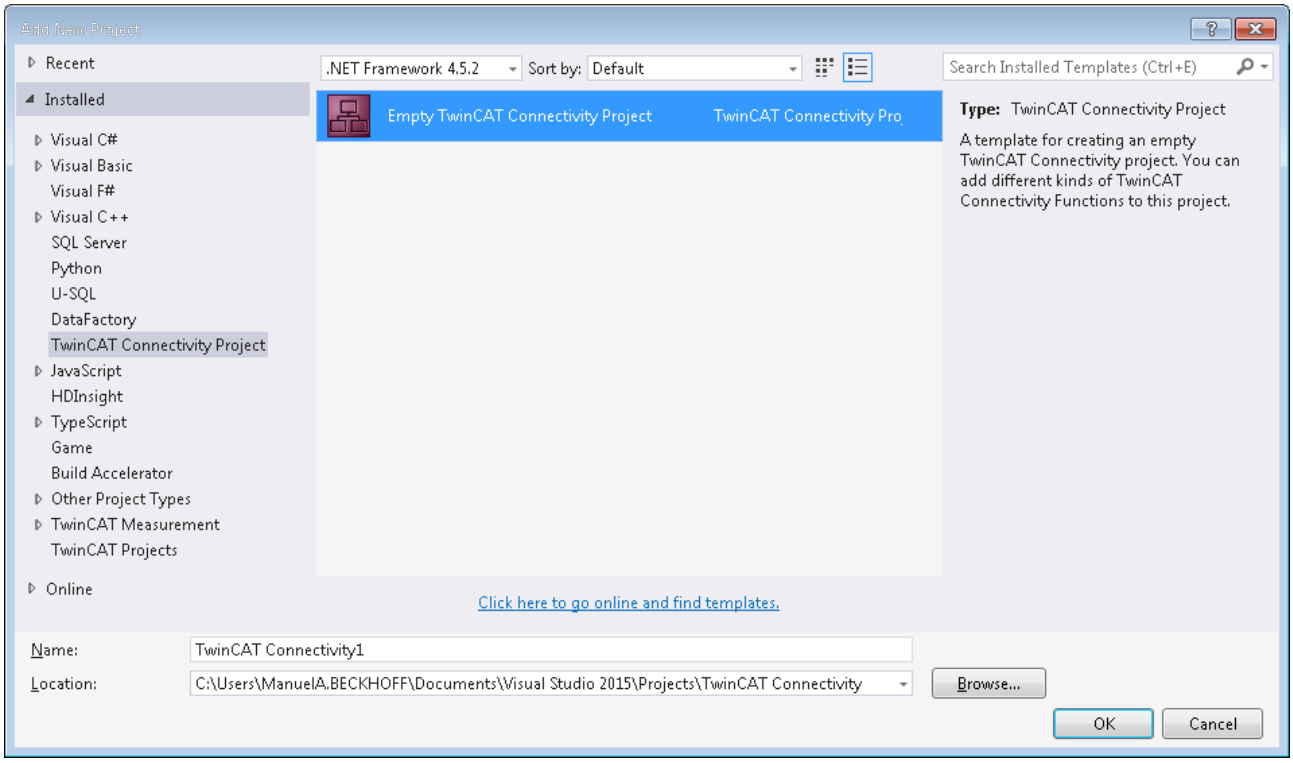
配置模式

在配置模式下，大部分工作在配置器中完成。务必为所需的数据库和 AutoLog 组设置配置。Target Browser 可用于配置 AutoLog 组、在线访问目标系统以及选择要传达的变量。如果使用 **AutoStart**（自动启动）选项，则在 TwinCAT 系统启动时会直接建立与配置的数据库的通信。如果选择 **Manual**（手动）选项，则必须通过功能块 FB_PLCDBAutoLog [▶ 156] 或 AutoLog 视图启用通信。

构建项目

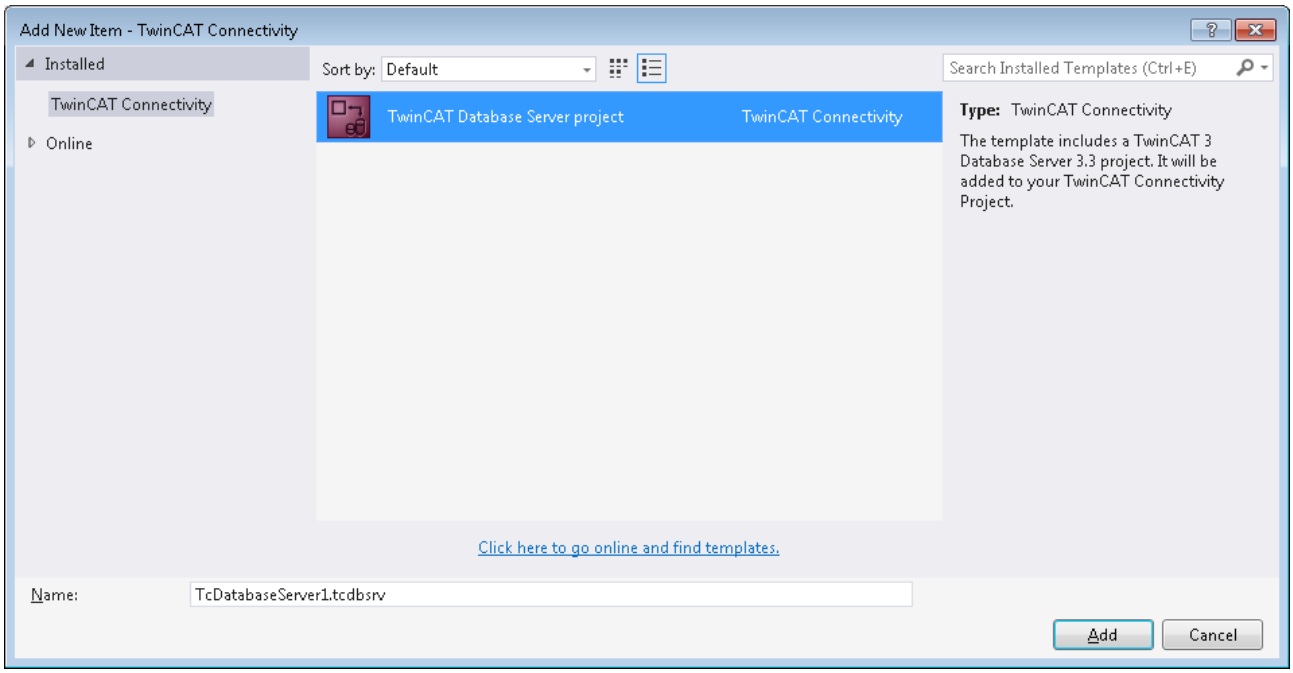
Visual Studio 的 TwinCAT Connectivity 扩展提供了一个新的项目模板。在创建新项目时，**TwinCAT Connectivity 项目**类别将作为一个选项出现。

要创建一个新的 TwinCAT Connectivity 项目，请选择 **Empty TwinCAT Connectivity Project**（空的 TwinCAT Connectivity 项目），指定项目名称和存储位置，然后点击 **OK**（确定）将其添加到解决方案中。这样可以方便地与 TwinCAT 或其他 Visual Studio 项目并行创建 TwinCAT Connectivity 项目或 TwinCAT 数据库服务器项目。

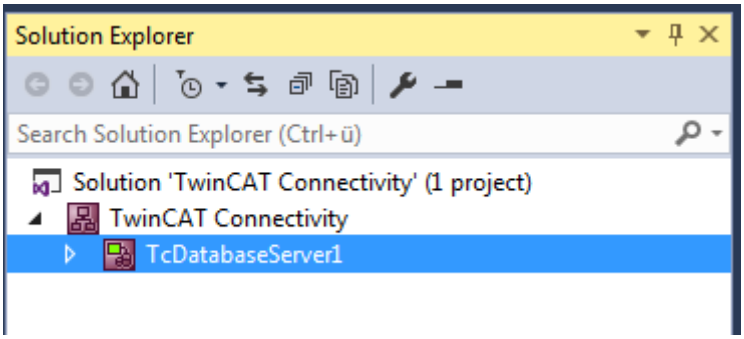


在解决方案中会出现一个新的项目节点。在 Connectivity 项目节点下方，您可以为支持的连接功能添加子项目。

使用 **Add**（添加）将一个新的 TwinCAT 数据库服务器项目添加到 TwinCAT Connectivity 项目中。在现有项目模板列表中可以找到 TwinCAT 数据库服务器项目。



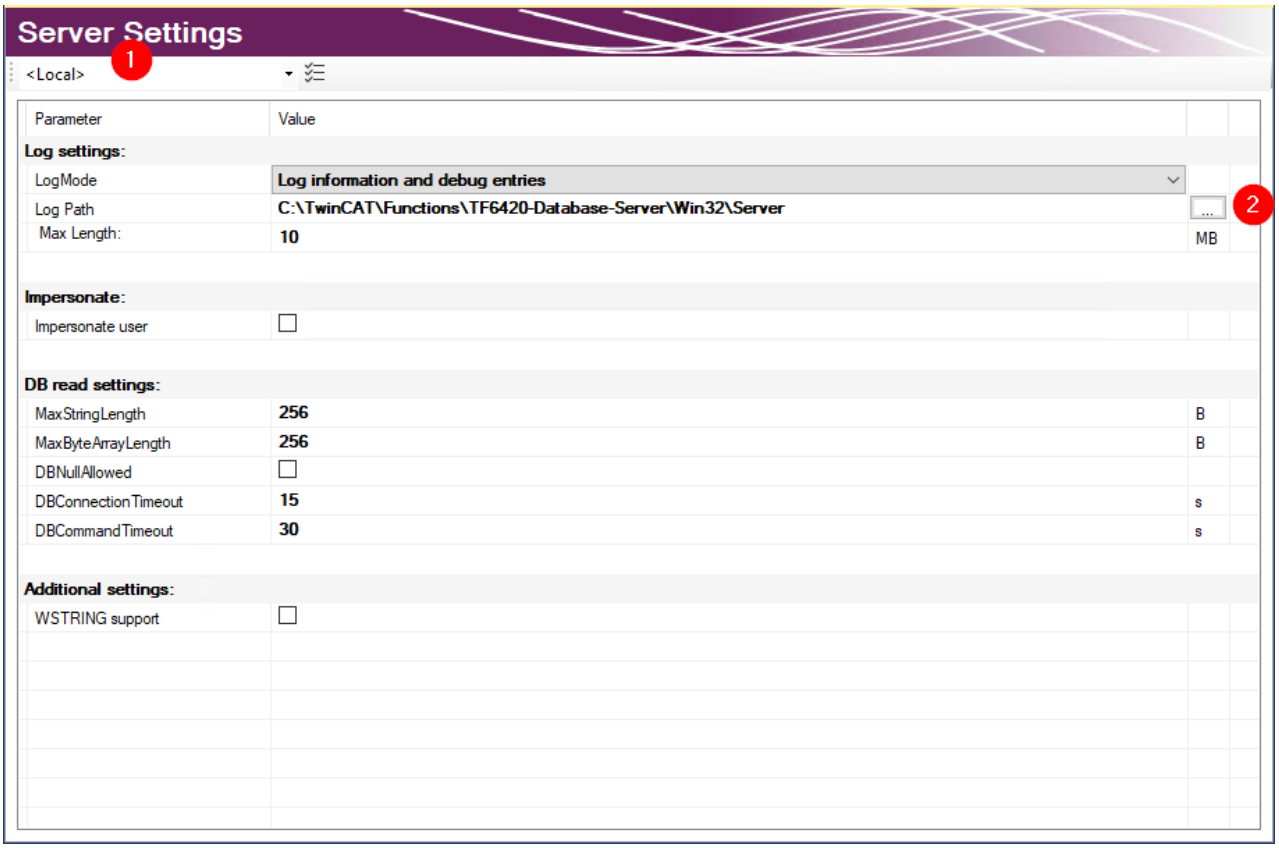
在 TwinCAT Connectivity 节点下，创建一个新的 TwinCAT 数据库服务器项目。



现在可以将其用作 TwinCAT 数据库服务器待配置的基础。通过属性窗口或编辑器可以编辑文档。

一个 Connectivity 项目可以与任意数量的 TwinCAT 数据库服务器项目或其他项目相关联，因此它可能包含多项配置。

服务器设置的编辑器



服务器设置编辑器可以用于编辑 TwinCAT 数据库服务器的设置。这些是与相应的服务器相关的常规设置。在下拉菜单（1）中，您可以通过 Ams NetID 选择目标系统。为此，您必须通过 TwinCAT 创建一条通往目标系统的路由。在传输已完成的配置时，设置将存储在该目标系统的 TwinCAT 数据库服务器中。

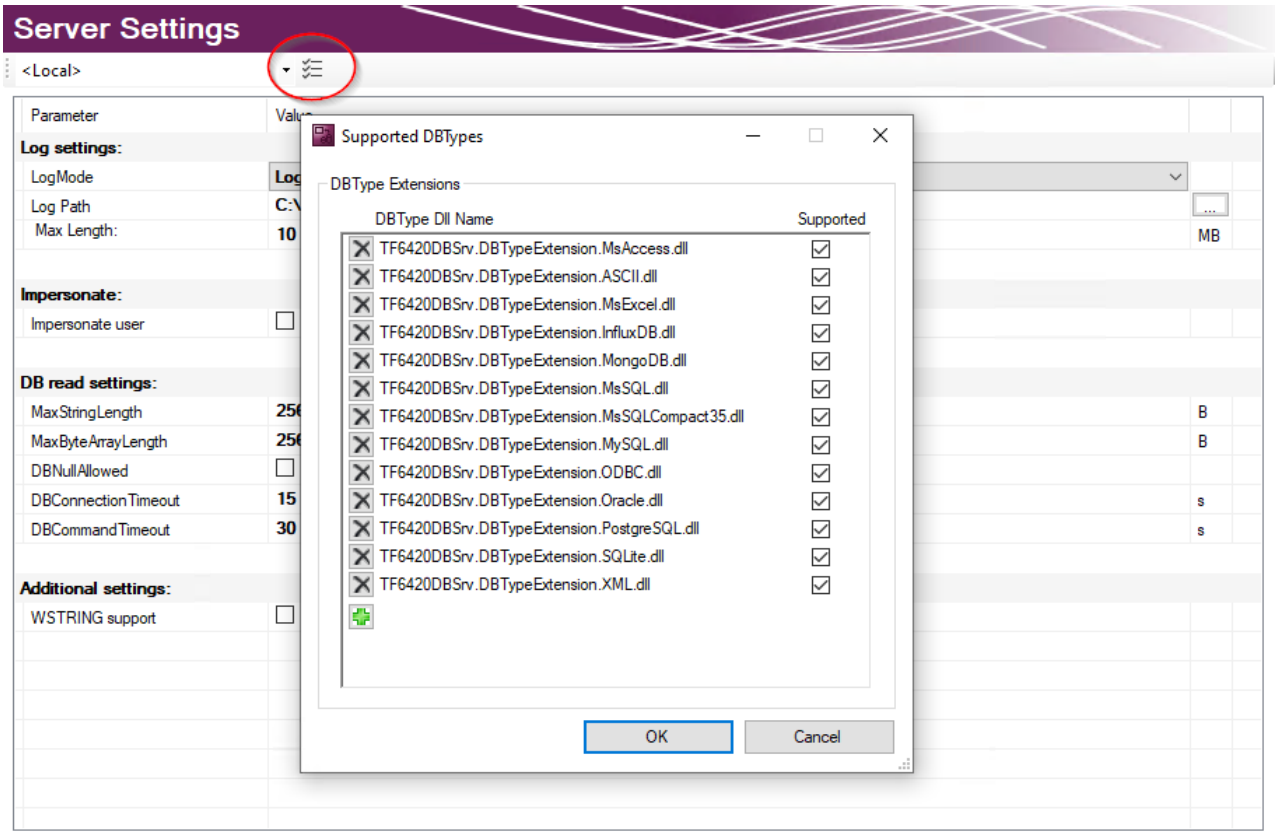
在 **Log settings**（日志设置）下可以配置关于记录故障或错误的设置。在发生故障或错误时，数据库服务器会在日志文件中生成一个详细的条目。通过信息日志查看器 [▶ 48]可以读取日志文件。在 **Log Settings**（日志设置）下，您可以指定文件位置的路径和最大文件大小。您还可以影响日志的准确性。出于性能方面的考虑，我们建议在不再需要的情况下，在错误分析之后再次停用日志记录。

对于基于文件的数据库（例如 Access 或 SQL Compact）的网络访问，必须设置 **Impersonate**（模拟）选项，以便 TwinCAT 数据库服务器可以与该网络驱动器连接。**Windows CE 目前不支持该功能。**

可以使用进一步的配置设置来控制从数据库读取数据的过程。这些设置指的是目标系统上的 TwinCAT 数据库服务器：

MaxStringLength	PLC 中的变量的最大字符串长度
MaxByteArrayLength	PLC 中的变量的最大字节数组长度
DBNullAllowed	表示 TwinCAT 数据库服务器是否接受零值。
DBConnectionTimeout	表示在尝试建立连接时，TwinCAT 数据库服务器出现连接错误的时间。
DBCommandTimeout	表示在发送命令时，TwinCAT 数据库服务器出现连接故障的时间。如果涉及大量数据，则处理一条命令可能需要相当长的时间，具体取决于数据库和基础设施。

支持的数据库类型

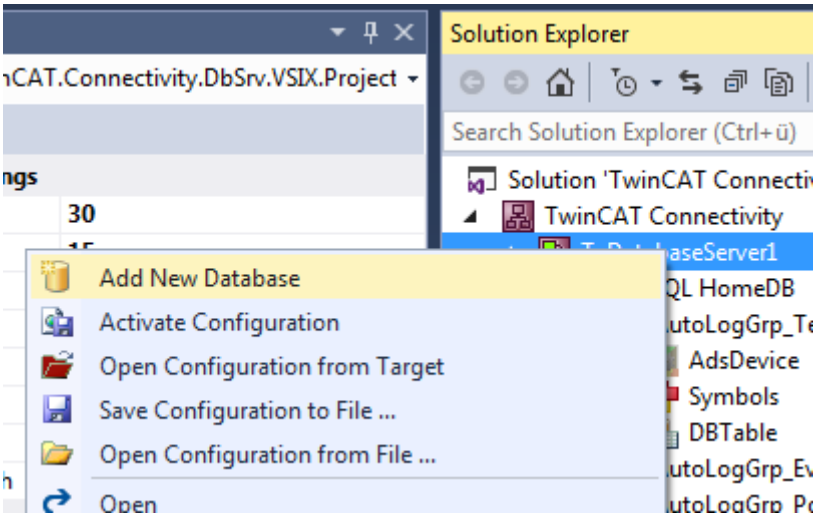


在服务器设置中可以选择已安装的数据库类型。默认情况下会选择所有已安装的数据库。TwinCAT 3 数据库服务器将加载相应的数据库接口。这样可以取消选择目标系统上未使用的数据库。

添加新的数据库配置

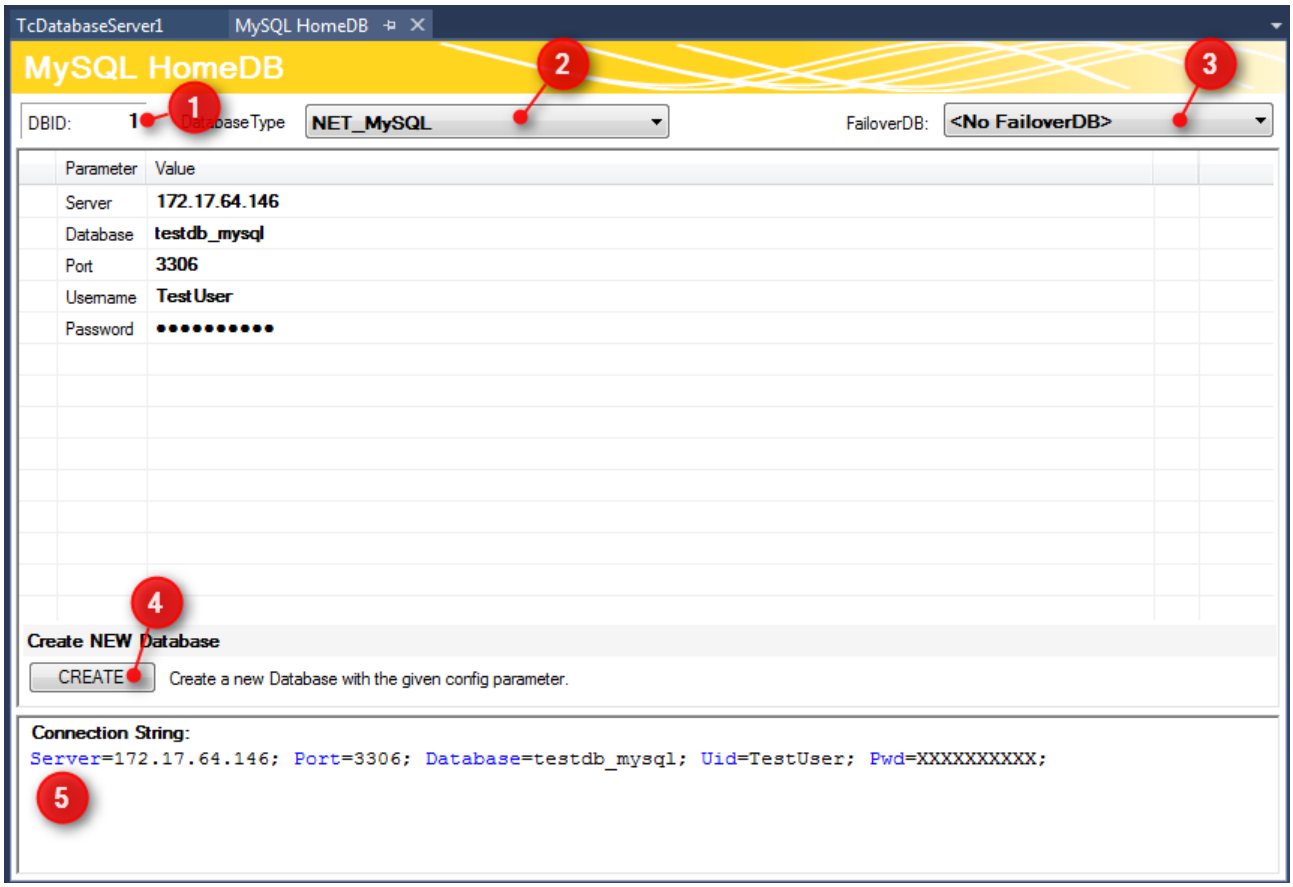
要向数据库服务器提供数据库连接所需的所有信息，必须进行数据库配置。

通过数据库服务器项目的上下文菜单中的 **Add New Database**（添加新数据库）命令或工具栏中的相应命令可以添加新的数据库配置。



新的数据库配置以文件的形式被添加到项目文件夹中，并集成到项目中。与所有 Visual Studio 项目一样，新文件的信息存储在 Connectivity 项目中。

数据库配置编辑器

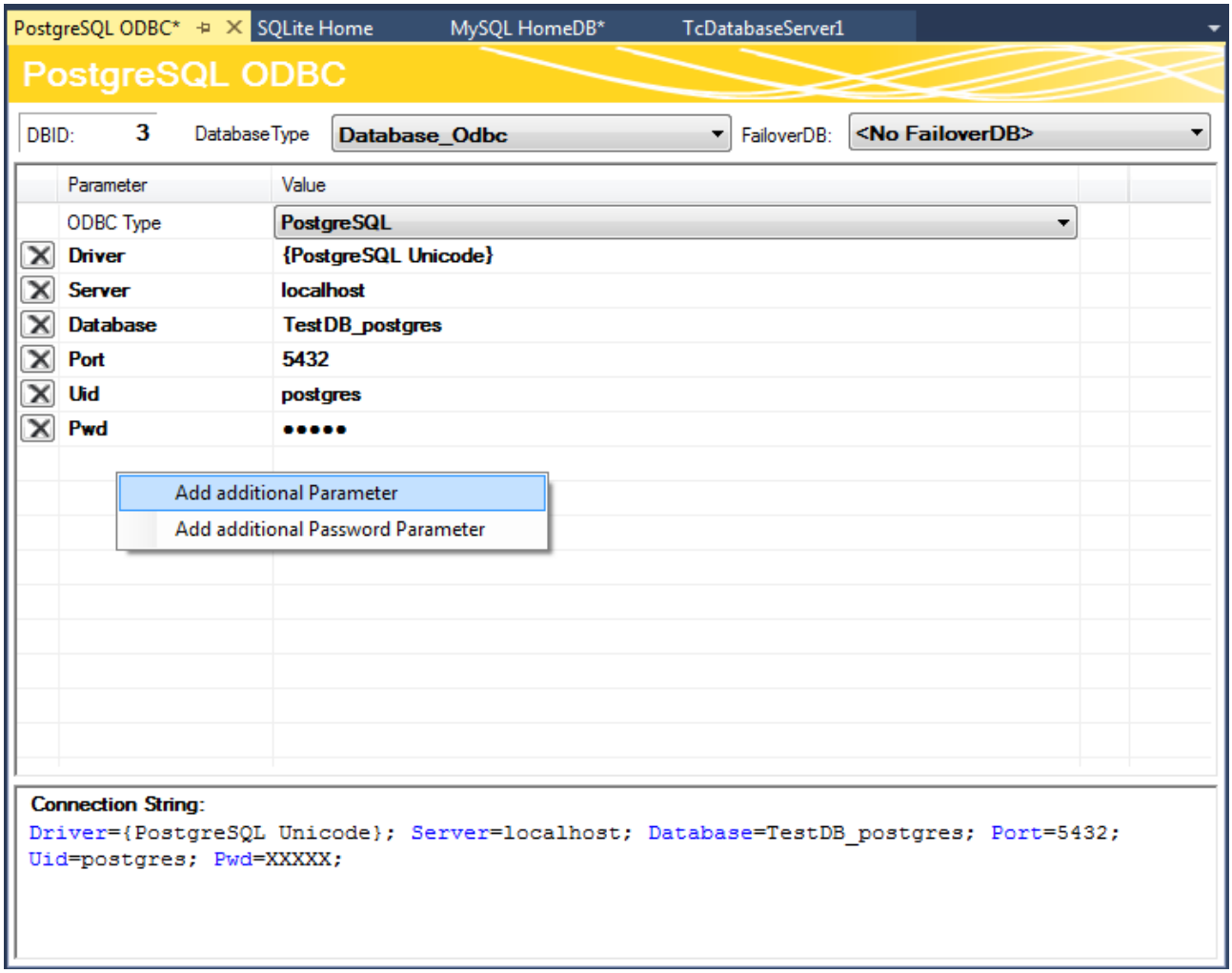


PLC 中的一些功能块所需的数据库 ID 会在编辑器的上部（1）显示。从下拉菜单（2）中可以选择目标数据库的数据库类型。另一个选项是数据库的 ODBC 接口，不过目前还不支持。请注意，根据数据库的不同，并非 TwinCAT 数据库服务器的所有功能都能得到保证。

您还可以选择所谓的故障转移数据库（3），在配置模式下遇到错误时就会触发故障转移数据库。在网络断开的情况下，该功能可以自动确保数据存储在其他地方，而不会丢失。

对于每个数据库 [▶ 120]，都有附加的可调整参数。根据数据库会创建一个连接字符串（5），用于描述与数据库的连接。这样旨在使您设置的参数更加透明。

CREATE（创建）（4）按钮可以用于创建新的数据库。只有在相关数据库支持该功能的情况下，才会显示它。



通过 ODBC 接口可以配置未知数据库。在 **ODBC Type**（ODBC 类型）下拉列表中，选择“Unknown Database”（未知数据库），然后通过上下文菜单中的命令添加参数。它们可能包含以加密形式存储的密码。利用这些参数可以组合成所需的连接字符串。请注意，只能使用 TwinCAT 数据库服务器的有限功能。仅支持 SQL 专家模式的显式功能块。

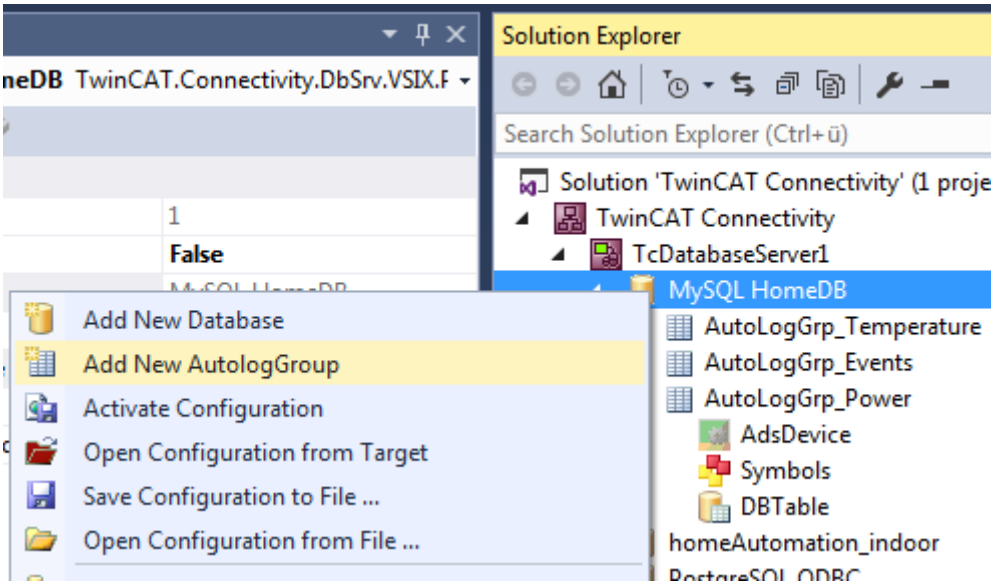
故障转移数据库

TwinCAT 3 数据库服务器具有故障转移数据库功能。该功能提供了一个选项，可在设置的数据库出现连接中断或其他问题时切换到另一个指定数据库，以避免可能发生的数据丢失。仅限配置模式使用该功能。在自动写入的情况下，如果出现错误，则使用相应的替代数据库。第一个数据库的表格格式必须与第二个匹配。

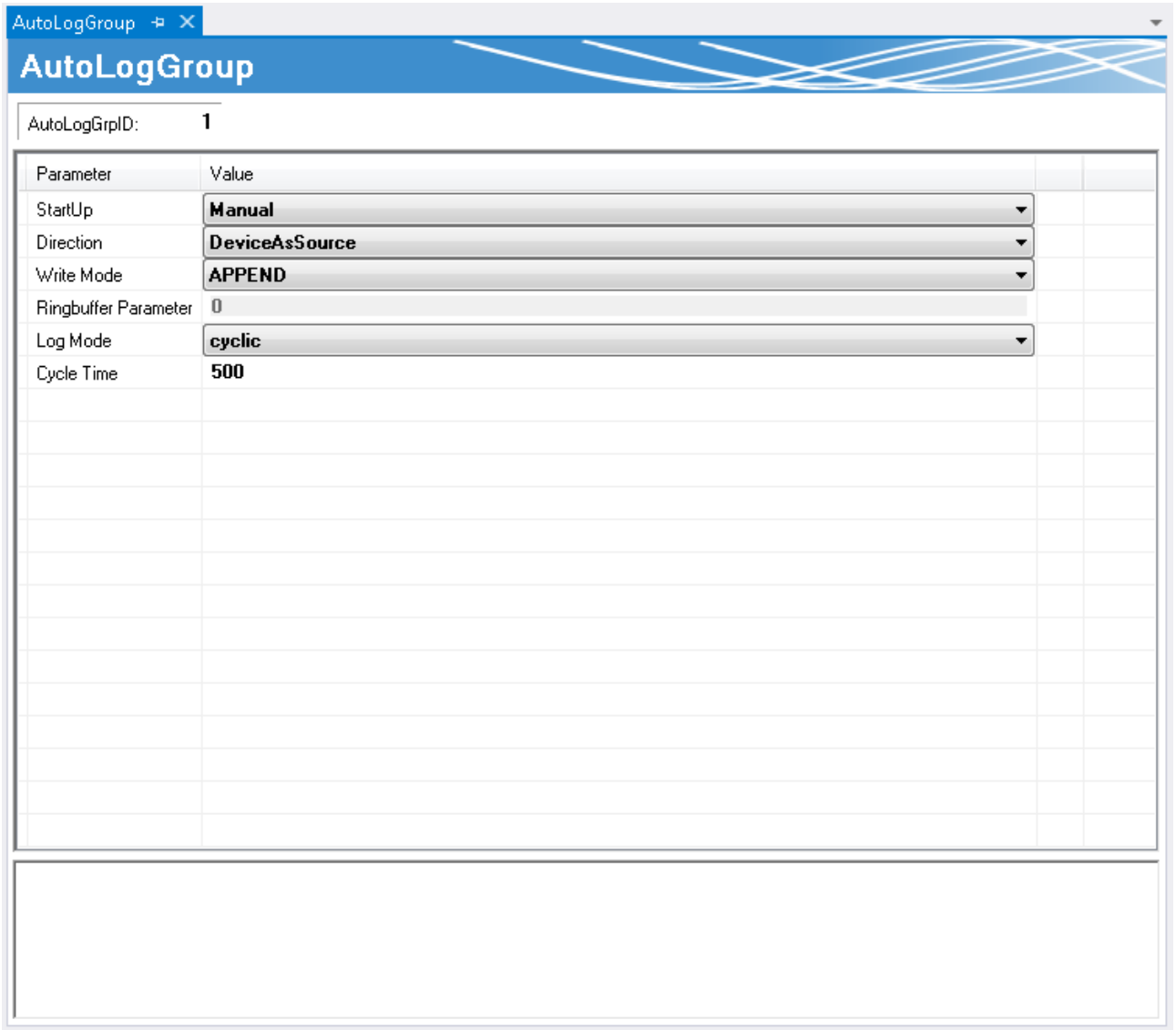
添加新的 AutoLog 组

AutoLog 组包含有关 PLC 的哪些变量要与数据库中的哪些变量同步的信息。此外，这里还会存储有关同步时间和同步类型的信息。

通过数据库配置的上下文菜单中的 **Add New AutologGroup**（添加新的 Autolog 组）命令或工具栏可以为数据库配置添加新的 AutoLog 组。这些 AutoLog 组指的是父级数据库。



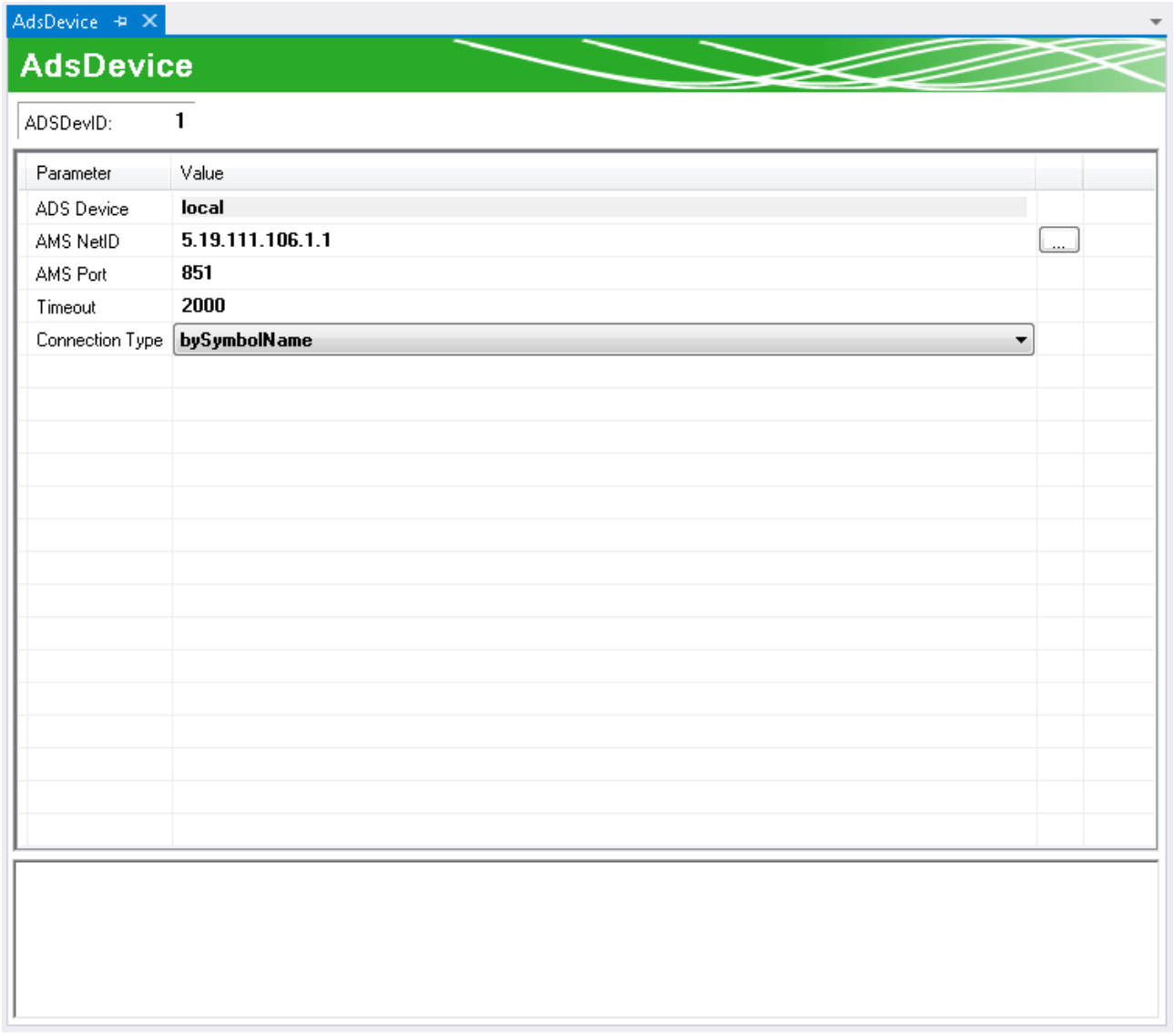
新的 AutoLog 组以及相应的组件作为文件被添加到项目文件夹中，并集成到项目中。它们包括 ADS 设备、符号组和表格设置。为了在项目中保存这些文件，您应保存 TwinCAT Connectivity 项目文件。然后，在编辑器或属性窗口中可以编辑这些文件。



启动	您可以手动启用 AutoLog 模式（通过 PLC 或配置器中的命令），或者在系统启动时自动启用该模式。
方向	设定的 ADS 设备可用作数据目标或数据源。
写入模式	这些数据可以逐行添加到数据库中，也可以按时间或数量保存在环形缓冲区中，或者只是在相应的位置进行更新。
环形缓冲区参数	根据设置的不同，该参数可表示环形缓冲区更新的时间或周期。
日志模式	在经过特定的循环时间后或在发生变化时，就会写入变量。
周期时间	写入变量后的周期时间。

配置 ADS 设备

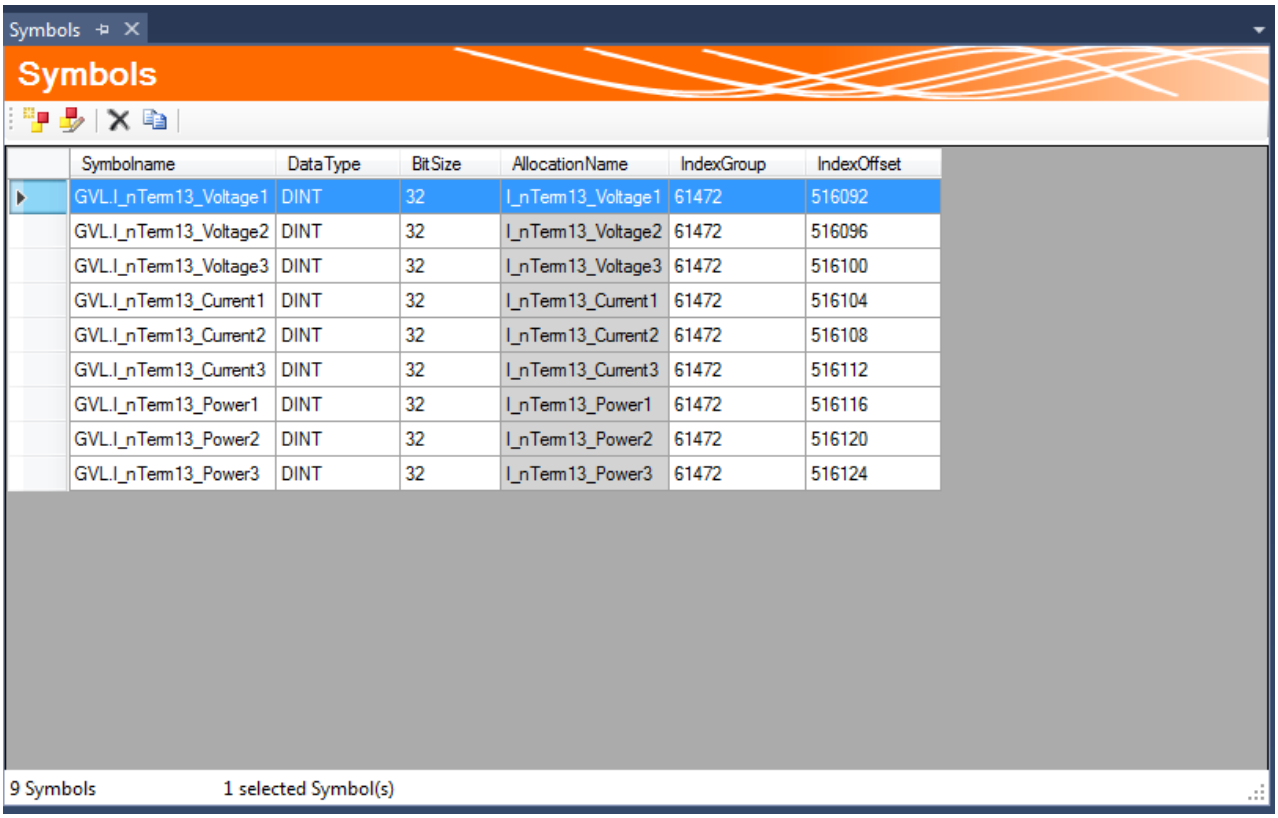
在 AutoLog 组下会自动创建 ADS 设备。在最常见的用例中，ADS 设备为 PLC Runtime。在编辑器中可以设置以下参数：



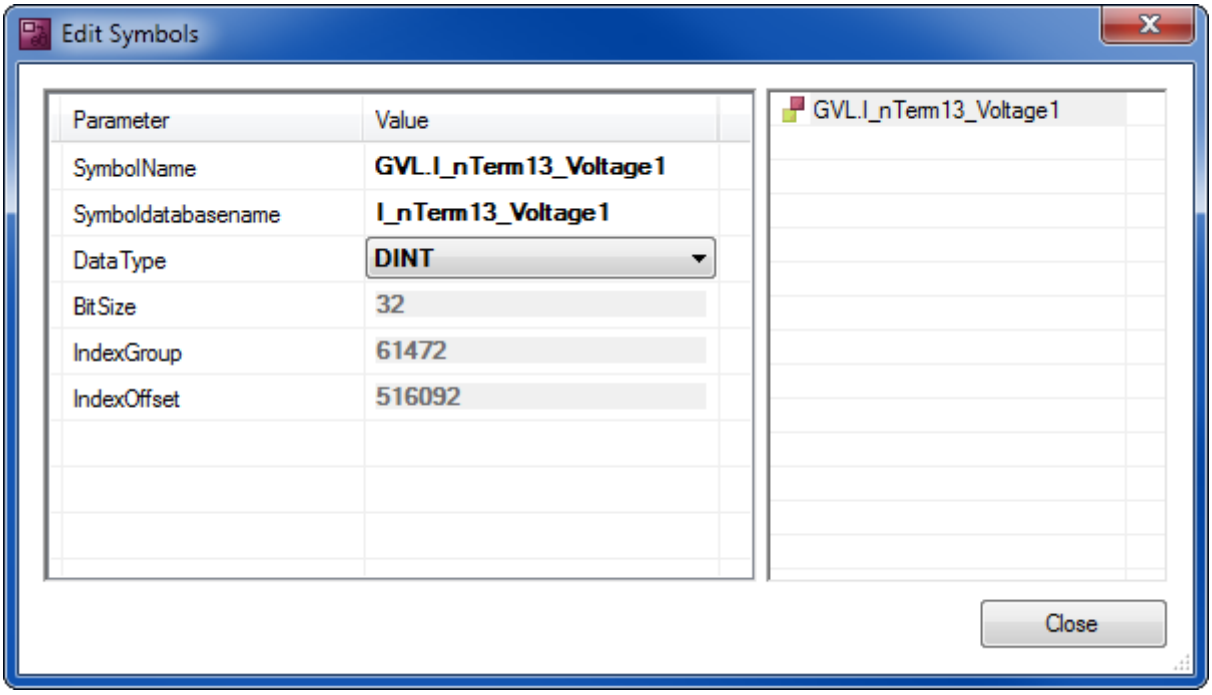
ADS 设备	ADS 目标设备的名称。
AMS NetID	TwinCAT 网络中的目标设备的地址。
AMS 端口	TwinCAT 网络中的目标设备的端口。
超时	与目标设备失去连接的时间。
连接类型	bySymbolName：根据符号名称建立连接。 byIndexGroup：根据内存索引建立连接。

配置符号

您在这里设置的符号会被写入数据库或从数据库读取，具体取决于 ADS 设备是数据目标还是数据源。TwinCAT Target Browser [▶ 49]可以用于便捷访问。在这里，您可以搜索目标系统上的符号，并通过拖放操作在 2 个工具之间进行通信。



您还可以将符号手动添加到符号组或进行编辑。所需的信息各不相同，具体取决于在 ADS 设备中是通过符号名称还是通过索引组来选择连接类型。起始点始终是 ADS 设备。

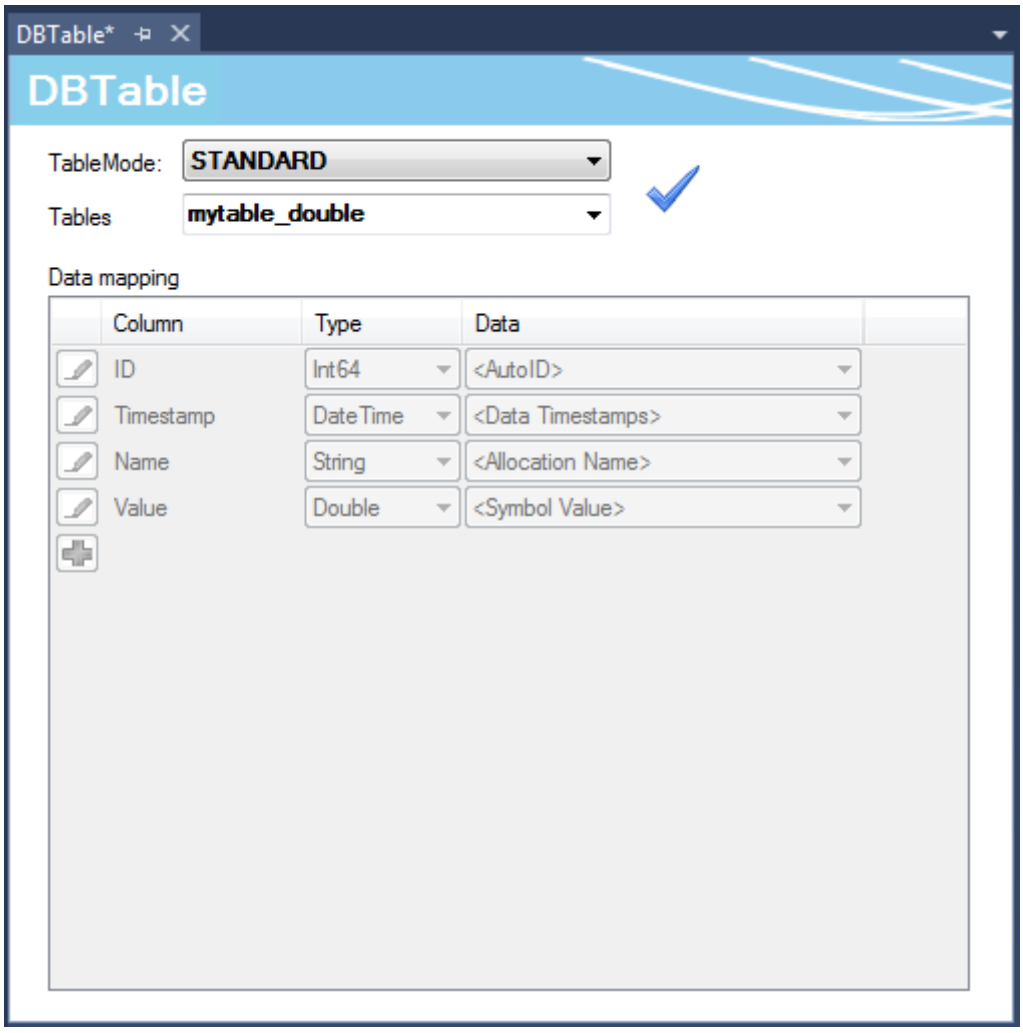


SymbolName	根据设置的 ADS 设备，对符号进行寻址
符号数据库名称	数据库表中的变量的名称
DataType	符号的 PLC 数据类型
BitSize	符号的位大小（为数据类型自动设置）
IndexGroup	TwinCAT 系统中的索引组
IndexOffset	TwinCAT 系统中的索引偏移

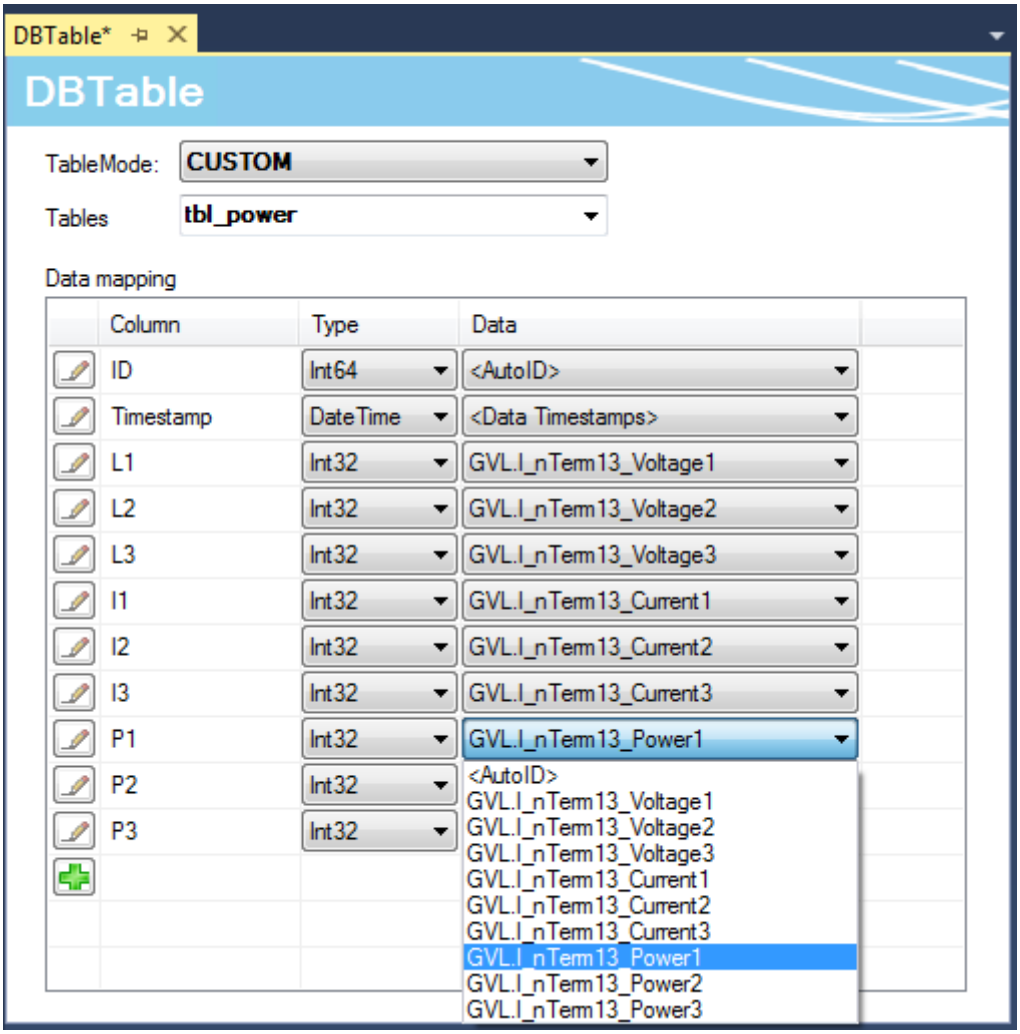
配置表格

数据库中的表格可以基于标准表结构，也可以基于个性化结构。

从可能的表格列表中可以选择相应的表格。如果表格尚不存在，您可以通过 SQL 查询编辑器创建它。如果您选择标准表结构，则蓝色勾号表示选定的表格是否与该结构相对应。

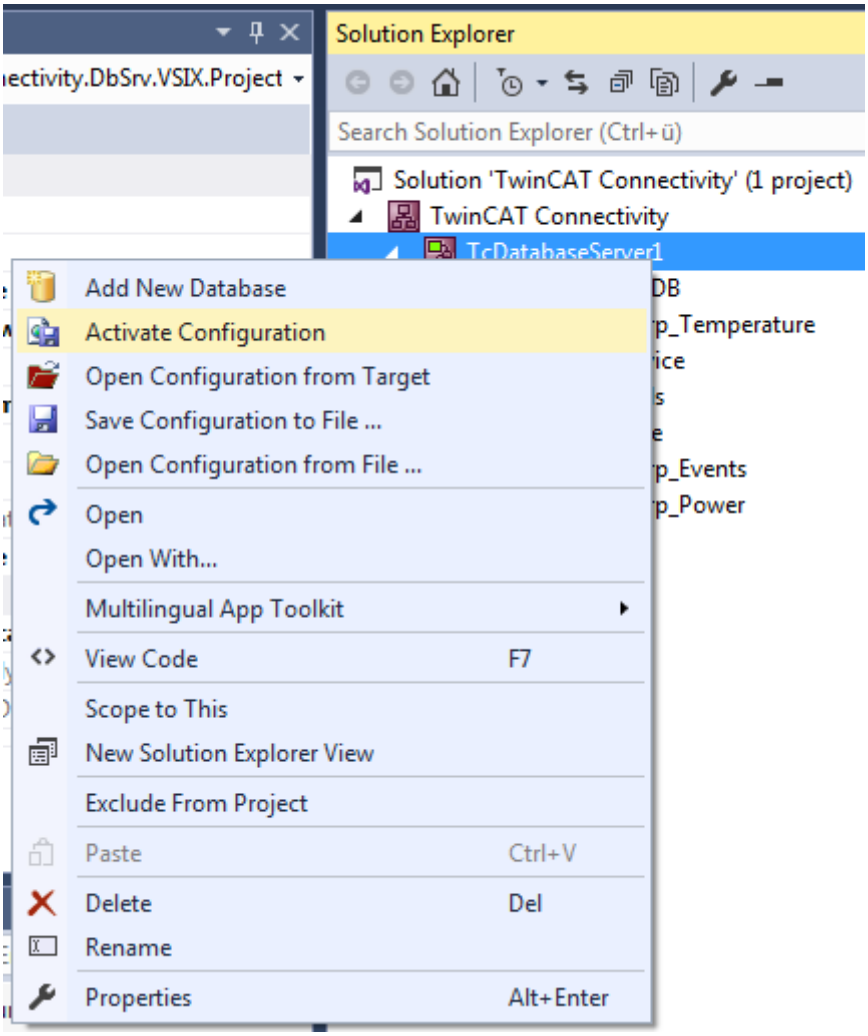


特定的表格类型提供了根据需要将在符号组中设置的各个符号分配到数据库中的表格列的选项。在 AutoLog 模式下将数据集写入数据库时，采样时符号组的当前值将保存在相应的表格列中。



激活项目

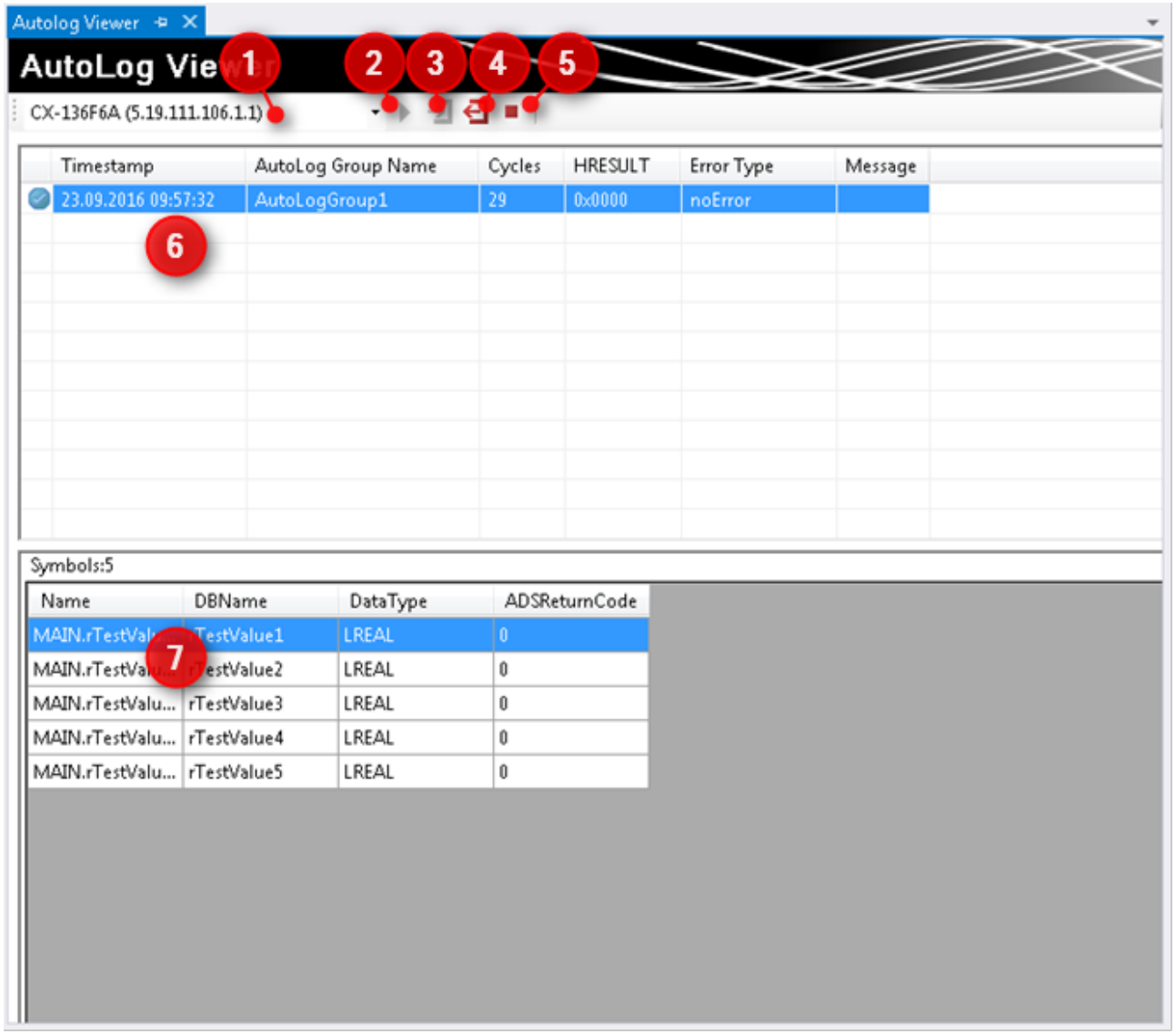
要在 TwinCAT 数据库服务器上激活一个已配置的项目，可使用 TwinCAT 数据库服务器项目的上下文菜单中的 **Activate Configuration**（激活配置）命令。



在 TwinCAT 系统启动时，变量的日志记录就会开始，具体取决于在 AutoLog 组中指定的启动行为。通过以下 AutoLog Viewer或 PLC 中的相应功能块可以手动启动该模式。

AutoLog Viewer

TwinCAT 数据库服务器的 AutoLog 查看器是一种用于控制和监控 AutoLog 模式的工具。您可以登录目标系统，这类似于 TwinCAT PLC。在登录状态下，可以启动或停止 AutoLog 模式。有关日志记录的当前状态的信息在窗口的下部显示。在选择 AutoLog 组时，通过记录的符号可显示更多信息。



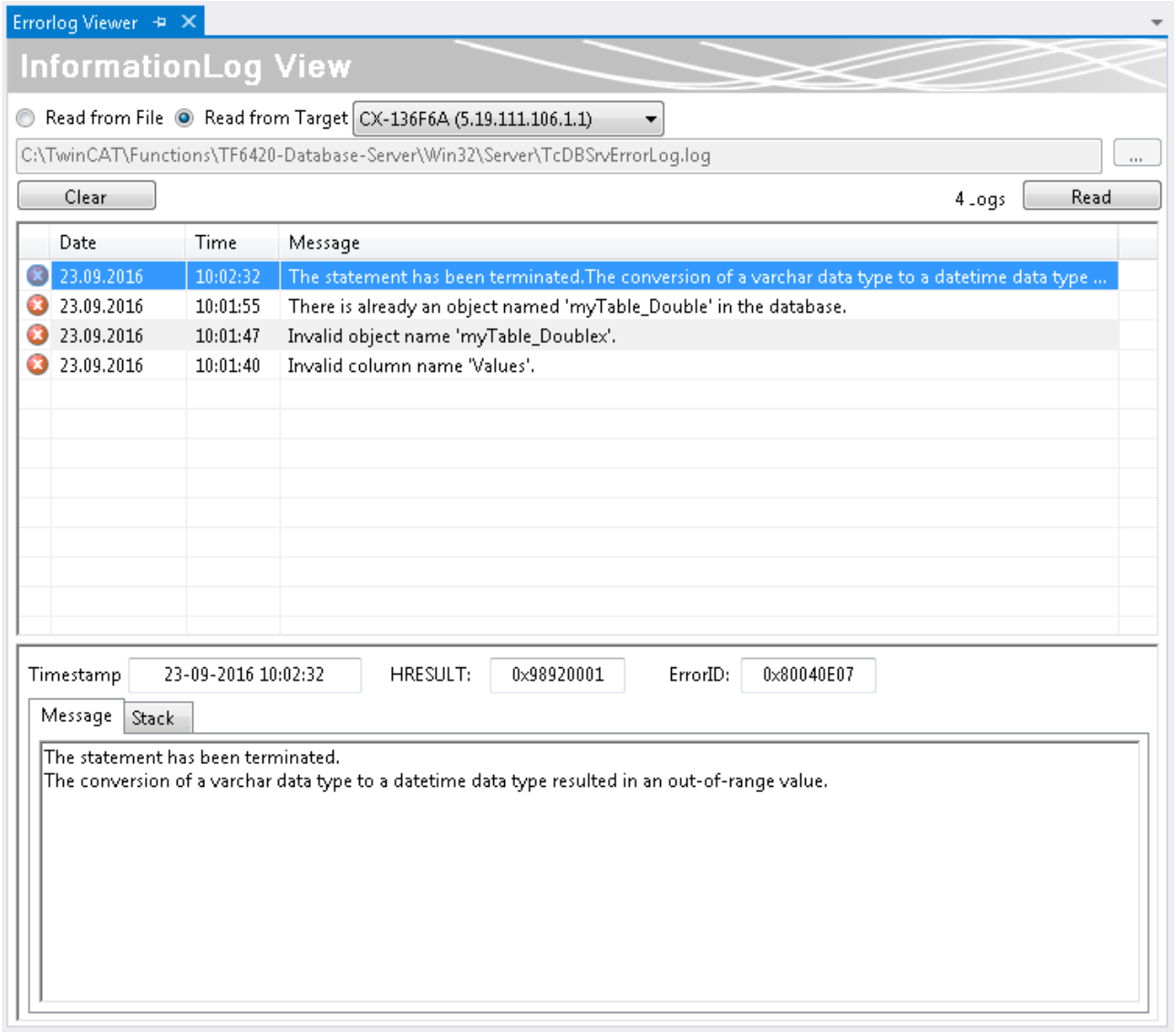
ID	Name	功能
1	目标系统	选择已安装 TwinCAT 数据库服务器的目标系统
2	启动	手动启动 AutoLog 模式
3	登录	登陆进入活动状态的 AutoLog 进程
4	退出	退出活动状态的 AutoLog 进程
5	停止	手动停止 AutoLog 模式
6	AutoLog 组	目标系统上已配置的 AutoLog 组的列表
7	符号	选定 AutoLog 组的已配置符号列表

AutoLog Viewer可以用于启动和监控已配置的应用程序。根据设置的不同，在登录并启动后，AutoLog 组的递增循环计数器会根据更新时间显示。更新错误也会在这里显示。如需更细节的处理，我们建议使用 InformationLog View。

使用 InformationLog View进行更细节的错误处理

InformationLog View是一种从 TwinCAT 数据库服务器读取日志文件的工具。记录的信息会以纯文本的形式显示，包含时间戳、ID 和错误消息。

不仅可以直接通过文件访问功能来查看或清空日志文件，还可以直接通过目标系统实现这些操作。这在同一网络下有分布式数据库服务器时特别有优势，可以借此快速、方便地访问日志文件。要进行这种访问，必须有一条连接目标设备的路由。



5.1.1.3 PLC 专家模式

本章汇集了使用 TwinCAT 数据库服务器的 PLC 专家模式所需的所有内容。与配置模式不同，在该模式下，数据的写入或读取并非基于周期或事件，而是基于程序序列中的特定时间。这需要 SQL 语言知识。

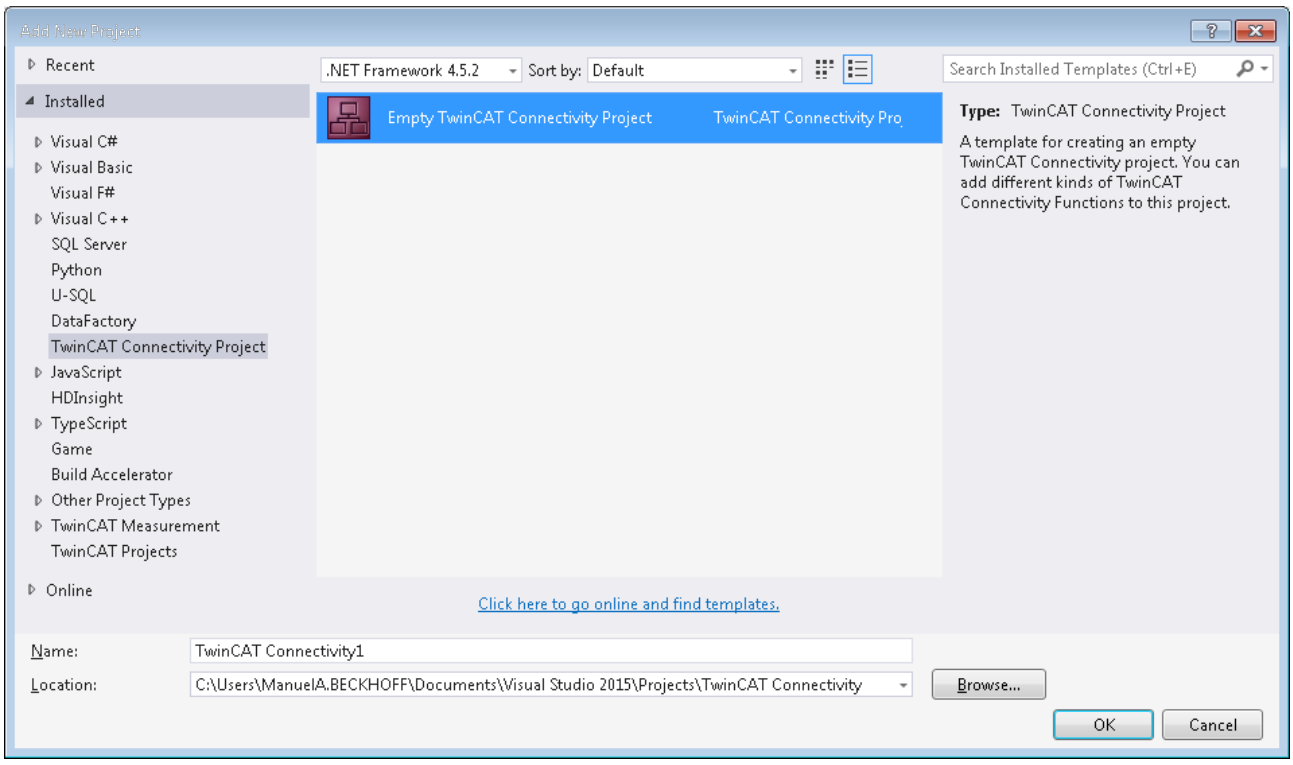
PLC 专家模式

在 PLC 专家模式下，在配置器中只能设置数据库配置。在应用程序的 PLC 代码中可以实现其他功能。使用功能块 FB_PLCDBCreate [▶ 166]，可以省去配置器，甚至可以从 PLC 配置数据库。如有需要，可提供用于读取和写入的功能块。功能块 FB_PLCDBCmd [▶ 177] 可以在 PLC 专家模式和 SQL 专家模式之间进行转换。在这里，可以轻松地将表结构映射为 PLC 结构，并且可以将带有当前结构值占位符的 SQL 命令传输到 TwinCAT 数据库服务器。然后，TwinCAT 数据库服务器会自动插入所有值，并将命令发送到数据库。

构建项目

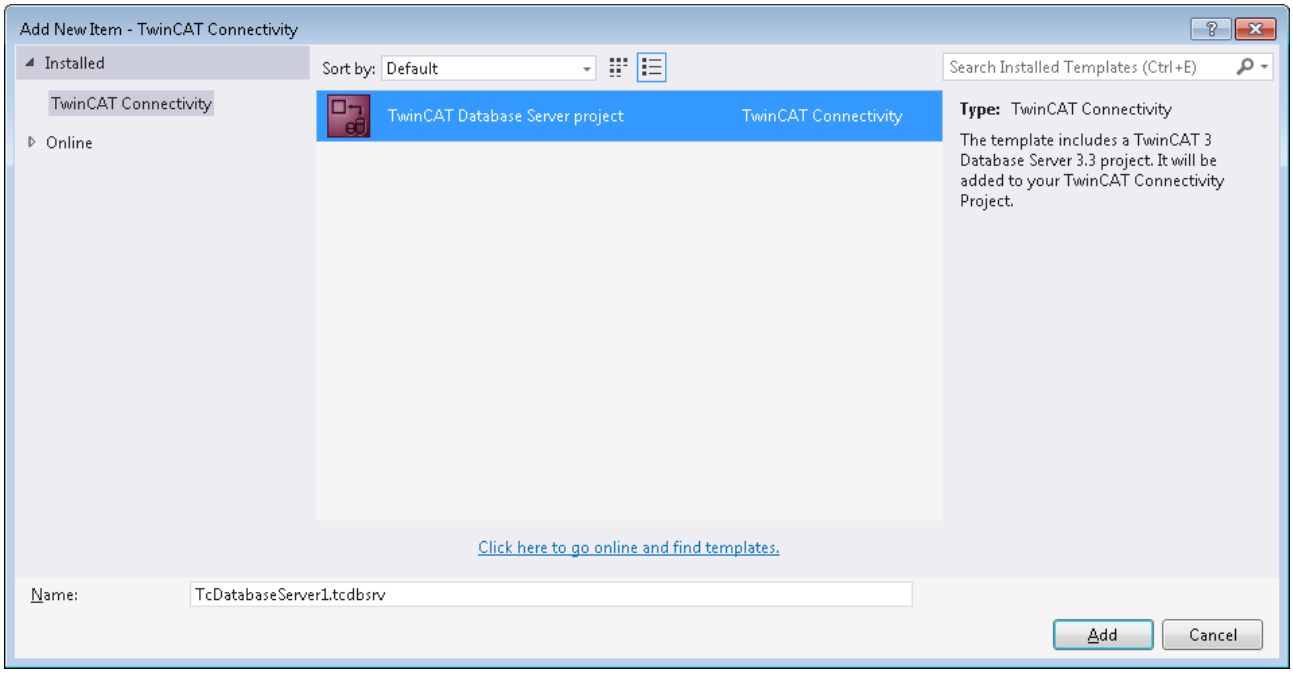
Visual Studio 的 TwinCAT Connectivity 扩展提供了一个新的项目模板。在创建新项目时，**TwinCAT Connectivity 项目**类别将作为一个选项出现。

要创建一个新的 TwinCAT Connectivity 项目，请选择 **Empty TwinCAT Connectivity Project**（空的 TwinCAT Connectivity 项目），指定项目名称和存储位置，然后点击 **OK**（确定）将其添加到解决方案中。这样可以方便地与 TwinCAT 或其他 Visual Studio 项目并行创建 TwinCAT Connectivity 项目或 TwinCAT 数据库服务器项目。

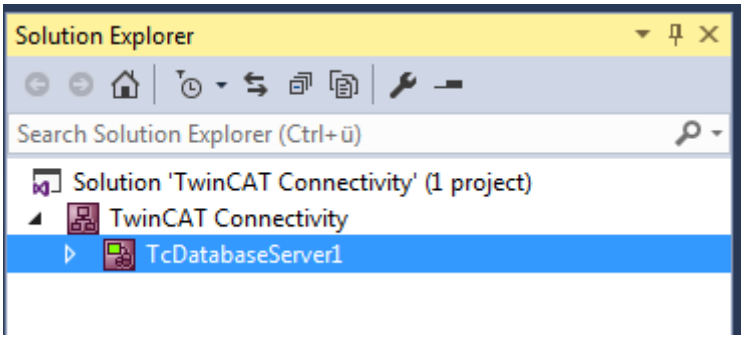


在解决方案中会出现一个新的项目节点。在 Connectivity 项目节点下方，您可以为支持的连接功能添加子项目。

使用 **Add**（添加）将一个新的 TwinCAT 数据库服务器项目添加到 TwinCAT Connectivity 项目中。在现有项目模板列表中可以找到 TwinCAT 数据库服务器项目。



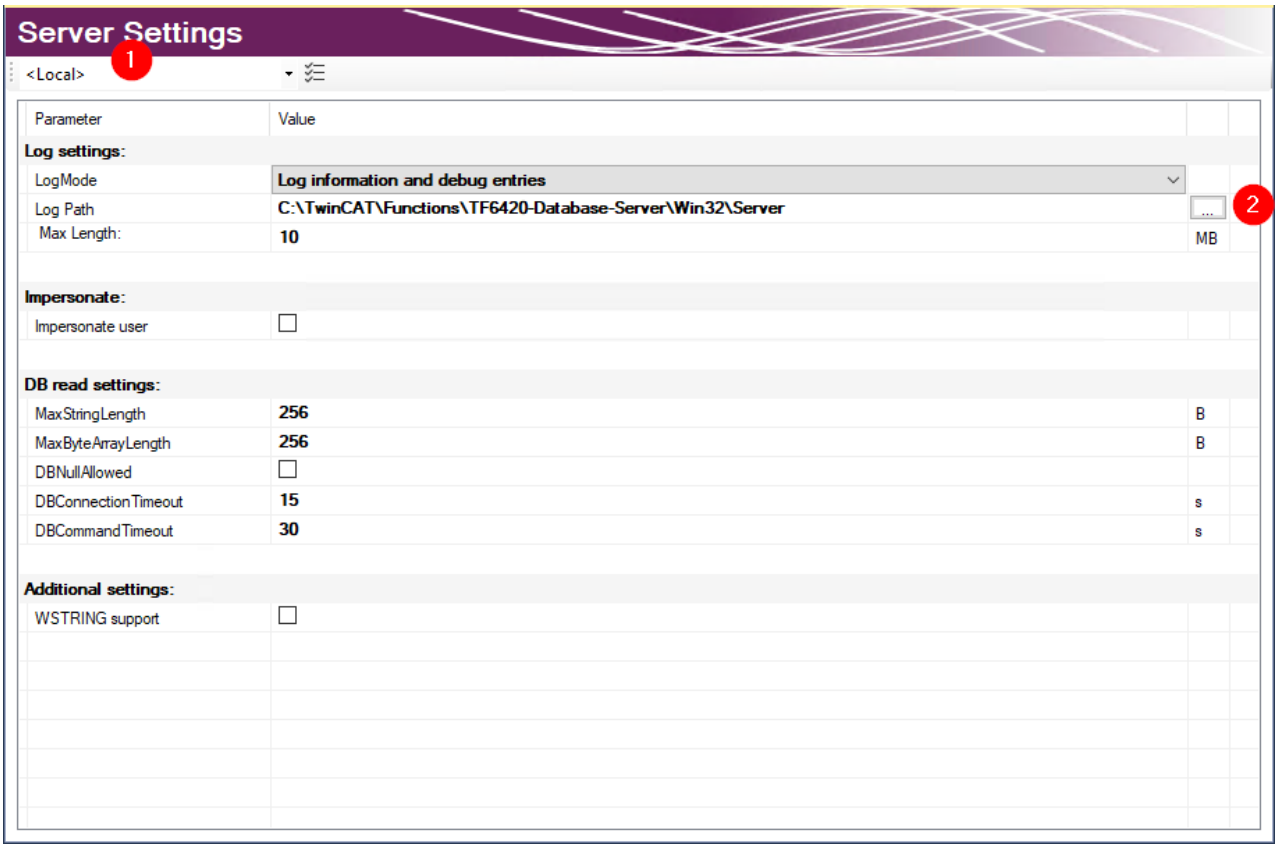
在 TwinCAT Connectivity 节点下，创建一个新的 TwinCAT 数据库服务器项目。



现在可以将其用作 TwinCAT 数据库服务器待配置的基础。通过属性窗口或编辑器可以编辑文档。

一个 Connectivity 项目可以与任意数量的 TwinCAT 数据库服务器项目或其他项目相关联，因此它可能包含多项配置。

服务器设置的编辑器



服务器设置编辑器可以用于编辑 TwinCAT 数据库服务器的设置。这些是与相应的服务器相关的常规设置。在下拉菜单（1）中，您可以通过 Ams NetID 选择目标系统。为此，您必须通过 TwinCAT 创建一条通往目标系统的路由。在传输已完成的配置时，设置将存储在该目标系统的 TwinCAT 数据库服务器中。

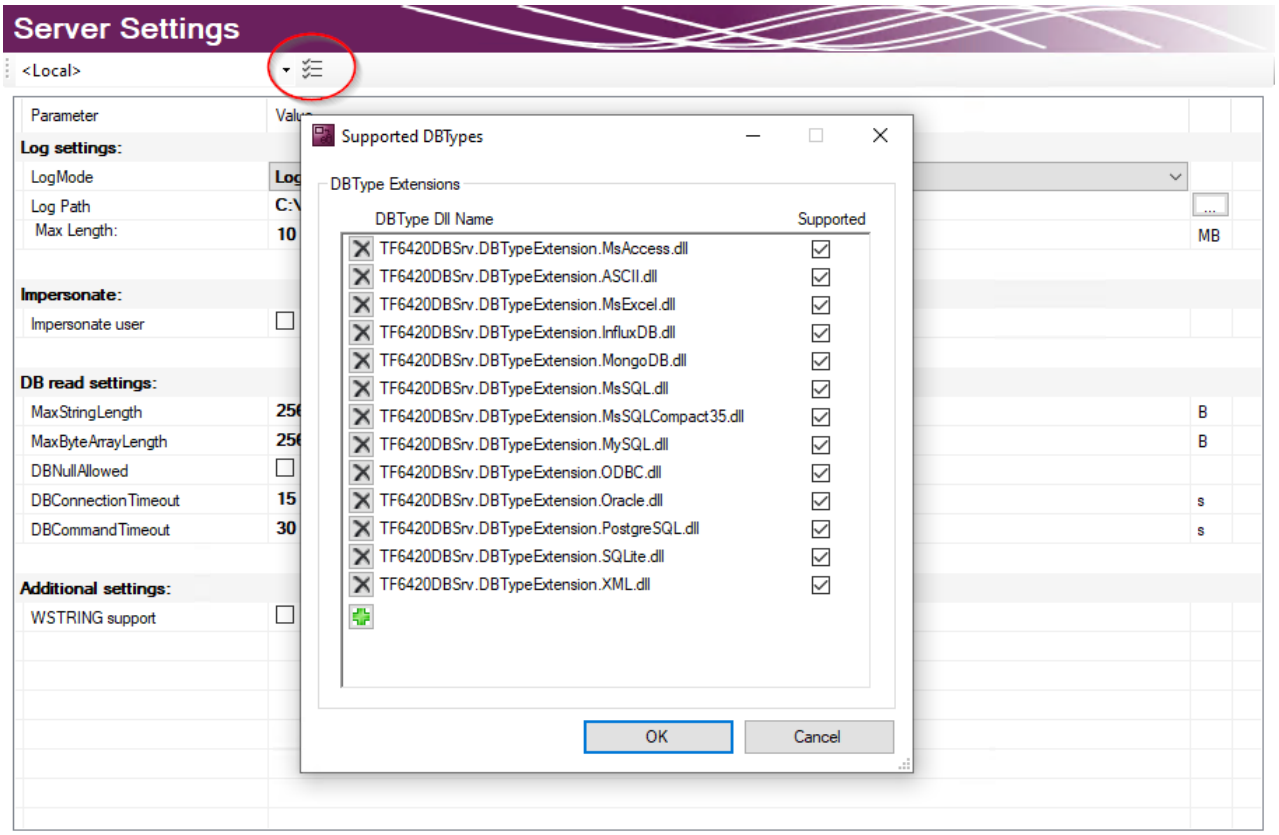
在 **Log settings**（日志设置）下可以配置关于记录故障或错误的设置。在发生故障或错误时，数据库服务器会在日志文件中生成一个详细的条目。通过信息日志查看器 [▶ 48]可以读取日志文件。在 **Log Settings**（日志设置）下，您可以指定文件位置的路径和最大文件大小。您还可以影响日志的准确性。出于性能方面的考虑，我们建议在不再需要的情况下，在错误分析之后再次停用日志记录。

对于基于文件的数据库（例如 Access 或 SQL Compact）的网络访问，必须设置 **Impersonate**（模拟）选项，以便 TwinCAT 数据库服务器可以与该网络驱动器连接。**Windows CE 目前不支持该功能。**

可以使用进一步的配置设置来控制从数据库读取数据的过程。这些设置指的是目标系统上的 TwinCAT 数据库服务器：

MaxStringLength	PLC 中的变量的最大字符串长度
MaxByteArrayLength	PLC 中的变量的最大字节数组长度
DBNullAllowed	表示 TwinCAT 数据库服务器是否接受零值。
DBConnectionTimeout	表示在尝试建立连接时，TwinCAT 数据库服务器出现连接错误的时间。
DBCommandTimeout	表示在发送命令时，TwinCAT 数据库服务器出现连接故障的时间。如果涉及大量数据，则处理一条命令可能需要相当长的时间，具体取决于数据库和基础设施。

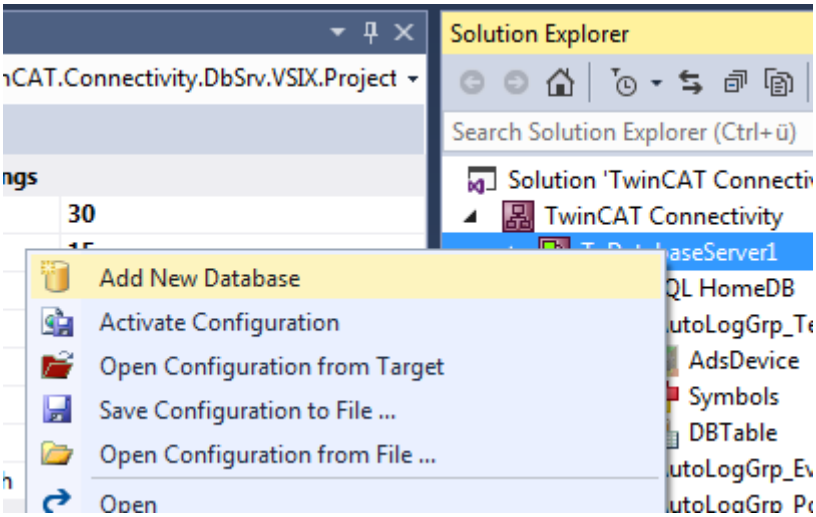
支持的数据库类型



在服务器设置中可以选择已安装的数据库类型。默认情况下会选择所有已安装的数据库。TwinCAT 3 数据库服务器将加载相应的数据库接口。这样可以取消选择目标系统上未使用的数据库。

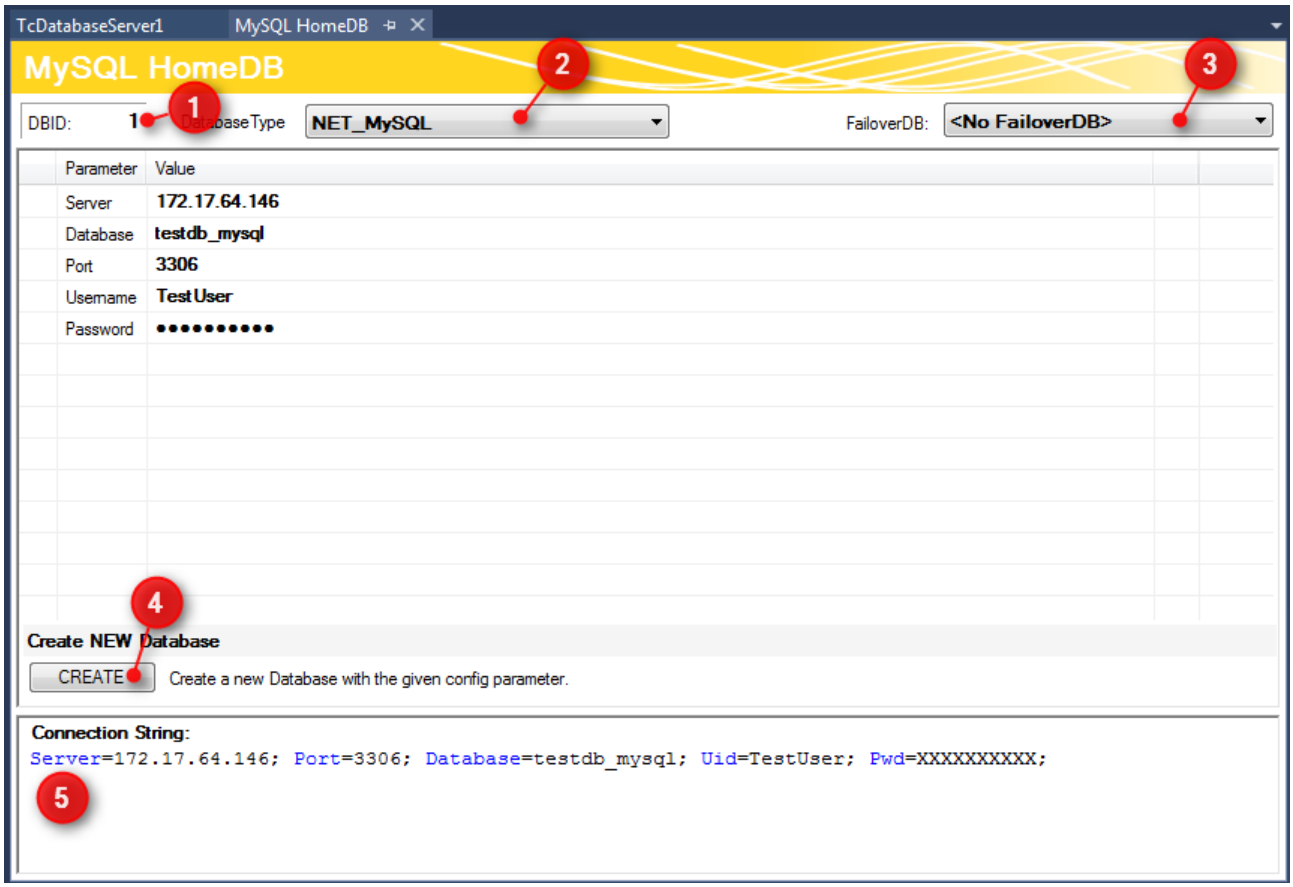
添加数据库配置

通过数据库服务器项目的上下文菜单中的 **Add New Database**（添加新数据库）命令或工具栏中的相应命令可以添加新的数据库配置。



新的数据库配置以文件的形式被添加到项目文件夹中，并集成到项目中。与所有 Visual Studio 项目一样，新文件的信息存储在 Connectivity 项目中。

数据库配置编辑器

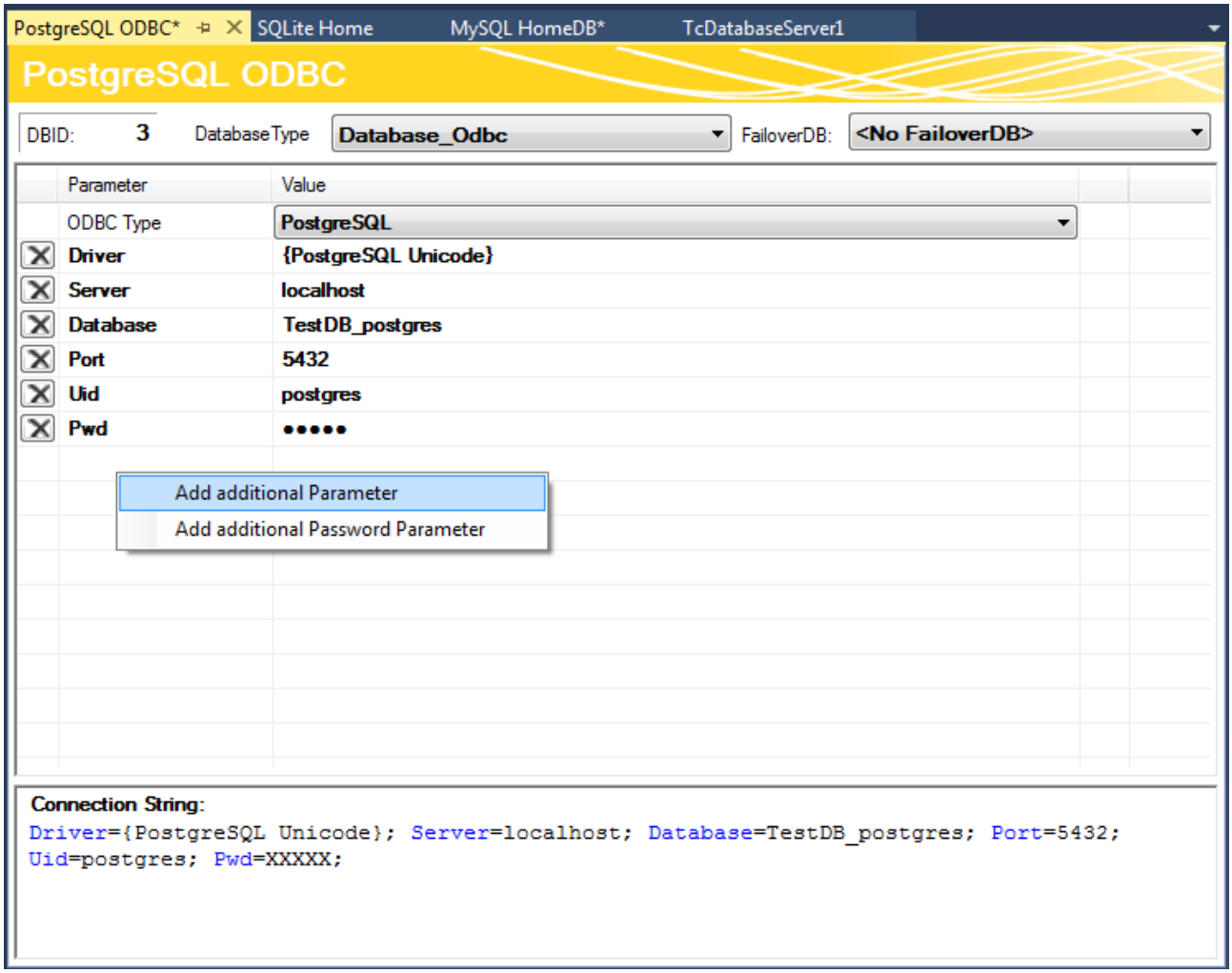


PLC 中的一些功能块所需的数据库 ID 会在编辑器的上部（1）显示。从下拉菜单（2）中可以选择目标数据库的数据库类型。另一个选项是数据库的 ODBC 接口，不过目前还不支持。请注意，根据数据库的不同，并非 TwinCAT 数据库服务器的所有功能都能得到保证。

您还可以选择所谓的故障转移数据库（3），在配置模式下遇到错误时就会触发故障转移数据库。在网络断开的情况下，该功能可以自动确保数据存储在其他地方，而不会丢失。

对于每个数据库 [▶ 120]，都有附加的可调整参数。根据数据库会创建一个连接字符串（5），用于描述与数据库的连接。这样旨在使您设置的参数更加透明。

CREATE（创建）（4）按钮可以用于创建新的数据库。只有在相关数据库支持该功能的情况下，才会显示它。

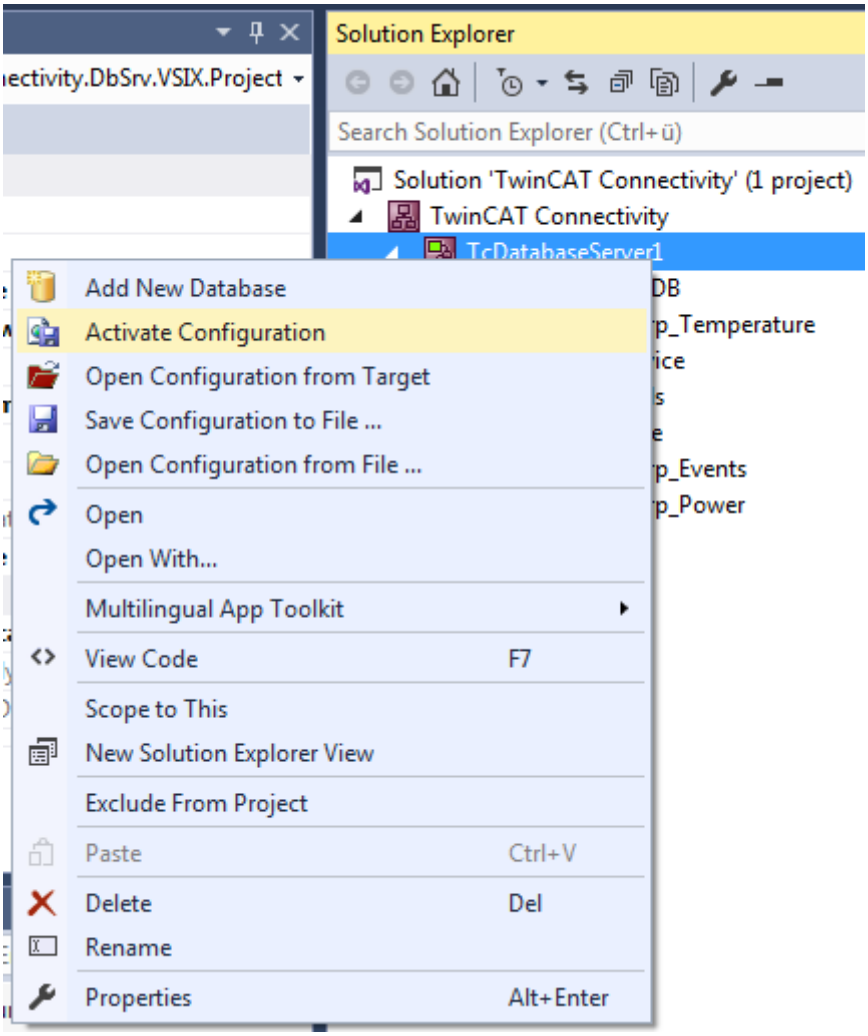


通过 ODBC 接口可以配置未知数据库。在 **ODBC Type**（ODBC 类型）下拉列表中，选择“Unknown Database”（未知数据库），然后通过上下文菜单中的命令添加参数。它们可能包含以加密形式存储的密码。利用这些参数可以组合成所需的连接字符串。请注意，只能使用 TwinCAT 数据库服务器的有限功能。仅支持 SQL 专家模式的显式功能块。

在该模式下不需要额外的 AutoLog 组配置，因为数据库和 PLC 之间的数据读取和写入均由 PLC 程序员手动调用。至此，配置部分已完成。

激活项目

要在 TwinCAT 数据库服务器上激活一个已配置的项目，可使用 TwinCAT 数据库服务器项目的上下文菜单中的 **Activate Configuration**（激活配置）命令。



在激活项目后， 可以用于进一步的开发步骤，例如，创建数据库或表格、为 PLC 生成与数据库的相应表结构相匹配的结构体，或使用已实施的信息测试与数据库的连接。

PLC 程序员可以使用可用的 PLC API [▶ 159] 功能块与 TwinCAT 数据库服务器进行通信。

5.1.1.4 SQL 专家模式

本章介绍了使用 SQL 专家模式所需的所有步骤。该模式专为具有个性化需求的用户量身定制。将对以下主题进行讨论：

1. 创建项目
2. 创建和设置数据库配置
3. 激活数据库服务器项目
4. 使用 SQL 查询编辑器创建 SQL 命令

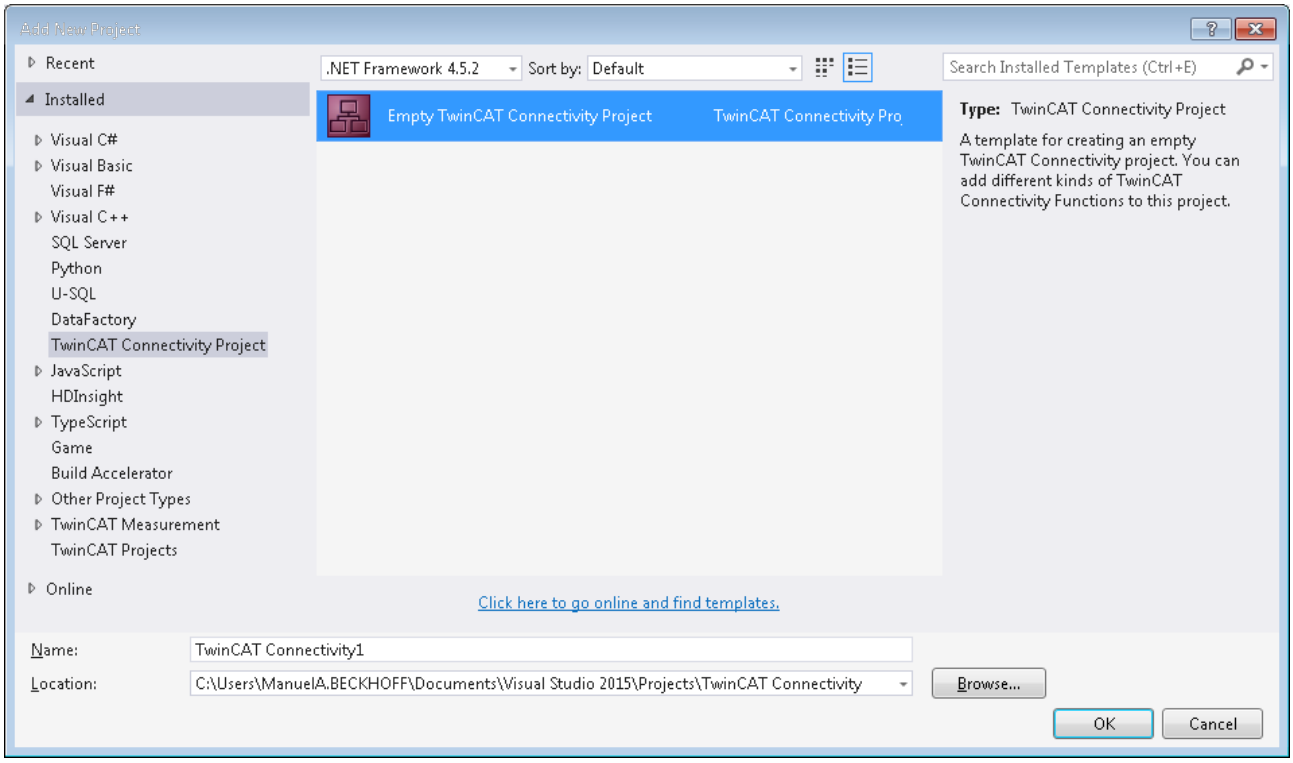
SQL 专家模式

在 SQL 专家模式下，用户可以在 PLC 中对插入、选择或更新等 SQL 命令进行构建，并通过 TwinCAT 数据库服务器将它们发送到数据库。这是一个非常灵活且功能强大的选项。此外，也可以从 PLC 调用数据库中的存储过程 [▶ 194]。

构建项目

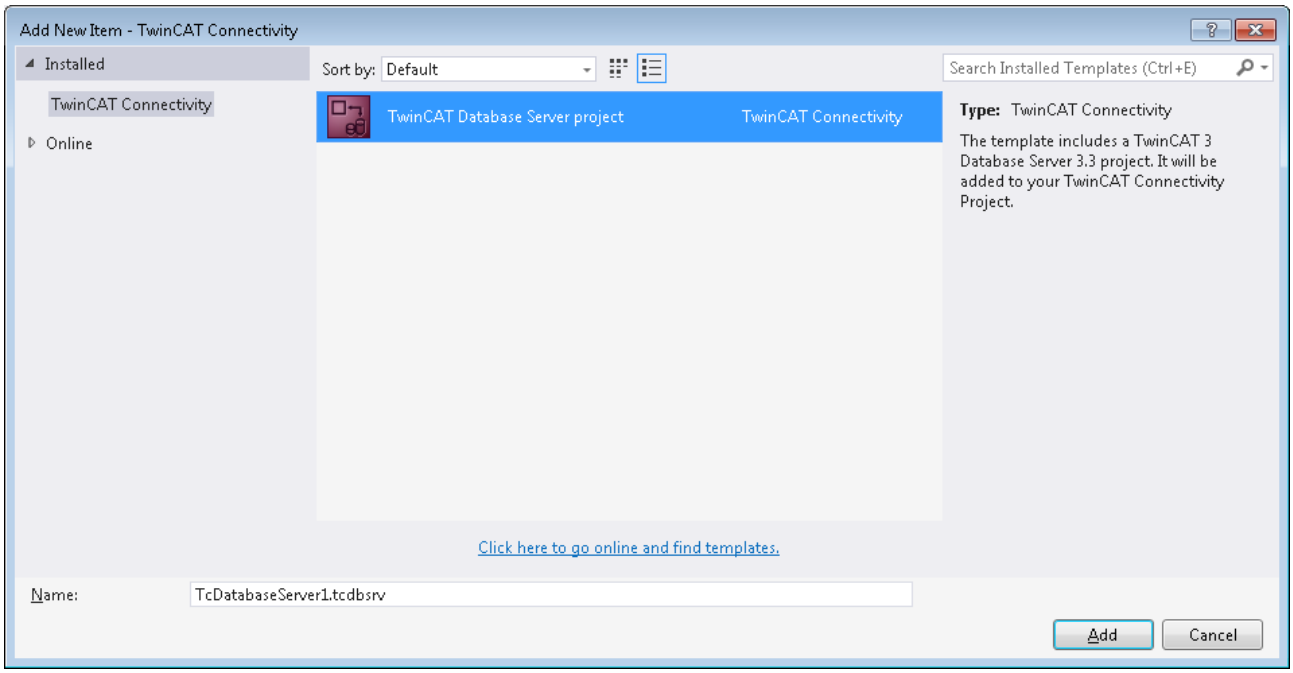
Visual Studio 的 TwinCAT Connectivity 扩展提供了一个新的项目模板。在创建新项目时，**TwinCAT Connectivity 项目**类别将作为一个选项出现。

要创建一个新的 TwinCAT Connectivity 项目，请选择 **Empty TwinCAT Connectivity Project**（空的 TwinCAT Connectivity 项目），指定项目名称和存储位置，然后点击 **OK**（确定）将其添加到解决方案中。这样可以方便地与 TwinCAT 或其他 Visual Studio 项目并行创建 TwinCAT Connectivity 项目或 TwinCAT 数据库服务器项目。

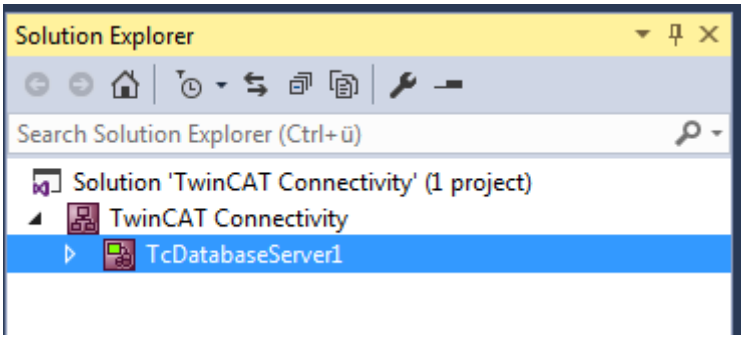


在解决方案中会出现一个新的项目节点。在 Connectivity 项目节点下方，您可以为支持的连接功能添加子项目。

使用 **Add**（添加）将一个新的 TwinCAT 数据库服务器项目添加到 TwinCAT Connectivity 项目中。在现有项目模板列表中可以找到 TwinCAT 数据库服务器项目。



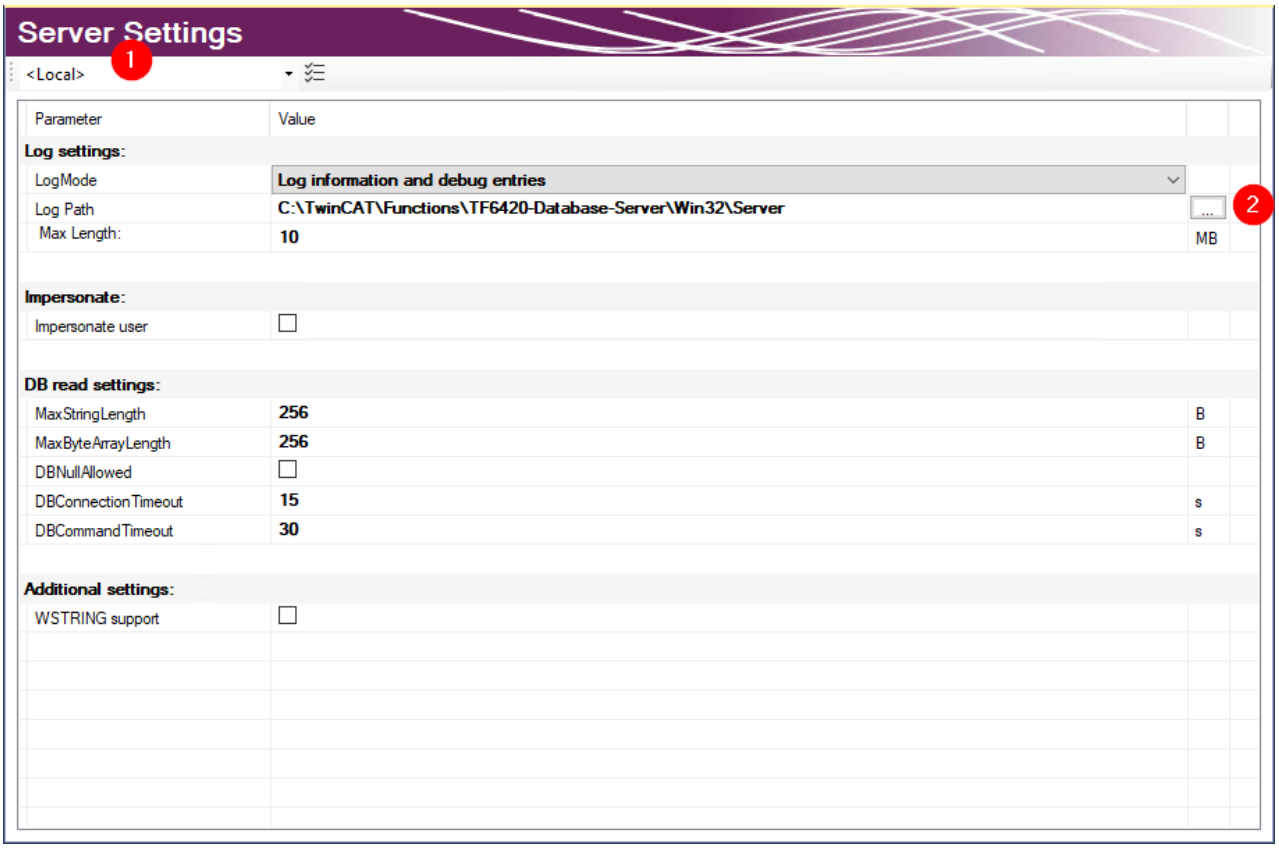
在 TwinCAT Connectivity 节点下，创建一个新的 TwinCAT 数据库服务器项目。



现在可以将其用作 TwinCAT 数据库服务器待配置的基础。通过属性窗口或编辑器可以编辑文档。

一个 Connectivity 项目可以与任意数量的 TwinCAT 数据库服务器项目或其他项目相关联，因此它可能包含多项配置。

服务器设置的编辑器



服务器设置编辑器可以用于编辑 TwinCAT 数据库服务器的设置。这些是与相应的服务器相关的常规设置。在下拉菜单（1）中，您可以通过 Ams NetID 选择目标系统。为此，您必须通过 TwinCAT 创建一条通往目标系统的路由。在传输已完成的配置时，设置将存储在该目标系统的 TwinCAT 数据库服务器中。

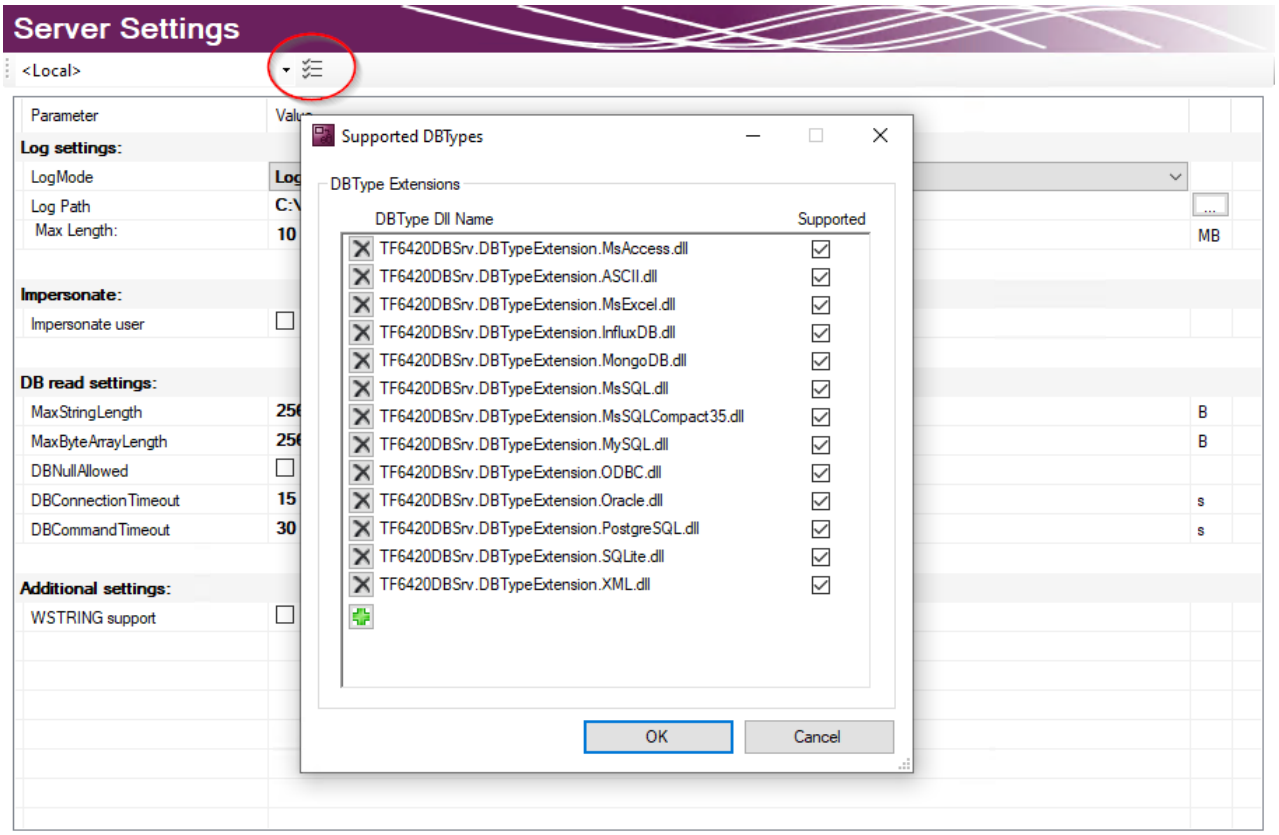
在 **Log settings**（日志设置）下可以配置关于记录故障或错误的设置。在发生故障或错误时，数据库服务器会在日志文件中生成一个详细的条目。通过信息日志查看器 [▶ 48]可以读取日志文件。在 **Log Settings**（日志设置）下，您可以指定文件位置的路径和最大文件大小。您还可以影响日志的准确性。出于性能方面的考虑，我们建议在不再需要的情况下，在错误分析之后再次停用日志记录。

对于基于文件的数据库（例如 Access 或 SQL Compact）的网络访问，必须设置 **Impersonate**（模拟）选项，以便 TwinCAT 数据库服务器可以与该网络驱动器连接。**Windows CE 目前不支持该功能。**

可以使用进一步的配置设置来控制从数据库读取数据的过程。这些设置指的是目标系统上的 TwinCAT 数据库服务器：

MaxStringLength	PLC 中的变量的最大字符串长度
MaxByteArrayLength	PLC 中的变量的最大字节数组长度
DBNullAllowed	表示 TwinCAT 数据库服务器是否接受零值。
DBConnectionTimeout	表示在尝试建立连接时，TwinCAT 数据库服务器出现连接错误的时间。
DBCommandTimeout	表示在发送命令时，TwinCAT 数据库服务器出现连接故障的时间。如果涉及大量数据，则处理一条命令可能需要相当长的时间，具体取决于数据库和基础设施。

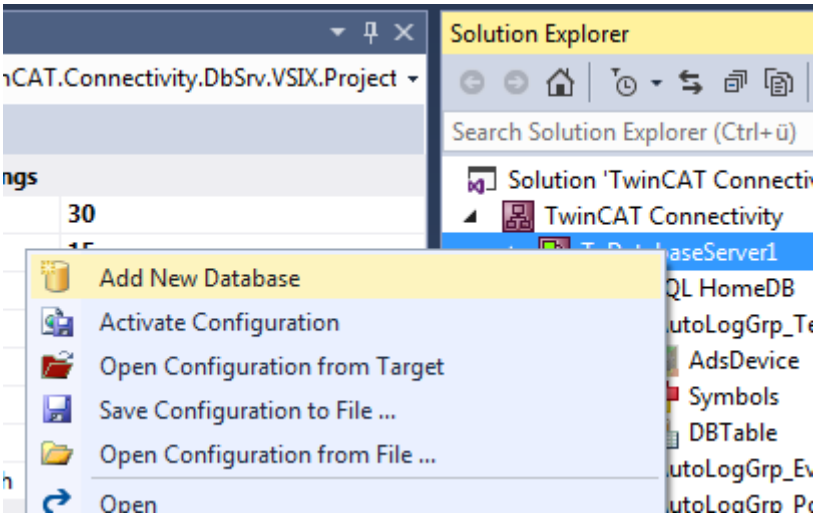
支持的数据库类型



在服务器设置中可以选择已安装的数据库类型。默认情况下会选择所有已安装的数据库。TwinCAT 3 数据库服务器将加载相应的数据库接口。这样可以取消选择目标系统上未使用的数据库。

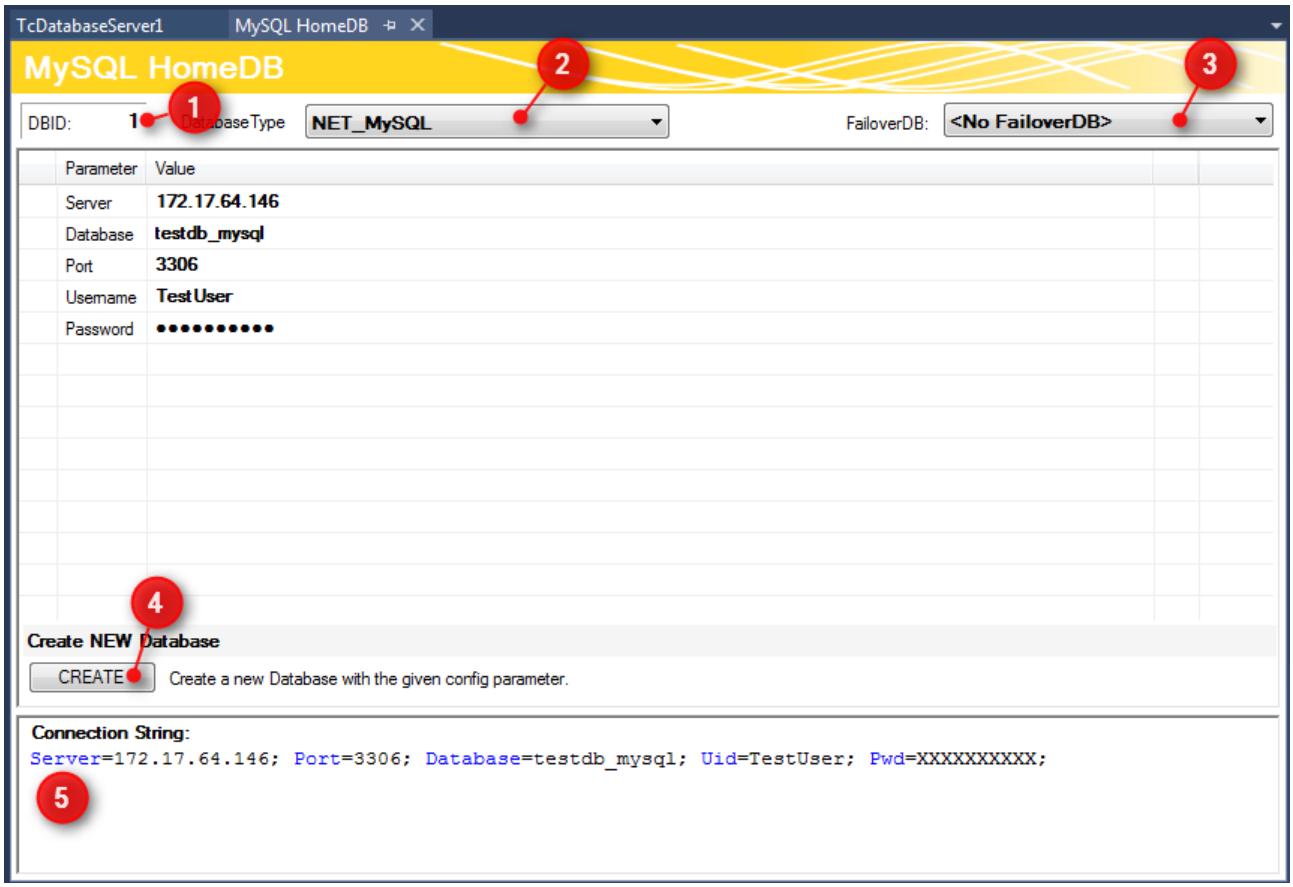
添加数据库配置

通过数据库服务器项目的上下文菜单中的 **Add New Database**（添加新数据库）命令或工具栏中的相应命令可以添加新的数据库配置。



新的数据库配置以文件的形式被添加到项目文件夹中，并集成到项目中。与所有 Visual Studio 项目一样，新文件的信息存储在 Connectivity 项目中。

数据库配置编辑器

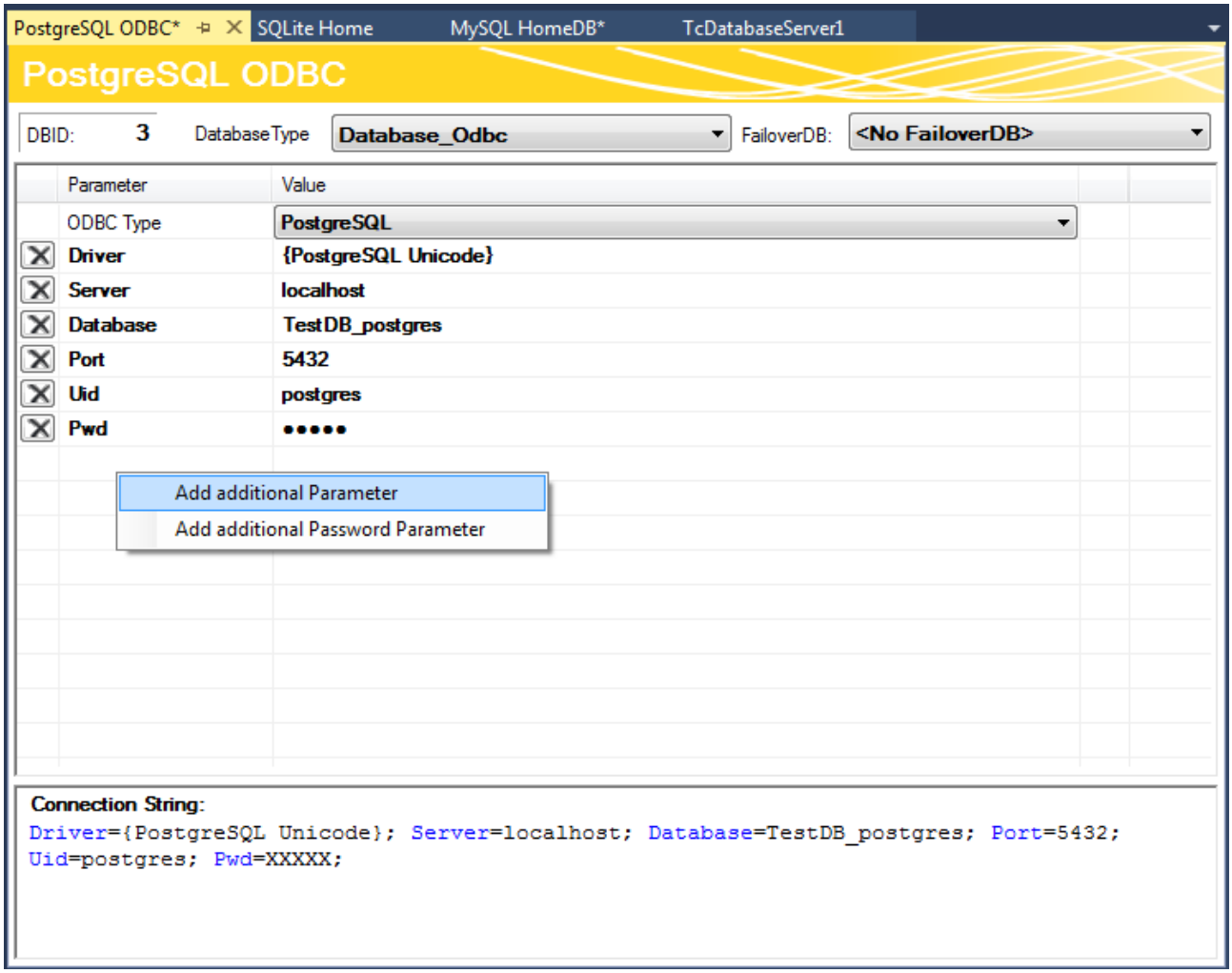


PLC 中的一些功能块所需的数据库 ID 会在编辑器的上部（1）显示。从下拉菜单（2）中可以选择目标数据库的数据库类型。另一个选项是数据库的 ODBC 接口，不过目前还不支持。请注意，根据数据库的不同，并非 TwinCAT 数据库服务器的所有功能都能得到保证。

您还可以选择所谓的故障转移数据库（3），在配置模式下遇到错误时就会触发故障转移数据库。在网络断开的情况下，该功能可以自动确保数据存储在其他地方，而不会丢失。

对于每个数据库 [▶ 120]，都有附加的可调整参数。根据数据库会创建一个连接字符串（5），用于描述与数据库的连接。这样旨在使您设置的参数更加透明。

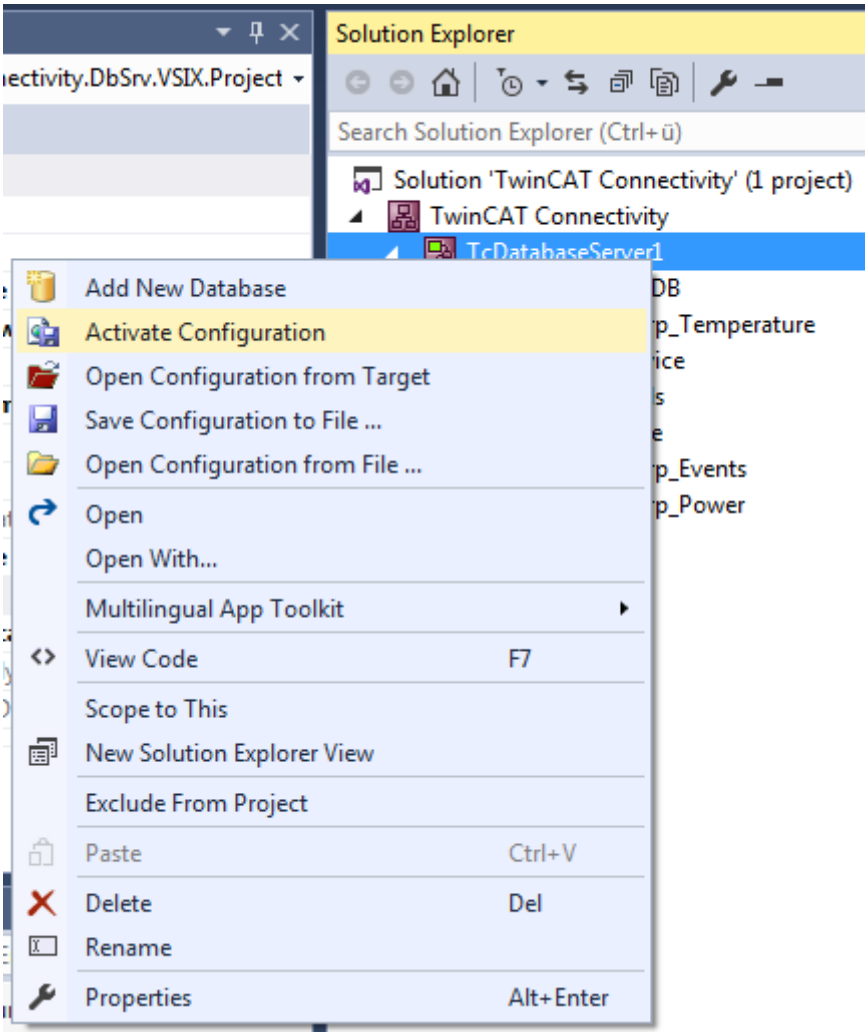
CREATE（创建）（4）按钮可以用于创建新的数据库。只有在相关数据库支持该功能的情况下，才会显示它。



通过 ODBC 接口可以配置未知数据库。在 **ODBC Type**（ODBC 类型）下拉列表中，选择“Unknown Database”（未知数据库），然后通过上下文菜单中的命令添加参数。它们可能包含以加密形式存储的密码。利用这些参数可以组合成所需的连接字符串。请注意，只能使用 TwinCAT 数据库服务器的有限功能。仅支持 SQL 专家模式的显式功能块。

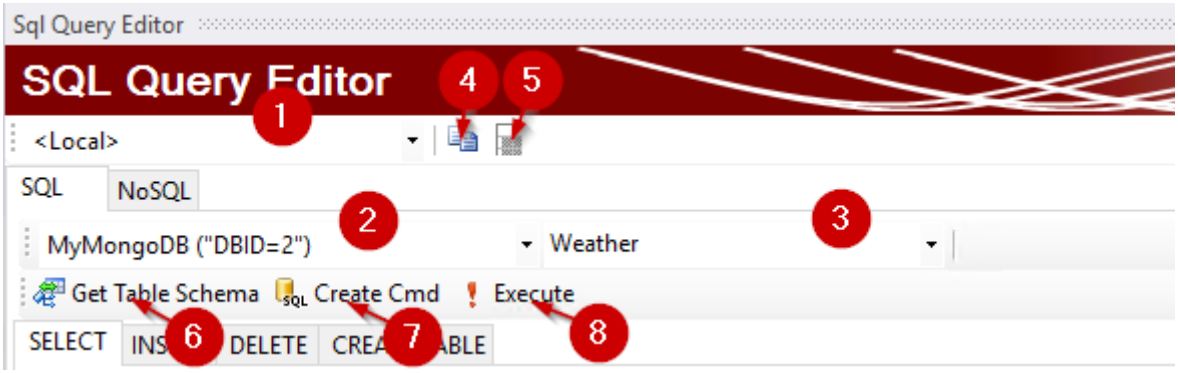
激活项目

要在 TwinCAT 数据库服务器上激活一个已配置的项目，可使用 TwinCAT 数据库服务器项目的上下文菜单中的 **Activate Configuration**（激活配置）命令。



SQL 查询编辑器

SQL 查询编辑器是一种数据库服务器工具，为您的应用程序开发提供支持。该工具可以用于测试连接状态和 SQL 命令，并检查 PLC 和数据库之间的兼容性。



ID	Name	功能
1	目标系统	选择已安装 TwinCAT 数据库服务器的目标系统
2	数据库	选择已配置的数据库连接
3	表格	选择数据库中的已有表格
4	为 PLC 进行复制	将 SQL 命令复制到 PLC 字符串。可将其复制到 PLC 源代码中。特殊字符会被自动捕获并格式化。
5	导出 TC3	将表格架构导出到 PLC 结构中。例如，这可以在使用 SQL 命令的程序中使用。
6	获取表格架构	读取表结构
7	创建命令	根据表结构，创建 SQL 命令
8	执行	执行 SQL 命令

首先，从 TwinCAT 系统的路由中选择目标系统（1）。在目标系统上必须安装 TwinCAT 数据库服务器。如果在配置中已存储 NoSQL 数据库，则会显示一个附加的 NoSQL 选项卡。您可以在下面的子项目中找到文档。

当您在目标系统上激活数据库配置后，将会显示所有已配置的数据库（2）。您也可以从数据库中选择一个可用的表格（3）。根据该表格，您可以从 SQL 查询编辑器中生成 SQL 命令，并将其发送到数据库。根据数据库类型的不同，SQL 命令有不同的语法。

有 3 种命令可用于生成单独的 SQL 命令：

- 获取表格架构：调用选定表格的结构。
  - 将会显示列名、PLC 数据类型和变量大小等信息。此外，还可以通过 **Copy for PLC**（为 PLC 进行复制）（4）或 **Export TC3**（导出 TC3）（5）命令为 PLC 应用程序准备检索到的结构。
- 创建命令：在命令文本框中生成一条 SQL 命令，具体取决于选定的选项卡。根据数据库类型的不同，命令语法可能会有所差异。在这里会使用先前读取的表格架构。
  - 可以选择修改已创建的 SQL 命令。
- 执行：执行在文本框中显示的 SQL 命令并返回值（如适用）。

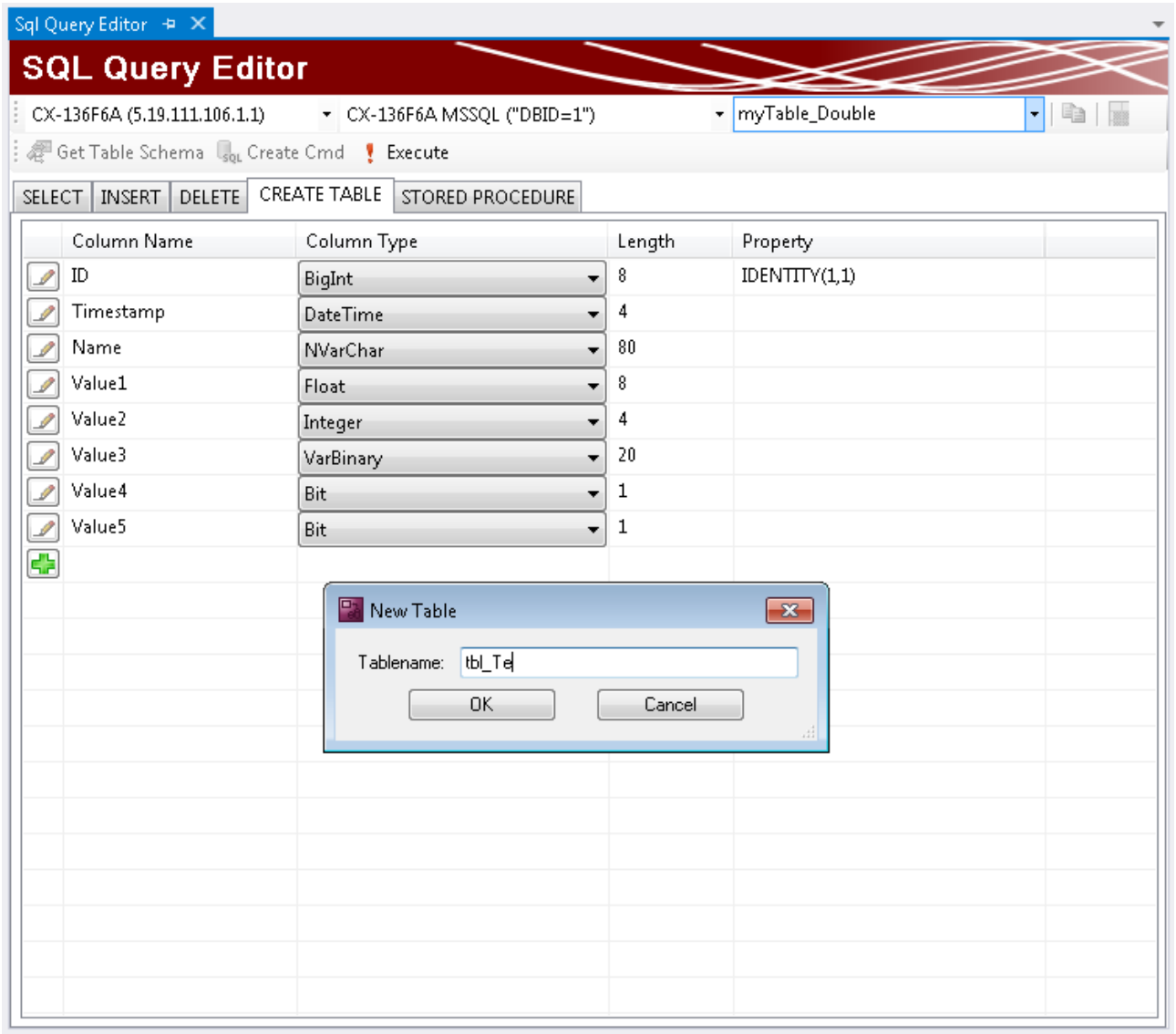
下文将解释各个 SQL 命令中的差异。

注释：由于 SQL 命令的语法经常与 TwinCAT 的 ST 代码中的语法相冲突，因此 SQL 查询编辑器提供了命令“Copy for PLC”（为 PLC 进行复制）（4）。该命令可用于将已创建且已测试的 SQL 命令复制到缓存中，并为 ST 程序代码的特殊字符设置正确的格式。

创建表格命令

CREATE TABLE（创建表格）选项卡可以用于在数据库中创建表格。根据需要可以使用（+）将更多列添加到表格中。在您已指定列名和类型后，您还可以指定其他属性，以便生成自动 ID 等。

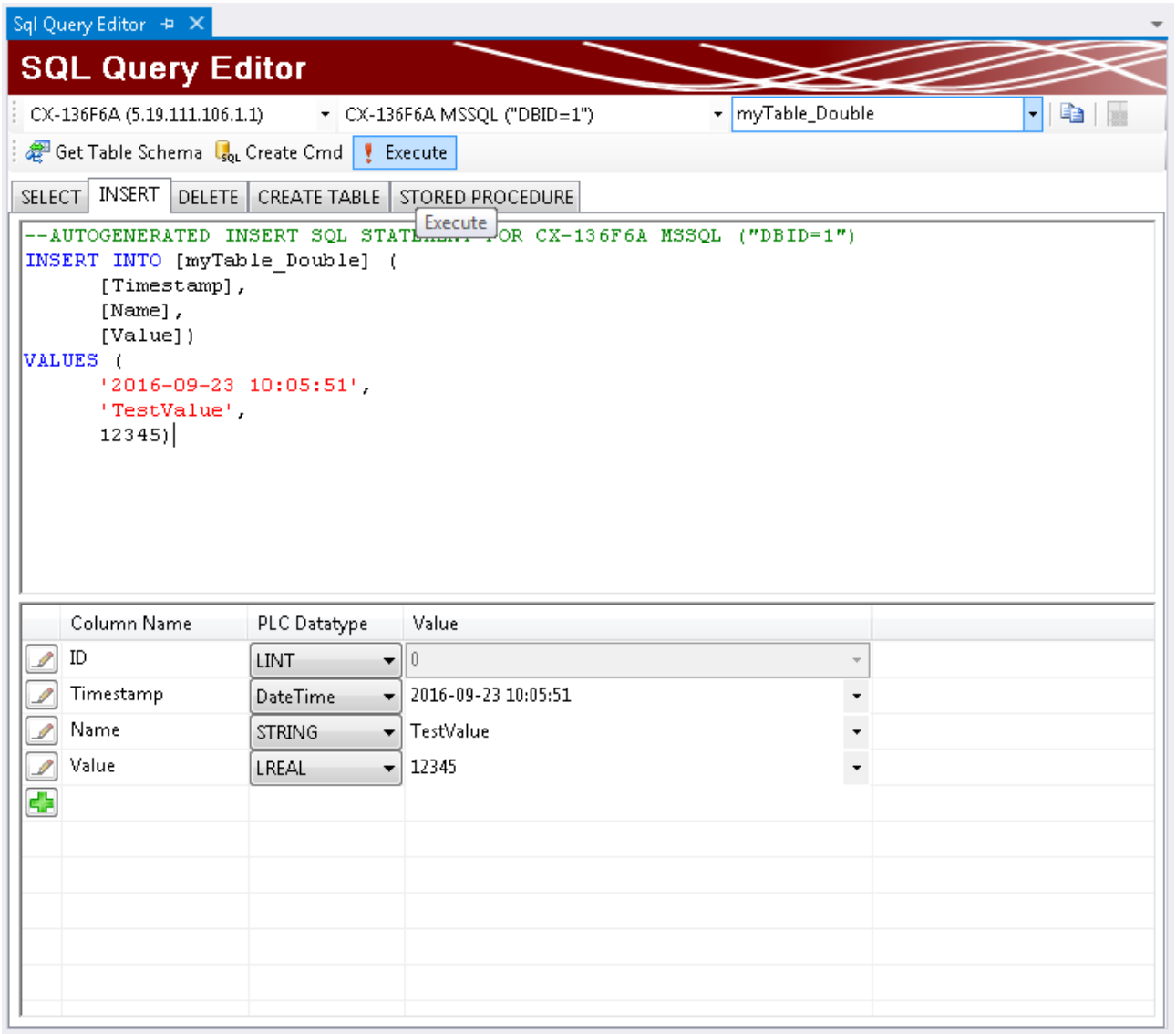
通过执行命令可以确定表格名称。您可以创建具有已配置的表结构的表格。



插入命令

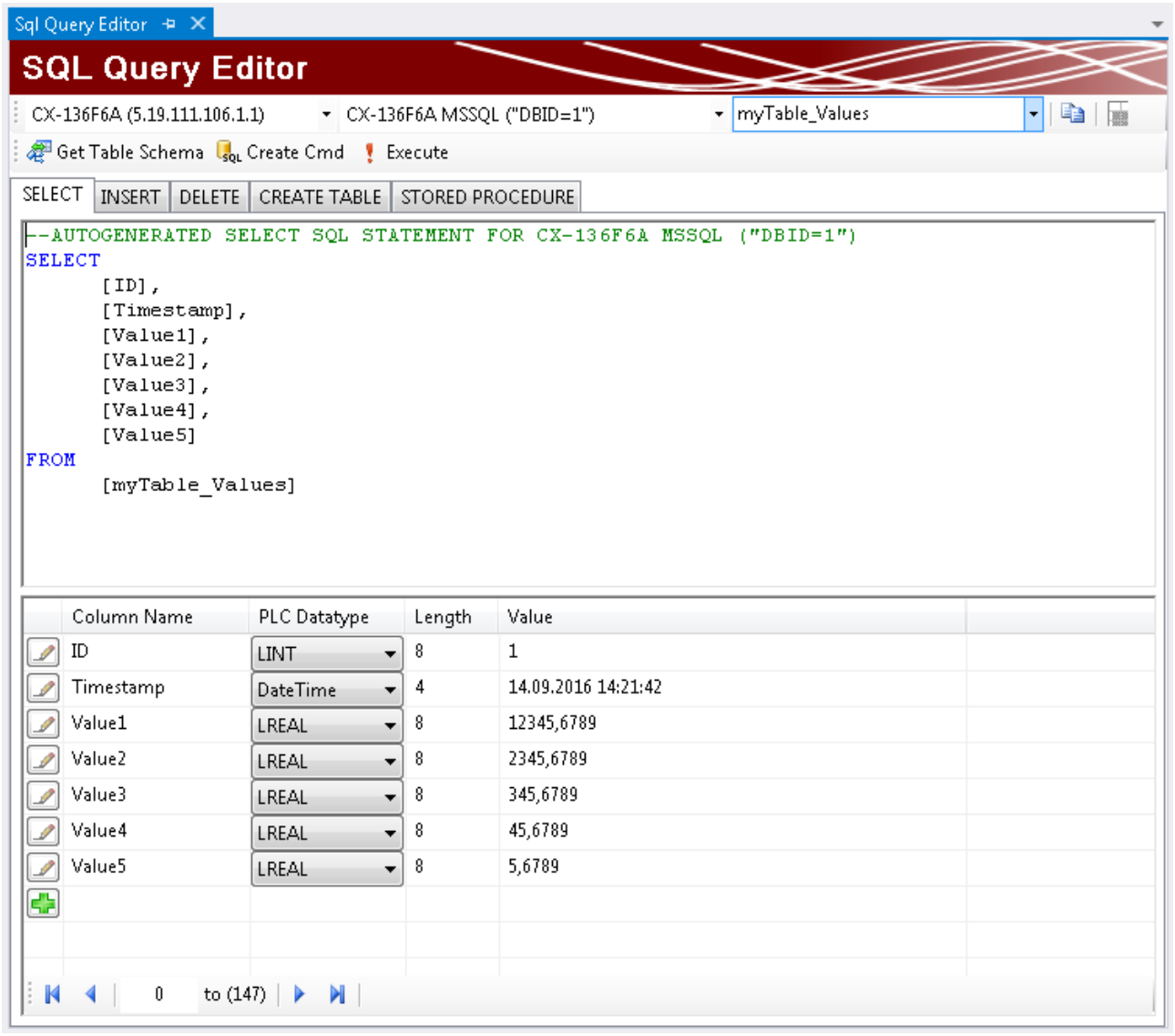
插入命令使得您可以将记录数据写入表格。在检索到表结构后，您可以修改“Value”（值）下的值。如果随后生成命令，则插入命令中的值将会自动采用正确的格式。在执行命令时，这些值将被写入表格。

如果使用自动 ID 生成功能，则无法对该值进行自定义修改。



选择命令

通过 **SELECT**（选择）选项卡可以创建和发送选择命令。选择命令使您可以从数据库中读取记录数据。在执行命令后，如果表格中存在值，则会返回这些值。它们会在表结构显示中的“Value”（值）下列出。使用显示下方的箭头可以浏览各个记录。

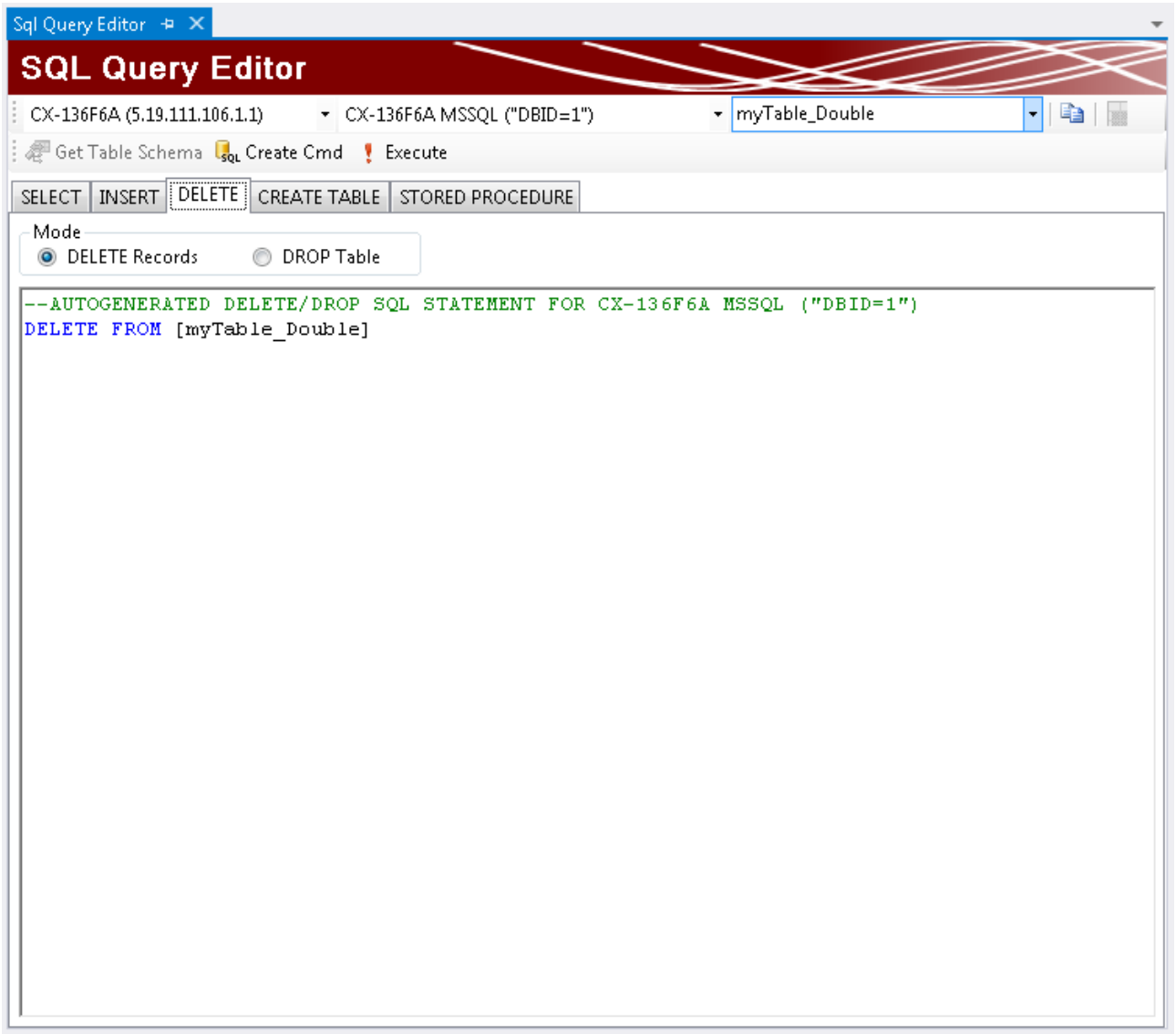


删除命令

删除命令有 2 个功能。

1. **删除记录**：删除表格的内容。
2. **丢弃表格**：删除整个表格。

此外，也可以对该 SQL 命令进行自定义，比如仅删除表格中的某个特定部分。

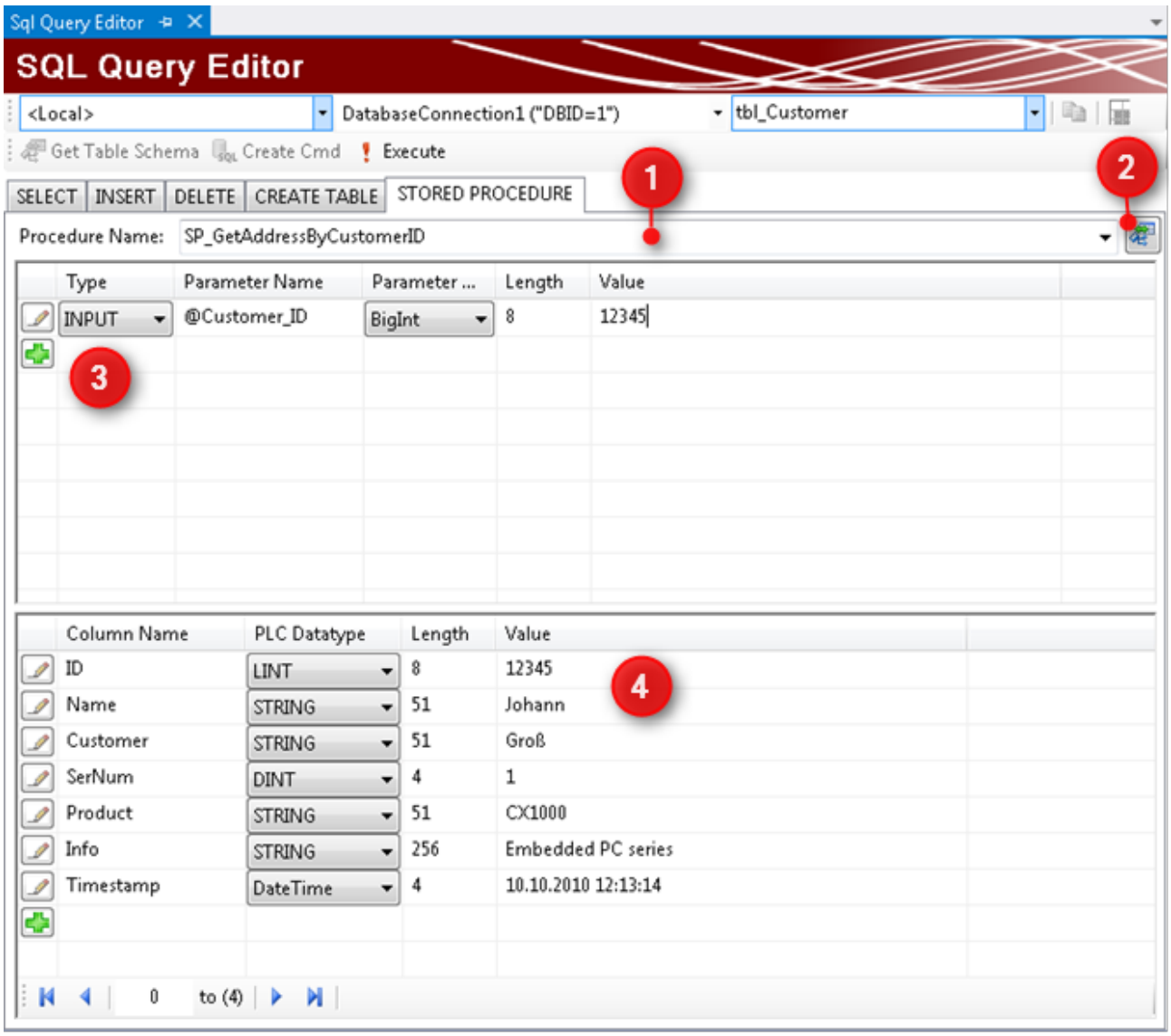


存储过程

TwinCAT 数据库服务器支持“存储过程”，它支持大量的数据库，用于在数据库层处理更加复杂的查询或提供简化的接口。

如果在数据库和表格中支持存储过程，则您可以列出并选择它们（1）。可以自动拾取输入和输出参数（2），并将其传输到显示的表格中（3）（4）。

在这里会显示参数类型、名称和数据类型。此外，您还可以在这里插入值，以便通过“Execute”（执行）使用输入值执行存储过程。结果会在输出值（4）中显示。如果返回多个记录，则可以使用箭头键在这些记录之间切换。该功能可作为在 PLC 中进行调用的开发辅助工具。通过调用相应的功能块 [▶ 194]可以返回结果。

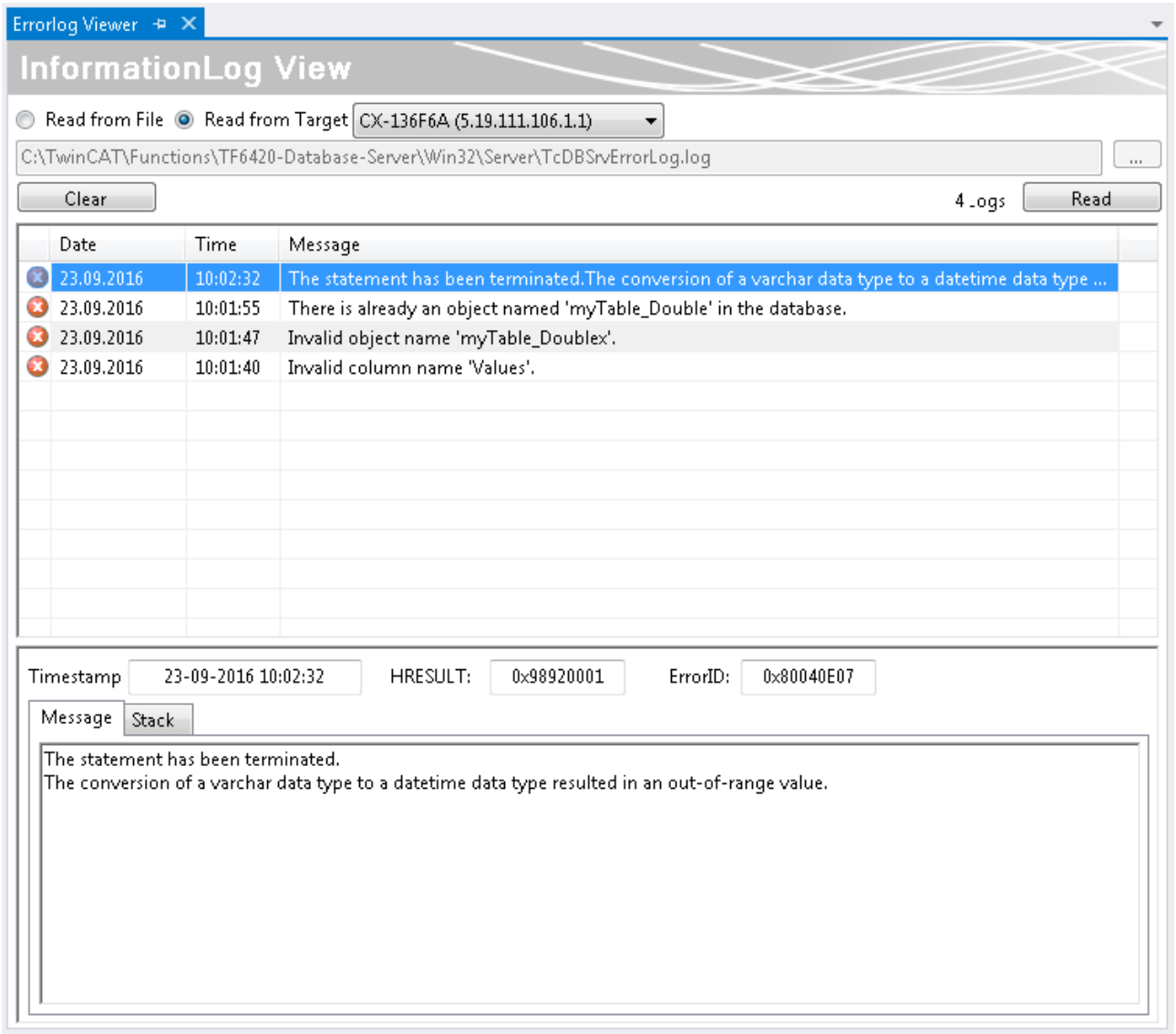


用于诊断的 InformationLog 视图

InformationLog 视图可用于故障排除。

InformationLog View是一种从 TwinCAT 数据库服务器读取日志文件的工具。记录的信息会以纯文本的形式显示，包含时间戳、ID 和错误消息。

不仅可以直接通过文件访问功能来查看或清空日志文件，还可以直接通过目标系统实现这些操作。这在同一网络下有分布式数据库服务器时特别有优势，可以借此快速、方便地访问日志文件。要进行这种访问，必须有一条连接目标设备的路由。



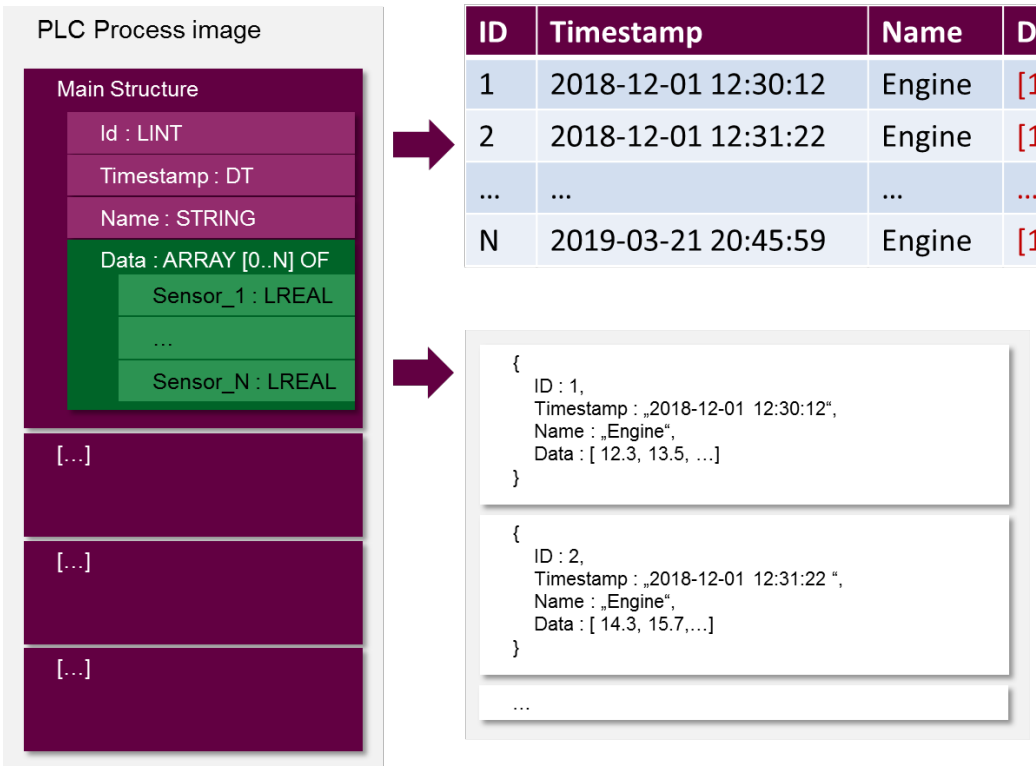
5.1.1.5 NoSql 专家模式

NoSQL

NoSQL 数据库（不仅仅是 Sequel）与传统的关系型数据存储不同。

基于文档的数据库：

记录数据作为文档存储在数据库中。这样做的好处是能够以更加灵活且层次分明的方式存档数据。



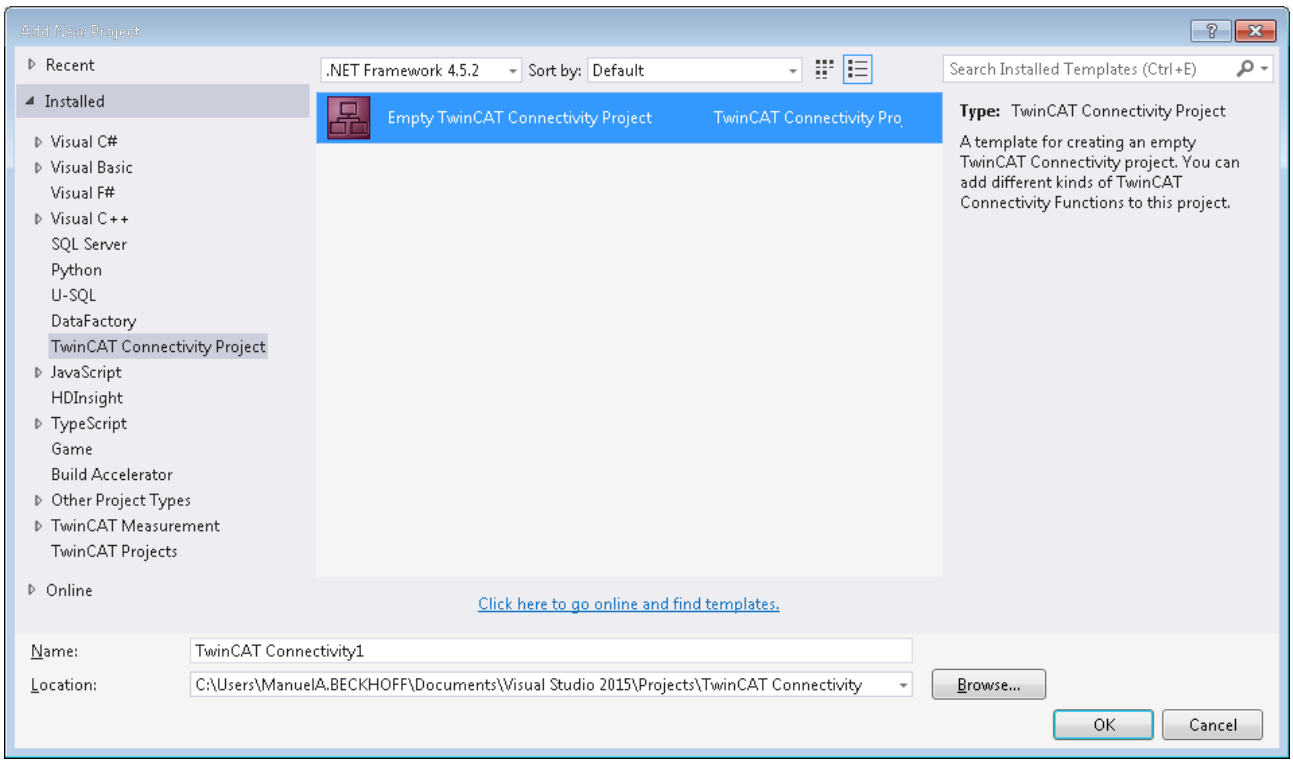
如果一个记录由多个基本数据类型的扁平化结构组成，则不能再通过关系型数据库进行直接映射。非关系型数据库具有这种灵活性。对程序代码和相应结构进行修改也很容易，而无需创建新的表格。

基于文档的数据库通常会将数据另存为 JSON 格式的记录。所有记录可能都不同。

构建项目

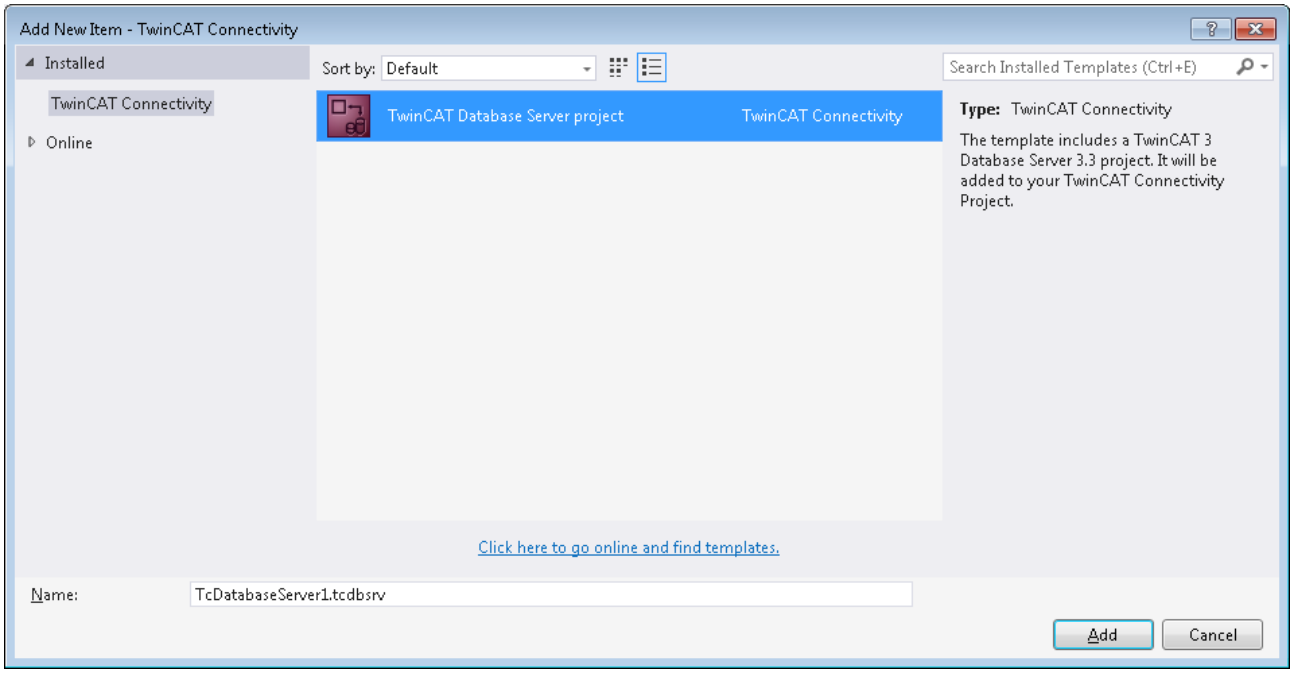
Visual Studio 的 TwinCAT Connectivity 扩展提供了一个新的项目模板。在创建新项目时，**TwinCAT Connectivity 项目**类别将作为一个选项出现。

要创建一个新的 TwinCAT Connectivity 项目，请选择 **Empty TwinCAT Connectivity Project**（空的 TwinCAT Connectivity 项目），指定项目名称和存储位置，然后点击 **OK**（确定）将其添加到解决方案中。这样可以方便地与 TwinCAT 或其他 Visual Studio 项目并行创建 TwinCAT Connectivity 项目或 TwinCAT 数据库服务器项目。

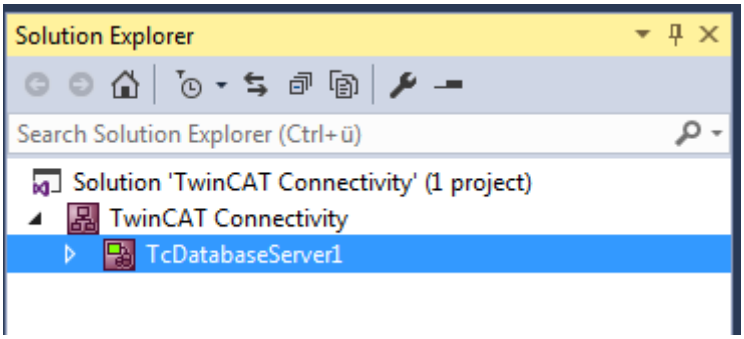


在解决方案中会出现一个新的项目节点。在 Connectivity 项目节点下方，您可以为支持的连接功能添加子项目。

使用 **Add**（添加）将一个新的 TwinCAT 数据库服务器项目添加到 TwinCAT Connectivity 项目中。在现有项目模板列表中可以找到 TwinCAT 数据库服务器项目。



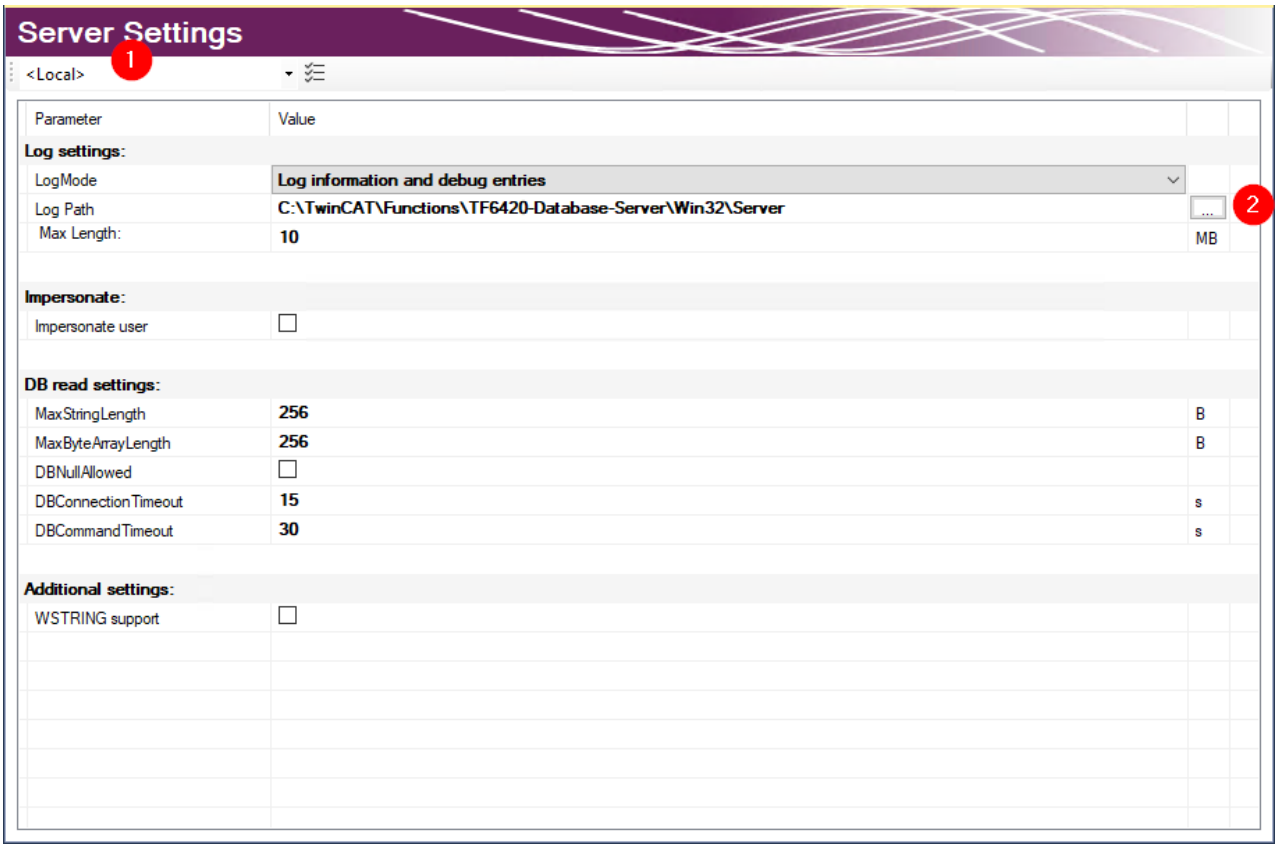
在 TwinCAT Connectivity 节点下，创建一个新的 TwinCAT 数据库服务器项目。



现在可以将其用作 TwinCAT 数据库服务器待配置的基础。通过属性窗口或编辑器可以编辑文档。

一个 Connectivity 项目可以与任意数量的 TwinCAT 数据库服务器项目或其他项目相关联，因此它可能包含多项配置。

服务器设置的编辑器



服务器设置编辑器可以用于编辑 TwinCAT 数据库服务器的设置。这些是与相应的服务器相关的常规设置。在下拉菜单（1）中，您可以通过 Ams NetID 选择目标系统。为此，您必须通过 TwinCAT 创建一条通往目标系统的路由。在传输已完成的配置时，设置将存储在该目标系统的 TwinCAT 数据库服务器中。

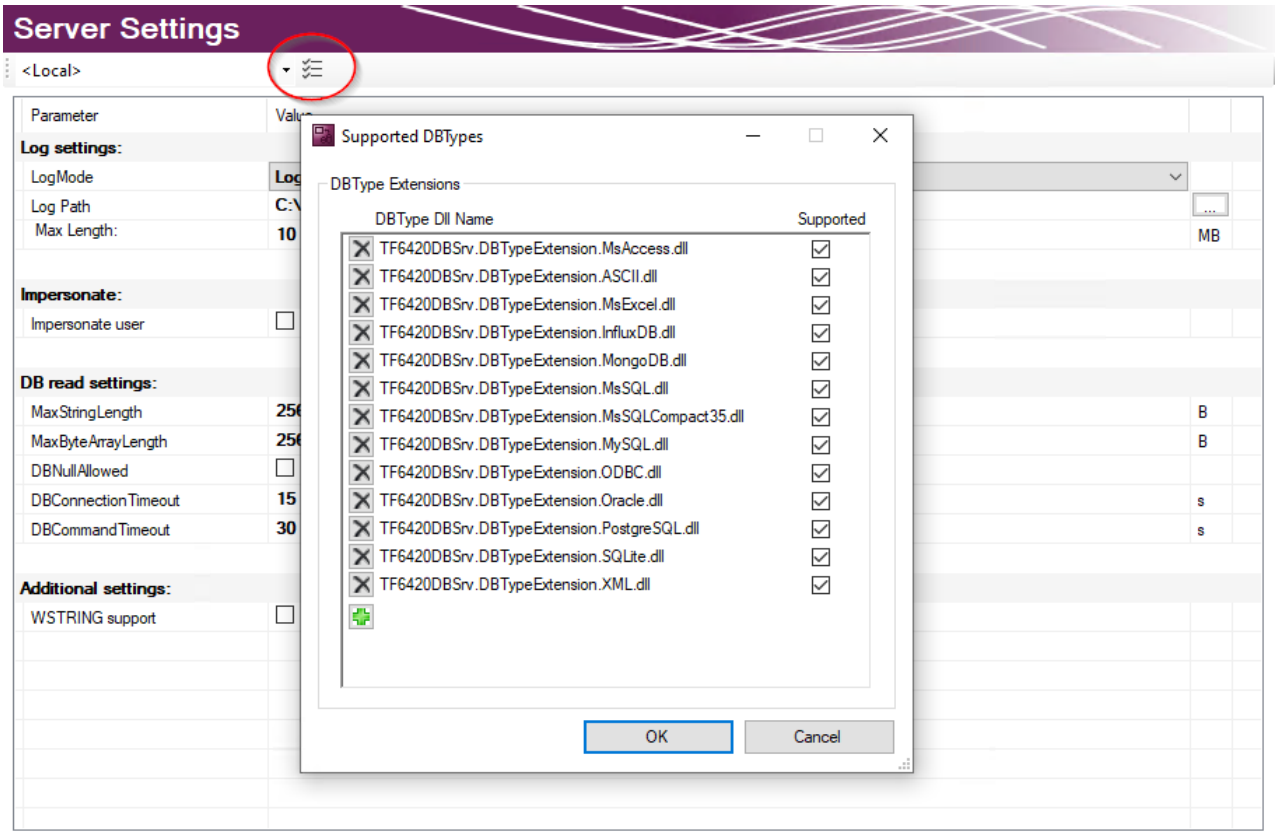
在 **Log settings**（日志设置）下可以配置关于记录故障或错误的设置。在发生故障或错误时，数据库服务器会在日志文件中生成一个详细的条目。通过信息日志查看器 [▶ 48]可以读取日志文件。在 **Log Settings**（日志设置）下，您可以指定文件位置的路径和最大文件大小。您还可以影响日志的准确性。出于性能方面的考虑，我们建议在不再需要的情况下，在错误分析之后再次停用日志记录。

对于基于文件的数据库（例如 Access 或 SQL Compact）的网络访问，必须设置 **Impersonate**（模拟）选项，以便 TwinCAT 数据库服务器可以与该网络驱动器连接。**Windows CE 目前不支持该功能。**

可以使用进一步的配置设置来控制从数据库读取数据的过程。这些设置指的是目标系统上的 TwinCAT 数据库服务器：

MaxStringLength	PLC 中的变量的最大字符串长度
MaxByteArrayLength	PLC 中的变量的最大字节数组长度
DBNullAllowed	表示 TwinCAT 数据库服务器是否接受零值。
DBConnectionTimeout	表示在尝试建立连接时，TwinCAT 数据库服务器出现连接错误的时间。
DBCommandTimeout	表示在发送命令时，TwinCAT 数据库服务器出现连接故障的时间。如果涉及大量数据，则处理一条命令可能需要相当长的时间，具体取决于数据库和基础设施。

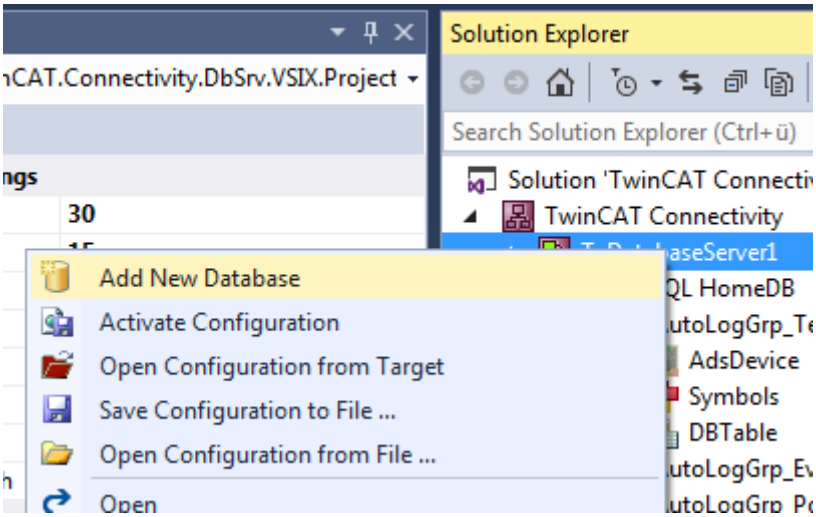
支持的数据库类型



在服务器设置中可以选择已安装的数据库类型。默认情况下会选择所有已安装的数据库。TwinCAT 3 数据库服务器将加载相应的数据库接口。这样可以取消选择目标系统上未使用的数据库。

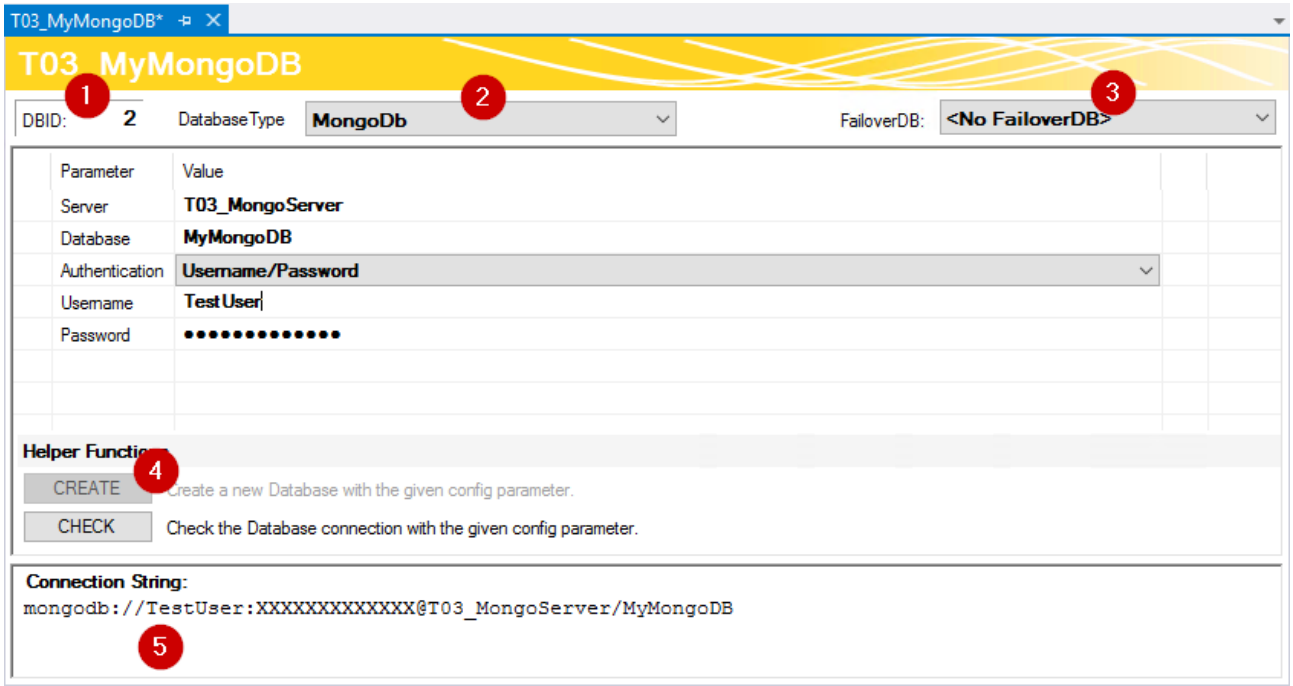
添加数据库配置

通过数据库服务器项目的上下文菜单中的 **Add New Database**（添加新数据库）命令或工具栏中的相应命令可以添加新的数据库配置。



新的数据库配置以文件的形式被添加到项目文件夹中，并集成到项目中。与所有 Visual Studio 项目一样，新文件的信息存储在 Connectivity 项目中。

数据库配置编辑器



PLC 中的一些功能块所需的数据库 ID 会在编辑器的上部（1）显示。从下拉菜单（2）中可以选择目标数据库的数据库类型。另一个选项是数据库的 ODBC 接口，不过目前还不支持。请注意，根据数据库的不同，并非 TwinCAT 数据库服务器的所有功能都能得到保证。

您还可以选择所谓的故障转移数据库（3），在配置模式下遇到错误时就会触发故障转移数据库。在网络断开的情况下，该功能可以自动确保数据存储在其他地方，而不会丢失。

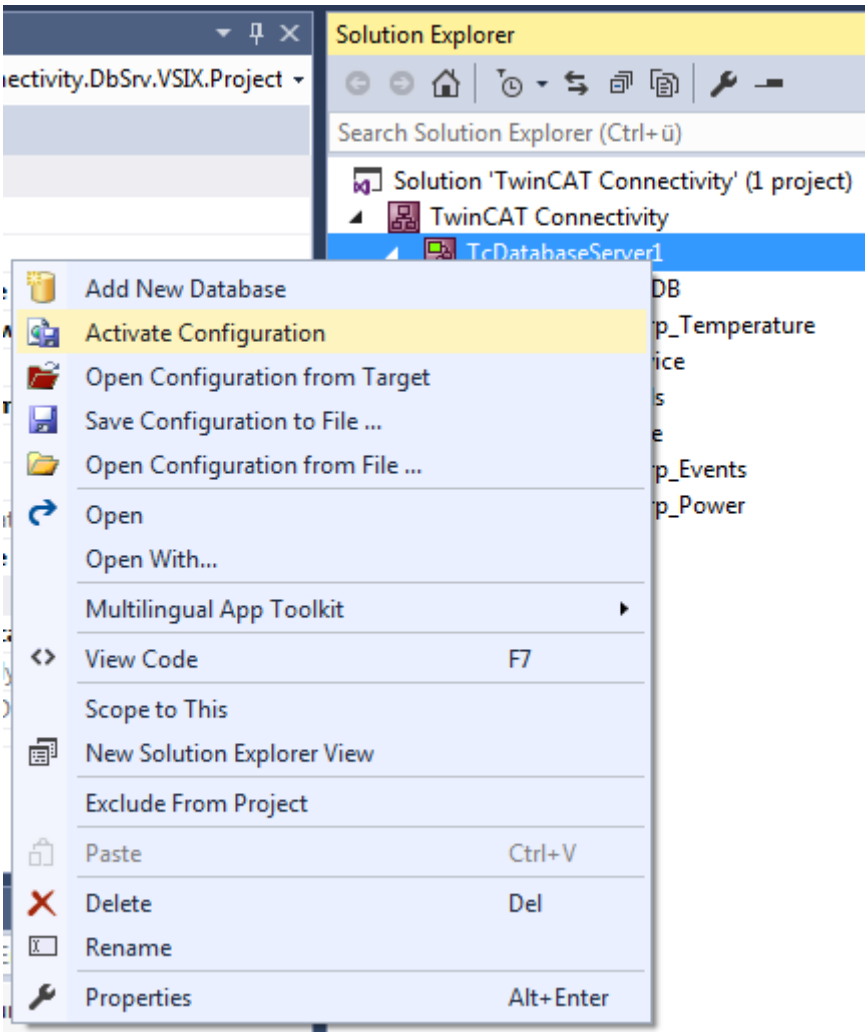
对于每个数据库 [▶ 120]，都有附加的可调整参数。根据数据库会创建一个连接字符串（5），用于描述与数据库的连接。这样旨在使您设置的参数更加透明。

CREATE（创建）（4）按钮可以用于创建新的数据库。只有在相关数据库支持该功能的情况下，才会显示它。**CHECK**（检查）可以用于检查与数据库的连接。

此外，也可以从目标数据库中选择 NoSQL 数据库。如果您使用 NoSQL 数据库激活项目，则会在 SQL 查询编辑器中启用 NoSQL 选项卡，这样便于使用 NoSQL 特定的功能。

激活项目

要在 TwinCAT 数据库服务器上激活一个已配置的项目，可使用 TwinCAT 数据库服务器项目的上下文菜单中的 **Activate Configuration**（激活配置）命令。



PLC 专家模式/配置模式下的 MongoDB

PLC 专家和配置模式会在其流程中使用预定义的数据库模式。通常情况下，所使用结构的模式在运行期间不会改变。不过，为了能够使用功能组件，TwinCAT 3 数据库服务器需要对表格架构进行描述。因此会为 MongoDB 模拟一个表格。
在 SQL 查询编辑器中，使用 **SQL** 选项卡和 **CREATE TABLE**（创建表格）子类别可以创建一个表格，或者如本示例中创建一个集合。此外，与关系型数据库不同的是，在元数据集合中会创建一个条目。在这里会存储有关 TwinCAT 3 数据库服务器的表格架构的信息。

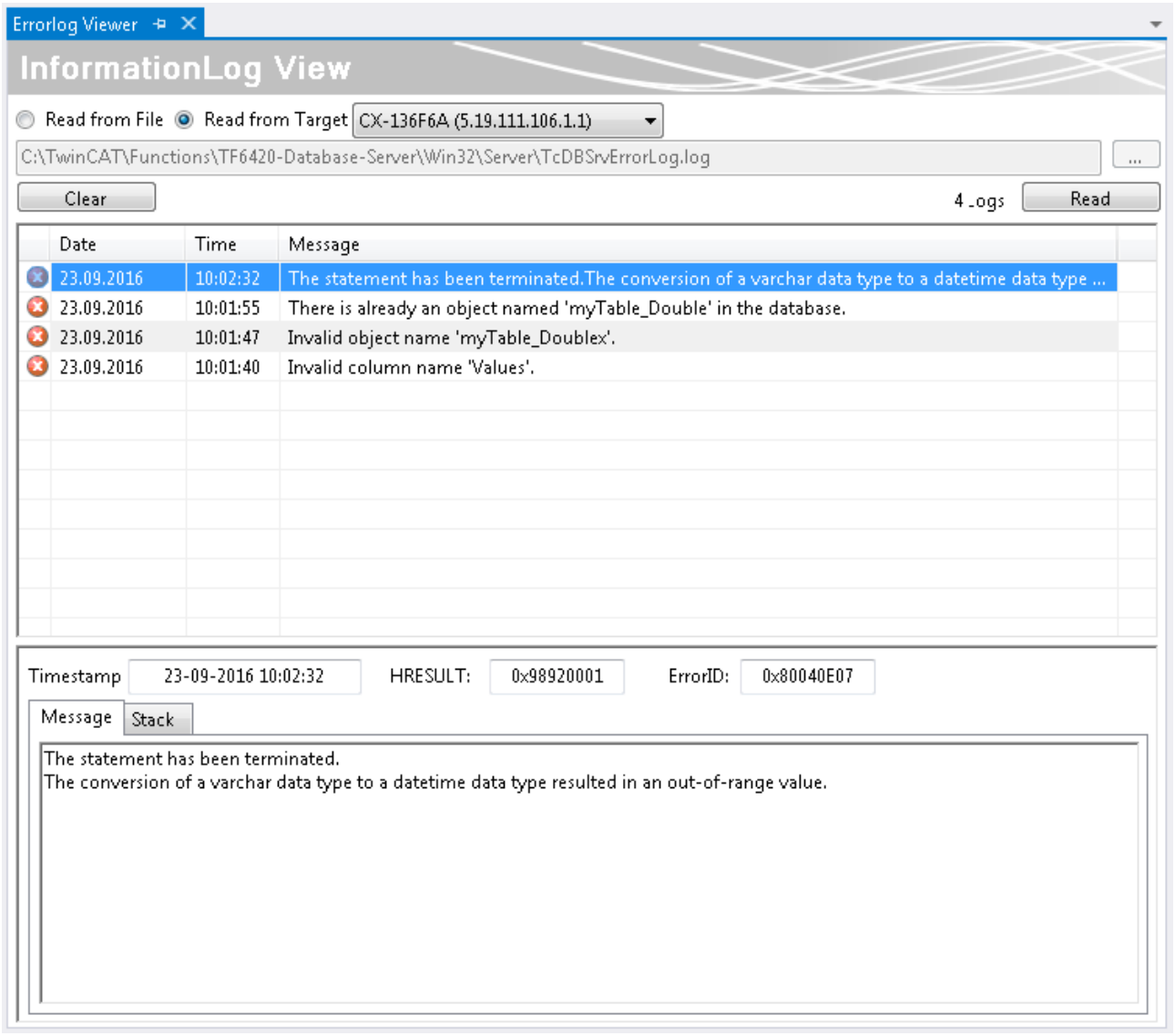
为了使用高级功能（例如，任何层次结构或灵活记录），我们建议使用 NoSQL 功能块。

用于诊断的 InformationLog 视图

InformationLog 视图可用于故障排除。

InformationLog View是一种从 TwinCAT 数据库服务器读取日志文件的工具。记录的信息会以纯文本的形式显示，包含时间戳、ID 和错误消息。

不仅可以直接通过文件访问功能来查看或清空日志文件，还可以直接通过目标系统实现这些操作。这在同一网络下有分布式数据库服务器时特别有优势，可以借此快速、方便地访问日志文件。要进行这种访问，必须有一条连接目标设备的路由。

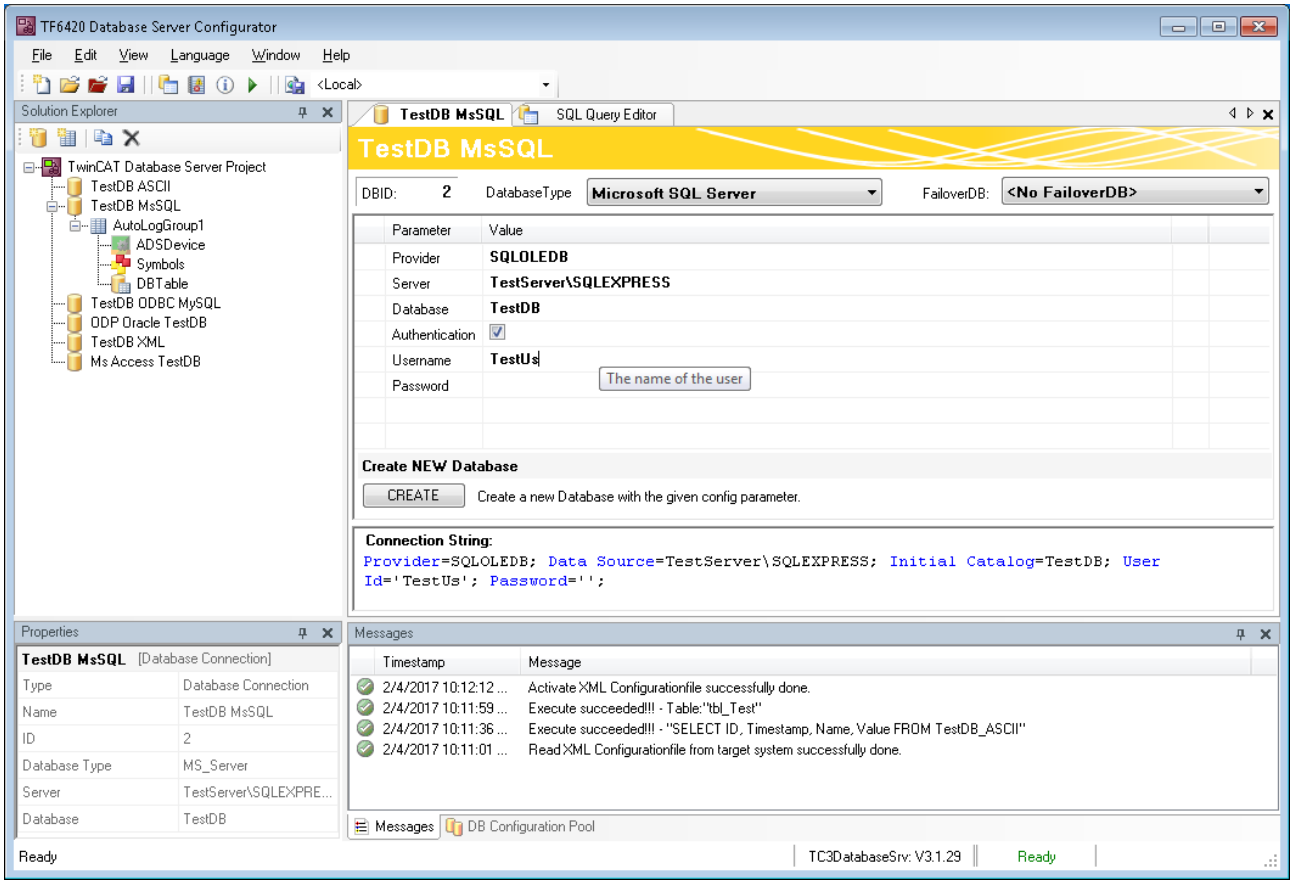


5.1.2 独立配置器

5.1.2.1 基本信息

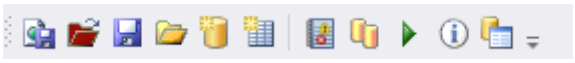
5.1.2.1.1 界面和基本功能

TwinCAT 3 数据库服务器是通过 XML 配置文件进行的配置 。
借助 XML 配置文件编辑器可以轻松地创建和修改配置文件中的设置。您可以创建新的配置文件，也可以读取和修改现有的配置文件。



工具栏和命令

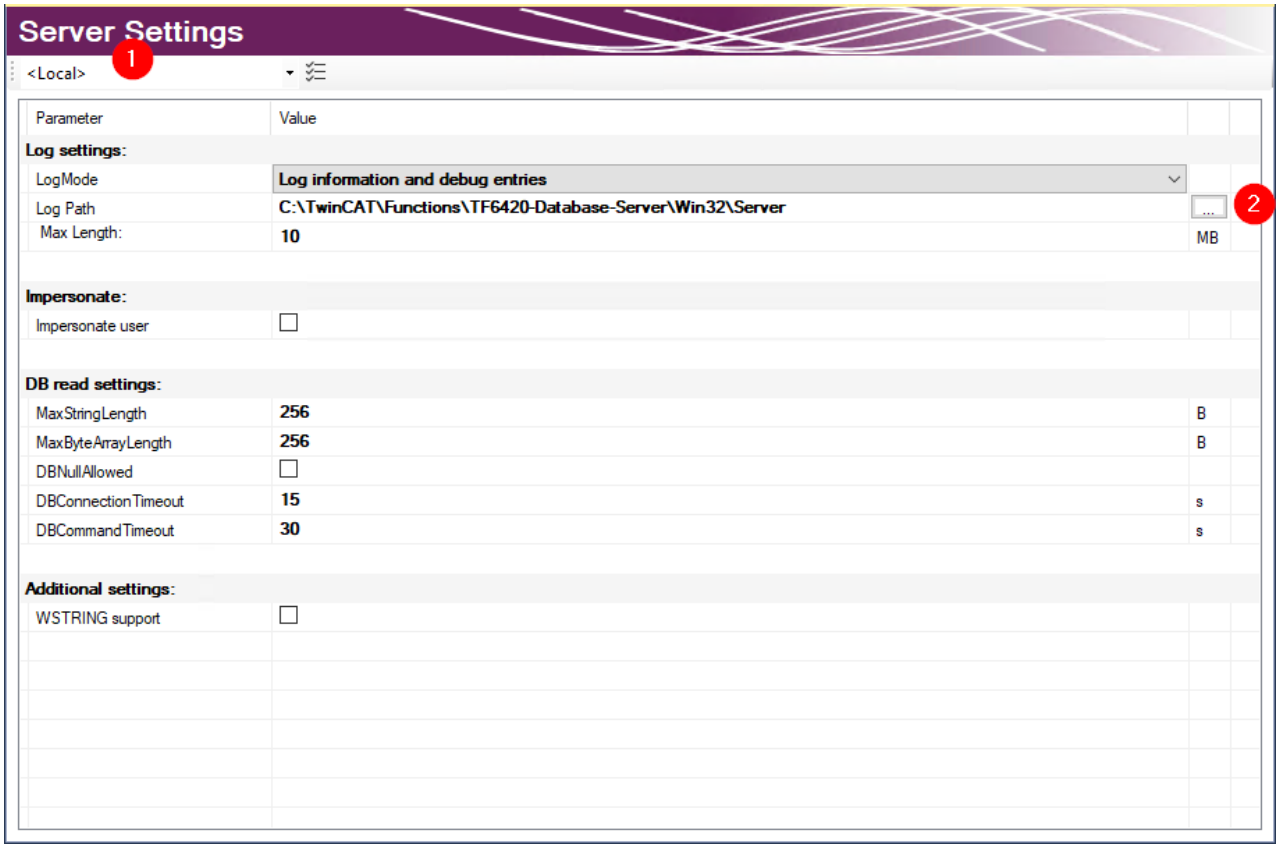
工具栏具有以下元素：



工具栏按钮	描述
	激活配置
	读取目标设备的配置
	在 XML 文件中保存配置
	从 XML 文件读取配置
	添加新的数据库配置
	添加新的 AutoLog 组
	事件显示
	数据库池
	AutoLog Viewer
	InformationLog View
	SQL 查询编辑器

5.1.2.1.2 项目属性

服务器设置的编辑器



服务器设置编辑器可以用于编辑 TwinCAT 数据库服务器的设置。这些是与相应的服务器相关的常规设置。在下拉菜单（1）中，您可以通过 Ams NetID 选择目标系统。为此，您必须通过 TwinCAT 创建一条通往目标系统的路由。在传输已完成的配置时，设置将存储在该目标系统的 TwinCAT 数据库服务器中。

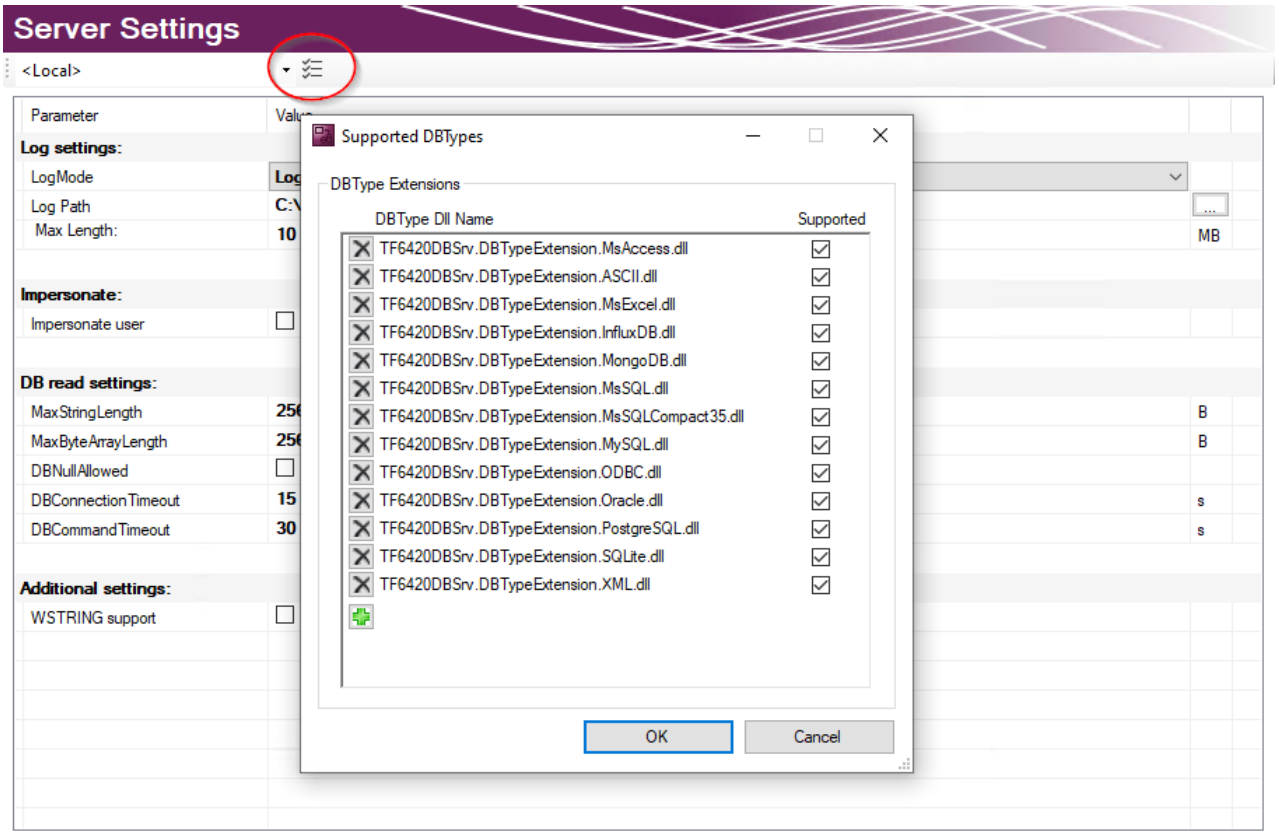
在 **Log settings**（日志设置）下可以配置关于记录故障或错误的设置。在发生故障或错误时，数据库服务器会在日志文件中生成一个详细的条目。通过信息日志查看器 [▶ 48]可以读取日志文件。在 **Log Settings**（日志设置）下，您可以指定文件位置的路径和最大文件大小。您还可以影响日志的准确性。出于性能方面的考虑，我们建议在不再需要的情况下，在错误分析之后再次停用日志记录。

对于基于文件的数据库（例如 Access 或 SQL Compact）的网络访问，必须设置 **Impersonate**（模拟）选项，以便 TwinCAT 数据库服务器可以与该网络驱动器连接。**Windows CE 目前不支持该功能。**

可以使用进一步的配置设置来控制从数据库读取数据的过程。这些设置指的是目标系统上的 TwinCAT 数据库服务器：

MaxStringLength	PLC 中的变量的最大字符串长度
MaxByteArrayLength	PLC 中的变量的最大字节数组长度
DBNullAllowed	表示 TwinCAT 数据库服务器是否接受零值。
DBConnectionTimeout	表示在尝试建立连接时，TwinCAT 数据库服务器出现连接错误的时间。
DBCommandTimeout	表示在发送命令时，TwinCAT 数据库服务器出现连接故障的时间。如果涉及大量数据，则处理一条命令可能需要相当长的时间，具体取决于数据库和基础设施。

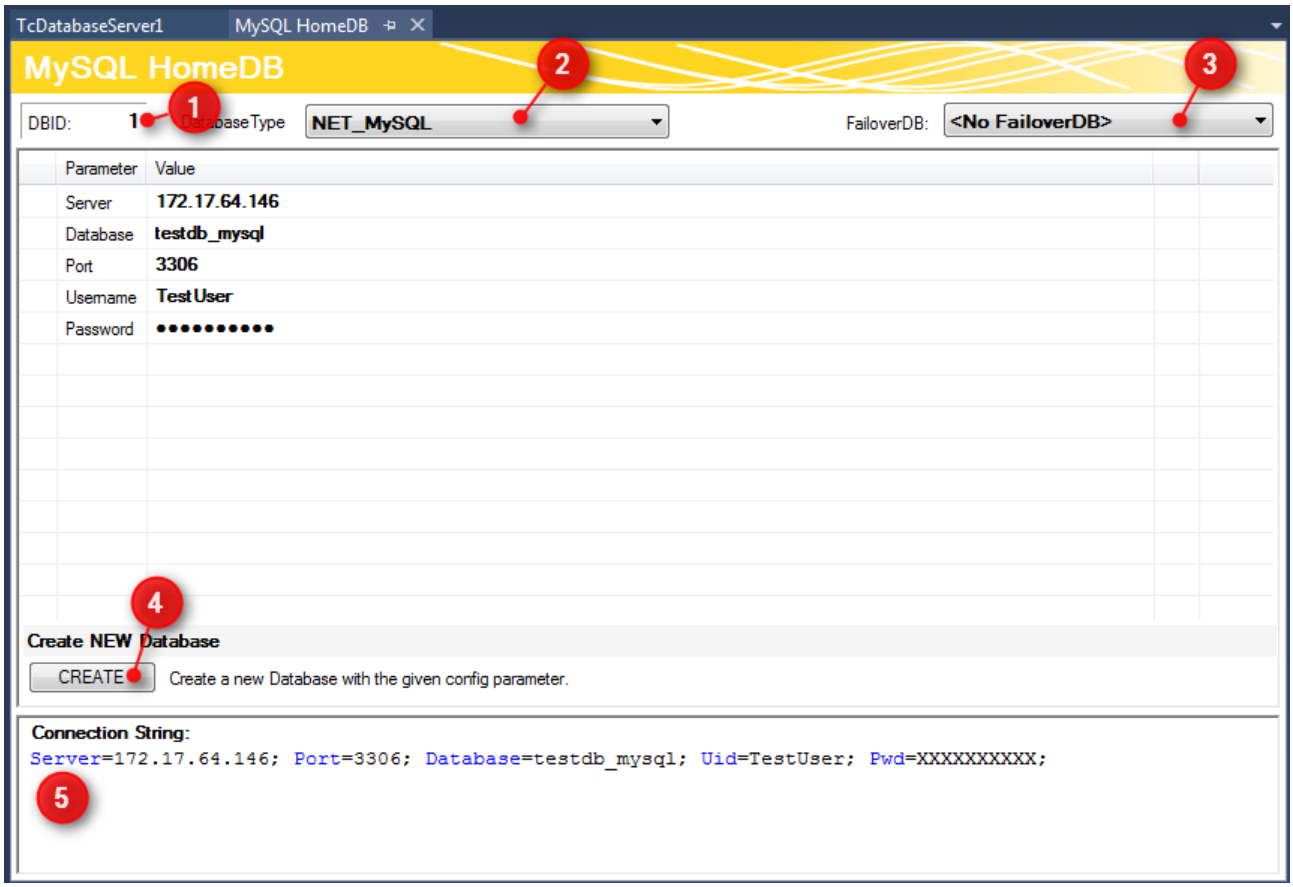
支持的数据库类型



在服务器设置中可以选择已安装的数据库类型。默认情况下会选择所有已安装的数据库。TwinCAT 3 数据库服务器将加载相应的数据库接口。这样可以取消选择目标系统上未使用的数据库。

5.1.2.1.3 配置数据库

数据库配置编辑器

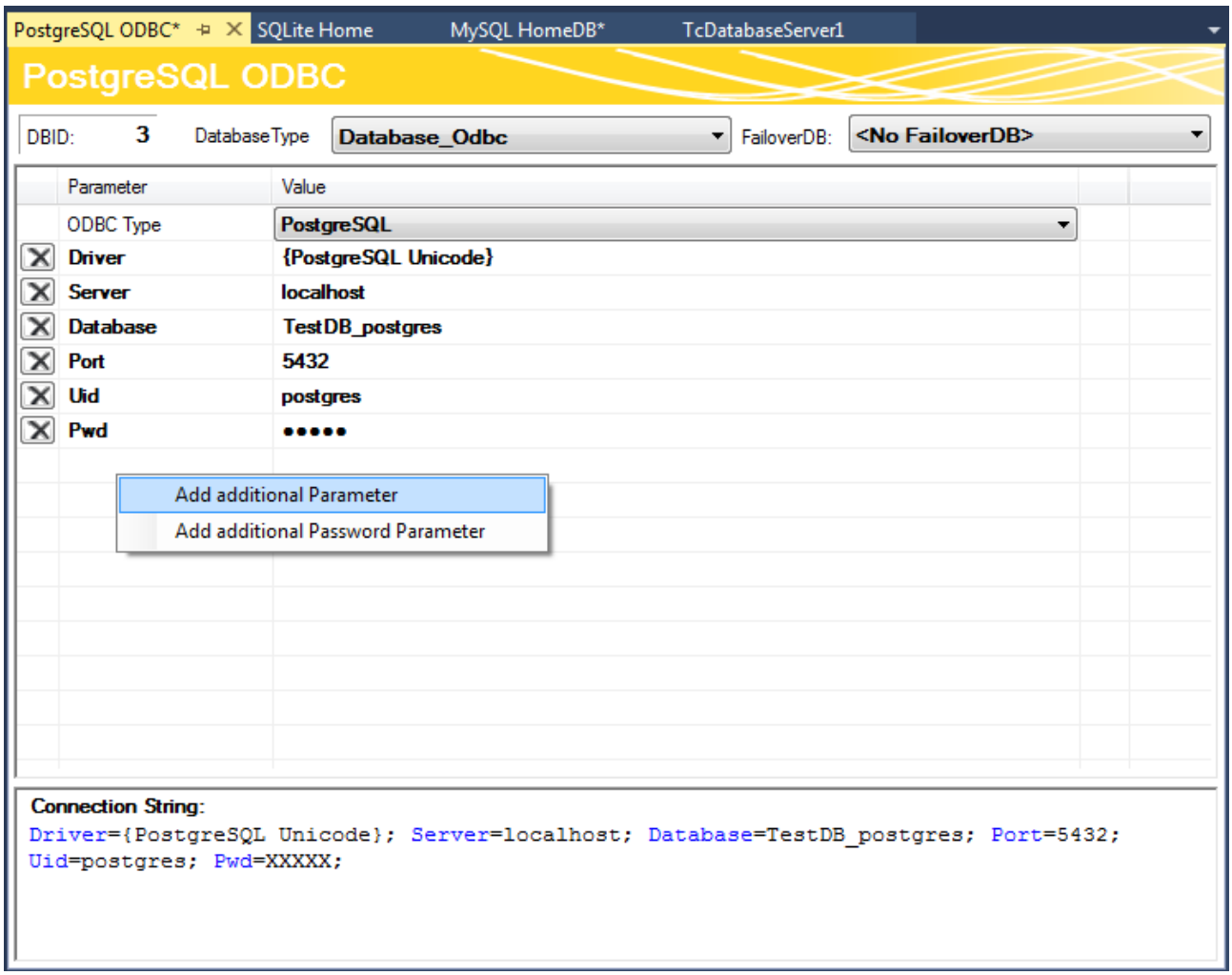


PLC 中的一些功能块所需的数据库 ID 会在编辑器的上部（1）显示。从下拉菜单（2）中可以选择目标数据库的数据库类型。另一个选项是数据库的 ODBC 接口，不过目前还不支持。请注意，根据数据库的不同，并非 TwinCAT 数据库服务器的所有功能都能得到保证。

您还可以选择所谓的故障转移数据库（3），在配置模式下遇到错误时就会触发故障转移数据库。在网络断开的情况下，该功能可以自动确保数据存储在其他地方，而不会丢失。

对于每个数据库 [▶ 120]，都有附加的可调整参数。根据数据库会创建一个连接字符串（5），用于描述与数据库的连接。这样旨在使您设置的参数更加透明。

CREATE（创建）（4）按钮可以用于创建新的数据库。只有在相关数据库支持该功能的情况下，才会显示它。

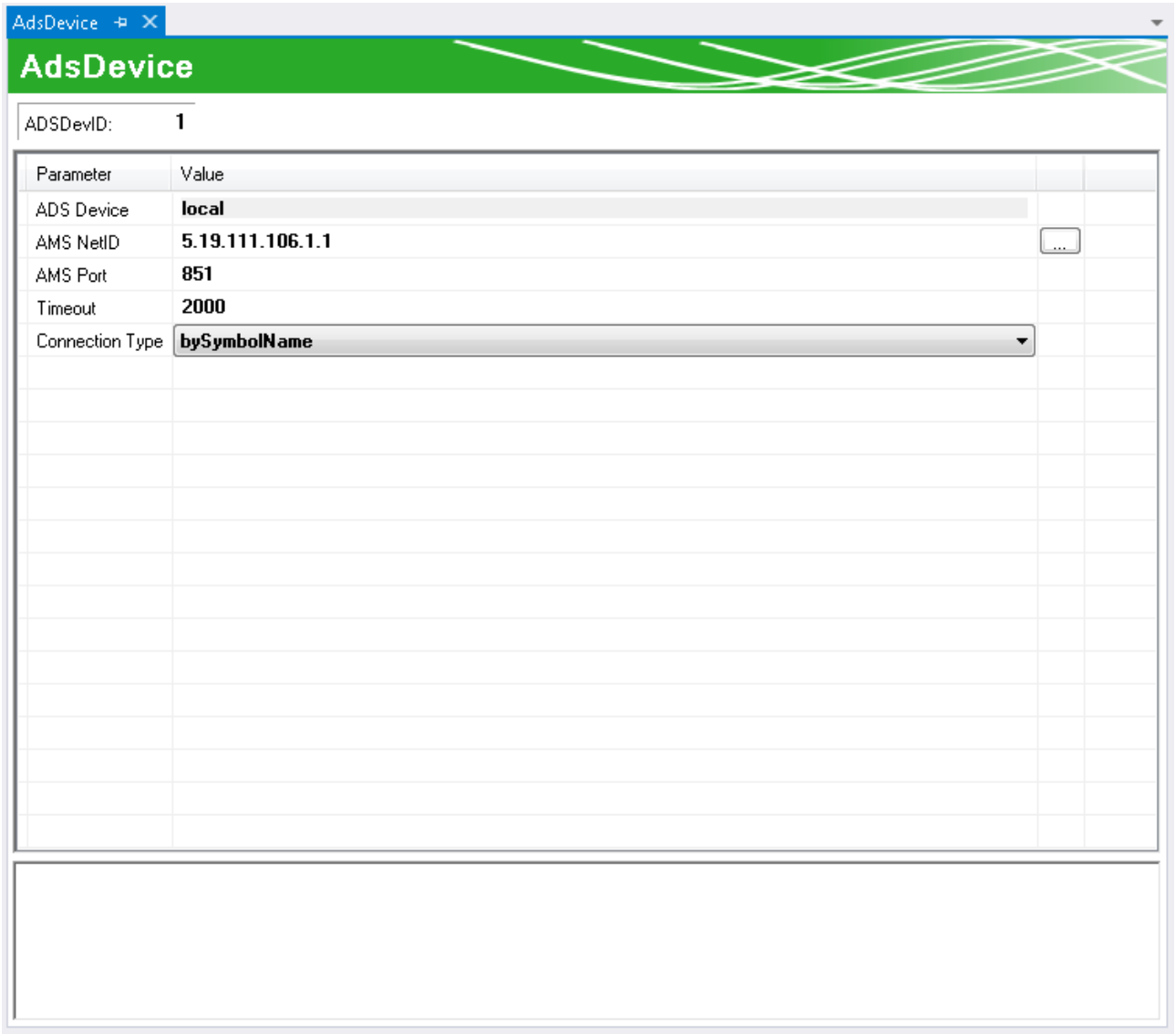


通过 ODBC 接口可以配置未知数据库。在 **ODBC Type**（ODBC 类型）下拉列表中，选择“Unknown Database”（未知数据库），然后通过上下文菜单中的命令添加参数。它们可能包含以加密形式存储的密码。利用这些参数可以组合成所需的连接字符串。请注意，只能使用 TwinCAT 数据库服务器的有限功能。仅支持 SQL 专家模式的显式功能块。

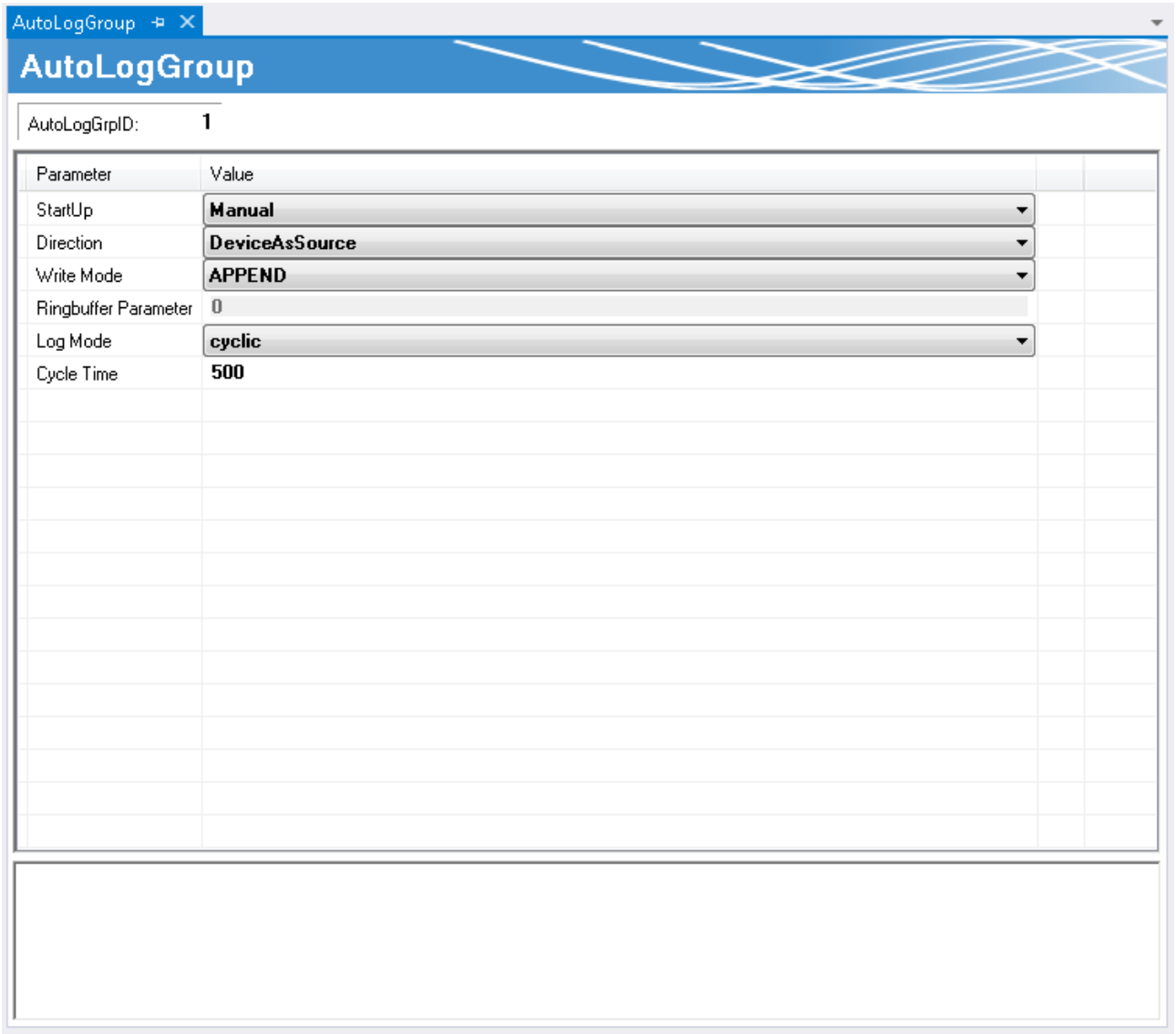
5.1.2.1.4 配置 AutoLog 组

配置 ADS 设备

在 AutoLog 组下会自动创建 ADS 设备。在最常见的用例中，ADS 设备为 PLC Runtime。在编辑器中可以设置以下参数：



ADS 设备	ADS 目标设备的名称。
AMS NetID	TwinCAT 网络中的目标设备的地址。
AMS 端口	TwinCAT 网络中的目标设备的端口。
超时	与目标设备失去连接的时间。
连接类型	bySymbolName：根据符号名称建立连接。 byIndexGroup：根据内存索引建立连接。

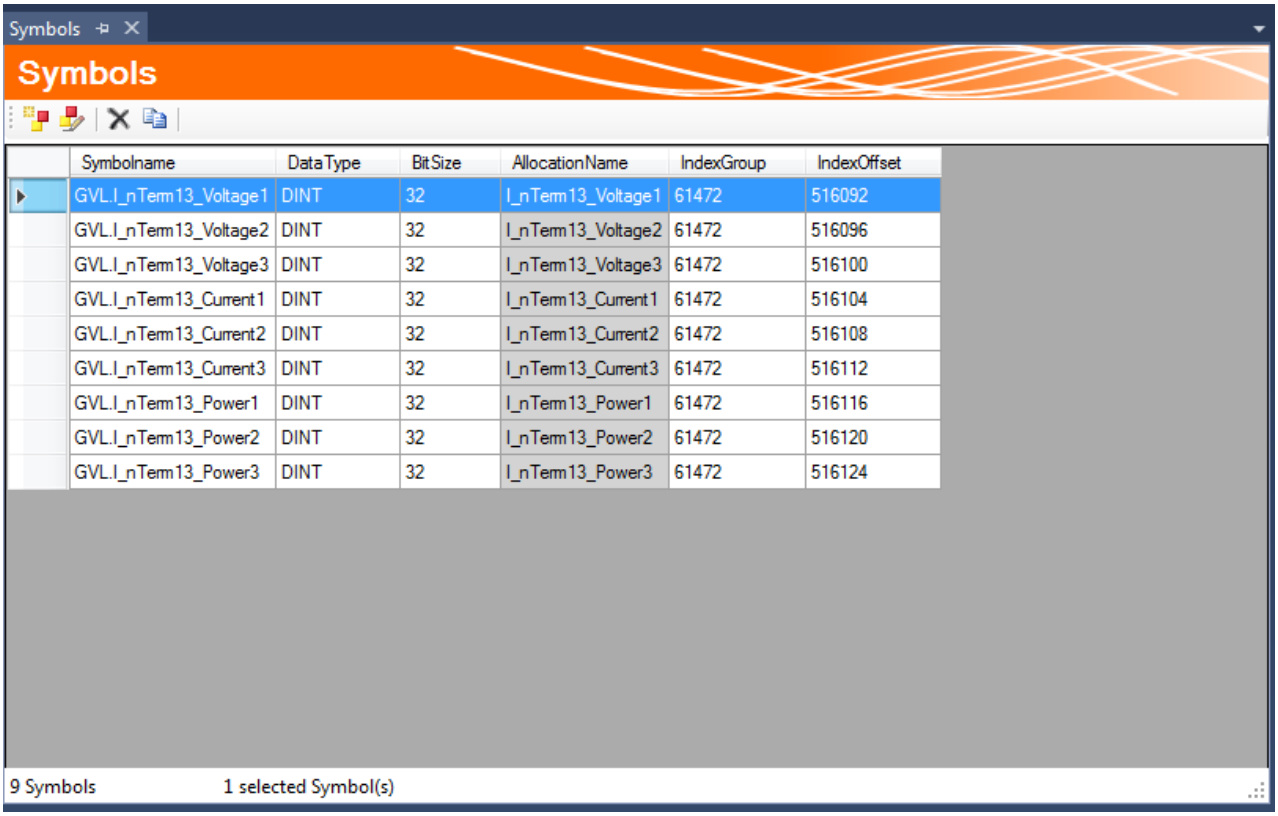


启动	您可以手动启用 AutoLog 模式（通过 PLC 或配置器中的命令），或者在系统启动时自动启用该模式。
方向	设定的 ADS 设备可用作数据目标或数据源。
写入模式	这些数据可以逐行添加到数据库中，也可以按时间或数量保存在环形缓冲区中，或者只是在相应的位置进行更新。
环形缓冲区参数	根据设置的不同，该参数可表示环形缓冲区更新的时间或周期。
日志模式	在经过特定的循环时间后或在发生变化时，就会写入变量。
周期时间	写入变量后的周期时间。

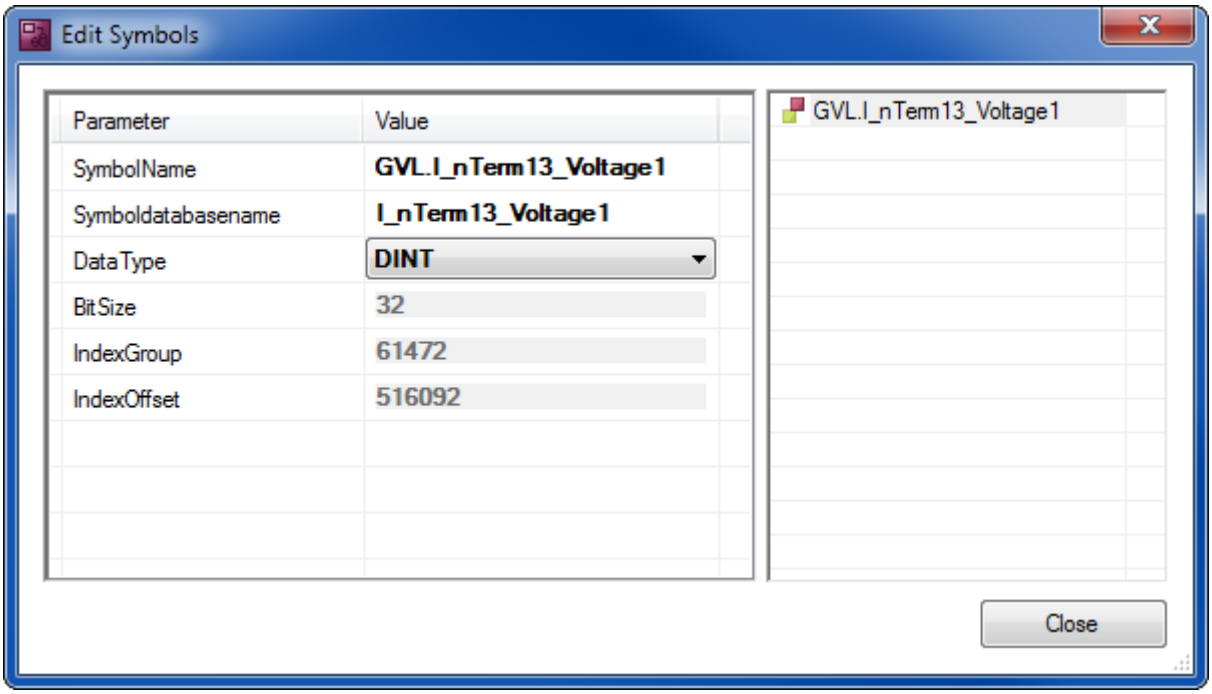
通过数据库配置的上下文菜单中的 **Add New AutologGroup**（添加新的 Autolog 组）命令或工具栏可以为数据库配置添加新的 AutoLog 组。这些 AutoLog 组指的是父级数据库。

配置符号

您在这里设置的符号会被写入数据库或从数据库读取，具体取决于 ADS 设备是数据目标还是数据源。TwinCAT Target Browser [▶ 49]可以用于便捷访问。在这里，您可以搜索目标系统上的符号，并通过拖放操作在 2 个工具之间进行通信。



您还可以将符号手动添加到符号组或进行编辑。所需的信息各不相同，具体取决于在 ADS 设备中是通过符号名称还是通过索引组来选择连接类型。起始点始终是 ADS 设备。



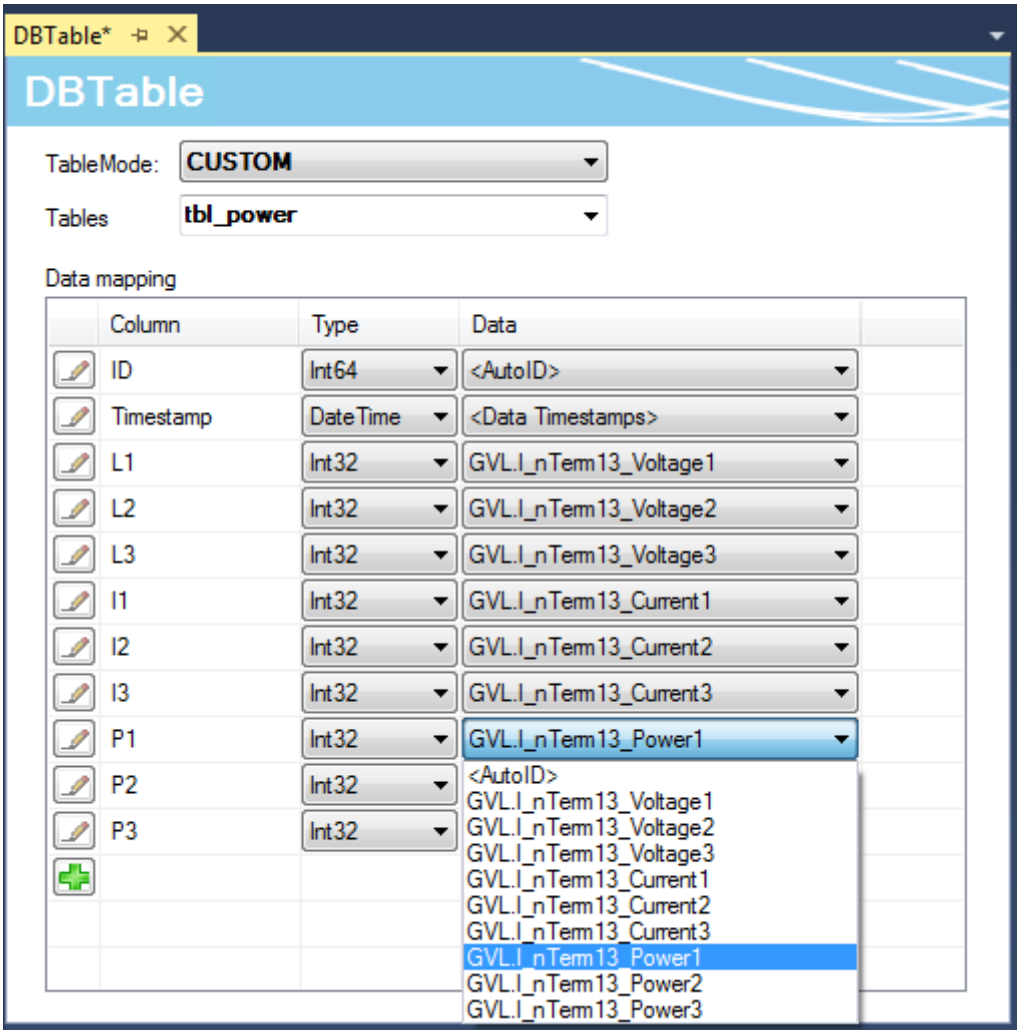
SymbolName	根据设置的 ADS 设备，对符号进行寻址
符号数据库名称	数据库表中的变量的名称
DataType	符号的 PLC 数据类型
BitSize	符号的位大小（为数据类型自动设置）
IndexGroup	TwinCAT 系统中的索引组
IndexOffset	TwinCAT 系统中的索引偏移

配置表格

数据库中的表格可以基于标准表结构，也可以基于个性化结构。

从可能的表格列表中可以选择相应的表格。如果表格尚不存在，您可以通过 SQL 查询编辑器创建它。如果您选择标准表结构，则蓝色勾号表示选定的表格是否与该结构相对应。

特定的表格类型提供了根据需要将在符号组中设置的各个符号分配到数据库中的表格列的选项。在 AutoLog 模式下将数据集写入数据库时，采样时符号组的当前值将保存在相应的表格列中。



5.1.写入方向模式
2.1.
4.1

TwinCAT 数据库服务器有 4 种不同的写入方向模式。下文将对此进行解释。

DB_TO_ADS

该写入模式用于循环读取数据库中的变量值，并将读取的值写入 PLC 变量。

ADS_TO_DB_APPEND

该写入模式可用于将 PLC 中的变量值循环写入数据库。每次都会创建一个新的记录，并将其添加到表格/文件的末尾。

ADS_TO_DB_UPDATE

该写入模式可用于循环读取 PLC 中的变量值，并将读取值与数据库记录进行比较。如果发现差异，则会使用新值修改相应的记录。

ADS_TO_DB_RINGBUFFER

该写入模式可以用于指定记录的数量或记录的存留时长。
在通过符号组进行循环日志记录期间，以及在使用功能块 FB_DBWrite 进行日志记录期间，都可以使用该写入模式。
RingBuffer 模式适用于所有数据库类型。该模式还可以用于影响 ASCII 文件中的日志记录。

RingBuffer-Arten

通过 2 种不同的方式可以使用 RingBuffer：

- “RingBuffer_Time”
- “RingBuffer_Count”

RingBuffer Time

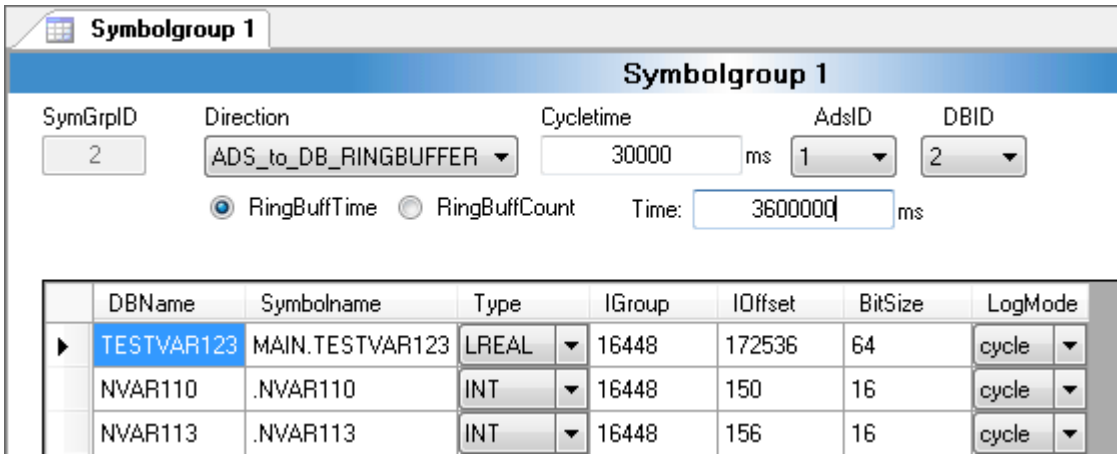
在该模式下，可以为记录的最大期限指定一个时间。如果超过该期限，则会删除相应的记录。

RingBuffer Count

在该模式下，可以指定记录的最大数量。当达到最大数量时，将会删除最旧的记录，以便存储新记录。

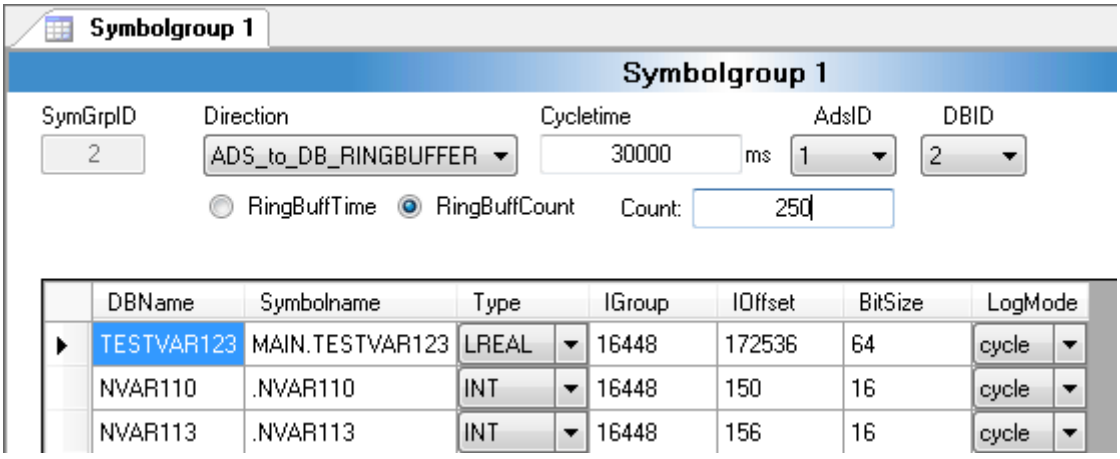
在 XML 配置文件编辑器中声明 RingBuffer 模式

RingBuffer_Time：



以毫秒为单位指定时间。

RingBuffer_Count：



在 FB_DBWrite 中声明 RingBuffer 模式：

RingBuffer_Time：

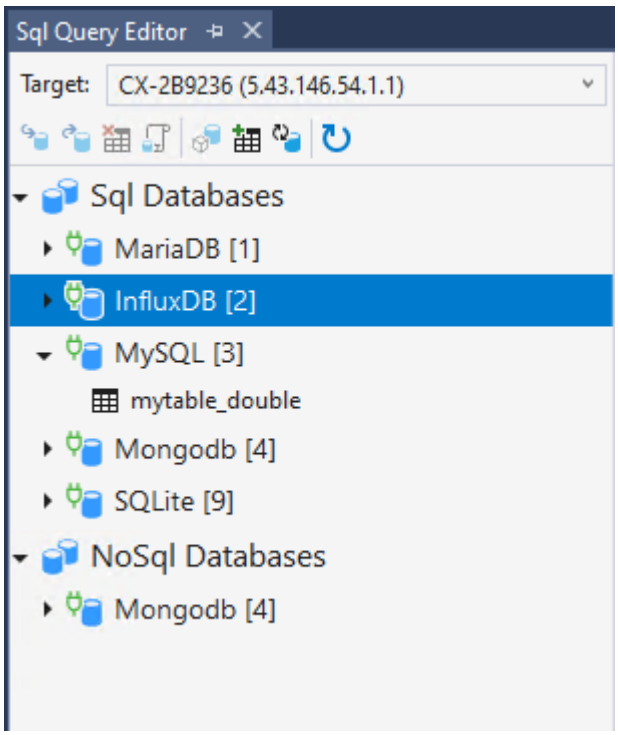
```
fbDBWrite(
    sNetID:= '',
    hDBID:= 2,
    hAdsID:= 1,
    sVarName:= 'MAIN.TESTVAR123',
    nIGroup:= ,
    nIOffset:= ,
    nVarSize:= ,
    sVarType:= ,
    sDBVarName:= 'TESTVAR123',
    eDBWriteMode:= eDBWriteMode_RingBuffer_Time,
    tRingBufferTime:= 3600000,
    nRingBufferCount:= ,
    bExecute:= TRUE,
    tTimeout:= T#15S,
    bBusy=> bBusy,
    bError=> bErr,
    nErrID=> nErrID,
    sSQLState=> stSQLState);
```

RingBuffer_Count:

```
fbDBWrite(
    sNetID:= '',
    hDBID:= 2,
    hAdsID:= 1,
    sVarName:= 'MAIN.TESTVAR123',
    nIGroup:= ,
    nIOffset:= ,
    nVarSize:= ,
    sVarType:= ,
    sDBVarName:= 'TESTVAR123',
    eDBWriteMode:= eDBWriteMode_RingBuffer_Count,
    tRingBufferTime:= ,
    nRingBufferCount:= 250,
    bExecute:= TRUE,
    tTimeout:= T#15S,
    bBusy=> bBusy,
    bError=> bErr,
    nErrID=> nErrID,
    sSQLState=> stSQLState);
```

5.1.2.1.5 SQL 查询编辑器

SQL 查询编辑器是一种数据库服务器工具，为您的应用程序开发提供支持。该工具可以用于测试连接状态和 SQL 命令，并检查 PLC 和数据库之间的兼容性。



在选择目标系统的 TwinCAT 数据库服务器之后，SQL 查询编辑器将加载当前的数据库配置以及已成功连接的数据库的表格。根据数据库是否支持 SQL 和 NoSQL 接口（来自 TwinCAT 数据库服务器），它将在相应的类别下列出。

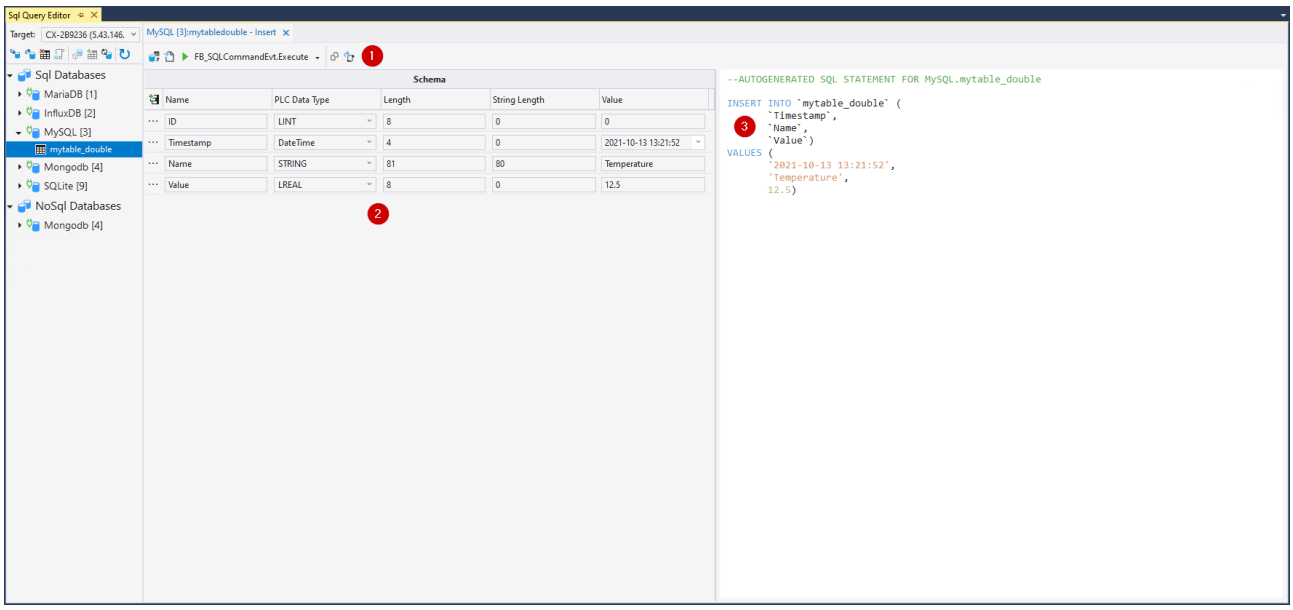
在选择目标系统下方，有一个带有可用命令的状态栏：

表格层	
插入工作区	打开插入工作区，使用 SQL 将数据集写入选定的表格。
选择工作区	打开选择工作区，使用 SQL 从选定的表格读取数据集。
删除/丢弃工作区	打开删除/丢弃工作区，使用 SQL 从选定的表格中删除数据集或删除整个表格。
NoSQL 工作区	打开 NoSQL 工作区，执行 NoSQL 特定的查询。
数据库层	
存储过程工作区	打开存储过程工作区，执行数据库的存储过程。
表格工作区	打开表格工作区，在选定的数据库中创建新的表格。
更新表格	更新选定数据库的可用表格。
一般信息	
更新数据库	更新整个数据库树。

在相应选项卡下的树形结构的右侧会打开工作区。此外，在同一个表格中还可以同时打开多个选项卡。

插入工作区

插入工作区允许通过 SQL 功能块的 TwinCAT 数据库服务器接口将数据写入选定的表格中。



在下部区域（2）有一个表格，其中列出了要写入的数据集中的各个数据符号。在这里可以确定名称、PLC 数据类型、字节长度以及值。然后，通过命令可以使用所输入的值生成 SQL 语句。

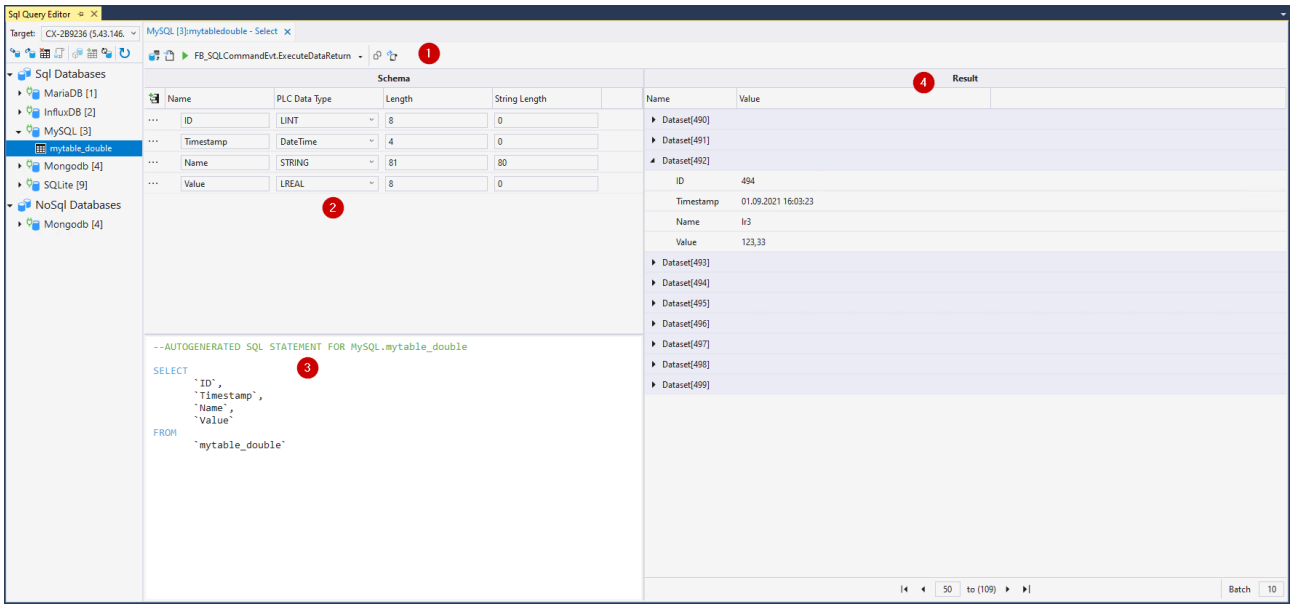
然后，该 SQL 语句就会在文本字段（3）中出现。根据数据库语法的不同，内容可能会有所差异。

上部状态栏包含与 TwinCAT 数据库服务器交互的命令（1）。

命令	描述
读取表格结构	读出工作区的表格的表格结构。
生成 SQL 语句	根据数据库语法的不同，从当前表格生成 SQL 语句。
执行	通过 TwinCAT 数据库服务器的相应接口执行文本字段（3）中的语句。
复制语句	将文本字段（3）中的语句复制为与 TwinCAT 兼容的语法。
导出为结构	将输入值表格的结构导出到与 TwinCAT 3 兼容的 DUT 中。

选择工作区

选择工作区允许通过 TwinCAT 数据库服务器接口提供的 SQL 功能块将数据读取到选定的表中。



在下部区域（2）有一个表格，其中列出了要写入读取的数据集中的各个数据符号。在这里可以确定名称、PLC 数据类型以及字节长度。然后，需要这些信息来解释数据。

然后，该 SQL 语句就会在文本字段（3）中出现。根据数据库语法的不同，内容可能会有所差异。

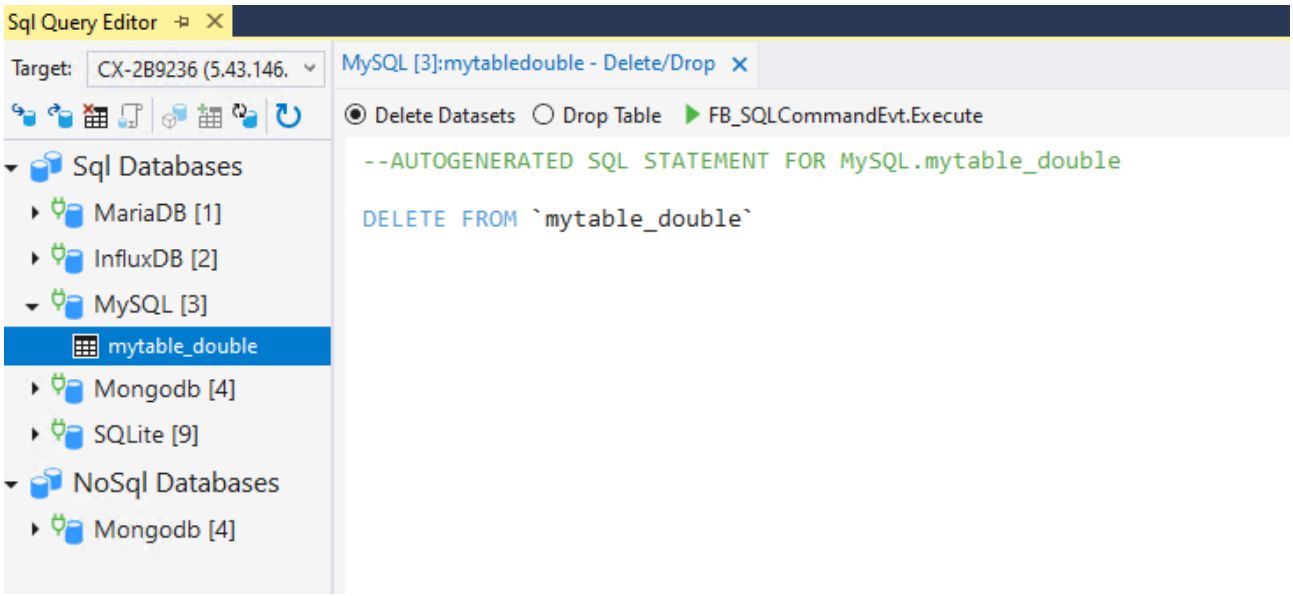
在执行语句后，结果字段（4）会显示数据。如果返回多个结果，则可以在页面中进行切换。

上部状态栏包含与 TwinCAT 数据库服务器交互的命令（1）。

命令	描述
读取表格结构	读出工作区的表格的表格结构。
生成 SQL 语句	根据数据库语法的不同，从当前表格生成 SQL 语句。
执行	通过 TwinCAT 数据库服务器的相应接口执行文本字段（3）中的语句。
复制语句	将文本字段（3）中的语句复制为与 TwinCAT 兼容的语法。
导出为结构	将输入值表格的结构导出到与 TwinCAT 3 兼容的 DUT 中。

删除/丢弃工作区

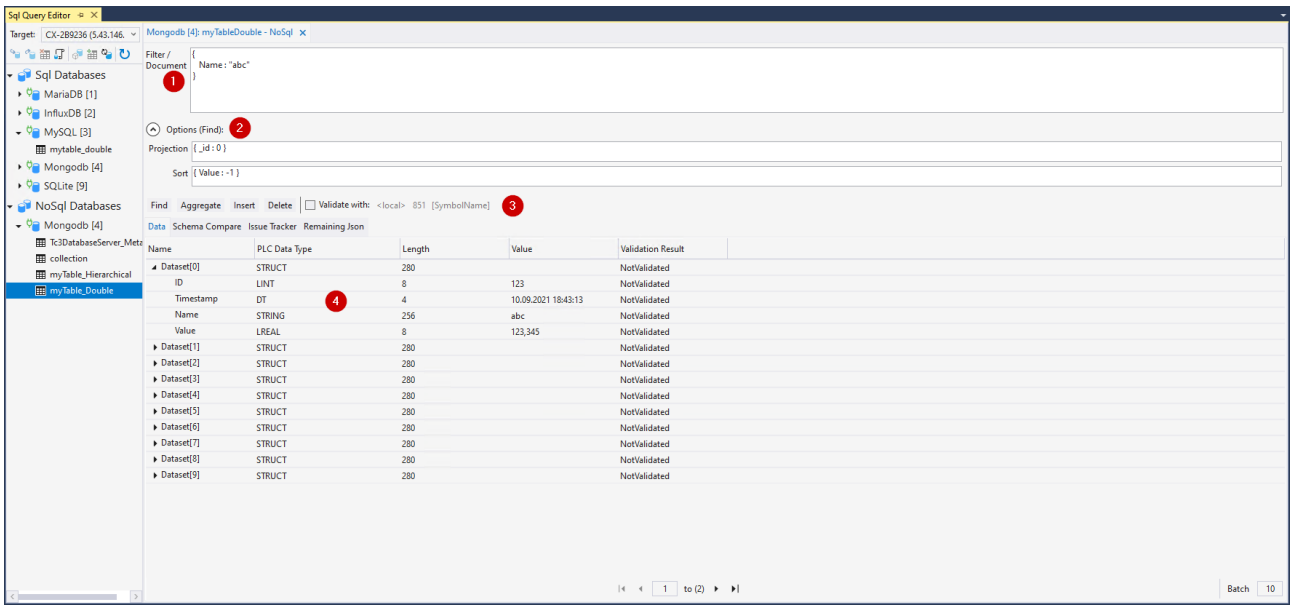
删除/丢弃工作区提供了发布 SQL 语句的选项，可从表格中删除数据或从数据库中删除整个表格。



为此，您可以在状态栏中选择这两个选项。然后，在语句字段中会生成与数据库相对应的语法。要使用 TwinCAT 数据库服务器接口执行此操作，可使用开关 **FB_SQLCommandEvt.Execute**。

NoSql 工作区

NoSql 工作区支持 NoSql 数据库或 TwinCAT 数据库服务器接口为 NoSQL 提供的特殊功能。



ID	Name	功能
1	筛选器/文档	根据所使用的功能的不同，该输入字段可充当文档或 JSON 格式的筛选器。如果您既要执行查找操作，又要执行投影或排序操作，您可以在下面的选项（查找）中填写这些字段。
2	选项（查找）	描述查找功能的其他参数，例如，投影或排序。
3	控制元素	用于与 NoSQL 的 TwinCAT 数据库服务器接口交互的控制元素。
4	数据显示	返回数据的列表。使用导航可以逐页浏览。

查找：使用在文本字段（1）中输入的筛选器执行搜索查询。或者，通过选项（查找）字段也可以执行投影或排序操作。数据将返回并在数据显示（4）中列出。筛选器的语法是数据库特定的。

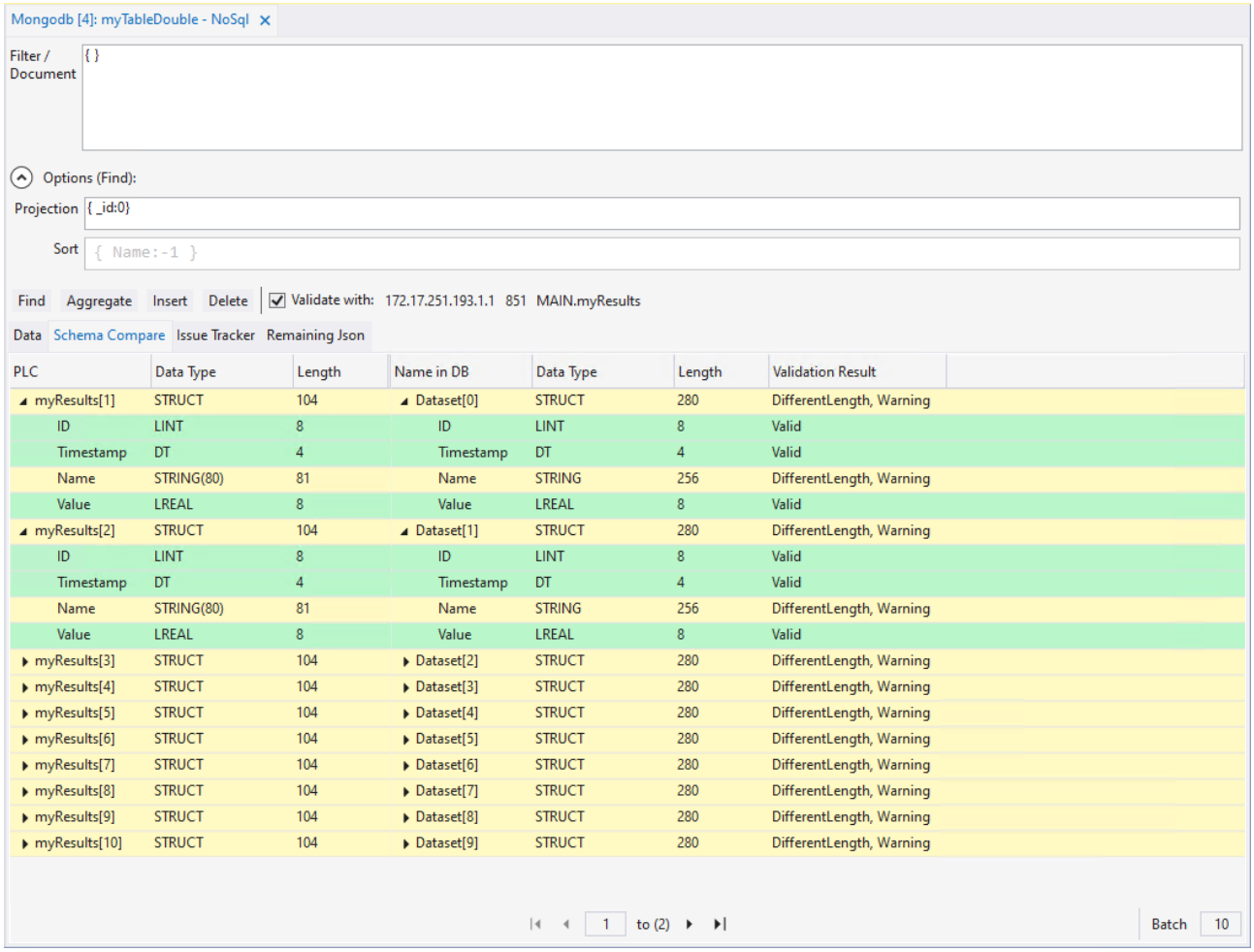
聚合：使用在文本字段（1）中输入的参数执行聚合。数据将返回并在数据显示（4）中列出。筛选器的语法是数据库特定的。

插入：执行在文本字段（1）中输入的（JSON）文档或文档数组的插入查询。然后将它们写入数据集。

删除：对使用文本字段（1）中的筛选器找到的数据执行删除查询。从数据集中删除任何找到的数据。

验证：如果选择该选项，则数据查询不会根据自己的结构自动进行解析，而会尝试将这些数据映射到通过这些参数指定的 PLC 中的符号结构。

使用后一种功能时，查找查询可能会导致冲突。与 PLC 过程映像中的结构相比，NoSQL 数据库中的数据集不必遵循固定的结构。查询的文档可能没有 PLC 结构中的特定元素的数据。或者数据集包含的数据没有在 PLC 结构中出现。在 PLC 中，这些数据通过名称或“ElementName”属性进行分配。



通过 **Schema Compare**（模式比较）选项卡可以检查数据中的差异。从上述示例中可以看出，在 PLC 结构中返回的文档中，变量“名称”的数据类型长度与数据库的数据类型长度不同。相应的颜色可显示冲突的权重：

红色：可用的数据过多或过少。

黄色：数据集的字节长度不匹配，或底层数据集剩余或缺失。

绿色：无冲突

这些冲突也在 **Issue Tracker**（问题跟踪）选项卡下列出。如果需要，也可以将它作为字符串数组读入 PLC。

Remaining Json（剩余 Json）选项卡以 JSON 形式返回任何剩余数据集。也可以将该信息作为字符串读入 PLC。

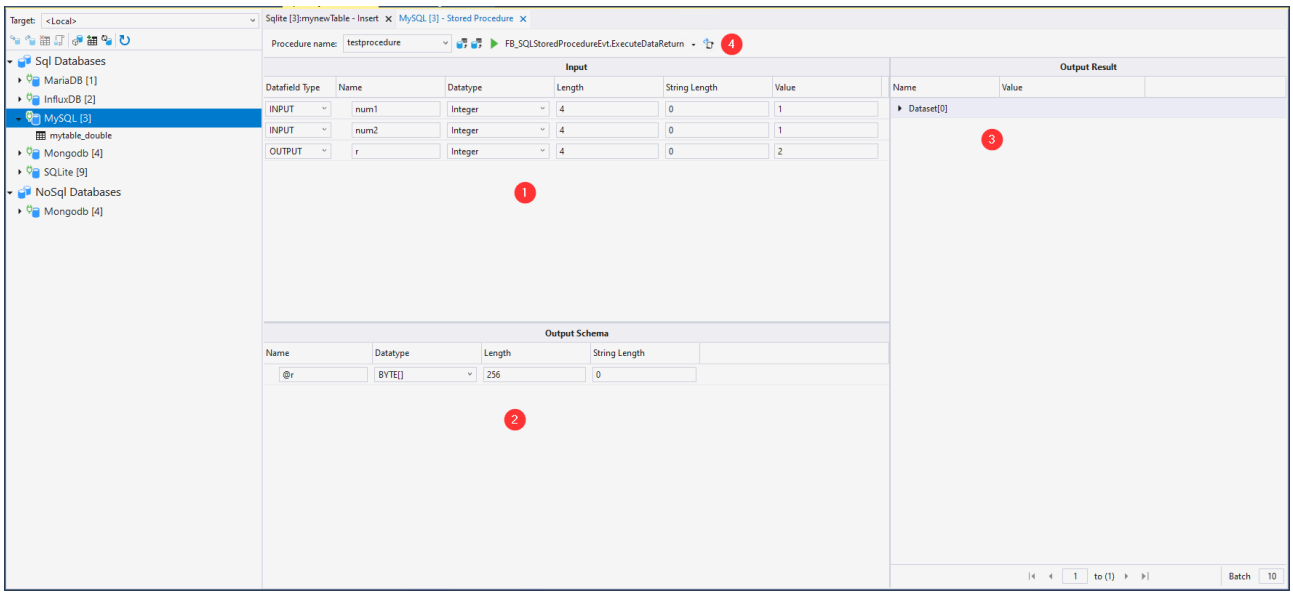
根据其他显示数据可知，状态栏中的控制元素可以用于遍历数据。可以指定同时显示的数据集的数量。

存储过程工作区

TwinCAT 数据库服务器支持“存储过程”，它支持大量的数据库，用于在数据库层处理更加复杂的查询或提供简化的接口。

如果数据库中存在**存储过程**，则它们将在状态栏（4）的下拉列表中列出。

下面是输入参数（1）和输出结构（2）的表格。此外，还有一个输出结果（3）的视图。如果成功执行**存储过程**，则会在这里显示结果。

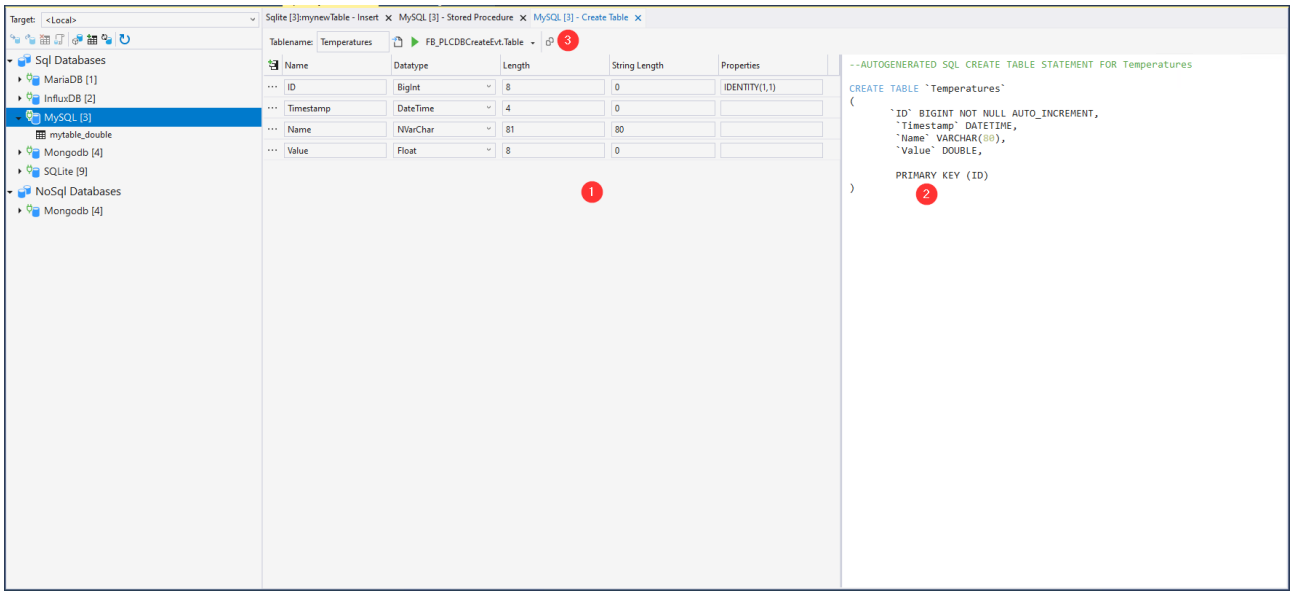


状态栏有以下命令：

命令	描述
读取存储过程的输入结构	读出输入参数结构。结果在表 1 中显示。
读取存储过程输出结构	读出输出参数结构。结果在表 2 中显示。**信息：** 这需要执行存储过程。根据编程的不同，在这里可以更改数据。
设计	通过 TwinCAT 数据库服务器的相应接口执行存储过程。
导出为结构	将表格的结构导出到与 TwinCAT 3 兼容的 DUT 中。

表格工作区

表格工作区可用于创建新的表格。

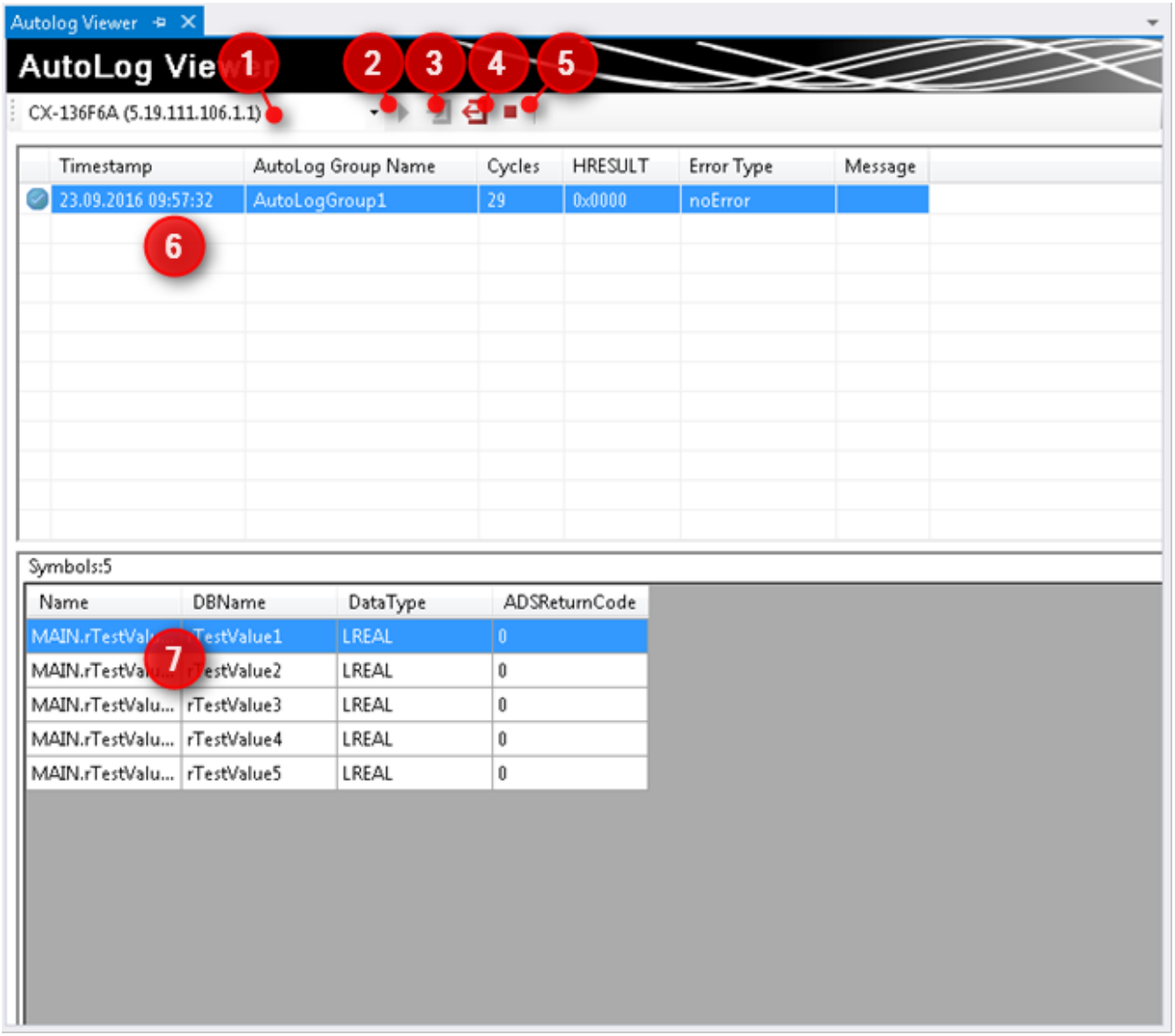


在这里可以创建表结构（1），并在相应的字段（2）中从其中生成 SQL 语句。为此，可以使用带有以下命令的状态栏（3）：

命令	描述
表格名称	指定新表格的表格名称。
生成 SQL 语句	根据数据库语法的不同，从当前表格生成 SQL 语句。
执行	通过 TwinCAT 数据库服务器的相应接口执行存储过程。
复制语句	将文本字段（2）中的语句复制为与 TwinCAT 兼容的语法。

5.1.2.1.6 AutoLog 实时视图

TwinCAT 数据库服务器的 AutoLog 查看器是一种用于控制和监控 AutoLog 模式的工具。您可以登录目标系统，这类似于 TwinCAT PLC。在登录状态下，可以启动或停止 AutoLog 模式。有关日志记录的当前状态的信息在窗口的下部显示。在选择 AutoLog 组时，通过记录的符号可显示更多信息。

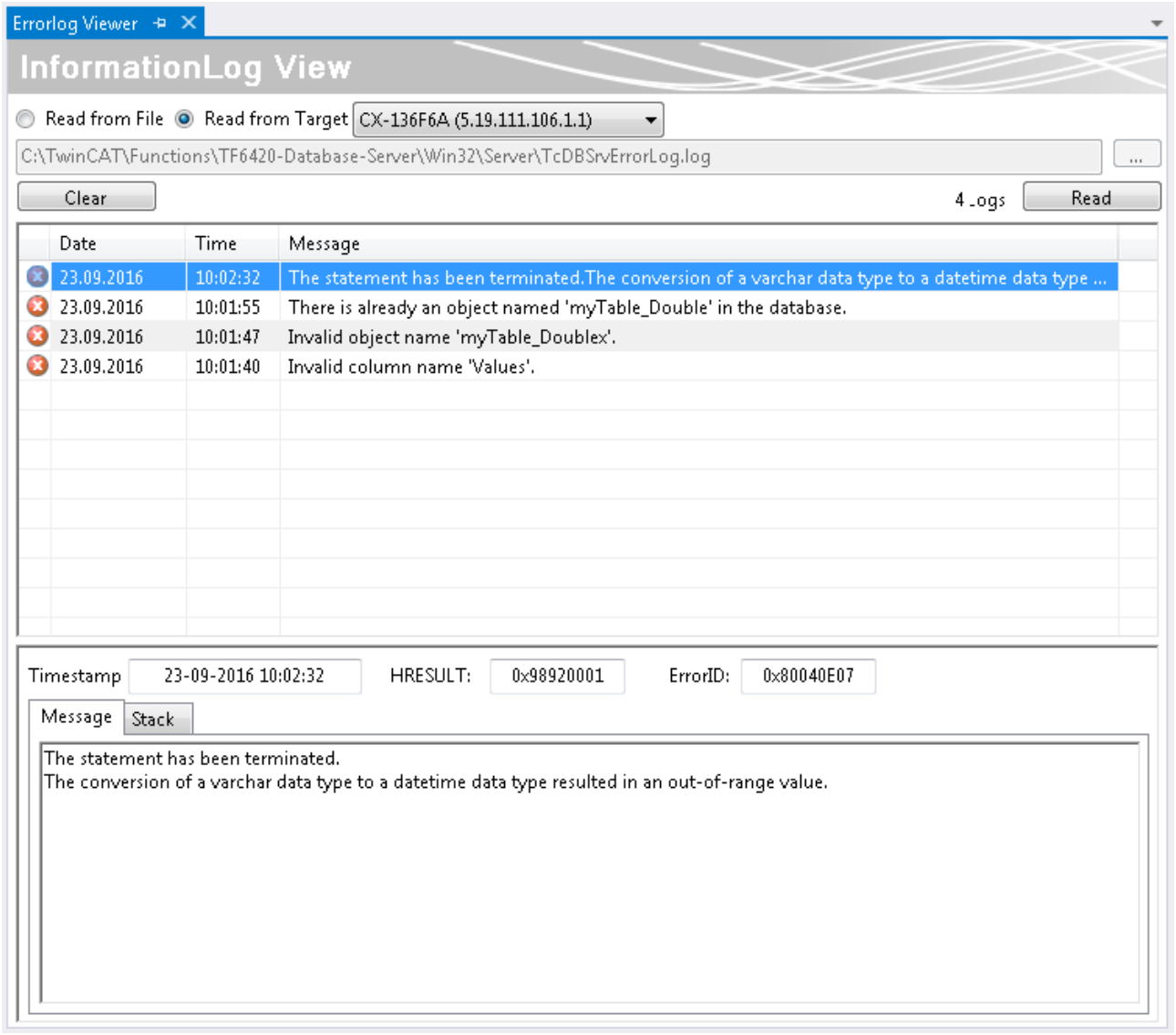


ID	Name	功能
1	目标系统	选择已安装 TwinCAT 数据库服务器的目标系统
2	启动	手动启动 AutoLog 模式
3	登录	登陆进入活动状态的 AutoLog 进程
4	退出	退出活动状态的 AutoLog 进程
5	停止	手动停止 AutoLog 模式
6	AutoLog 组	目标系统上已配置的 AutoLog 组的列表
7	符号	选定 AutoLog 组的已配置符号列表

5.1.2.1.7 InformationLog View

InformationLog View是一种从 TwinCAT 数据库服务器读取日志文件的工具。记录的信息会以纯文本的形式显示，包含时间戳、ID 和错误消息。

不仅可以直接通过文件访问功能来查看或清空日志文件，还可以直接通过目标系统实现这些操作。这在同一网络下有分布式数据库服务器时特别有优势，可以借此快速、方便地访问日志文件。要进行这种访问，必须有一条连接目标设备的路由。



5.2 数据库

TwinCAT 数据库服务器用于连接 TwinCAT PLC 和数据库系统。它支持各种数据库。除了诸如 Microsoft SQL 或 Oracle 等传统数据库外，XML 和 ASCII 文件也可以用作数据库。使用 ODBC 数据库，甚至可以输入数据库连接字符串，以便与未被列为支持的数据库类型的数据库进行通信。

下面的 2 个表格概述了在哪些操作系统平台上支持哪些数据库，以及哪些数据库可用于 TwinCAT Scope 的数据导出和导入。

KaPlatform 支持

关于哪个平台支持哪些数据库连接的概述。

数据库	Windows		Windows CE		TwinCAT/BSD	
	本地	远程	本地	远程	本地	远程
MS SQL	X	X	-	X	-	X
MS SQL Compact	X	-	X	-	-	-
MySQL	X	X	-	X*	X	X
Oracle DB	X	X	-	-	-	-
SQLite	X	-	X**	-	X*	-
ASCII 文件	X	-	X	-	X	-
XML	X	-	X	-	X	-
ODBC	X*	X*	-	-	X*	X*
MS Access	X*	-	-	-	-	-
MS Excel	X*	-	-	-	-	-
MongoDB	X	X	-	-	X	X
PostgreSQL	X	X	-	-	-	X
InfluxDB 1.7 | 1.8	X	X	-	-	X	X
InfluxDB 2	X	X	-	-	-	X

*在设备上必须为数据库安装额外的服务器或客户端驱动程序

**仅适用于具有 ARM 架构的设备

TwinCAT Scope 支持

关于在 TwinCAT Scope 中支持哪些数据库进行数据导入和导出的概述。TwinCAT Scope 总是与 TwinCAT 数据库服务器一起工作。

数据库	Scope 导出	Scope 导入
MS SQL	X	X
MS SQL Compact	-	X
MySQL	-	X
Oracle DB	-	X
SQLite	-	X
ASCII 文件	X	X
XML	-	X
ODBC	-	X
MS Access	-	X
MS Excel	-	X
MongoDB	-	-
PostgreSQL	-	X
InfluxDB 1.7 | 1.8	-	X
InfluxDB 2	-	X

在以下部分中将会介绍 PLC 中的各个数据库的配置以及数据集的映射。

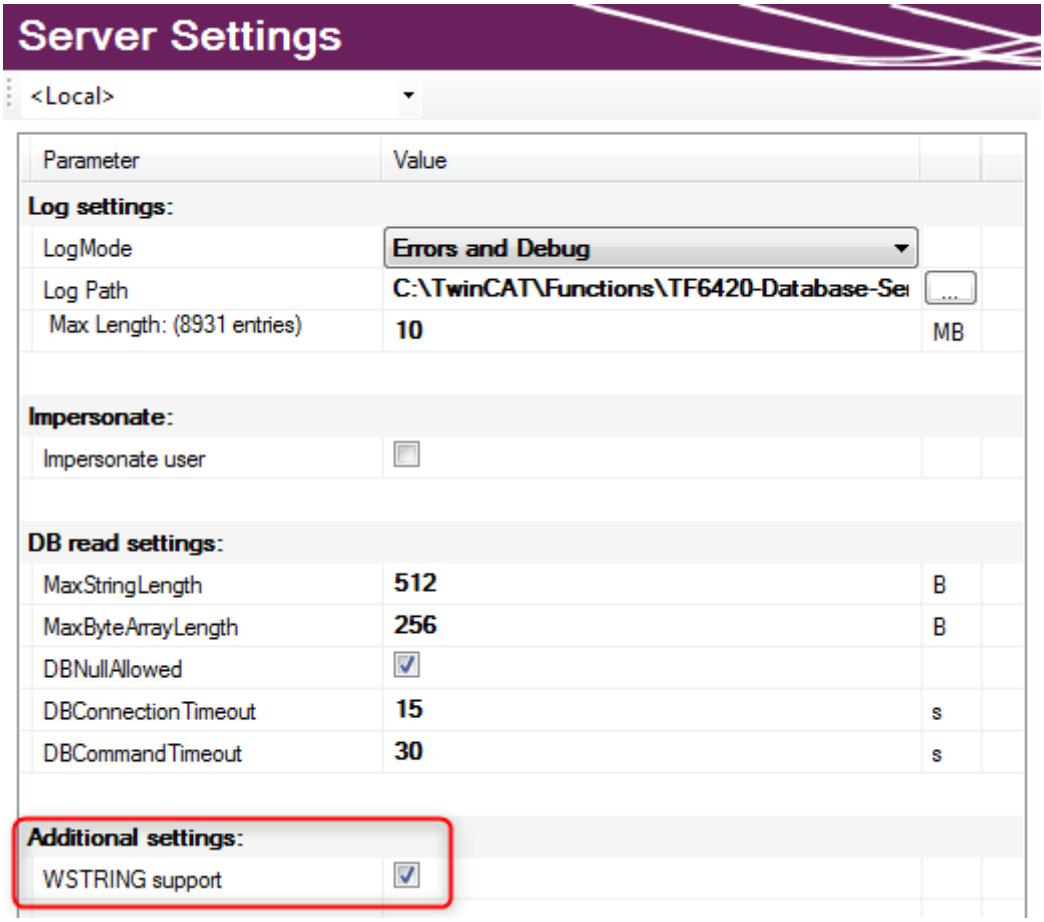
5.2.1 常规信息

您将在以下页面中找到一些有关支持的数据库的常规信息。这些信息是通用的，即不限于某个特定的数据库，并且涵盖网络访问、数据类型支持和操作系统支持等主题。

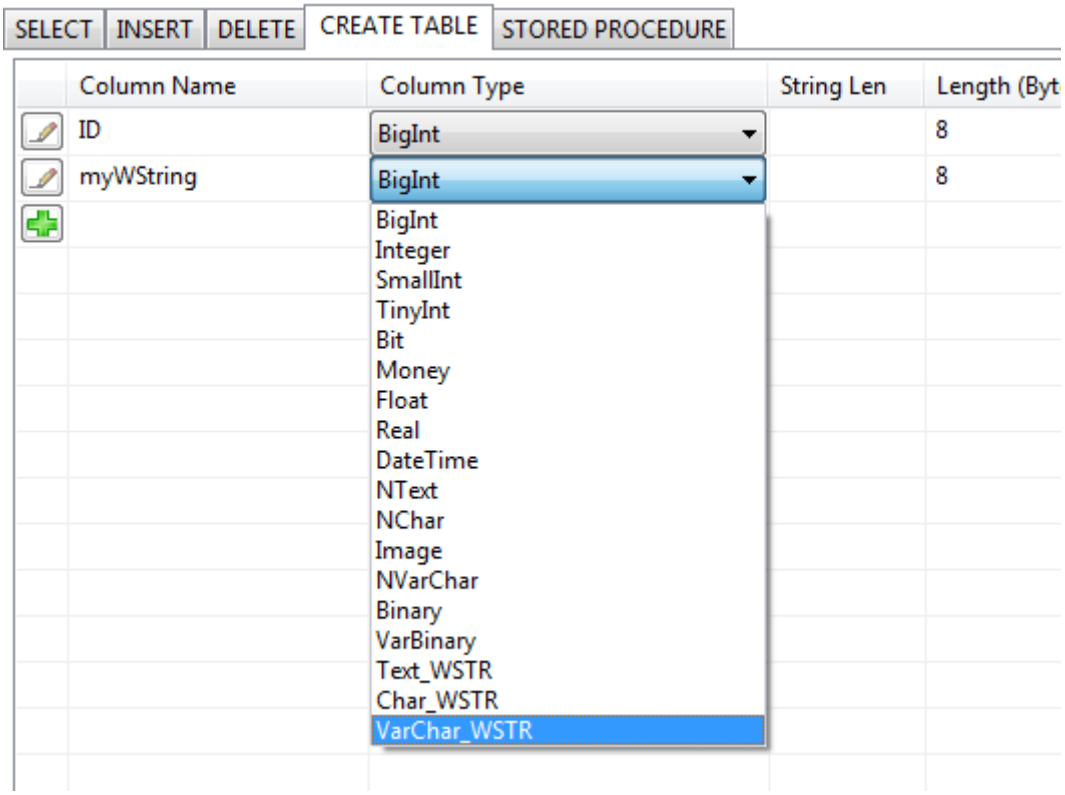
5.2.1.1 WString 支持

WSTRING 可用于使用 Unicode 字符集。

在服务器设置中必须激活该数据类型，以便在 PLC 专家模式下能够使用 TwinCAT 数据库服务器将其写入数据库。



对于该数据类型，每个字符需要 2 个字节。请在创建表结构时牢记这一点。为了能够以 UTF16 格式将其保存在数据库中，必须根据字符集创建列。为此，也可以使用 SQL 查询编辑器。



数据库服务器支持以下数据库：

数据库	UTF8	UTF16	字符集定义	性能损失
MySQL	x	x	特定于列	x
MSSQL	x	x	特定于列	
Oracle	x	x	特定于列	
PostgreSQL	x	x	跨数据库	x
其他	x			

性能损失
一些数据库会导致性能问题，因为必须发送额外的 SQL 命令才能读取字符集。

从 3.1.31.4 版本开始提供 WString 支持。

5.2.1.2 BULK 支持

TwinCAT 数据库服务器也支持所谓的 BULK 命令，用于选择数据库。BULK 命令是将已收集的数据插入表格中的多个行的 SQL 语句。相较于针对每个要添加的行向数据库发送单个插入语句的过程，使用 BULK 插入命令通常可以带来更好的性能表现。

目前，Microsoft SQL 数据库已经可以支持 TwinCAT 数据库服务器通过 FB_PLCDBCmdEvt 功能块发出的 BULK 命令。

命令示例：“SQLBULK<INSERT>#MyTable”

5.2.2 MS SQL 数据库

本部分包含有关 Microsoft SQL 数据库的配置和数据类型映射的信息。

兼容版本： Microsoft SQL 数据库 20xx。

TwinCAT 数据库服务器配置器中的声明

Microsoft SQL 数据库	
Database Type	从下拉菜单中选择“Microsoft SQL Server”（Microsoft SQL 服务器）。
提供程序	“SQLOLEDB”或 SQL 本地客户端的提供程序，例如，“SQLNCLI10”
Server	在这里输入您的 SQL 服务器的名称。示例： “TESTSERVER\SQLEXPRESS”
数据库	输入数据库的名称。如果数据库尚不存在，则可以使用创建按钮进行创建。必须具备相应的权限。
Authentication	以特定用户的身份登录数据库的选项。
用户名	在这里输入用户名。
密码	输入相应的密码。

Windows CE 支持

Windows CE 版本的 TwinCAT 数据库服务器也支持该数据库。这种连接不是本地的，但是通过网络连接可以建立连接。

DB 和 PLC 之间的数据类型映射

E_ColumnTypes	MS SQL	TwinCAT PLC
BigInt	bigint	T_ULARGE_INTEGER (TcUtilities.lib)
Integer	integer	DINT
SmallInt	smallint	INT
TinyInt	tinyint	SINT
Bit_	bit	BYTE
Money	Money	LREAL
Float	float	LREAL
Real_	real	REAL
DateTime	DateTime	DT
NText	ntext	STRING
NChar	nchar	STRING
Image	Image	ARRAY OF BYTE
NVarChar	nvarchar	STRING
Binary	Binary	ARRAY OF BYTE
VarBinary	varbinary	ARRAY OF BYTE

数据类型支持

该数据库支持数据类型 WSTRING。（请参见 WString 支持 [▶ 122]）

注意

数据安全

在闪存设备中，写入访问操作的次数是有限的。闪存设备可能会出现故障，存在丢失数据的风险。

- 务必定期备份系统。使用 IPC 诊断功能，以便确定闪存设备的状态。

5.2.2.1 关于 Microsoft SQL 服务器的说明

Windows Eventlog 中的日志

错误事件 “报告服务器 Windows 服务（SQLEXPRESS）无法连接到报告服务器数据库。”	在 SQL Server 2005 下的 SQL 配置管理器中，停止 SQL Server 报告服务（SQLEXPRESS），并将 **Start Mode**（启动模式）设置为“手动”。数据库服务器不需要报告服务。
信息事件 “‘TcDataLogger’ 数据库已启动”	在 **Databases/TcDataLogger**下，通过 SQL Server Management Studio Express 中的上下文菜单打开 **Properties**（属性），并在 **Options**（选项）下将 **Close automatically**（自动关闭）选项设置为 FALSE。不需要此选项，因为数据库服务器会自动打开和关闭数据库。

可以禁止向 Windows Eventlog 记录日志。然后，不会再记录事件。无法区分不同类型的事件。

在 SQL Server 2005 服务下的 SQL 配置管理器中选择 SQL Server（SQLEXPRESS），并通过上下文菜单打开 **Properties**（属性）。**Advanced**（高级）选项卡包含一个 **Startup parameters**（启动参数）子项目。各个参数之间以分号分隔。添加参数“-n”，并重新启动服务。

从此以后，SQL 服务器将不再记录任何事件。

5.2.3 MS SQL Compact 数据库

本部分包含有关 Microsoft SQL Compact 数据库的配置和数据类型映射的信息。MS SQL Compact 是嵌入式应用程序的理想数据库。它占用空间小，但却能提供关系型数据库所需的功能。

兼容版本：Microsoft SQL Compact 数据库 3.5

TwinCAT 数据库服务器配置器中的声明

Microsoft SQL Compact 数据库	
Database Type	从下拉菜单中选择“Microsoft Compact SQL”。
数据库 URL	输入数据库的名称和路径。如果数据库尚不存在，则可以使用创建按钮进行创建。必须具备相应的权限。
Authentication	使用密码登录数据库的选项。
密码	输入密码。

Windows CE 支持

Windows CE 版本的 TwinCAT 数据库服务器也支持该数据库。在本地可以建立连接。

DB 和 PLC 之间的数据类型映射

E_ColumnTypes	MS SQL Compact	TwinCAT PLC
BigInt	bigint	T_ULARGE_INTEGER (TcUtilities.lib)
Integer	integer	DINT
SmallInt	smallint	INT
TinyInt	tinyint	SINT
Bit_	bit	BYTE
Money	Money	LREAL
Float	float	LREAL
Real_	real	REAL
DateTime	DateTime	DT
NText	ntext	STRING
NChar	nchar	STRING
Image	Image	ARRAY OF BYTE
NVarChar	nvarchar	STRING
Binary	Binary	ARRAY OF BYTE
VarBinary	varbinary	ARRAY OF BYTE

数据类型支持

该数据库不支持 WSTRING。（请参见 WString 支持 [▶ 122]）

注意

数据安全

在闪存设备中，写入访问操作的次数是有限的。闪存设备可能会出现故障，存在丢失数据的风险。

- 务必定期备份系统。使用 IPC 诊断功能，以便确定闪存设备的状态。

5.2.4 MySQL 数据库

本部分包含有关 MySQL 数据库的配置和数据类型映射的信息。

TwinCAT 数据库服务器配置器中的声明

MySQL 数据库	
Database Type	从下拉菜单中选择“MySQL”。
Server	在这里输入您的服务器的名称或 IP 地址。
数据库	输入数据库的名称。如果数据库尚不存在，则可以使用创建按钮进行创建。必须具备相应的权限。
Port	在这里输入与 MySQL 数据库通信的端口。默认值：3306。
用户名	在这里输入用户名。
密码	输入相应的密码。

Windows CE 支持

Windows CE 版本的 TwinCAT 数据库服务器也支持该数据库。这种连接不是本地的，但是通过网络连接可以建立连接。

必要组件**并非**自动安装，必须手动进行安装。要安装它们，**还**需要将 6.7.8.0 版本的“MySql.Data.CF.dll”库复制到 CE 设备的 TwinCAT 数据库服务器文件夹中。 *(\Hard Disk\TwinCAT\Functions\TF6420-Database-Server\Server)*

该 CE 库支持与 MySQL 服务器的连接，最高版本为 8.0.28。

DB 和 PLC 之间的数据类型映射

E_ColumnTypes	MySQL	TwinCAT PLC
BigInt	BIGINT	T_ULARGE_INTEGER (TcUtilities.lib)
Integer	INT	DINT
SmallInt	SMALLINT	INT
TinyInt	TINYINT	SINT
Bit_	CHAR(1)	STRING
Money	DECIMAL(18,4)	LREAL
Float	DOUBLE	LREAL
Real_	FLOAT	REAL
DateTime	DATETIME	DT
NText	TEXT	STRING
NChar	CHAR	STRING
Image	BLOB	ARRAY OF BYTE
NVarChar	VARCHAR	STRING
Binary	BLOB	ARRAY OF BYTE
VarBinary	BLOB	ARRAY OF BYTE

“大型 Windows”系统的驱动程序

所使用的 MySQLConnector 经 MIT 授权使用，并受版权保护：

版权所有 (c) 2016 Bradley Grainger

特此授予任何获得本软件副本及相关文档文件（以下简称为“软件”）的人员个人免费许可，使其可以免费且无限制地处理本软件，包括但不限于使用、复制、修改、合并、出版、分发、转授权和/或出售本软件副本的权利，并允许向其提供软件的人员这样做，但须遵守以下条件：

上述版权声明和本许可声明应包含在本软件的所有副本或重要部分中。

本软件按“原样”提供，不附带任何明示或暗示的保证，包括但不限于关于适销性、针对特定用途的适用性和非侵权性的保证。在任何情况下，作者或版权持有者均不对因本软件或者使用或以其他方式处理本软件而产生、引起或与之相关的任何索赔、损害赔偿或其他责任（无论是合同诉讼、侵权诉讼还是其他诉讼）承担责任。

数据类型支持

该数据库支持数据类型 WSTRING。（请参见 WString 支持 [▶ 122]）

注意

数据安全

在闪存设备中，写入访问操作的次数是有限的。闪存设备可能会出现故障，存在丢失数据的风险。

- 务必定期备份系统。使用 IPC 诊断功能，以便确定闪存设备的状态。

5.2.5 Oracle 数据库

本部分包含有关 Oracle 数据库的配置和数据类型映射的信息。为了与 Oracle 数据库连接，可以使用所谓的 ODP 驱动程序。

兼容版本：Oracle 9i、10g、11g 以及更高版本

TwinCAT 数据库服务器需要 32 位版本的 .NET ODP 组件。

TwinCAT 数据库服务器配置器中的声明

Oracle 数据库	
Database Type	从下拉菜单中选择“Oracle ODP”。
主机	输入数据库的 IP 或主机名。
服务名称	输入服务或数据库的名称。
Port	输入通信端口（可选）。默认值：1521。
Protocal	输入协议（可选）。默认值：TCPIP。
模式	输入数据库架构。
用户名	在这里输入用户名。
密码	输入相应的密码。

Windows CE 支持

在 Windows CE 下，TwinCAT 数据库服务器不支持该数据库。

DB 和 PLC 之间的数据类型映射

E_ColumnTypes	Oracle	TwinCAT PLC
BigInt	DECIMAL(15,0)	T_ULARGE_INTEGER (TcUtilities.lib)
Integer	INTEGER	DINT
SmallInt	SMALLINT	INT
TinyInt	SMALLINT	SINT
Bit_	CHAR(1)	BYTE
Money	DECIMAL(18,4)	LREAL
Float	DOUBLE PRECISION	LREAL
Real_	FLOAT	REAL
DateTime	DATE	DT
NText	VARCHAR(254)	STRING
NChar	CHAR(254)	STRING
Image	BLOB	ARRAY OF BYTE
NVarChar	NVARCHAR(254)	STRING
Binary	BLOB	ARRAY OF BYTE
VarBinary	BLOB	ARRAY OF BYTE

数据类型支持

该数据库支持数据类型 WSTRING。（请参见 WString 支持 [▶ 122]）

注意

数据安全

在闪存设备中，写入访问操作的次数是有限的。闪存设备可能会出现故障，存在丢失数据的风险。

- 务必定期备份系统。使用 IPC 诊断功能，以便确定闪存设备的状态。

5.2.6 SQLite

本部分包含有关 SQLite 数据库的配置和数据类型映射的信息。SQLite 是嵌入式应用程序的理想数据库。这种基于文件的 SQL 数据库无需安装，因为它已经集成在 TwinCAT 数据库服务器中。关系型数据库可以提供 SQL 数据库的大部分功能，并支持 SQL92 标准的命令。该数据库可以实现可靠且快速的数据存储。但是，该数据库不允许区分用户。因此，它非常适合在本地系统中安全地存储变量。

TwinCAT 数据库服务器配置器中的声明

SQLite 数据库	
Database Type	从下拉菜单中选择“SQLite”。
SQLite database file	输入数据库的名称和路径。您还可以使用浏览器对话框。如果数据库尚不存在，则可以使用创建按钮进行创建。必须具备相应的权限。
Authentication	以特定用户的身份登录数据库的选项。
密码	输入相应的密码。

Windows CE 支持

Windows CE 版本的 TwinCAT 数据库服务器也支持该数据库，但仅限于配备 Arm® 处理器的设备。在本地可以建立连接。

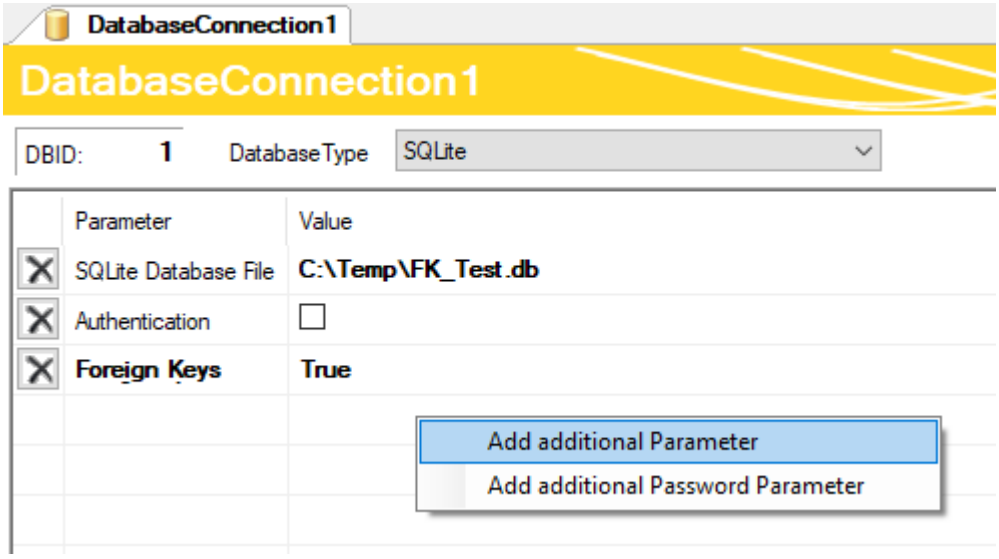
TwinCAT/BSD 支持

TwinCAT/BSD 上的 TwinCAT 数据库服务器支持该数据库。不过，还需要安装软件包库中的“sqlite3”软件包才能使用。

主键/外键的使用

在配置中必须指定一个附加的参数，以便能够在 SQLite 数据库中使用主键和外键。

“Foreign Keys”（外键）-> “True”（真）



DB 和 PLC 之间的数据类型映射

SQLite 有 5 种内部基本数据类型。为了更精确地解释数据，还支持其他的数据类型，在数据库制造商的文档中列出了这些数据类型。

E_ColumnTypes	SQLite	TwinCAT PLC
BigInt	BIGINT	T_ULARGE_INTEGER (TcUtilities.lib)
Integer	INT	DINT
SmallInt	SMALLINT	INT
TinyInt	TINYINT	BYTE
Bit_	BOOLEAN	BOOL
Money	DOUBLE	LREAL
Float	FLOAT	LREAL
Real_	REAL	REAL
DateTime	DATETIME	DT
NText	TEXT	STRING
NChar	NCHAR	STRING
Image	BLOB	ARRAY OF BYTE
NVarChar	NVARCHAR	STRING
Binary	BLOB	ARRAY OF BYTE
VarBinary	BLOB	ARRAY OF BYTE

特殊功能：在 Sqlite 中，字符串或二进制数据类型不受限制。不过，TwinCAT 3 数据库服务器需要固定的限制，这可以在常规服务器设置中进行设置。

不兼容的数据类型

第三方软件可能会在数据库中创建不兼容的数据类型名称，TwinCAT3 数据库服务器无法解释这些命名。在这种情况下，使用 SQL 查询构建器十分有帮助。

数据类型支持

该数据库支持数据类型 WSTRING。（请参见 WString 支持 [▶ 122]）

注意

数据安全

在闪存设备中，写入访问操作的次数是有限的。闪存设备可能会出现故障，存在丢失数据的风险。

- 务必定期备份系统。使用 IPC 诊断功能，以便确定闪存设备的状态。

5.2.7 ASCII 文件

将 ASCII 文件配置为数据库的相关信息。该文件由 TwinCAT 数据库服务器自动生成。不需要创建数据库的过程。在其他电子表格程序（例如 Microsoft Excel）中可以导入已创建的文件，并对其进行处理。

TwinCAT 数据库服务器配置器中的声明

ASCII 文件作为数据库	
Database Type	从下拉菜单中选择“ASCII”。
ASCII 文件	输入 ASCII 文件的路径。该文件由 TwinCAT 数据库服务器自动生成。
值分隔符	在这里，您可以指定值（即列）的分隔符。默认值：“;”
旧的 ASCII DB 格式	出于兼容性考虑，您可以选择切换到 TwinCAT 数据库服务器 3.0.x 版本所使用的旧的 ASCII 格式。使用默认的表结构。
DBValue 类型	仅在启用 **Old ASCII DB format**（旧的 ASCII DB 格式）时有效。您可以选择 BYTES 或 DOUBLE。使用 DOUBLE 时，值为纯文本；使用 BYTES 时，值为字节流。

不支持的功能

该数据库不支持自动 ID 生成功能。如果在配置模式下使用标准表结构，则不会设置 ID 的值。

数据类型支持

该数据库不支持 WSTRING。（请参见 WString 支持 [▶ 122]）

Windows CE 支持

Windows CE 版本的 TwinCAT 数据库服务器也支持该数据库。在本地可以建立连接。

注意

数据安全

在闪存设备中，写入访问操作的次数是有限的。闪存设备可能会出现故障，存在丢失数据的风险。

- 务必定期备份系统。使用 IPC 诊断功能，以便确定闪存设备的状态。

5.2.8 XML 数据库

本部分包含关于将 XML 文件配置为数据库以及数据类型映射的信息。在 XSD 文件中定义了数据库结构、表格和列。使用 TwinCAT 数据库服务器配置器（**创建**命令）可以创建包含样式信息的 XML 文件、XSD 文件和 XSL 文件。基于 XSL 文件，在网络浏览器中可以打开 XML 文件，并在其中显示数据库或表格的图形增强视图。

关于将 XML 文件作为数据库使用的更多信息，请参见“XML - 信息 [▶ 132]”部分。

TwinCAT 数据库服务器配置器中的声明

XML 数据库	
Database Type	从下拉菜单中选择“XML”。
XML 数据库文件	输入 XML 文件的名称和路径。
XML 模式文件	输入 XSD 文件的名称和路径。
数据库	输入数据库的名称。如果数据库尚不存在，则可以使用创建按钮进行创建。必须具备相应的权限。在 XML 数据库中会自动创建 XML、XSD 和 XSL 文件。

Windows CE 支持

Windows CE 版本的 TwinCAT 数据库服务器也支持该数据库。在本地可以建立连接。

DB 和 PLC 之间的数据类型映射

E_ColumnTypes	XML	TwinCAT PLC
BigInt	bigint	T_ULARGE_INTEGER (TcUtilities.lib)
Integer	integer	DINT
SmallInt	smallint	INT
TinyInt	tinyint	BYTE
Bit_	bit	BOOL
Money	Money	LREAL
Float	float	LREAL
Real_	real	LREAL
DateTime	DateTime	DT
NText	ntext	STRING
NChar	nchar	STRING
Image	Image	ARRAY OF BYTE
NVarChar	nvarchar	STRING
Binary	Binary	ARRAY OF BYTE
VarBinary	varbinary	ARRAY OF BYTE

数据类型支持

该数据库不支持 WSTRING。（请参见 WString 支持 [▶ 122]）

注意

数据安全

在闪存设备中，写入访问操作的次数是有限的。闪存设备可能会出现故障，存在丢失数据的风险。

- 务必定期备份系统。使用 IPC 诊断功能，以便确定闪存设备的状态。

5.2.8.1 XML - 信息

1. 使用 XML 文件作为 TwinCAT 3 数据库服务器的数据库

2. 使用 TwinCAT 3 数据库服务器对 XML 文件应用 XPath 查询

有关 XML 模式的更多信息，请访问：http://www.edition-w3.de/TR/2001/REC-xmlschema-0-20010502/

1. XML 作为数据库

标准表结构的 XSD 模式：

```
<?xmlversion="1.0"?>
<xsd:schemaxmlns:xsd="http://www.w3.org/2001/XMLSchema">
<xsd:simpleTypename="bigint">
<xsd:restrictionbase="xsd:long" />
</xsd:simpleType>
<xsd:simpleTypename="datetime">
<xsd:restrictionbase="xsd:dateTime" />
</xsd:simpleType>
<xsd:simpleTypename="ntext_80">
<xsd:restrictionbase="xsd:string">
<xsd:maxLengthvalue="80" />
</xsd:restriction>
</xsd:simpleType>
<xsd:simpleTypename="float">
<xsd:restrictionbase="xsd:double" />
</xsd:simpleType>
<xsd:complexTypename="myTable_Double_Type">
<xsd:sequence>
<xsd:elementminOccurs="0"maxOccurs="unbounded"name="row">
<xsd:complexType>
<xsd:attributename="ID"type="bigint" />
<xsd:attributename="Timestamp"type="datetime" />
```

```
<xsd:attributename="Name"type="ntext_80" />
<xsd:attributename="Value" type="float" />
</xsd:complexType>
</xsd:element>
</xsd:sequence>
</xsd:complexType>
<xsd:elementname="TestDB_XML">
<xsd:complexType>
<xsd:sequenceminOccurs="1"maxOccurs="1">
<xsd:elementname="myTable_Double"type="myTable_Double_Type" />
</xsd:sequence>
</xsd:complexType>
</xsd:element>
</xsd:schema>
```

标准表结构的 XML 文件（示例）：

```
<?xmlversion="1.0"encoding="UTF-8"?>
<TestDB_XMLxmlns:xs="http://www.w3.org/2001/XMLSchema-
nstance"xs:noNamespaceSchemaLocation="TestDB_XML.xsd">
<myTable_Double>
<rowID="1"Timestamp="2012-03-08T12:45:08"Name="TestValue1"Value="222.222" />
<rowID="2"Timestamp="2012-03-08T12:45:14"Name="TestValue1"Value="222.222" />
<rowID="3"Timestamp="2012-03-08T12:45:18"Name="TestValue1"Value="222.222" />
<rowID="4"Timestamp="2012-03-08T12:45:22"Name="TestValue1"Value="222.222" />
<rowID="5"Timestamp="2012-03-08T12:45:23"Name="TestValue1"Value="222.222" />
</myTable_Double>
</TestDB_XML>
```

XML 表格的数据类型：

```
<xsd:simpleTypename="bigint">
<xsd:restrictionbase="xsd:long" />
</xsd:simpleType>

<xsd:simpleTypename="datetime">
<xsd:restrictionbase="xsd:dateTime" />
</xsd:simpleType>

<xsd:simpleTypename="ntext_80"> //Länge kann individuell angegeben werden
<xsd:restrictionbase="xsd:string">
<xsd:maxLengthvalue="80" />
</xsd:restriction>
</xsd:simpleType>

<xsd:simpleTypename="float">
<xsd:restrictionbase="xsd:double" />
</xsd:simpleType>

<xsd:simpleTypename="binary_1"> //Länge kann individuell angegeben werden
<xsd:restrictionbase="xsd:hexBinary">
<xsd:maxLengthvalue="1" />
</xsd:restriction>
</xsd:simpleType>

<xsd:simpleTypename="bit">
<xsd:restrictionbase="xsd:boolean" />
</xsd:simpleType>

<xsd:simpleTypename="image_1"> //Länge kann individuell angegeben werden
<xsd:restrictionbase="xsd:hexBinary">
<xsd:maxLengthvalue="1" />
</xsd:restriction>
</xsd:simpleType>

<xsd:simpleTypename="integer">
<xsd:restrictionbase="xsd:int" />
</xsd:simpleType>

<xsd:simpleTypename="money">
<xsd:restrictionbase="xsd:double" />
</xsd:simpleType>

<xsd:simpleTypename="nchar_50"> //Länge kann individuell angegeben werden
<xsd:restrictionbase="xsd:string">
<xsd:maxLengthvalue="50" />
</xsd:restriction>
</xsd:simpleType>

<xsd:simpleTypename="nvarchar_50"> //Länge kann individuell angegeben
werden
<xsd:restrictionbase="xsd:string">
<xsd:maxLengthvalue="50" />
</xsd:restriction>
</xsd:simpleType>

<xsd:simpleTypename="real">
<xsd:restrictionbase="xsd:double" />
</xsd:simpleType>

<xsd:simpleTypename="smallint">
<xsd:restrictionbase="xsd:short" />
</xsd:simpleType>
```

```
<xsd:simpleTypename="tinyint">
<xsd:restrictionbase="xsd:byte" />
</xsd:simpleType>

<xsd:simpleTypename="varbinary_1"> //Länge kann individuell angegeben werden
<xsd:restrictionbase="xsd:hexBinary">
<xsd:maxLengthvalue="1" />
</xsd:restriction>
</xsd:simpleType>
```

DB 和 PLC 之间的数据类型映射

E_ColumnTypes	XML	TwinCAT PLC
BigInt	bigint	T_ULARGE_INTEGER (TcUtilities.lib)
Integer	integer	DINT
SmallInt	smallint	INT
TinyInt	tinyint	BYTE
Bit_	bit	BOOL
Money	Money	LREAL
Float	float	LREAL
Real_	real	LREAL
DateTime	DateTime	DT
NText	ntext	STRING
NChar	nchar	STRING
Image	Image	ARRAY OF BYTE
NVarChar	nvarchar	STRING
Binary	Binary	ARRAY OF BYTE
VarBinary	varbinary	ARRAY OF BYTE

在/从 XML 文件中创建/读取记录

标准 SQL 命令可以用于生成记录。TwinCAT 数据库服务器可以解析 SQL INSERT 命令，将其转换为 XML 节点的形式并写入XML 文件。TwinCAT 数据库服务器以 XPath 查询的形式转换 XML 文件的 SQL SELECT 命令。

支持的 INSERT 命令示例：

- INSERT INTO myTable_Double (ID, Timestamp, Name, Value) VALUES(1, CURRENT_TIMESTAMP, 'TestValue1' , 1234.5678)
- INSERT INTO myTable_Double (Timestamp, Name) VALUES(CURRENT_TIMESTAMP, 'TestValue1');
- INSERT INTO myTable_Double VALUES(1, CURRENT_TIMESTAMP, 'TestValue1', 1234.5678);
- INSERT INTO myTable_Double VALUES(1, '2010-01-06 12:13:14', 'TestValue1', 1234.5678);

支持的 SELECT 命令示例：

- SELECTID, Timestamp, Name, Value FROM myTable_Double;
- SELECT* FROM myTable_Double;
- SELECTTimestamp, Name FROM myTable_Double
- SELECT* FROM myTable_Double WHERE Name = 'TestValue1';
- SELECT* FROM myTable_Double WHERE ID > 1;

支持的功能块：

- FB_DBCreate
- FB_DBCyclicRdWrt
- FB_DBRead
- FB_DBRecordArraySelect

- FB_DBRecordDelete
- FB_DBRecordInsert
- FB_DBRecordInsert_EX
- FB_DBRecordSelect
- FB_DBRecordSelect_EX
- FB_DBTableCreate
- FB_DBWrite

2. XML 标准 XPath 函数

XPath 类型

TwinCAT 数据库服务器中的 XPath 的前缀的语法如下：XPATH_[Type]<[Position]>#[Path]

XPath 有 4 种不同的类型：

- SEL
  - 从 XML 读取数据并将其返回到 PLC
- ADD
  - 在选定的位置将传输的数据添加到 XML 中。
- UPD
  - 使用新数据在选定的位置上替换现有的 XML 信息。
- DEL
  - 删除 XML 中选定位置的数据。

3 种不同的数据适用于各个位置：

- ATTR
  - 适用于选定 XML 标签中的所有属性值。
- TAG
  - 适用于选定 XML 标签的 InnerText 值。
- SUBTAG
  - 适用于选定 XML 标签的所有子标签的 InnerText 值。
  - 如果存在 XML 模式，则属性将转换为正确的数据类型。
    如果不存在 XML 模式，则属性将作为 T_MaxString 返回。

示例：

XML 文件：

```
<?xmlversion="1.0"encoding="utf-8" ?>
<TestXML>
<Nodeattr1="1"attr2="Node1">
<SubNode1>SubNodeWert1</SubNode1>
<SubNode2>200</SubNode2>
<SubNode3>SubNodeWert3</SubNode3>
<SubNode4>400.5</SubNode4>
<SubNode5>SubNodeWert5</SubNode5>
</Node>
<Nodeattr1="2"attr2="Node2">
<SubNode1>SubNodeWert1</SubNode1>
<SubNode2>200</SubNode2>
<SubNode3>SubNodeWert3</SubNode3>
<SubNode4>400.5</SubNode4>
<SubNode5>SubNodeWert5</SubNode5>
</Node>
</TestXML>
```

XML 模式：

```
<?xmlversion="1.0"encoding="utf-8"?>
<xs:schemaattributeFormDefault="unqualified"elementFormDefault="qualified"xmlns:xs="http://
www.w3.org/2001/XMLSchema">
<xs:elementname="TestXML">
<xs:complexType>
```

```
<xs:sequence>
<xs:elementmaxOccurs="unbounded"name="Node">
<xs:complexType>
<xs:sequence>
<xs:elementname="SubNode1"type="xs:string" />
<xs:elementname="SubNode2"type="xs:short" />
<xs:elementname="SubNode3"type="xs:string" />
<xs:elementname="SubNode4"type="xs:double" />
<xs:elementname="SubNode5"type="xs:string" />
</xs:sequence>
<xs:attributename="attr1" type="xs:integer"use="required" />
<xs:attributename="attr2" type="xs:string"use="required" />
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:schema>
```

XPATH<ATTR> 示例

XPath => XPATH_SEL<ATTR>#TestXML/Node[@attr1=2]

如果**不**存在模式，则返回结构：

```
TYPEST_Record :
STRUCT
attr1 : T_MaxString := '2';
attr2 : T_MaxString := 'Node2';
END_STRUCT
END_TYPE
```

如果存在**一个**模式，则返回结构：

```
TYPEST_Record :
STRUCT
attr1 : DINT := 2;
attr2 : T_MaxString := 'Node2';
END_STRUCT
END_TYPE
```

XPATH<TAG> 示例

XPath => XPATH_SEL<TAG>#TestXML/Node[@attr1=2]/SubNode2

如果**不**存在模式，则返回值：SubNode2 : T_MaxString := '200';

如果存在**一个**模式，则返回值：SubNode2 : INT := 200;

XPATH<SUBTAG> 示例

XPath => XPATH_SEL<SUBTAG>#TestXML/Node[@attr1=2]

如果**不**存在模式，则返回结构：

```
TYPEST_Record :
STRUCT
SubNode1 : T_MaxString := 'SubNodeWert1';
SubNode2 : T_MaxString := '200';
SubNode3 : T_MaxString := 'SubNodeWert3';
SubNode4 : T_MaxString := '400.5';
SubNode5 : T_MaxString := 'SubNodeWert5';
END_STRUCT
END_TYPE
```

如果存在**一个**模式，则返回结构：

```
TYPEST_Record :
STRUCT
SubNode1 : T_MaxString := 'SubNodeWert1';
SubNode2 : INT := 200;
SubNode3 : T_MaxString := 'SubNodeWert3';
SubNode4 : LREAL := 400.5;
SubNode5 : T_MaxString := 'SubNodeWert5';
END_STRUCT
END_TYPE
```

使用 **FB_PLCDBCmd** 的特殊功能：

与 FB_PLCDBCmd 的通常实现不同，设置参数（ST_ExpParameter）不会指定相应指令的占位符，而会指定传输或返回数据的模式。

支持的功能块

- FB_DBRecordSelect
- FB_DBRecordSelect_EX
- FB_DBRecordArraySelect

5.2.9 ODBC 数据库

许多数据库都提供 ODBC 接口。TwinCAT 数据库服务器也有该接口。因此，在 TwinCAT 数据库配置器中，一般可以在数据库配置菜单中选择 ODBC 数据库。所谓的“自由连接字符串 [▶ 137]”可以用于通过**添加附加参数**来形成您自己的连接字符串。

更多已知和经常使用的 ODBC 数据库可用作模板。其中包括：

- MySQL [▶ 138]
- Oracle [▶ 139]
- PostgreSQL [▶ 140]
- IBM DB2 [▶ 141]
- Firebird [▶ 142]

Windows CE 支持

在 Windows CE 下，TwinCAT 数据库服务器不支持该数据库。

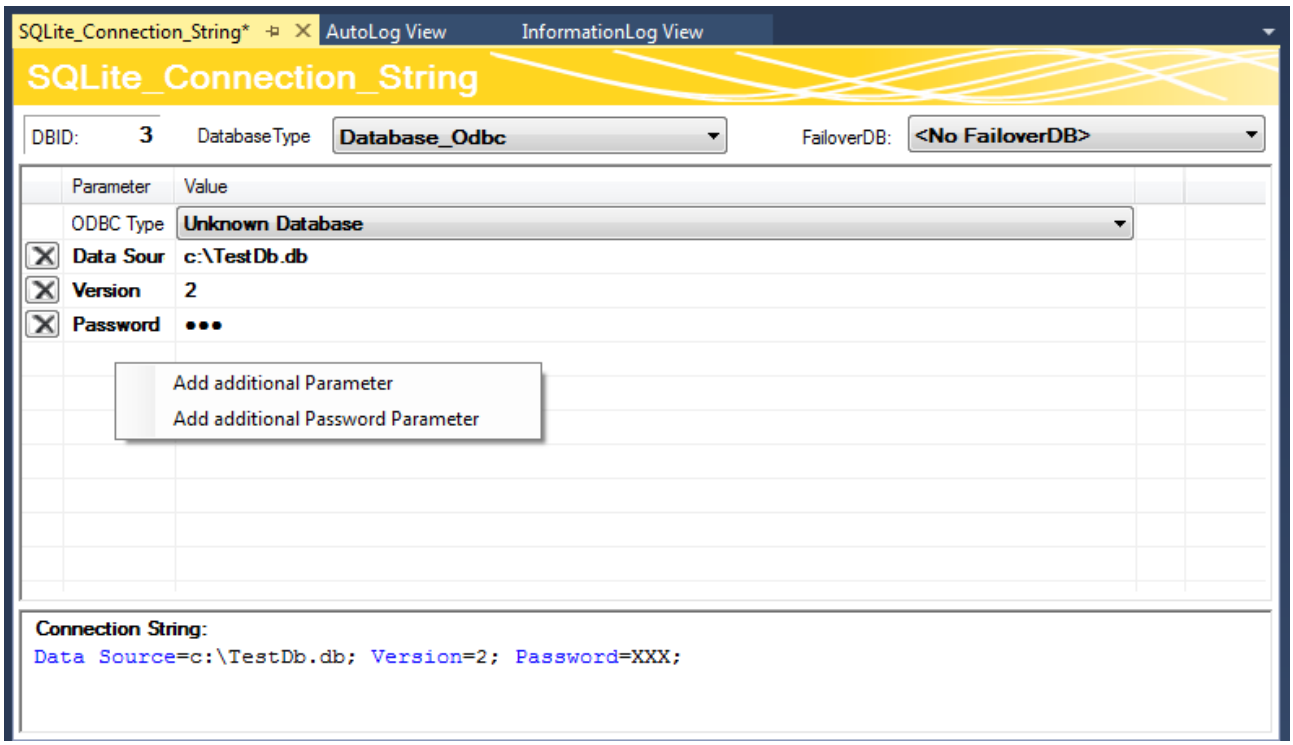
5.2.9.1 自由连接字符串

要使用 TwinCAT 数据库服务器默认不支持的 ODBC 接口数据库，您可以选择“未知数据库”作为 **ODBC 类型**。

TwinCAT 数据库服务器配置器中的声明

ODBC 自由连接字符串数据库	
Database Type	“Odbc_Database”
ODBC Type	从下拉菜单中选择“Unknown Database”（未知数据库）。

只需找到 ODBC 数据库的连接字符串，并在 TwinCAT 数据库服务器的配置窗口中进行修改即可。



为此，在上下文菜单中提供了 2 条命令：

- **添加附加参数**
  为连接字符串添加常规参数。根据连接字符串的要求，它可以有任何名称。
- **添加附加密码参数**
  添加特殊密码参数，其值在配置器和配置文件中不可识读（加密）。

使用自由连接字符串的工作原理

为了确保 TwinCAT 数据库服务器能够使用自由连接字符串，在 TwinCAT 数据库服务器的目标系统上必须安装相应的驱动程序。只能使用“SQL 专家模式 [▶ 17]”！

注意

数据安全

在闪存设备中，写入访问操作的次数是有限的。闪存设备可能会出现故障，存在丢失数据的风险。

- 务必定期备份系统。使用 IPC 诊断功能，以便确定闪存设备的状态。

5.2.9.2 MySQL

本部分包含有关使用 ODBC 的 MySQL 数据库的配置和数据类型映射的信息。

TwinCAT 数据库服务器配置器中的声明

ODBC MySQL 数据库	
Database Type	“Odbc_Database”
ODBC Type	从下拉菜单中选择“MySQL”。
Driver	输入实际安装的驱动程序。
Server	输入您的服务器的名称或 IP 地址。
数据库	输入数据库的名称。
Port	在这里输入与 MySQL 数据库通信的端口。默认值：3306。
Option	默认值：2“返回匹配的行，而不是受影响的行”
Uid	输入用户名。
Pwd	输入相应的密码。

DB 和 PLC 之间的数据类型映射

E_ColumnTypes	MySQL	TwinCAT PLC
BigInt	BIGINT	T_ULARGE_INTEGER (TcUtilities.lib)
Integer	INT	DINT
SmallInt	SMALLINT	INT
TinyInt	TINYINT	SINT
Bit_	CHAR(1)	STRING
Money	DECIMAL(18,4)	LREAL
Float	DOUBLE	LREAL
Real_	FLOAT	REAL
DateTime	DATETIME	DT
NText	TEXT	STRING
NChar	CHAR	STRING
Image	BLOB	ARRAY OF BYTE
NVarChar	VARCHAR	STRING
Binary	BLOB	ARRAY OF BYTE
VarBinary	BLOB	ARRAY OF BYTE

数据类型支持

该数据库支持数据类型 WSTRING。（请参见 WString 支持 [▶ 122]）

功能

TwinCAT 数据库服务器的所有功能均可应用于 ODBC 模板。这不适用于“自由连接字符串 [▶ 137]”。

注意

数据安全

在闪存设备中，写入访问操作的次数是有限的。闪存设备可能会出现故障，存在丢失数据的风险。

- 务必定期备份系统。使用 IPC 诊断功能，以便确定闪存设备的状态。

5.2.9.3 Oracle

本部分包含有关使用 ODBC 的 Oracle 数据库的配置和数据类型映射的信息。

TwinCAT 数据库服务器配置器中的声明

ODBC Oracle 数据库	
Database Type	“Odbc_Database”
ODBC Type	从下拉菜单中选择“Oracle”。
Driver	在这里选择实际安装的驱动程序。
Server	输入您的服务器的名称或 IP 地址。
Uid	在这里输入用户名。
Pwd	输入相应的密码。

DB 和 PLC 之间的数据类型映射

E_ColumnTypes	Oracle	TwinCAT PLC
BigInt	DECIMAL(15,0)	T_ULARGE_INTEGER (TcUtilities.lib)
Integer	INTEGER	DINT
SmallInt	SMALLINT	INT
TinyInt	SMALLINT	SINT
Bit_	CHAR(1)	BYTE
Money	DECIMAL(18,4)	LREAL
Float	DOUBLE PRECISION	LREAL
Real_	FLOAT	REAL
DateTime	DATE	DT
NText	VARCHAR(254)	STRING
NChar	CHAR(254)	STRING
Image	BLOB	ARRAY OF BYTE
NVarChar	NVARCHAR(254)	STRING
Binary	BLOB	ARRAY OF BYTE
VarBinary	BLOB	ARRAY OF BYTE

数据类型支持

该数据库支持数据类型 WSTRING。（请参见 WString 支持 [▶ 122]）

功能

TwinCAT 数据库服务器的所有功能均可应用于 ODBC 模板。这不适用于“自由连接字符串 [▶ 137]”。

注意

数据安全

在闪存设备中，写入访问操作的次数是有限的。闪存设备可能会出现故障，存在丢失数据的风险。

- 务必定期备份系统。使用 IPC 诊断功能，以便确定闪存设备的状态。

5.2.9.4 PostgreSQL

本部分包含有关使用 ODBC 的 PostgreSQL 数据库的配置和数据类型映射的信息。

TwinCAT 数据库服务器配置器中的声明

ODBC PostgreSQL 数据库	
Database Type	“Odbc_Database”
ODBC Type	从下拉菜单中选择“PostgreSQL”。
Driver	在这里选择实际安装的驱动程序。
Server	输入您的服务器的名称或 IP 地址。
数据库	输入数据库的名称。
Port	输入与 PostgreSQL 数据库通信的端口。默认值：5432。
Uid	在这里输入用户名。
Pwd	在这里输入相应的密码。

DB 和 PLC 之间的数据类型映射

E_ColumnTypes	PostgreSQL	TwinCAT PLC
BigInt	BIGINT	T_ULARGE_INTEGER (TcUtilities.lib)
Integer	integer	DINT
SmallInt	smallint	INT
TinyInt	smallint	INT
Bit_	bit	BYTE
Money	Money	LREAL
Float	Double precision	LREAL
Real_	real	REAL
DateTime	时间戳	DT
NText	text	STRING
NChar	character	STRING
Image	byte	ARRAY OF BYTE
NVarChar	Character varying	STRING
Binary	byte	ARRAY OF BYTE
VarBinary	byte	ARRAY OF BYTE

数据类型支持

该数据库支持数据类型 WSTRING。创建数据库时必须设置字符集。

功能

TwinCAT 数据库服务器的所有功能均可应用于 ODBC 模板。这不适用于“自由连接字符串 [▶ 137]”。

注意

数据安全

在闪存设备中，写入访问操作的次数是有限的。闪存设备可能会出现故障，存在丢失数据的风险。

- 务必定期备份系统。使用 IPC 诊断功能，以便确定闪存设备的状态。

5.2.9.5 IBM DB2

本部分包含有关使用 ODBC 的 IBM DB2 数据库的配置和数据类型映射的信息。

TwinCAT 数据库服务器配置器中的声明

ODBC IBM DB2 数据库	
Database Type	“Odbc_Database”
ODBC Type	从下拉菜单中选择“IBM DB2”。
Driver	在这里输入实际安装的驱动程序。
Host name	输入您的服务器的名称或 IP 地址。
数据库	输入数据库的名称。
Port	输入与 IBM DB2 数据库通信的端口。默认值：50000。
Protocal	默认值：TCPIP
Uid	在这里输入用户名。
Pwd	在这里输入相应的密码。
LONGDATACOMPAT	默认值：1

DB 和 PLC 之间的数据类型映射

E_ColumnTypes	IBM DB2	TwinCAT PLC
BigInt	BIGINT	T_ULARGE_INTEGER (TcUtilities.lib)
Integer	INT	DINT
SmallInt	SMALLINT	INT
TinyInt	SMALLINT	INT
Bit_	VARCHAR(1)	STRING(1)
Money	DECIMAL(18,4)	LREAL
Float	DOUBLE PRECISION	LREAL
Real_	FLOAT	LREAL
DateTime	TIMESTAMP	DT
NText	LONG VARCHAR	STRING
NChar	CHAR(254)	STRING
Image	BLOB	ARRAY OF BYTE
NVarChar	NVARCHAR(254)	STRING
Binary	BLOB	ARRAY OF BYTE
VarBinary	BLOB	ARRAY OF BYTE

数据类型支持

该数据库不支持 WSTRING。（请参见 WString 支持 [▶ 122]）

功能

TwinCAT 数据库服务器的所有功能均可应用于 ODBC 模板。这不适用于“自由连接字符串 [▶ 137]”。

注意

数据安全

在闪存设备中，写入访问操作的次数是有限的。闪存设备可能会出现故障，存在丢失数据的风险。

- 务必定期备份系统。使用 IPC 诊断功能，以便确定闪存设备的状态。

5.2.9.6 Firebird

本部分包含有关使用 ODBC 的 Firebird 数据库的配置和数据类型映射的信息。

TwinCAT 数据库服务器配置器中的声明

ODBC Firebird 数据库	
Database Type	“Odbc_Database”
ODBC Type	从下拉菜单中选择“Firebird”。
Driver	在这里选择实际安装的驱动程序。
数据库	输入数据库的名称。
Client	
Uid	在这里输入用户名。
Pwd	输入相应的密码。

DB 和 PLC 之间的数据类型映射

E_ColumnTypes	Firebird	TwinCAT PLC
BigInt	BIGINT	T_ULARGE_INTEGER (TcUtilities.lib)
Integer	INTEGER	DINT
SmallInt	SMALLINT	INT
TinyInt	TINYINT	INT
Bit_	CHAR(1)	STRING
Money	DECIMAL(18,4)	LREAL
Float	FLOAT	REAL
Real_	DOUBLE PRECISION	LREAL
DateTime	TIMESTAMP	DT
NText	VARCHAR(254)	STRING
NChar	CHAR(254)	STRING
Image	BLOB	ARRAY OF BYTE
NVarChar	VARCHAR(254)	STRING
Binary	BLOB	ARRAY OF BYTE
VarBinary	BLOB	ARRAY OF BYTE

数据类型支持

该数据库不支持 WSTRING。（请参见 WString 支持 [▶ 122]）

功能

TwinCAT 数据库服务器的所有功能均可应用于 ODBC 模板。这不适用于“自由连接字符串 [▶ 137]”。

注意

数据安全

在闪存设备中，写入访问操作的次数是有限的。闪存设备可能会出现故障，存在丢失数据的风险。

- 务必定期备份系统。使用 IPC 诊断功能，以便确定闪存设备的状态。

5.2.10 MS Access 数据库

变量的值保存在 Microsoft Access 数据库中。

Access 2000 和 Access 2003（*.mdb）数据库文件以及Access 2007（*.accdb）文件都兼容。您只需在 XML 配置文件的声明中指定不同的提供程序即可。

TwinCAT 数据库服务器配置器中的声明

Microsoft Access 数据库	
DBValueType	选择“Double”（双精度），可将日志记录限制为字母数字和布尔数据类型。 选择“Bytes”（字节），也可记录结构和字符串。
DBType	选择“MS Access”。PLC：eDBType_Access。
DBServer	无需配置。
DBProvider	Access 2000 - Access 2003： 提供程序是“Microsoft.Jet.OLEDB.4.0”。 Access 2007： 提供程序是“Microsoft.ACE.OLEDB.12.0”。
DBUrl	DBUrl 包含 MDB 文件的路径。 例如， *C:\TwinCAT\TcDatabaseSrv\Samples\TestDB.mdb*
DBTable	DBTable 包含表格的名称。

Windows CE 支持

在 Windows CE 下，TwinCAT 数据库服务器不支持该数据库。

TwinCAT/BSD 支持

TwinCAT/BSD 平台上的 TwinCAT 数据库服务器不支持该数据库。

DB 和 PLC 之间的数据类型映射

E_DBColumnTypes	MS Access	PLC 控制
eDBColumn_BigInt	Integer4	DINT
eDBColumn_Integer	Integer2	INT
eDBColumn_SmallInt	Integer2	SINT
eDBColumn_TinyInt	Integer1	SINT
eDBColumn_Bit	YESNO	BYTE
eDBColumn_Money	Currency	LREAL
eDBColumn_Float	Double	LREAL
eDBColumn_Real	Single	REAL
eDBColumn_DateTime	DATETIME	DT
eDBColumn_NText	Text	STRING
eDBColumn_NChar	VarChar	STRING
eDBColumn_Image	OLEOBJECT	ARRAY OF BYTE
eDBColumn_NVarChar	VarChar	STRING
eDBColumn_Binary	OLEOBJECT	ARRAY OF BYTE
eDBColumn_VarBinary	OLEOBJECT	ARRAY OF BYTE

数据类型支持

该数据库不支持 WSTRING。（请参见 WString 支持 [▶ 122]）

注意

数据安全

在闪存设备中，写入访问操作的次数是有限的。闪存设备可能会出现故障，存在丢失数据的风险。

- 务必定期备份系统。使用 IPC 诊断功能，以便确定闪存设备的状态。

5.2.11 MS Excel 数据库

变量值存储在 Microsoft Excel 数据库中。

TwinCAT 数据库服务器配置器中的声明

Microsoft Excel 数据库	
DBValueType	选择“Double”（双精度），可将日志记录限制为字母数字和布尔数据类型。 选择“Bytes”（字节），也可记录结构和字符串。
DBType	选择“MS Excel”。PLC：eDBType_MSExcel。
DBServer	无需配置。
DBProvider	“Microsoft.Jet.OLEDB.4.0”或“Microsoft.ACE.OLEDB.12.0”
DBUrl	DBUrl 包含 Excel 文件的路径。 例如， *C:\TwinCAT\TcDatabaseSrv\Samples\TestDB.xls*
DBTable	DBTable 包含表格的名称。

Windows CE 支持

在 Windows CE 下，TwinCAT 数据库服务器不支持该数据库。

TwinCAT/BSD 支持

TwinCAT/BSD 平台上的 TwinCAT 数据库服务器不支持该数据库。

DB 和 PLC 之间的数据类型映射

E_DBColumnTypes	MS Excel	PLC 控制
eDBColumn_BigInt	Number	LREAL
eDBColumn_Integer	Number	LREAL
eDBColumn_SmallInt	Number	LREAL
eDBColumn_TinyInt	Number	LREAL
eDBColumn_Bit	BOOLEAN	BOOL
eDBColumn_Money	Currency	LREAL
eDBColumn_Float	Number	LREAL
eDBColumn_Real	Number	LREAL
eDBColumn_DateTime	日期	DT
eDBColumn_NText	Text	STRING(255)
eDBColumn_NChar	Text	STRING(255)
eDBColumn_NVarChar	Text	STRING(255)

不支持的功能

该数据库不支持自动 ID 生成功能。如果在配置模式下使用标准表结构，则不会设置 ID 的值。

不支持的数据类型

Excel 数据库不支持 Binary（二进制）、VarBinary（可变二进制） 和 Image（图像）。

数据类型支持

该数据库不支持 WSTRING。（请参见 WString 支持 [▶ 122]）

注意

数据安全

在闪存设备中，写入访问操作的次数是有限的。闪存设备可能会出现故障，存在丢失数据的风险。

- 务必定期备份系统。使用 IPC 诊断功能，以便确定闪存设备的状态。

5.2.12 MongoDB

本部分包含有关 MongoDB 数据库的配置和数据类型映射的信息。

TwinCAT 数据库服务器配置器中的声明

MongoDB	
Database Type	从下拉菜单中选择“MongoDB”。
Server	输入 MongoDB 服务器的名称。
数据库	输入数据库的名称。如果数据库尚不存在，则可以在首次访问时进行创建。
Authentication	**None：**无身份验证 **用户名/密码：**使用用户名和密码登录 **x509 证书：** 用户名：证书用户的 ID 证书颁发机构：签名证书（*.crt）的路径 客户端证书：客户端证书（*.pfx）的路径 客户端私钥：客户端证书的密码 **GSSAPI/Kerberos：**使用用户名和密码登录 **LDAP（PLAIN）：**使用用户名和密码登录 （由于用户名和密码以纯文本的形式传输，因此不建议使用此选项）

Windows CE 支持

在 Windows CE 下，TwinCAT 数据库服务器不支持该数据库。

DB 和 PLC 之间的数据类型映射

MongoDB	TwinCAT PLC
long	LINT
int	DINT
BOOL	BYTE
double	LREAL
时间戳	DT
string	STRING
binData	ARRAY OF BYTE
objectId	T_ObjectId_MongoDB
array	ARRAY
object	STRUCT

数据类型支持

该数据库不支持 WSTRING。（请参见 WString 支持 [▶ 122]）

PLC 专家模式/配置模式下的 MongoDB

PLC 专家和配置模式会在其流程中使用预定义的数据库模式。通常情况下，所使用结构的模式在运行期间不会改变。不过，为了能够使用功能组件，TwinCAT 3 数据库服务器需要对表格架构进行描述。因此会为 MongoDB 模拟一个表格。
在 SQL 查询编辑器中，使用 **SQL** 选项卡和 **CREATE TABLE**（创建表格）子类别可以创建一个表格，或者如本示例中创建一个集合。此外，与关系型数据库不同的是，在元数据集合中会创建一个条目。在这里会存储有关 TwinCAT 3 数据库服务器的表格架构的信息。

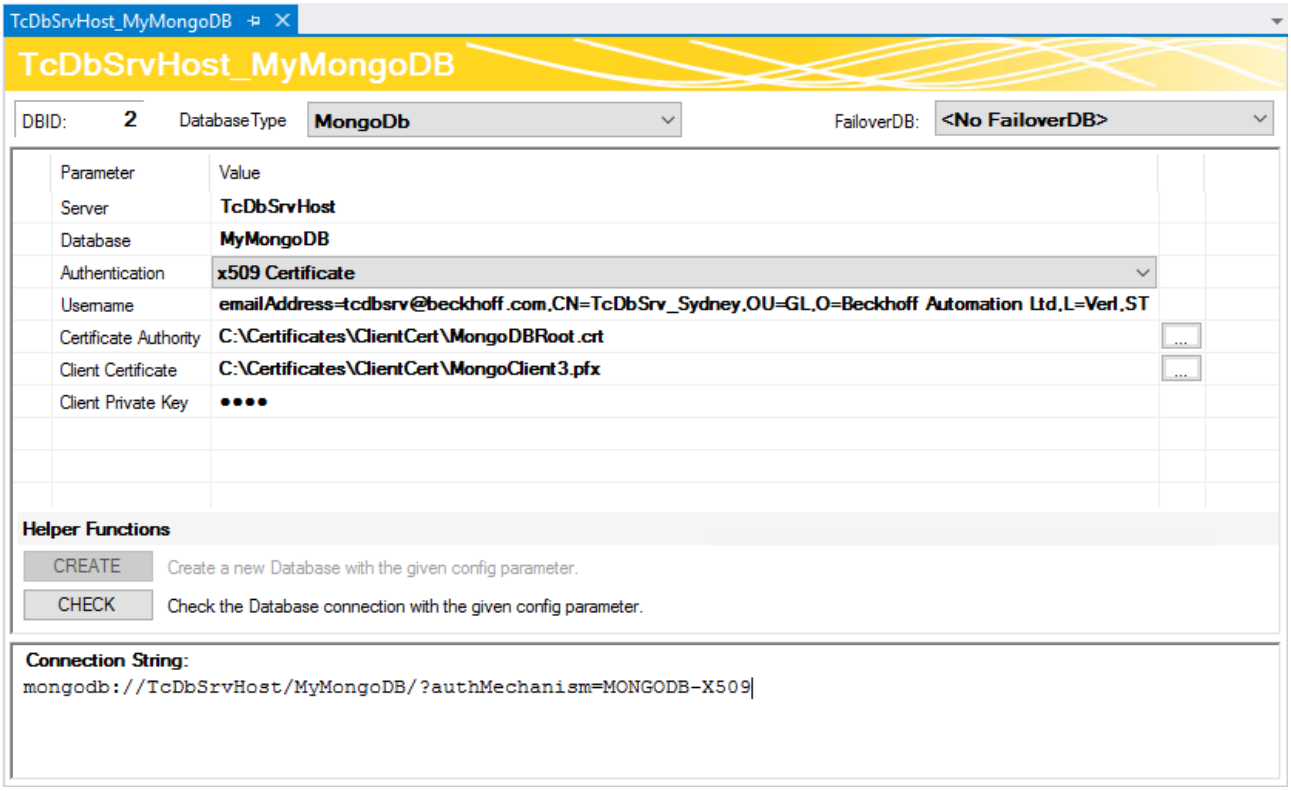
为了使用高级功能（例如，任何层次结构或灵活记录），我们建议使用 NoSQL 功能块。

证书的使用

除其他功能外，MongoDB 还支持通过证书进行身份验证。为此，可在身份验证下选择“x509 证书”方法。以下字段将会出现：

用户名	相应证书的用户名
证书授权	证书授权的 SSL 证书的路径。这可能是一个自签名证书。
客户端证书	由 SSL 证书签名的客户端证书。
客户端证书密码	客户端证书的密码。

使用证书配置与 MongoDB 的数据库连接：



注意

数据安全

在闪存设备中，写入访问操作的次数是有限的。闪存设备可能会出现故障，存在丢失数据的风险。

- 务必定期备份系统。使用 IPC 诊断功能，以便确定闪存设备的状态。

5.2.13 PostgreSQL

本部分提供有关 PostgreSql 数据库的配置和应用的内容。PostgreSQL 是一个具有客户端-服务器基础架构的对象关系型开源数据库。TwinCAT 3 数据库服务器使用 Npgsql API 进行连接。

TwinCAT 数据库服务器配置器中的声明

PostgreSql 数据库	
Database Type	从下拉菜单中选择“PostgreSql”。
Server	数据库服务器的名称或 IP
数据库	服务器上数据库的名称
Port	数据库的端口
Authentication	数据库身份验证方法
用户名	输入用户名。
密码	输入相应的密码。
客户端证书	使用的客户端证书（.pfx）的路径
客户端证书密码	引用的客户端证书的密码

Windows CE 支持

在 Windows CE 下，TwinCAT 数据库服务器不支持该数据库。

DB 和 PLC 之间的数据类型映射

E_ColumnTypes	PostgreSQL	TwinCAT PLC
BigInt	Bigint	T_ULARGE_INTEGER (TcUtilities.lib)
Integer	Integer	DINT
SmallInt	Smallint	INT
TinyInt		SINT
Bit_	Bit	BYTE
Money	Money	LREAL
Float	Double precision	LREAL
Real_	Real	REAL
DateTime	不带时区的Timestamp	DT
NText	Text	STRING
NChar	Character	STRING
Image		ARRAY OF BYTE
NVarChar	Character varying	STRING
Binary	Bytea	ARRAY OF BYTE
VarBinary		ARRAY OF BYTE

注意

数据安全

在闪存设备中，写入访问操作的次数是有限的。闪存设备可能会出现故障，存在丢失数据的风险。

- 务必定期备份系统。使用 IPC 诊断功能，以便确定闪存设备的状态。

数据类型支持

该数据库不支持 WSTRING。（请参见 WString 支持 [▶ 122]）

5.2.14 InfluxDB

本部分包含有关 InfluxDB 的配置和数据类型映射的信息。

支持的版本：1.7.x、1.8.x

TwinCAT 数据库服务器配置器中的声明

InfluxDB	
Database Type	从下拉菜单中选择“InfluxDB”。
Server	在这里输入所需的数据库服务器的地址。
数据库	输入数据库的名称。如果数据库尚不存在，则您可以使用“Create”（创建）进行创建。
Authentication	**None：**无身份验证 **用户名/密码：**使用用户名和密码登录

Windows CE 支持

在 Windows CE 下，TwinCAT 数据库服务器不支持该数据库。

DB 和 PLC 之间的数据类型映射

E_ColumnTypes	InfluxDB	TwinCAT PLC
BigInt	Integer	LINT
Integer	Integer	DINT
SmallInt	Integer	INT
TinyInt	Integer	BYTE
Bit_	Boolean	BOOL
Money	Float	LREAL
Float	Float	LREAL
Real_	Float	LREAL
DateTime		DT
NText	String	STRING
NChar	String	STRING
Image		ARRAY OF BYTE
NVarChar	String	STRING
Binary		ARRAY OF BYTE
VarBinary		ARRAY OF BYTE
	Tag	STRING
	"time"	LINT

Time

作为时间序列数据库，InfluxDB 具有一些特殊功能。一个序列的每个测量（表格）都包含时间列、标签列和字段列。时间列以 UNIX 纪元时间的形式存储在数据库中。数据库服务器的功能块使用 TwinCAT 时间（自 1601 年 1 月 1 日以来的 100 ns 时间步的数量）。它们被转换为 UNIX 纪元时间。该转换不包括 FB_SQLDBCommand。在这里，您可以传输自己的自由时间戳，无需转换。时间被设置为“ns”精度。在 InfluxDB 中，时间作为 ID 与标签列都是唯一的。如果标签和时间都相同，该数据集将会被覆盖。

标准表结构

在 PLC 中，InfluxDB 的标准表结构如下所示：

ColumnName	InfluxDB	TwinCAT PLC
时间	Integer	LINT
Name	Tag	T_MaxString
值	Float	LREAL

数据类型支持

该数据库不支持 WSTRING。（请参见 WString 支持 [▶ 122]）

注意

数据安全

在闪存设备中，写入访问操作的次数是有限的。闪存设备可能会出现故障，存在丢失数据的风险。

- 务必定期备份系统。使用 IPC 诊断功能，以便确定闪存设备的状态。

5.2.15 InfluxDB2

本部分包含有关 InfluxDB2 的配置和数据类型映射的内容。

支持的版本：2.x

TwinCAT 数据库服务器配置器中的声明

InfluxDB	
Database Type	从下拉菜单中选择“InfluxDB2”。
Server	在这里输入所需的数据库服务器的地址。
数据库	输入数据库的名称。如果数据库尚不存在，则您可以使用“Create”（创建）进行创建。
Authentication	**None：**无身份验证 **用户名/密码：**使用用户名和密码登录
Token	用于访问数据库的访问令牌。可以通过必要的权限在数据库软件中进行创建。

Windows CE 支持

在 Windows CE 下，TwinCAT 数据库服务器不支持该数据库。

DB 和 PLC 之间的数据类型映射

E_ColumnTypes	InfluxDB2	TwinCAT PLC
BigInt	Integer	LINT
Integer	Integer	DINT
SmallInt	Integer	INT
TinyInt	Integer	BYTE
Bit_	Boolean	BOOL
Money	Float	LREAL
Float	Float	LREAL
Real_	Float	LREAL
DateTime	DateTime	DT
NText	String	STRING
NChar	String	STRING
Image	-	ARRAY OF BYTE
NVarChar	String	STRING
Binary	-	ARRAY OF BYTE
VarBinary	-	ARRAY OF BYTE
Tag	Tag	STRING
BigInt	“_time”	LINT

Time

作为时间序列数据库，InfluxDB2 具有一些特殊功能。一个序列的每个测量（表格）都包含时间列、标签列和字段列。时间列以 UNIX 纪元时间的形式存储在数据库中。数据库服务器的功能块使用 TwinCAT 时间（自 1601 年 1 月 1 日以来的 100 ns 时间步的数量）。它们被转换为 UNIX 纪元时间。该转换不包括 FB_SQLDBCommand。在这里，您可以传输自己的自由时间戳，无需转换。时间被设置为“ns”精度。在 InfluxDB 中，时间作为 ID 与标签列都是唯一的。如果标签和时间都相同，该数据集将会被覆盖。

数据类型支持

该数据库不支持 WSTRING。（请参见 WString 支持 [▶ 122]）

注意

数据安全

在闪存设备中，写入访问操作的次数是有限的。闪存设备可能会出现故障，存在丢失数据的风险。

- 务必定期备份系统。使用 IPC 诊断功能，以便确定闪存设备的状态。

6 PLC API

6.1 Tc3_Database

6.1.1 功能块

根据基本概念 [▶ 17]，Tc3_Database.compiled 库的功能块分为 3 个部分：

- 配置模式：
  包含用于在配置器中定义的 AutoLog 组中进行读取和写入的功能块。
- PLC 专家模式：
  包含传统 PLC 程序员编程的功能块。
- SQL 专家模式：
  具备高级数据库知识的 IT 和 PLC 专家可以使用这些功能块在 PLC 中集成 SQL 命令。
- NoSql 专家模式：
  具备扩展数据库知识的 IT 和 PLC 专家可以使用这些功能块，通过 NoSQL 数据库创建指令，并将它们发送到数据库。

使用 Tc3_Eventlogger

TwinCAT 3 数据库服务器支持 Tc3_Eventlogger API。有关更多信息，请参见 Tc3_EventLogger支持 [▶ 210] 下的内容

6.1.1.1 配置模式

6.1.1.1.1 FB_ConfigTcDBSrvEvt

用于创建、读取和删除 TwinCAT 数据库服务器的配置条目的功能块。

语法

定义：

```
FUNCTION_BLOCK FB_ConfigTcDBSrvEvt
VAR_INPUT
    sNetID: T_AmsNetID := '';
    tTimeout: TIME := T#5S;
END_VAR
VAR_OUTPUT
    bBusy: BOOL;
    bError: BOOL;
    ipTcResult: Tc3_EventLogger.I_TcMessage;
END_VAR
```

输入

Name	Type	描述
sNetID	T_AmsNetID	ADS 命令指向的目标设备的 AMS 网络 ID。
tTimeout	TIME	表示函数被取消前的时间。

输出

Name	Type	描述
bBusy	BOOL	功能块的方法处于活动状态时即为 TRUE。
bError	BOOL	发生错误时即为 TRUE。
ipTcResult	Tc3_EventLogger.I_TcMessage [▶ 215]	TwinCAT 3 EventLogger 的消息接口，其中提供了有关返回值的详细信息。

属性

Name	Type	Access	描述
eTraceLevel	TcEventSeverity [▶ 216]	获取，设置	指定事件的重要性权重。只有权重高于该值的事件才会被发送到 TwinCAT 系统。

方法

Name	定义位置	描述
创建 [▶ 153]	本地	在 XML 配置文件中为 TwinCAT 数据库服务器创建新条目
读取 [▶ 154]	本地	读取 TwinCAT 数据库服务器的当前配置
删除 [▶ 155]	本地	从 TwinCAT 数据库服务器的配置中删除数据库和 AutoLog 组

要求

开发环境	目标平台	要引入的 PLC 库
TwinCAT v3.1 版本 4022.20	PC 或 CX（x86）	Tc3_Database

6.1.创建

1.1.

1.1

该方法可在 XML 配置文件中为 TwinCAT 数据库服务器创建新条目。TwinCAT 数据库服务器可以选择临时使用新条目。在这种情况下，不会将任何数据写入 XML 文件。

语法

```
METHOD Create : BOOL
VAR_INPUT
    pTcDBSrvConfig: POINTER TO BYTE;
    cbTcDBSrvConfig: UDINT;
    bTemporary: BOOL := TRUE;
    pConfigID: POINTER TO UDINT;
END_VAR
```

输入

Name	Type	描述
pTcDBSrvConfig	POINTER TO BYTE	要创建的配置结构的指针。
cbTcDBSrvConfig	UDINT	配置结构的长度
bTemporary	BOOL	指定是否要将配置存储在 XML 文件中。
pConfigID	POINTER TO UDINT	返回配置 ID（hDBID 或 hAutoLogGrpID）的指针

目前不支持创建 AutoLog 组。

返回值

Name	Type	描述
创建	BOOL	显示方法的状态。在方法执行完毕时，即使出现错误，也会返回 TRUE。

示例

```
VAR
    fbConfigTcDBSrv : FB_ConfigTcDBSrvEvt(sNetId := '', tTimeout:=T#5S);
    myConfigHandle  : INT;
    // Any other ConfigType can be used here
    stConfigDB      : T_DBConfig_MsCompactSQL;
    tcMessage       : I_TcMessage;
END_VAR
```

```
stConfigDB.bAuthentification := FALSE;
stConfigDB.sServer := 'C:\Recipes.sdf';

IF fbConfigTcDBSrv.Create(
    pTcDBSrvConfig:= ADR(stConfigDB),
    cbTcDBSrvConfig:= SIZEOF(stConfigDB),
    bTemporary:= TRUE,
    pConfigID:= ADR(myConfigHandle))
THEN
    IF fbConfigTcDBSrv.bError THEN
        tcMessage := fbConfigTcDBSrv.ipTcResult;
        nState := 255;
    ELSE
        nState := 0;
    END_IF
END_IF
```

6.1.读取

1.1.

1.2

该方法可以用于读取 TwinCAT 数据库服务器的当前配置。任何可能被包含的临时配置都会对其进行相应的标记。

语法

```
METHOD Read : BOOL
VAR_INPUT
    pDBConfig: POINTER TO ARRAY [1..MAX_CONFIGURATIONS] OF ST_ConfigDB;
    cbDBConfig: UDINT;
    pAutoLogGrpConfig: POINTER TO ARRAY[1..MAX_CONFIGURATIONS] OF
ST_ConfigAutoLogGrp;
    cbAutoLogGrpConfig: UDINT;
    pDBCount: POINTER TO UDINT;
    pAutoLogGrpCount: POINTER TO UDINT;
END_VAR
```

输入

Name	Type	描述
pDBConfig	POINTER TO ARRAY [1..MAX_CONFIGURATIONS [▶ 236]] OF ST_ConfigDB [▶ 219]	用于写入数据库配置的数组的指针地址。
cbDBConfig	UDINT	数据库配置数组的长度
pAutoLogGrpConfig	POINTER TO ARRAY[1..MAX_CONFIGURATIONS [▶ 236]] OF ST_ConfigAutoLogGrp [▶ 218]	用于写入 AutoLogGrp 配置的数组的指针地址。
cbAutoLogGrpConfig	UDINT	AutoLogGrp 配置数组的长度
pDBCount	POINTER TO UDINT	用于存储数据库配置数量的指针地址。
pAutoLogGrpCount	POINTER TO UDINT	用于存储 AutoLogGrp 配置数量的指针地址。

返回值

Name	Type	描述
读取	BOOL	显示方法的状态。在方法执行完毕时，即使出现错误，也会返回 TRUE。

示例

```
VAR
    fbConfigTcDBSrv    : FB_ConfigTcDBSrvEvt(sNetId := '', tTimeout:=T#5S);
    aDBConfig          : ARRAY[0..MAX_CONFIGURATIONS] OF ST_ConfigDB;
    aAutoGrpConfig     : ARRAY[0..MAX_CONFIGURATIONS] OF ST_ConfigAutoLogGrp;
    nDbCount           : UDINT;
    nAutoGrpCount      : UDINT;
    tcMessage          : I_TcMessage;
END_VAR
```

```
IF fbConfigTcDBSrv.Read(
    pDBConfig := ADR(aDBConfig),
    cbDBConfig := SIZEOF(aDBConfig),
    pAutologGrpConfig := ADR(aAutoGrpConfig),
    cbAutoLogGrpConfig := SIZEOF(aAutoGrpConfig),
    pDBCount := ADR(nDbCount),
    pAutoLogGrpCount := ADR(nAutoGrpCount))
 THEN
    IF fbConfigTcDBSrv.bError THEN
        tcMessage := fbConfigTcDBSrv.ipTcResult;
        nState := 255;
    ELSE
        nState := 0;
    END_IF
END_IF
```

6.1.删除

1.1.

1.3

该方法可以用于从 TwinCAT 数据库服务器的配置中删除数据库和 AutoLog 组。

语法

```
METHOD Delete : BOOL
VAR_INPUT
    eTcDBSrvConfigType: E_TcDBSrvConfigType;
    hConfigID: UDINT;
END_VAR
```

输入

Name	Type	描述
eTcDBSrvConfigType	E_TcDBSrvConfigType	要删除的配置的类型（数据库/AutoLog 组）
hConfigID	UDINT	要删除的配置的 ID（hDBID 或 hAutoLogGrpID）

返回值

Name	Type	描述
删除	BOOL	显示方法的状态。在方法执行完毕时，即使出现错误，也会返回 TRUE。

示例

```
VAR
    fbConfigTcDBSrv : FB_ConfigTcDBSrvEvt(sNetId := '', tTimeout:=T#5S);
    myConfigHandle  : INT;
    tcMessage       : I_TcMessage;
END_VAR
```

```
IF fbConfigTcDBSrv.Delete(
    eTcDBSrvConfigType := E_TcDBSrvConfigType.Database,
    hConfigID := myConfigHandle) THEN
IF fbConfigTcDBSrv.bError THEN
        tcMessage := fbConfigTcDBSrv.ipTcResult;
        nState := 255;
    ELSE
        nState := 0;
    END_IF
END_IF
```

6.1.1.1.2 FB_PLCDBAutoLogEvt

功能块具有 4 种方法，用于启动和停止已定义的 AutoLog 组，以及读取相应的组状态。

语法

定义：

```
FUNCTION_BLOCK FB_PLCDBAutoLogEvt
VAR_INPUT
    sNetID: T_AmsNetID := '';
    tTimeout: TIME := T#5S;
END_VAR
VAR_OUTPUT
    bBusy: BOOL;
    bError: BOOL;
    ipTcResult: Tc3_EventLogger.I_TcMessage;
    bBusy_Status: BOOL;
END_VAR
```

输入

名称	类型	描述
sNetID	T_AmsNetID	ADS 命令指向的目标设备的 AMS 网络 ID。
tTimeout	TIME	表示函数被取消前的时间。

输出

名称	类型	描述
bBusy	BOOL	除Status方法外，功能块的方法处于活动状态时为 TRUE。
bError	BOOL	发生错误时即为 TRUE。
ipTcResult	Tc3_EventLogger.I_TcMessage	结果界面，提供有关返回值的详细信息。
bBusy_Status	BOOL	Status方法可以独立于功能块的其他 3 种方法执行，因此该方法有自己独立的繁忙标志。Status方法处于活动状态时为 TRUE。

方法

名称	定义位置	描述
RunOnce [▶ 157]	本地	执行一次 AutoLog 组
启动 [▶ 157]	本地	使用相应配置的 AutoLog 组启动 AutoLog 模式
Status [▶ 157]	本地	查询 AutoLog 组的状态。
停止 [▶ 158]	本地	停止 AutoLog 模式

要求

开发环境	目标平台	要引入的 PLC 库
TwinCAT v3.1 版本 4022.20	PC 或 CX（x86）	Tc3_Database

6.1.RunOnce

1.1.

2.1

该方法可以用于执行一次 AutoLog 组，例如，基于控制器中的一个事件。

语法

```
METHOD RunOnce : BOOL
VAR_INPUT
    hAutoLogGrpID: UDINT;
    bAll: BOOL;
END_VAR
```

输入

名称	类型	描述
hAutoLogGrpID	UDINT	要执行一次的 AutoLog 组的 ID。
bAll	BOOL	如果为 TRUE，则所有 AutoLog 组都执行一次。

返回值

名称	类型	描述
RunOnce	BOOL	显示方法的状态。在方法执行完毕时，即使出现错误，也会返回 TRUE。

示例

```
VAR
    fbPLCDBAutoLog    : FB_PLCDBAutoLogEvt (sNetID:='', tTimeout := T#5S);
END_VAR
```

```
IF fbPLCDBAutoLog.RunOnce(hAutologGrpID := 1, bAll := FALSE) THEN
    ; // ...
END_IF
```

6.1.启动

1.1.

2.2

该方法使用相应配置的 AutoLog 组启动 AutoLog 模式。

语法

```
METHOD Start : BOOL
```

返回值

名称	类型	描述
启动	BOOL	显示方法的状态。在方法执行完毕时，即使出现错误，也会返回 TRUE。

示例

```
VAR
    fbPLCDBAutoLog    : FB_PLCDBAutoLogEvt (sNetID:='', tTimeout := T#5S);
END_VAR
```

```
IF fbPLCDBAutoLog.Start() THEN
    ; // ...
END_IF
```

6.1.Status

1.1.

2.3

该方法可以用于查询 AutoLog 组的状态。由于该方法可以独立于功能块的其他方法调用，因此在功能块主体中为该方法提供了一个单独的繁忙标志：bBusy_Status。

语法

```
METHOD Status : BOOL
VAR_INPUT
    tCheckCycle: TIME;
    pError: POINTER TO BOOL;
    pAutoLogGrpStatus: POINTER TO ARRAY [1..MAX_CONFIGURATIONS] OF ST_AutoLogGrpStatus;
    cbAutoLogGrpStatus: UDINT;
END_VAR
```

输入

名称	类型	描述
tCheckCycle	TIME	更新状态数组的间隔时间。
pError	POINTER TO BOOL	如果在 AutoLog 模式下发生错误，则为 TRUE。
pAutoLogStatus	POINTER TO ARRAY [1..MAX_CONFIGURATIONS [▶ 236]] OF ST_AutoLogGrpStatus [▶ 233]	包含所有组的状态数组的地址。
cbAutoLogStatus	UDINT	状态数组的长度

返回值

名称	类型	描述
Status	BOOL	显示方法的状态。在方法执行完毕时，即使出现错误，也会返回 TRUE。

示例

```
VAR
    fbPLCDBAutoLog    : FB_PLCDBAutoLogEvt(sNetID:='', tTimeout := T#5S);
    bError            : BOOL;
    aAutologGrpStatus : ARRAY[0..MAX_CONFIGURATIONS] OF ST_AutoLogGrpStatus;
END_VAR
```

```
IF fbPLCDBAutoLog.Status(tCheckCycle := T#30S, ADR(bError), ADR(aAutologGrpStatus), SIZEOF(aAutologG
rpStatus)) THEN
    ; // ...
END_IF
```

6.1.停止

1.1.

2.4

该方法可停止 AutoLog 模式。

语法

```
METHOD Stop : BOOL
```

返回值

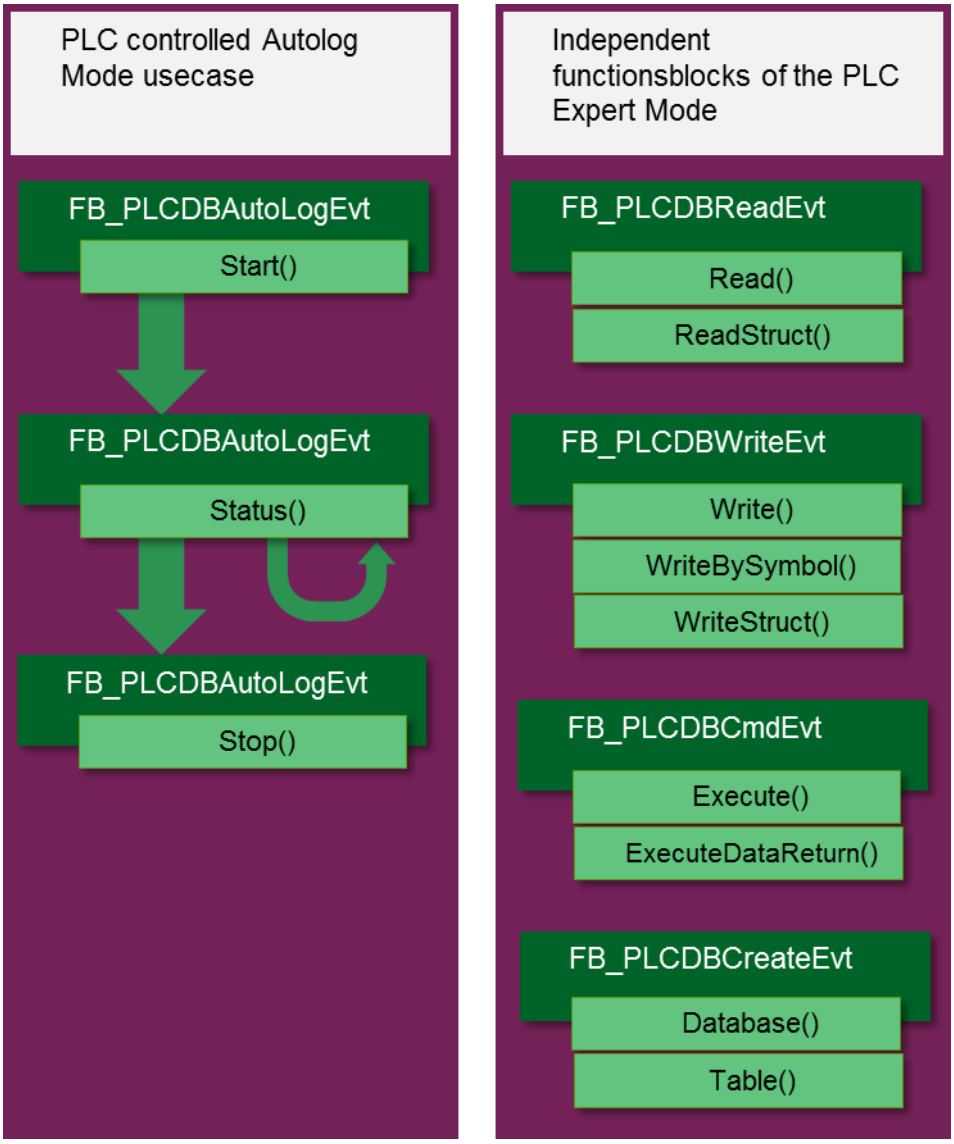
名称	类型	描述
停止	BOOL	显示方法的状态。在方法执行完毕时，即使出现错误，也会返回 TRUE。

示例

```
VAR
    fbPLCDBAutoLog    : FB_PLCDBAutoLogEvt (sNetID:='', tTimeout := T#5S);
END_VAR
```

```
IF fbPLCDBAutoLog.Stop() THEN
    ; // ...
END_IF
```

6.1.1.2 PLC 专家模式



6.1.1.2.1 FB_ConfigTcDBSrvEvt

用于创建、读取和删除 TwinCAT 数据库服务器的配置条目的功能块。

语法

定义：

```
FUNCTION_BLOCK FB_ConfigTcDBSrvEvt
VAR_INPUT
    sNetID: T_AmsNetID := '';
    tTimeout: TIME := T#5S;
END_VAR
VAR_OUTPUT
    bBusy: BOOL;
    bError: BOOL;
    ipTcResult: Tc3_EventLogger.I_TcMessage;
END_VAR
```

输入

Name	Type	描述
sNetID	T_AmsNetID	ADS 命令指向的目标设备的 AMS 网络 ID。
tTimeout	TIME	表示函数被取消前的时间。

输出

Name	Type	描述
bBusy	BOOL	功能块的方法处于活动状态时即为 TRUE。
bError	BOOL	发生错误时即为 TRUE。
ipTcResult	Tc3_EventLogger.I_TcMessage [▶ 215]	TwinCAT 3 EventLogger 的消息接口，其中提供了有关返回值的详细信息。

属性

Name	Type	Access	描述
eTraceLevel	TcEventSeverity [▶ 216]	获取，设置	指定事件的重要性权重。只有权重高于该值的事件才会被发送到 TwinCAT 系统。

方法

Name	定义位置	描述
创建 [▶ 160]	本地	在 XML 配置文件中为 TwinCAT 数据库服务器创建新条目
读取 [▶ 161]	本地	读取 TwinCAT 数据库服务器的当前配置
删除 [▶ 162]	本地	从 TwinCAT 数据库服务器的配置中删除数据库和 AutoLog 组

要求

开发环境	目标平台	要引入的 PLC 库
TwinCAT v3.1 版本 4022.20	PC 或 CX（x86）	Tc3_Database

6.1.创建
1.2.
1.1

该方法可在 XML 配置文件中为 TwinCAT 数据库服务器创建新条目。TwinCAT 数据库服务器可以选择临时使用新条目。在这种情况下，不会将任何数据写入 XML 文件。

语法

```
METHOD Create : BOOL
VAR_INPUT
    pTcDBSrvConfig: POINTER TO BYTE;
    cbTcDBSrvConfig: UDINT;
    bTemporary: BOOL := TRUE;
    pConfigID: POINTER TO UDINT;
END_VAR
```

输入

Name	Type	描述
pTcDBSrvConfig	POINTER TO BYTE	要创建的配置结构的指针。
cbTcDBSrvConfig	UDINT	配置结构的长度
bTemporary	BOOL	指定是否要将配置存储在 XML 文件中。
pConfigID	POINTER TO UDINT	返回配置 ID（hDBID 或 hAutoLogGrpID）的指针

目前不支持创建 AutoLog 组。

返回值

Name	Type	描述
创建	BOOL	显示方法的状态。在方法执行完毕时，即使出现错误，也会返回 TRUE。

示例

```
VAR
    fbConfigTcDBSrv : FB_ConfigTcDBSrvEvt(sNetId := '', tTimeout:=T#5S);
    myConfigHandle  : INT;
    // Any other ConfigType can be used here
    stConfigDB      : T_DBConfig_MsCompactSQL;
    tcMessage       : I_TcMessage;
END_VAR
```

```
stConfigDB.bAuthentification := FALSE;
stConfigDB.sServer := 'C:\Recipes.sdf';

IF fbConfigTcDBSrv.Create(
    pTcDBSrvConfig:= ADR(stConfigDB),
    cbTcDBSrvConfig:= SIZEOF(stConfigDB),
    bTemporary:= TRUE,
    pConfigID:= ADR(myConfigHandle))
THEN
    IF fbConfigTcDBSrv.bError THEN
        tcMessage := fbConfigTcDBSrv.ipTcResult;
        nState := 255;
    ELSE
        nState := 0;
    END_IF
END_IF
```

6.1.读取

1.2.

1.2

该方法可以用于读取 TwinCAT 数据库服务器的当前配置。任何可能被包含的临时配置都会对其进行相应的标记。

语法

```
METHOD Read : BOOL
VAR_INPUT
    pDBConfig: POINTER TO ARRAY [1..MAX_CONFIGURATIONS] OF ST_ConfigDB;
    cbDBConfig: UDINT;
    pAutoLogGrpConfig: POINTER TO ARRAY[1..MAX_CONFIGURATIONS] OF
ST_ConfigAutoLogGrp;
    cbAutoLogGrpConfig: UDINT;
    pDBCount: POINTER TO UDINT;
    pAutoLogGrpCount: POINTER TO UDINT;
END_VAR
```

输入

Name	Type	描述
pDBConfig	POINTER TO ARRAY [1..MAX_CONFIGURATIONS [▶ 236]] OF ST_ConfigDB [▶ 219]	用于写入数据库配置的数组的指针地址。
cbDBConfig	UDINT	数据库配置数组的长度
pAutoLogGrpConfig	POINTER TO ARRAY[1..MAX_CONFIGURATIONS [▶ 236]] OF ST_ConfigAutoLogGrp [▶ 218]	用于写入 AutoLogGrp 配置的数组的指针地址。
cbAutoLogGrpConfig	UDINT	AutoLogGrp 配置数组的长度
pDBCount	POINTER TO UDINT	用于存储数据库配置数量的指针地址。
pAutoLogGrpCount	POINTER TO UDINT	用于存储 AutoLogGrp 配置数量的指针地址。

返回值

Name	Type	描述
读取	BOOL	显示方法的状态。在方法执行完毕时，即使出现错误，也会返回 TRUE。

示例

```
VAR
    fbConfigTcDBSrv    : FB_ConfigTcDBSrvEvt(sNetId := '', tTimeout:=T#5S);
    aDBConfig          : ARRAY[0..MAX_CONFIGURATIONS] OF ST_ConfigDB;
    aAutoGrpConfig     : ARRAY[0..MAX_CONFIGURATIONS] OF ST_ConfigAutoLogGrp;
    nDbCount           : UDINT;
    nAutoGrpCount      : UDINT;
    tcMessage          : I_TcMessage;
END_VAR
```

```
IF fbConfigTcDBSrv.Read(
    pDBConfig := ADR(aDBConfig),
    cbDBConfig := SIZEOF(aDBConfig),
    pAutologGrpConfig := ADR(aAutoGrpConfig),
    cbAutoLogGrpConfig := SIZEOF(aAutoGrpConfig),
    pDBCount := ADR(nDbCount),
    pAutoLogGrpCount := ADR(nAutoGrpCount))
 THEN
    IF fbConfigTcDBSrv.bError THEN
        tcMessage := fbConfigTcDBSrv.ipTcResult;
        nState := 255;
    ELSE
        nState := 0;
    END_IF
END_IF
```

6.1.删除

1.2.

1.3

该方法可以用于从 TwinCAT 数据库服务器的配置中删除数据库和 AutoLog 组。

语法

```
METHOD Delete : BOOL
VAR_INPUT
    eTcDBSrvConfigType: E_TcDBSrvConfigType;
    hConfigID: UDINT;
END_VAR
```

输入

Name	Type	描述
eTcDBSrvConfigType	E_TcDBSrvConfigType	要删除的配置的类型（数据库/AutoLog 组）
hConfigID	UDINT	要删除的配置的 ID（hDBID 或 hAutoLogGrpID）

返回值

Name	Type	描述
删除	BOOL	显示方法的状态。在方法执行完毕时，即使出现错误，也会返回 TRUE。

示例

```
VAR
    fbConfigTcDBSrv : FB_ConfigTcDBSrvEvt(sNetId := '', tTimeout:=T#5S);
    myConfigHandle  : INT;
    tcMessage       : I_TcMessage;
END_VAR
```

```
IF fbConfigTcDBSrv.Delete(
    eTcDBSrvConfigType := E_TcDBSrvConfigType.Database,
    hConfigID := myConfigHandle) THEN
IF fbConfigTcDBSrv.bError THEN
        tcMessage := fbConfigTcDBSrv.ipTcResult;
        nState := 255;
    ELSE
        nState := 0;
    END_IF
END_IF
```

6.1.1.2.2 FB_PLCDBAutoLogEvt

功能块具有 4 种方法，用于启动和停止已定义的 AutoLog 组，以及读取相应的组状态。

语法

定义：

```
FUNCTION_BLOCK FB_PLCDBAutoLogEvt
VAR_INPUT
    sNetID: T_AmsNetID := '';
    tTimeout: TIME := T#5S;
END_VAR
VAR_OUTPUT
    bBusy: BOOL;
    bError: BOOL;
    ipTcResult: Tc3_EventLogger.I_TcMessage;
    bBusy_Status: BOOL;
END_VAR
```

输入

名称	类型	描述
sNetID	T_AmsNetID	ADS 命令指向的目标设备的 AMS 网络 ID。
tTimeout	TIME	表示函数被取消前的时间。

输出

名称	类型	描述
bBusy	BOOL	除Status方法外，功能块的方法处于活动状态时为 TRUE。
bError	BOOL	发生错误时即为 TRUE。
ipTcResult	Tc3_EventLogger.I_TcMessage	结果界面，提供有关返回值的详细信息。
bBusy_Status	BOOL	Status方法可以独立于功能块的其他 3 种方法执行，因此该方法有自己独立的繁忙标志。Status 方法处于活动状态时为 TRUE。

方法

名称	定义位置	描述
RunOnce [▶ 164]	本地	执行一次 AutoLog 组
启动 [▶ 165]	本地	使用相应配置的 AutoLog 组启动 AutoLog 模式
Status [▶ 165]	本地	查询 AutoLog 组的状态。
停止 [▶ 166]	本地	停止 AutoLog 模式

要求

开发环境	目标平台	要引入的 PLC 库
TwinCAT v3.1 版本 4022.20	PC 或 CX（x86）	Tc3_Database

6.1.RunOnce
1.2.
2.1

该方法可以用于执行一次 AutoLog 组，例如，基于控制器中的一个事件。

语法

```
METHOD RunOnce : BOOL
VAR_INPUT
    hAutoLogGrpID: UDINT;
    bAll: BOOL;
END_VAR
```

输入

名称	类型	描述
hAutoLogGrpID	UDINT	要执行一次的 AutoLog 组的 ID。
bAll	BOOL	如果为 TRUE，则所有 AutoLog 组都执行一次。

返回值

名称	类型	描述
RunOnce	BOOL	显示方法的状态。在方法执行完毕时，即使出现错误，也会返回 TRUE。

示例

```
VAR
    fbPLCDBAutoLog    : FB_PLCDBAutoLogEvt (sNetID:='', tTimeout := T#5S);
END_VAR
```

```
IF fbPLCDBAutoLog.RunOnce(hAutologGrpID := 1, bAll := FALSE) THEN
    ; // ...
END_IF
```

6.1.启动
1.2.
2.2

该方法使用相应配置的 AutoLog 组启动 AutoLog 模式。

语法

```
METHOD Start : BOOL
```

返回值

名称	类型	描述
启动	BOOL	显示方法的状态。在方法执行完毕时，即使出现错误，也会返回 TRUE。

示例

```
VAR
    fbPLCDBAutoLog    : FB_PLCDBAutoLogEvt (sNetID:='', tTimeout := T#5S);
END_VAR
```

```
IF fbPLCDBAutoLog.Start() THEN
    ; // ...
END_IF
```

6.1.Status
1.2.
2.3

该方法可以用于查询 AutoLog 组的状态。由于该方法可以独立于功能块的其他方法调用，因此在功能块主体中为该方法提供了一个单独的繁忙标志：bBusy_Status。

语法

```
METHOD Status : BOOL
VAR_INPUT
    tCheckCycle: TIME;
    pError: POINTER TO BOOL;
    pAutoLogGrpStatus: POINTER TO ARRAY [1..MAX_CONFIGURATIONS] OF ST_AutoLogGrpStatus;
    cbAutoLogGrpStatus: UDINT;
END_VAR
```

输入

名称	类型	描述
tCheckCycle	TIME	更新状态数组的间隔时间。
pError	POINTER TO BOOL	如果在 AutoLog 模式下发生错误，则为 TRUE。
pAutoLogStatus	POINTER TO ARRAY [1..MAX_CONFIGURATIONS [▶ 236]] OF ST_AutoLogGrpStatus [▶ 233]	包含所有组的状态数组的地址。
cbAutoLogStatus	UDINT	状态数组的长度

返回值

名称	类型	描述
Status	BOOL	显示方法的状态。在方法执行完毕时，即使出现错误，也会返回 TRUE。

示例

```
VAR
    fbPLCDBAutoLog    : FB_PLCDBAutoLogEvt(sNetID:='', tTimeout := T#5S);
    bError            : BOOL;
    aAutologGrpStatus : ARRAY[0..MAX_CONFIGURATIONS] OF ST_AutoLogGrpStatus;
END_VAR
```

```
IF fbPLCDBAutoLog.Status(tCheckCycle := T#30S, ADR(bError), ADR(aAutologGrpStatus), SIZEOF(aAutologG
rpStatus)) THEN
    ; // ...
END_IF
```

6.1.停止

1.2.

2.4

该方法可停止 AutoLog 模式。

语法

```
METHOD Stop : BOOL
```

返回值

名称	类型	描述
停止	BOOL	显示方法的状态。在方法执行完毕时，即使出现错误，也会返回 TRUE。

示例

```
VAR
    fbPLCDBAutoLog    : FB_PLCDBAutoLogEvt (sNetID:='', tTimeout := T#5S);
END_VAR
```

```
IF fbPLCDBAutoLog.Stop() THEN
    ; // ...
END_IF
```

6.1.1.2.3 FB_PLCDBCreateEvt

具有 2 种方法的功能块。一种可以用于在 PLC 中指定的数据库服务器上从 PLC 创建数据库的方法。另一种方法可用于在指定的数据库中生成一个新的表格。

语法

定义：

```
FUNCTION_BLOCK FB_PLCDBCreateEvt
VAR_INPUT
    sNetID: T_AmsNetID := '';
    tTimeout: TIME := T#5S;
END_VAR
VAR_OUTPUT
    bBusy: BOOL;
    bError: BOOL;
    ipTcResult: Tc3_EventLogger.I_TcMessage
END_VAR
```

输入

Name	Type	描述
sNetID	T_AmsNetID	ADS 命令指向的目标设备的 AMS 网络 ID。
tTimeout	TIME	表示函数被取消前的时间。

输出

Name	Type	描述
bBusy	BOOL	功能块的方法处于活动状态时即为 TRUE。
bError	BOOL	发生错误时即为 TRUE。
ipTcResult	Tc3_EventLogger.I_TcMessage [▶ 215]	TwinCAT 3 EventLogger 的消息接口，其中提供了有关返回值的详细信息。

属性

Name	Type	Access	描述
eTraceLevel	TcEventSeverity [▶ 216]	获取，设置	指定事件的重要性权重。只有权重高于该值的事件才会被发送到 TwinCAT 系统。

方法

Name	定义位置	描述
数据库 [▶ 167]	本地	创建一个新的数据库
表格 [▶ 168]	本地	通过 PLC 中一个包含 x 个元素（即 x 列）的数组来定义表结构，并创建一个新表。

要求

开发环境	目标平台	要引入的 PLC 库
TwinCAT v3.1 版本 4022.20	PC 或 CX（x86）	Tc3_Database

6.1.数据库
1.2.
3.1

该方法可创建一个新的数据库。您可以指定已创建的数据库是否也应该用于 TwinCAT 数据库服务器的配置。

语法

```
METHOD Database : BOOL
VAR_INPUT
    pDatabaseConfig: POINTER TO BYTE;
    cbDatabaseConfig: UDINT;
    bCreateXMLConfig: BOOL;
    pDBID: POINTER TO UDINT;
END_VAR
```

输入

Name	Type	描述
pDatabaseConfig	POINTER TO BYTE	数据库配置结构 [▶ 219]的地址
cbDatabaseConfig	UDINT	数据库配置结构的长度
bCreateXMLConfig	BOOL	指定是否将新创建的数据库作为新的配置条目录入到 XML 文件。
pDBID	UDINT	如果/当创建了新的配置条目时，则返回 hDBID。

返回值

Name	Type	描述
数据库	BOOL	显示方法的状态。在方法执行完毕时，即使出现错误，也会返回 TRUE。

示例

```
VAR
    fbPLCDBCreate : FB_PLCDBCreateEvt(sNetID := '', tTimeout := T#5S);
    stConfigDB    : T_DBConfig_MsCompactSQL;
    hDBID         : UDINT;
    tcMessage     : I_TcMessage;
END_VAR
```

```
stConfigDB.bAuthentification := FALSE;
stConfigDB.sServer := 'C:\Test.sdf';

IF fbPLCDBCreate.Database(
    pDatabaseConfig:= ADR(stConfigDB),
    cbDatabaseConfig := SIZEOF(stConfigDB),
    bCreateXMLConfig := TRUE,
    pDBID := ADR(hDBID))
THEN
    IF fbPLCDBCreate.bError THEN
        tcMessage := fbPLCDBCreate.ipTcResult;
        nState := 255;
    ELSE
        nState := 0;
    END_IF
END_IF
```

6.1.表格

1.2.

3.2

通过 PLC 中一个包含 x 个元素（即 x 列）的数组来定义表结构，并创建一个新表。

语法

```
METHOD Table : BOOL
VAR_INPUT
    hDBID : UDINT;
    sTableName : T_MaxString;
    pTableCfg : POINTER TO ARRAY[0..MAX_DBCOLUMNS] OF ST_ColumnInfo;
    cbTableCfg : UDINT;
END_VAR
```

输入

Name	Type	描述
hDBID	UDINT	表示要使用的数据库的 ID。
sTableName	MaxString	要创建的表格的名称。
pTableCfg	POINTER TO ARRAY[0..MAX_DBCOLUMNS [▶ 236]] OF ST_ColumnInfo [▶ 234]	表示表结构数组的指针地址。将各个列写入该数组。
cbTableCfg	UDINT	表示配置列信息的数组的长度。

返回值

Name	Type	描述
表格	BOOL	显示方法的状态。在方法执行完毕时，即使出现错误，也会返回 TRUE。

示例

```
VAR
    fbPLCDBCreate : FB_PLCDBCreateEvt(sNetID := '', tTimeout := T#5S);
    ColumnInfo    : ARRAY [0..14] OF ST_ColumnInfo;
    tcMessage     : I_TcMessage;
END_VAR
```

```
ColumnInfo[0].sName := 'colBigInt';     ColumnInfo[0].eType := E_ColumnType.BigInt;     ColumnInfo[0
].nLength := 8;     ColumnInfo[0].sProperty := 'IDENTITY(1,1)';
ColumnInfo[1].sName := 'colInteger';    ColumnInfo[1].eType := E_ColumnType.Integer;    ColumnInfo[1
].nLength := 4;
ColumnInfo[2].sName := 'colSmallInt';   ColumnInfo[2].eType := E_ColumnType.SmallInt;   ColumnInfo[2
].nLength := 2;
ColumnInfo[3].sName := 'colTinyInt';    ColumnInfo[3].eType := E_ColumnType.TinyInt;    ColumnInfo[3
].nLength := 1;
ColumnInfo[4].sName := 'colBit';        ColumnInfo[4].eType := E_ColumnType.BIT_;       ColumnInfo[4
].nLength := 1;
```

```
ColumnInfo[5].sName := 'colMoney';      ColumnInfo[5].eType := E_ColumnType.Money;      ColumnInfo[5
].nLength := 8;
ColumnInfo[6].sName := 'colFloat';      ColumnInfo[6].eType := E_ColumnType.Float;      ColumnInfo[6
].nLength := 8;
ColumnInfo[7].sName := 'colReal';       ColumnInfo[7].eType := E_ColumnType.REAL_;      ColumnInfo[7
].nLength := 4;
ColumnInfo[8].sName := 'colDateTime';   ColumnInfo[8].eType := E_ColumnType.DateTime;   ColumnInfo[8
].nLength := 4;
ColumnInfo[9].sName := 'colNText';      ColumnInfo[9].eType := E_ColumnType.NText;      ColumnInfo[9
].nLength := 256;
ColumnInfo[10].sName := 'colNChar';     ColumnInfo[10].eType := E_ColumnType.NChar;     ColumnInfo[1
0].nLength := 10;
ColumnInfo[11].sName := 'colImage';     ColumnInfo[11].eType := E_ColumnType.Image;     ColumnInfo[1
1].nLength := 256;
ColumnInfo[12].sName := 'colNVarChar';  ColumnInfo[12].eType := E_ColumnType.NVarChar;  ColumnInfo[1
2].nLength := 50;
ColumnInfo[13].sName := 'colBinary';    ColumnInfo[13].eType := E_ColumnType.Binary;    ColumnInfo[1
3].nLength := 30;
ColumnInfo[14].sName := 'colVarBinary'; ColumnInfo[14].eType := E_ColumnType.VarBinary; ColumnInfo[1
4].nLength := 20;

IF fbPLCDBCreate.Table(
    hDBID:= 1,
    sTableName:= 'myNewTable',
    pTableCfg:= ADR(ColumnInfo),
    cbTableCfg:= SIZEOF(ColumnInfo))
THEN
    IF fbPLCDBCreate.bError THEN
        TcMessage:= fbPLCDBCreate.ipTcResult;
        nState := 255;
    ELSE
        nState := 0;
    END_IF
END_IF
```

6.1.1.2.4 FB_PLCDBReadEvt

用于从数据库读取记录的功能块。

语法

定义：

```
FUNCTION_BLOCK FB_PLCDBReadEvt
VAR_INPUT
    sNetID: T_AmsNetID := '';
    tTimeout: TIME := T#5S;
END_VAR
VAR_OUTPUT
    bBusy: BOOL;
    bError: BOOL;
    ipTcResult: Tc3_EventLogger.I_TcMessage
END_VAR
```

输入

Name	Type	描述
sNetID	T_AmsNetID	ADS 命令指向的目标设备的 AMS 网络 ID。
tTimeout	TIME	表示函数被取消前的时间。

输出

Name	Type	描述
bBusy	BOOL	功能块的方法处于活动状态时即为 TRUE。
bError	BOOL	发生错误时即为 TRUE。
ipTcResult	Tc3_EventLogger.I_TcMessage [▶ 215]	TwinCAT 3 EventLogger 的消息接口，其中提供了有关返回值的详细信息。

属性

Name	Type	描述
nRecords	UDINT	依据 sDBSymbolName，输出可以收集的最大记录数。
eTraceLevel	TcEventSeverity [▶ 216]	指定事件的重要性权重。只有权重高于该值的事件才会被发送到 TwinCAT 系统。

Methods

Name	定义位置	描述
读取 [▶ 170]	本地	从采用倍福指定的标准表结构的数据库表中读取指定数量的记录。
ReadStruct [▶ 171]	本地	从采用任何表结构的数据库表中读取指定数量的记录。

要求

开发环境	目标平台	要引入的 PLC 库
TwinCAT v3.1 版本 4022.20	PC 或 CX（x86）	Tc3_Database

6.1.读取

1.2.

4.1

该方法从采用倍福指定的标准表结构的数据库表中读取指定数量的记录。例如，标准表结构可用于 AutoLog 模式和 FB_DBWriteEvt 功能块。

语法

```
METHOD Read : BOOL
VAR_INPUT
    hDBID: UDINT;
    sTableName: T_MaxString;
    sDBSymbolName: T_MaxString;
    eOrderBy: E_OrderColumn := E_OrderColumn.eColumnID;
    eOrderType: E_OrderType := E_OrderType.eOrder_ASC;
    nStartIndex: UDINT;
    nRecordCount: UDINT;
    pData: POINTER TO ST_StandardRecord;
    cbData: UDINT;
END_VAR
```

输入

Name	Type	描述
hDBID	UDINT	表示要使用的数据库的 ID。
sTableName	T_MaxString	要读取的表格的名称。
sDBSymbolName	T_MaxString	要从标准表结构中读取的符号名称。
eOrderBy	E_OrderColumn.eColumnID	排序列（ID、时间戳、名称或值）
eOrderType	E_OrderType.eOrder_ASC	排序方向（ASC 或 DESC）
nStartIndex	UDINT	表示要读取的第一个记录的索引。
nRecordCount	UDINT	表示要读取的记录数。
pData	POINTER TO ST_StandardRecord	要写入记录的结构数组的地址。
cbData	UDINT	表示结构数组的大小，以字节为单位。

返回值

Name	Type	描述
读取	BOOL	显示方法的状态。在方法执行完毕时，即使出现错误，也会返回 TRUE。

示例

```
VAR
    fbPLCDBRead    : FB_PLCDBReadEvt (sNetID := '', tTimeout := T#5S);
    ReadStruct     : ST_StandardRecord;
    tcMessage      : I_TcMessage;
END_VAR
```

```
IF fbPLCDBRead.Read(
    hDBID:= 1,
    sTableName:= 'MyTable_WithLReal',
    sDBSymbolName:= 'MyValue',
    eOrderBy:= E_OrderColumn.ID,
    eOrderType:= E_OrderType.DESC,
    nStartIndex:= 0,
    nRecordCount:= 1,
    pData:= ADR(ReadStruct),
    cbData:= SIZEOF(ReadStruct))
THEN
    IF fbPLCDBRead.bError THEN
        tcMessage := fbPLCDBRead.ipTcResult;
        nState := 255;
    ELSE
        nState := 0;
    END_IF
END_IF
```

PLC 中的结果：

Expression	Type	Value
ReadStruct	ST_StandardRecord	
nID	LINT	2
dtTimestamp	DATE_AND_TIME	DT#2018-2-1-16:8:8
sName	STRING(80)	'MyValue'
rValue	LREAL	15.9

6.1.ReadStruct

1.2.

4.2

此方法可从具有任意表结构的数据库表中读取指定数量的记录。

语法

```
METHOD ReadStruct : BOOL
VAR_INPUT
    hDBID: UDINT;
    sTableName: T_MaxString;
    pColumnNames: POINTER TO ARRAY [0..MAX_DBCOLUMNS] OF STRING(50);
    cbColumnNames: UDINT;
    sOrderByColumn: STRING(50);
    eOrderType: E_OrderType := E_OrderType.eOrder_ASC
    nStartIndex: UDINT;
    nRecordCount: UDINT;
    pData: POINTER TO BYTE;
    cbData: UDINT;
END_VAR
```

输入

Name	Type	描述
hDBID	UDINT	表示要使用的数据库的 ID。
sTableName	T_MaxString	要读取的表格的名称。
pColumnNames	POINTER TO ARRAY [0..MAX_DB COLUMNS] OF STRING(50)	包含待读取列名的数组的地址。
cbColumnNames	UDINT	列名数组的长度
sOrderByColumn	STRING(50)	指定排序列的名称。
eOrderType	E_OrderType	排序方向（ASC 或 DESC）
nStartIndex	UDINT	表示要读取的第一个记录的索引。
nRecordCount	UDINT	表示要读取的记录数。
pData	POINTER TO BYTE	要写入记录的结构数组的地址。
cbData	UDINT	表示结构数组的大小，以字节为单位。

返回值

Name	Type	描述
ReadStruct	BOOL	显示方法的状态。在方法执行完毕时，即使出现错误，也会返回 TRUE。

示例

```
VAR
    fbPLCDBRead    : FB_PLCDBReadEvt (sNetID := '', tTimeout := T#5S);
    myCustomStruct : ST_Record;
    tcMessage      : I_TcMessage;
END_VAR
```

```
TYPE ST_Record :
STRUCT
    nID        : LINT;
    dtTimestamp: DATE_AND_TIME;
    sName      : STRING;
    nSensor1   : LREAL;
    nSensor2   : LREAL;
END_STRUCT
END_TYPE
```

```
// set columnnames
ColumnNames[0] := 'ID';
ColumnNames[1] := 'Timestamp';
ColumnNames[2] := 'Name';
ColumnNames[3] := 'Sensor1';
ColumnNames[4] := 'Sensor2';

IF fbPLCDBRead.ReadStruct(
    hDBID:= 1,
    sTableName:= 'MyTable_Struct',
    pColumnNames:= ADR(ColumnNames),
    cbColumnNames:= SIZEOF(ColumnNames),
    sOrderByColumn:= ColumnNames[0],
    eOrderType:= E_OrderType.DESC,
    nStartIndex:= 0,
    nRecordCount:= 1,
    pData:= ADR(myCustomStruct),
    cbData:= SIZEOF(myCustomStruct))
THEN
    IF fbPLCDBRead.bError THEN
        tcMessage:= fbPLCDBRead.ipTcResult;
        nState := 255;
    ELSE
        nState := 0;
    END_IF
END_IF
```

PLC 中的结果：

Expression	Type	Value
myCustomStruct	ST_Record	
nID	LINT	1
dtTimestamp	DATE_AND_TIME	DT#2018-2-1-15:17:54
sName	STRING	'MyStructVal'
nSensor1	LREAL	12.34
nSensor2	LREAL	102.5

6.1.1.2.5 FB_PLCDBWriteEvt

用于将记录写入数据库的功能块。

语法

定义：

```
FUNCTION_BLOCK FB_PLCDBWriteEvt
VAR_INPUT
    sNetID: T_AmsNetID := '';
    tTimeout: TIME := T#5S;
END_VAR
VAR_OUTPUT
    bBusy: BOOL;
    bError: BOOL;
    ipTcResult: Tc3_EventLogger.I_TcMessage;
END_VAR
```

输入

Name	Type	描述
sNetID	T_AmsNetID	ADS 命令指向的目标设备的 AMS 网络 ID。
tTimeout	TIME	表示函数被取消前的时间。

输出

Name	Type	描述
bBusy	BOOL	功能块的方法处于活动状态时即为 TRUE。
bError	BOOL	发生错误时即为 TRUE。
ipTcResult	Tc3_EventLogger.I_TcMessage [▶ 215]	TwinCAT 3 EventLogger 的消息接口，其中提供了有关返回值的详细信息。

属性

Name	Type	Access	描述
eTraceLevel	TcEventSeverity [▶ 216]	获取，设置	指定事件的重要性权重。只有权重高于该值的事件才会被发送到 TwinCAT 系统。

方法

Name	定义位置	描述
Write [▶ 174]	本地	在倍福指定的标准表结构中创建一条记录。
WriteBySymbol [▶ 175]	本地	读取指定的 ADS 符号的值，并将其保存在倍福指定的标准表结构中。
WriteStruct [▶ 176]	本地	创建具有任何表结构的记录。

要求

开发环境	目标平台	要引入的 PLC 库
TwinCAT v3.1 版本 4022.20	PC 或 CX（x86）	Tc3_Database

6.1.Write
1.2.
5.1

该方法在倍福指定的标准表结构中创建记录。

语法

```
METHOD Write : BOOL
VAR_INPUT
    hDBID: UDINT;
    sTableName: T_MaxString;
    pValue: POINTER TO BYTE;
    cbValue: UDINT;
    sDBSymbolName: T_MaxString;
    eDBWriteMode: E_WriteMode := E_WriteMode.eADS_TO_DB_Append;
    nRingBuffParameter: UDINT;
END_VAR
```

输入

Name	Type	描述
hDBID	UDINT	表示要使用的数据库的 ID。
sTableName	T_MaxString	要读取的表格的名称。
pValue	POINTER TO BYTE	在标准表结构中要记录的变量的地址。
cbValue	UDINT	要记录的变量的长度。
sDBSymbolName	T_MaxString	在表格中记录的名称。
eDBWriteMode	E_WriteMode	表示写入模式。（添加、更新、环形缓冲区）
nRingBuffParameter	UDINT	“环形缓冲区”写入模式的附加参数。

返回值

Name	Type	描述
Write	BOOL	显示方法的状态。在方法执行完毕时，即使出现错误，也会返回 TRUE。

示例

本示例显示了如何使用 FB_PLCDBWriteEvt.Write 方法：

```
VAR
    fbPLCDBWrite    : FB_PLCDBWriteEvt(sNetID := '', tTimeout := T#5S);
    myValue         : LREAL := 43.23;
    tcMessage       : I_TcMessage;
END_VAR
```

```
IF fbPLCDBWrite.Write(
    hDBID:= 1,
    sTableName:= 'myTable_WithLReal',
    pValue:= ADR(myValue),
    cbValue:= SIZEOF(myValue),
    sDBSymbolName:= 'MyValue',
```

```
    eDBWriteMode:= E_WriteMode.eADS_TO_DB_RingBuff_Count,
    nRingBuffParameter:= 3)
THEN
    IF fbPLCDBWrite.bError THEN
        tcMessage := fbPLCDBWrite.ipTcResult;
        nState := 255;
    ELSE
        nState := 0;
    END_IF
END_IF
```

数据库中的结果：

ID	Timestamp	Name	值
27	已丢弃		
28	'2018-01-30 14:04:19'	'MyValue'	41.23
29	'2018-01-30 14:04:29'	'MyValue'	42.23
30	'2018-01-30 14:04:39'	'MyValue'	43.23

使用环形缓冲区选项时，在任何一个时间在数据库中只能存在 3 个此名称的条目。较旧的条目会被删除。

6.1.WriteBySymbol

1.2.

5.2

该方法读取指定的 ADS 符号的值，并将其保存在倍福指定的标准表结构中。此外，还可以读取其他 ADS 设备中的 ADS 符号。

语法

```
METHOD WriteBySymbol : BOOL
VAR_INPUT
    hDBID: UDINT;
    sTableName: T_MaxString;
    stADSDevice: ST_ADSDevice;
    stSymbol: ST_Symbol;
    eDBWriteMode: E_WriteMode := E_WriteMode.eADS_TO_DB_Append;
    nRingBuffParameter: UDINT;
END_VAR
```

输入

Name	Type	描述
hDBID	UDINT	表示要使用的数据库的 ID。
sTableName	T_MaxString	要读取的表格的名称。
stADSDevice	ST_ADSDevice	要在标准表结构中记录符号的 ADS 设备。
stSymbol	ST_Symbol	要写入的变量的符号名称
eDBWriteMode	E_WriteMode	表示写入模式。（添加、更新、环形缓冲区）
nRingBuffParameter	UDINT	“环形缓冲区”写入模式的额外参数

返回值

Name	Type	描述
WriteBySymbol	BOOL	显示方法的状态。在方法执行完毕时，即使出现错误，也会返回 TRUE。

示例

本示例展示了如何使用 FB_PLCDBWriteEvt.WriteBySymbol 方法：

```
VAR
    fbPLCDBWrite    :  FB_PLCDBWriteEvt(sNetID := '', tTimeout := T#5S);
    myValue         :  LREAL := 43.23;
    myAdsDevice     :  ST_ADSDevice;
    mySymbol        :  ST_Symbol;
    tcMessage       :  I_TcMessage;
END_VAR
```

```
// Set ADSDevice Information
myAdsDevice.sDevNetID     := '127.0.0.1.1.1';
myAdsDevice.nDevPort      := 851;
myAdsDevice.eADSRdWrtMode := E_ADSRdWrtMode.bySymbolName;
myAdsDevice.tTimeout      := T#5S;

// Set Symbol Information
mySymbol.eDataType        := E_PLCDataType.eType_LREAL;
mySymbol.sDBSymbolName    := 'MySymbol';
mySymbol.sSymbolName      := 'MAIN.myValue';
mySymbol.nBitSize         := 8;

// Call Functionblock
IF fbPLCDBWrite.WriteBySymbol(
    hDBID:= 1,
    sTableName:= 'myTable_WithLReal',
    stADSDevice:= myAdsDevice,
    stSymbol:= mySymbol,
    eDBWriteMode:= E_WriteMode.eADS_TO_DB_Append,
    nRingBuffParameter:= 1)
THEN
    IF fbPLCDBWrite.bError THEN
        tcMessage := fbPLCDBWrite.ipTcResult;
        nState := 255;
    ELSE
        nState := 0;
    END_IF
END_IF
```

数据库中的结果：

ID	Timestamp	Name	值
28	'2018-01-30 14:04:19'	'MyValue'	41.23
29	'2018-01-30 14:04:29'	'MyValue'	42.23
30	'2018-01-30 14:04:39'	'MyValue'	43.23
31	'2018-01-30 14:06:12’	‘MySymbol’	86.2

6.1.WriteStruct

1.2.

5.3

该方法创建具有可自由选择的表结构的记录。

语法

```
METHOD WriteStruct : BOOL
VAR_INPUT
    hDBID: UDINT;
    sTableName: T_MaxString;
    pRecord: POINTER TO BYTE;
    cbRecord: UDINT;
    pColumnNames: POINTER TO ARRAY [0..MAX_DBCOLUMNS] OF STRING(50);
    cbColumnNames: UDINT;
END_VAR
```

输入

Name	Type	描述
hDBID	UDINT	表示要使用的数据库的 ID。
sTableName	T_MaxString	要读取的表格的名称。
pRecord	POINTER TO BYTE	在可自由选择的表结构中要记录的结构体的地址。
cbRecord	UDINT	要写入的结构的长度
pColumnNames	POINTER TO ARRAY [0 ..MAX_DBCOLUMNS] OF STRING(50)	包含要填写的列名的数组的地址。
cbColumnNames	UDINT	列名数组的长度

返回值

Name	Type	描述
WriteStruct	BOOL	显示方法的状态。在方法执行完毕时，即使出现错误，也会返回 TRUE。

示例

本示例显示了如何使用方法 FB_PLCDBWriteEvt.WriteStruct：

```
VAR
    fbPLCDBWrite    :  FB_PLCDBWriteEvt(sNetID := '', tTimeout := T#5S);
    myRecord        :  ST_Record;
    ColumnNames     :  ARRAY[0..4] OF STRING(50);

    systime         :  GETSYSTEMTIME;
    currentTime     :  T_FILETIME;
    tcMessage       :  I_TcMessage;
END_VAR
```

```
TYPE ST_Record :
STRUCT
    nID        : LINT;
    dtTimestamp: DATE_AND_TIME;
    sName      : STRING;
    nSensor1   : LREAL;
    nSensor2   : LREAL;
END_STRUCT
END_TYPE
```

```
// set Values
systime(timeLoDw => currentTime.dwLowDateTime, timeHiDW => currentTime.dwHighDateTime );
myRecord.dtTimestamp := FILETIME_TO_DT(currentTime);
myRecord.sName       := 'MyStructVal';
myRecord.nSensor1    := 12.34;
myRecord.nSensor2    := 102.5;

// set columnnames
ColumnNames[0] := 'ID';
ColumnNames[1] := 'Timestamp';
ColumnNames[2] := 'Name';
ColumnNames[3] := 'Sensor1';
ColumnNames[4] := 'Sensor2';

// Call Functionblock
IF fbPLCDBWrite.WriteStruct(
    hDBID:= 1,
    sTableName:= 'myTable_Struct',
    pRecord:= ADR(myRecord),
    cbRecord:= SIZEOF(myRecord),
    pColumnNames:= ADR(ColumnNames) ,
    cbColumnNames:= SIZEOF(ColumnNames))
THEN
    IF fbPLCDBWrite.bError THEN
        tcMessage := fbPLCDBWrite.ipTcResult;
        nState := 255;
    ELSE
        nState := 0;
    END_IF
END_IF
```

数据库中的结果：

ID	Timestamp	Name	Sensor1	Sensor2
5	'2018-01-30 15:23:26'	‘MyStructVal’	12.34	102.5

6.1.1.2.6 FB_PLCDBCmdEvt

具有 2 种方法的功能块。用户可以定义并传输自己的 SQL 命令。SQL 命令中的占位符可以与 PLC 中的结构相关联，从而反映表结构。数据库服务器将结构的当前数据输入 SQL 命令。

语法

定义：

```
FUNCTION_BLOCK FB_PLCDBCmdEvt
VAR_INPUT
    sNetID: T_AmsNetID := '';
    tTimeout: TIME := T#5S;
END_VAR
VAR_OUTPUT
    bBusy: BOOL;
    bError: BOOL;
    ipTcResult: Tc3_EventLogger.I_TcMessage
END_VAR
```

输入

Name	Type	描述
sNetID	T_AmsNetID	ADS 命令指向的目标设备的 AMS 网络 ID。
tTimeout	TIME	表示函数被取消前的时间。

输出

Name	Type	描述
bBusy	BOOL	功能块的方法处于活动状态时即为 TRUE。
bError	BOOL	发生错误时即为 TRUE。
ipTcResult	Tc3_EventLogger.I_TcMessage [▶ 215]	TwinCAT 3 EventLogger 的消息接口，其中提供了有关返回值的详细信息。

属性

Name	Type	Access	描述
eTraceLevel	TcEventSeverity [▶ 216]	获取，设置	指定事件的重要性权重。只有权重高于该值的事件才会被发送到 TwinCAT 系统。

Methods

Name	定义位置	描述
执行 [▶ 178]	本地	向数据库发送任何 SQL 命令。 无法读取返回的记录。
ExecuteDataReturn [▶ 180]	本地	向数据库发送任何 SQL 命令。 可以读取指定数量的记录。

要求

开发环境	目标平台	要引入的 PLC 库
TwinCAT v3.1 版本 4022.20	PC 或 CX（x86）	Tc3_Database

6.1.执行

1.2.

6.1

该方法可以用于向数据库发送 SQL 命令。每次调用时都会打开数据库连接，然后再次关闭。在命令中可以定义占位符，在执行之前，TwinCAT 数据库服务器会使用相应的值替换这些占位符。无法读取返回的记录。

语法

```
METHOD Execute : BOOL
VAR_INPUT
    hDBID: UDINT;
    pExpression: POINTER TO BYTE;
    cbExpression: UDINT;
    pData: POINTER TO BYTE;
    cbData: UDINT;
    pParameter: POINTER TO ARRAY[0..MAX_DBCOLUMNS] OF ST_ExpParameter;
    cbParameter: UDINT;
END_VAR
```

示例

```
VAR
    fbPLCDBCmd    : FB_PLCDBCmdEvt(sNetID := '', tTimeout := T#5S);
    sCmd          : STRING (1000);
    myStruct      : ST_DataAll;
    aPara         : ARRAY[0..14] OF ST_ExpParameter;
    tcMessage     : I_TcMessage;
END_VAR
```

```
TYPE ST_DataAll :
STRUCT
    colBigInt: LINT;
    colInteger: DINT;
    colSmallInt: INT;
    colTinyInt: BYTE;
    colBit: BOOL;
    colMoney: LREAL;
    colFloat: LREAL;
    colReal: REAL;
    colDateTime: DT;
    colNText: STRING(255);
    colNChar: STRING(10);
    colImage: ARRAY[0..255] OF BYTE;
    colNVarChar: STRING(50);
    colBinary: ARRAY[0..29] OF BYTE;
    colVarBinary: ARRAY[0..19] OF BYTE;
END_STRUCT
END_TYPE
```

```
// set Parameter configuration
aPara[0].sParaName := 'colBigInt';    aPara[0].eParaType :=
E_ExpParameterType.Int64;      aPara[0].nParaSize := 8;
aPara[1].sParaName := 'colInteger';   aPara[1].eParaType :=
E_ExpParameterType.Int32;      aPara[1].nParaSize := 4;
aPara[2].sParaName := 'colSmallInt';  aPara[2].eParaType :=
E_ExpParameterType.Int16;      aPara[2].nParaSize := 2;
aPara[3].sParaName := 'colTinyInt';   aPara[3].eParaType :=
E_ExpParameterType.Byte_;      aPara[3].nParaSize := 1;
aPara[4].sParaName := 'colBit';       aPara[4].eParaType :=
E_ExpParameterType.Boolean;    aPara[4].nParaSize := 1;
aPara[5].sParaName := 'colMoney';     aPara[5].eParaType :=
E_ExpParameterType.Double64;   aPara[5].nParaSize := 8;
aPara[6].sParaName := 'colFloat';     aPara[6].eParaType :=
E_ExpParameterType.Double64;   aPara[6].nParaSize := 8;
aPara[7].sParaName := 'colReal';      aPara[7].eParaType :=
E_ExpParameterType.Float32;    aPara[7].nParaSize := 4;
aPara[8].sParaName := 'colDateTime';  aPara[8].eParaType :=
E_ExpParameterType.DateTime;   aPara[8].nParaSize := 4;
aPara[9].sParaName := 'colNText';     aPara[9].eParaType :=
E_ExpParameterType.STRING_;    aPara[9].nParaSize := 256;
aPara[10].sParaName:= 'colNChar';     aPara[10].eParaType :=
E_ExpParameterType.STRING_;   aPara[10].nParaSize := 10;
aPara[11].sParaName:= 'colImage';     aPara[11].eParaType :=
E_ExpParameterType.ByteArray; aPara[11].nParaSize := 256;
aPara[12].sParaName:= 'colNVarChar';  aPara[12].eParaType :=
E_ExpParameterType.STRING_;   aPara[12].nParaSize := 50;
aPara[13].sParaName:= 'colBinary';    aPara[13].eParaType :=
E_ExpParameterType.ByteArray; aPara[13].nParaSize := 30;
aPara[14].sParaName:= 'colVarBinary'; aPara[14].eParaType :=
E_ExpParameterType.ByteArray; aPara[14].nParaSize := 20;

// set command
sCmd := 'INSERT INTO MyTableName (colInteger, colSmallInt, colTinyInt, colBit, colMoney, colFloat,
colReal, colDateTime, colNText, colNChar, colImage, colNVarChar, colBinary, colVarBinary) VALUES
({colInteger}, {colSmallInt}, {colTinyInt}, {colBit}, {colMoney}, {colFloat}, {colReal},
{colDateTime}, {colNText}, {colNChar}, {colImage}, {colNVarChar}, {colBinary}, {colVarBinary})';

// call functionblock
IF fbPLCDBCmd.Execute(
    hDBID:= 1,
    pExpression:= ADR(sCmd),
    cbExpression:= SIZEOF(sCmd),
    pData:= ADR(myStruct),
    cbData:= SIZEOF(myStruct),
    pParameter:= ADR(aPara),
    cbParameter:= SIZEOF(aPara))
THEN
    IF fbPLCDBCmd.bError THEN
        tcMessage := fbPLCDBCmd.ipTcResult;
        nState := 255;
```

```
    ELSE
        nState := 0;
    END_IF
END_IF
```

6.1.ExecuteDataReturn

1.2.

6.2

该方法可以用于向数据库发送 SQL 命令。每次调用时都会打开数据库连接，然后再次关闭。在命令中可以定义占位符，在执行之前，TwinCAT 数据库服务器会使用相应的值替换这些占位符。可以读取指定数量的记录。

语法

```
METHOD ExecuteDataReturn : BOOL
VAR_INPUT
    hDBID: UDINT;
    pExpression: POINTER TO BYTE;
    cbExpression: UDINT;
    pData: POINTER TO BYTE;
    cbData: UDINT;
    pParameter: POINTER TO ARRAY[0..MAX_DBCOLUMNS] OF ST_ExpParameter;
    cbParameter: UDINT;
    nStartIndex: UDINT;
    nRecordCount: UDINT;
    pReturnData: POINTER TO BYTE;
    cbReturnData: UDINT;
    pRecords: POINTER TO UDINT;
END_VAR
```

输入

Name	Type	描述
hDBID	UDINT	表示要使用的数据库的 ID。
pExpression	POINTER TO BYTE	包含 SQL 命令的字符串变量的地址
cbExpression	UDINT	使用 SQL 命令的字符串变量的长度
pData	POINTER TO BYTE	带有参数值的结构的地址
cbData	UDINT	带有参数值的结构的长度
pParameter	POINTER TO ARRAY[0..MAX_DBCOLUMNS] OF ST_ExpParameter	带有参数信息的结构数组的地址
cbParameter	UDINT	带有参数信息的结构数组的长度
nStartIndex	UDINT	表示要读取的第一个记录的索引。
nRecordCount	UDINT	表示要读取的记录数。
pReturnData	POINTER TO BYTE	要写入记录的结构数组的地址。
cbReturnData	UDINT	表示结构数组的大小，以字节为单位。
pRecords	POINTER TO BYTE	读取的记录数。

返回值

Name	Type	描述
ExecuteDataReturn	BOOL	显示方法的状态。在方法执行完毕时，即使出现错误，也会返回 TRUE。

对命令进行参数设置

在 SQL 命令中的大括号内指定各个参数的列名。
示例：‚SELECT * FROM MyHouse_Temperatures WHERE Room = {SelectedRoom}‘ 。
因此，在结构 ST_ExpParameter 中必须指定 SelectedRoom 作为参数名称。

一些数据库不支持 SQL 语句的参数设置。（TOP/LIMIT/ROWNUM/...）通常不支持可参数化的表格名称。

示例

```
VAR
    fbPLCDBCmd       : FB_PLCDBCmdEvt(sNetID := '', tTimeout := T#5S);
    sCmd             : STRING (1000);
    stPara           : ST_ExpParameter;
    RecordAmt        : ULINT := 3;
    ReturnDataStruct : ARRAY [0..9] OF ST_DataAll;
    nRecords         : UDINT;
    tcMessage        : I_TcMessage;
END_VAR
```

```
// set Parameter configuration
stPara.eParaType := E_ExpParameterType.Int64;
stPara.nParaSize := 8;
stPara.sParaName := 'RecordAmt';

// set command with placeholder
sCmd := 'SELECT TOP ({RecordAmt}) * FROM MyTableName';

// call functionblock
IF fbPLCDBCmd.ExecuteDataReturn(
    hDBID:= 1,
    pExpression:= ADR(sCmd),
    cbExpression:= SIZEOF(sCmd),
    pData:= ADR(RecordAmt),
    cbData:= SIZEOF(RecordAmt),
    pParameter:= ADR(stPara),
    cbParameter:= SIZEOF(stPara),
    nStartIndex:= 0,
    nRecordCount:= 10,
    pReturnData:= ADR(ReturnDataStruct),
    cbReturnData:= SIZEOF(ReturnDataStruct),
    pRecords:= ADR(nRecords))
THEN
    IF fbPLCDBCmd.bError THEN
        tcMessage := fbPLCDBCmd.ipTcResult;
        nState := 255;
    ELSE
        nState := 0;
    END_IF
END_IF
```

6.1.1.3 SQL 专家模式

6.1.1.3.1 FB_ConfigTcDBSrvEvt

用于创建、读取和删除 TwinCAT 数据库服务器的配置条目的功能块。

语法

定义：

```
FUNCTION_BLOCK FB_ConfigTcDBSrvEvt
VAR_INPUT
    sNetID: T_AmsNetID := '';
    tTimeout: TIME := T#5S;
END_VAR
VAR_OUTPUT
    bBusy: BOOL;
    bError: BOOL;
    ipTcResult: Tc3_EventLogger.I_TcMessage;
END_VAR
```

输入

Name	Type	描述
sNetID	T_AmsNetID	ADS 命令指向的目标设备的 AMS 网络 ID。
tTimeout	TIME	表示函数被取消前的时间。

输出

Name	Type	描述
bBusy	BOOL	功能块的方法处于活动状态时即为 TRUE。
bError	BOOL	发生错误时即为 TRUE。
ipTcResult	Tc3_EventLogger.I_TcMessage [▶ 215]	TwinCAT 3 EventLogger 的消息接口，其中提供了有关返回值的详细信息。

属性

Name	Type	Access	描述
eTraceLevel	TcEventSeverity [▶ 216]	获取，设置	指定事件的重要性权重。只有权重高于该值的事件才会被发送到 TwinCAT 系统。

方法

Name	定义位置	描述
创建 [▶ 183]	本地	在 XML 配置文件中为 TwinCAT 数据库服务器创建新条目
读取 [▶ 183]	本地	读取 TwinCAT 数据库服务器的当前配置
删除 [▶ 184]	本地	从 TwinCAT 数据库服务器的配置中删除数据库和 AutoLog 组

要求

开发环境	目标平台	要引入的 PLC 库
TwinCAT v3.1 版本 4022.20	PC 或 CX（x86）	Tc3_Database

6.1.创建

1.3.

1.1

该方法可在 XML 配置文件中为 TwinCAT 数据库服务器创建新条目。TwinCAT 数据库服务器可以选择临时使用新条目。在这种情况下，不会将任何数据写入 XML 文件。

语法

```
METHOD Create : BOOL
VAR_INPUT
    pTcDBSrvConfig: POINTER TO BYTE;
    cbTcDBSrvConfig: UDINT;
    bTemporary: BOOL := TRUE;
    pConfigID: POINTER TO UDINT;
END_VAR
```

输入

Name	Type	描述
pTcDBSrvConfig	POINTER TO BYTE	要创建的配置结构的指针。
cbTcDBSrvConfig	UDINT	配置结构的长度
bTemporary	BOOL	指定是否要将配置存储在 XML 文件中。
pConfigID	POINTER TO UDINT	返回配置 ID（hDBID 或 hAutoLogGrpID）的指针

目前不支持创建 AutoLog 组。

返回值

Name	Type	描述
创建	BOOL	显示方法的状态。在方法执行完毕时，即使出现错误，也会返回 TRUE。

示例

```
VAR
    fbConfigTcDBSrv : FB_ConfigTcDBSrvEvt(sNetId := '', tTimeout:=T#5S);
    myConfigHandle  : INT;
    // Any other ConfigType can be used here
    stConfigDB      : T_DBConfig_MsCompactSQL;
    tcMessage       : I_TcMessage;
END_VAR
```

```
stConfigDB.bAuthentification := FALSE;
stConfigDB.sServer := 'C:\Recipes.sdf';

IF fbConfigTcDBSrv.Create(
    pTcDBSrvConfig:= ADR(stConfigDB),
    cbTcDBSrvConfig:= SIZEOF(stConfigDB),
    bTemporary:= TRUE,
    pConfigID:= ADR(myConfigHandle))
THEN
    IF fbConfigTcDBSrv.bError THEN
        tcMessage := fbConfigTcDBSrv.ipTcResult;
        nState := 255;
    ELSE
        nState := 0;
    END_IF
END_IF
```

6.1.读取

1.3.

1.2

该方法可以用于读取 TwinCAT 数据库服务器的当前配置。任何可能被包含的临时配置都会对其进行相应的标记。

语法

```
METHOD Read : BOOL
VAR_INPUT
    pDBConfig: POINTER TO ARRAY [1..MAX_CONFIGURATIONS] OF ST_ConfigDB;
    cbDBConfig: UDINT;
    pAutoLogGrpConfig: POINTER TO ARRAY[1..MAX_CONFIGURATIONS] OF
ST_ConfigAutoLogGrp;
    cbAutoLogGrpConfig: UDINT;
    pDBCount: POINTER TO UDINT;
    pAutoLogGrpCount: POINTER TO UDINT;
END_VAR
```

输入

Name	Type	描述
pDBConfig	POINTER TO ARRAY [1..MAX_CONFIGURATIONS [▶ 236]] OF ST_ConfigDB [▶ 219]	用于写入数据库配置的数组的指针地址。
cbDBConfig	UDINT	数据库配置数组的长度
pAutoLogGrpConfig	POINTER TO ARRAY[1..MAX_CONFIGURATIONS [▶ 236]] OF ST_ConfigAutoLogGrp [▶ 218]	用于写入 AutoLogGrp 配置的数组的指针地址。
cbAutoLogGrpConfig	UDINT	AutoLogGrp 配置数组的长度
pDBCount	POINTER TO UDINT	用于存储数据库配置数量的指针地址。
pAutoLogGrpCount	POINTER TO UDINT	用于存储 AutoLogGrp 配置数量的指针地址。

返回值

Name	Type	描述
读取	BOOL	显示方法的状态。在方法执行完毕时，即使出现错误，也会返回 TRUE。

示例

```
VAR
    fbConfigTcDBSrv    : FB_ConfigTcDBSrvEvt(sNetId := '', tTimeout:=T#5S);
    aDBConfig          : ARRAY[0..MAX_CONFIGURATIONS] OF ST_ConfigDB;
    aAutoGrpConfig     : ARRAY[0..MAX_CONFIGURATIONS] OF ST_ConfigAutoLogGrp;
    nDbCount           : UDINT;
    nAutoGrpCount      : UDINT;
    tcMessage          : I_TcMessage;
END_VAR
```

```
IF fbConfigTcDBSrv.Read(
    pDBConfig := ADR(aDBConfig),
    cbDBConfig := SIZEOF(aDBConfig),
    pAutologGrpConfig := ADR(aAutoGrpConfig),
    cbAutoLogGrpConfig := SIZEOF(aAutoGrpConfig),
    pDBCount := ADR(nDbCount),
    pAutoLogGrpCount := ADR(nAutoGrpCount))
 THEN
    IF fbConfigTcDBSrv.bError THEN
        tcMessage := fbConfigTcDBSrv.ipTcResult;
        nState := 255;
    ELSE
        nState := 0;
    END_IF
END_IF
```

6.1.删除

1.3.

1.3

该方法可以用于从 TwinCAT 数据库服务器的配置中删除数据库和 AutoLog 组。

语法

```
METHOD Delete : BOOL
VAR_INPUT
    eTcDBSrvConfigType: E_TcDBSrvConfigType;
    hConfigID: UDINT;
END_VAR
```

输入

Name	Type	描述
eTcDBSrvConfigType	E_TcDBSrvConfigType	要删除的配置的类型（数据库/AutoLog 组）
hConfigID	UDINT	要删除的配置的 ID（hDBID 或 hAutoLogGrpID）

返回值

Name	Type	描述
删除	BOOL	显示方法的状态。在方法执行完毕时，即使出现错误，也会返回 TRUE。

示例

```
VAR
    fbConfigTcDBSrv : FB_ConfigTcDBSrvEvt(sNetId := '', tTimeout:=T#5S);
    myConfigHandle  : INT;
    tcMessage       : I_TcMessage;
END_VAR
```

```
IF fbConfigTcDBSrv.Delete(
    eTcDBSrvConfigType := E_TcDBSrvConfigType.Database,
    hConfigID := myConfigHandle) THEN
IF fbConfigTcDBSrv.bError THEN
        tcMessage := fbConfigTcDBSrv.ipTcResult;
        nState := 255;
    ELSE
        nState := 0;
    END_IF
END_IF
```

6.1.1.3.2 FB_SQLDatabaseEvt

用于打开、关闭和管理数据库连接的功能块。

语法

定义：

```
FUNCTION_BLOCK FB_SQLDatabaseEvt
VAR_INPUT
    sNetID: T_AmsNetID := '';
    tTimeout: TIME := T#5S;
END_VAR
VAR_OUTPUT
    bBusy: BOOL;
    bError: BOOL;
    ipTcResult: Tc3_EventLogger.I_TcMessage
END_VAR
```

输入

Name	Type	描述
sNetID	T_AmsNetID	ADS 命令指向的目标设备的 AMS 网络 ID。
tTimeout	TIME	表示函数被取消前的时间。

输出

Name	Type	描述
bBusy	BOOL	功能块的方法处于活动状态时即为 TRUE。
bError	BOOL	发生错误时即为 TRUE。
ipTcResult	Tc3_EventLogger.I_TcMessage [▶ 215]	TwinCAT 3 EventLogger 的消息接口，其中提供了有关返回值的详细信息。

属性

Name	Type	Access	描述
eTraceLevel	TcEventSeverity [▶ 216]	获取，设置	指定事件的重要性权重。只有权重高于该值的事件才会被发送到 TwinCAT 系统。

方法

Name	定义位置	描述
连接 [▶ 186]	本地	打开与已声明数据库的连接。
CreateCmd [▶ 187]	本地	使用功能块 FB_SQLDatabaseEvt 的已打开的数据库连接，对功能块 FB_SQLCommandEvt [▶ 189] 的实例进行初始化。
CreateSP [▶ 187]	本地	使用功能块 FB_SQLDatabaseEvt 的已打开的数据库连接，对功能块 FB_SQLStoredProcedureEvt [▶ 194] 的实例进行初始化。
Disconnect [▶ 188]	本地	关闭该功能块实例打开的数据库连接。

要求

开发环境	目标平台	要引入的 PLC 库
TwinCAT v3.1 版本 4022.20	PC 或 CX（x86）	Tc3_Database

6.1.连接
1.3.
2.1

该方法打开与已声明数据库的连接。

语法

```
METHOD Connect : BOOL
VAR_INPUT
    hDBID: UDINT := 1;
END_VAR
```

输入

Name	Type	描述
hDBID	UDINT	表示要使用的数据库的 ID。

返回值

Name	Type	描述
连接	BOOL	显示方法的状态。在方法执行完毕时，即使出现错误，也会返回 TRUE。

示例

```
VAR
    fbSqlDatabase : FB_SQLDatabaseEvt(sNetID := '', tTimeout := T#5S);
END_VAR
```

```
// open connection
IF fbSqlDatabase.Connect(1) THEN
    IF fbSqlDatabase.bError THEN
        nState := 255;
    ELSE
        nState := nState+1;
    END_IF
END_IF
```

6.1.CreateCmd

1.3.

2.2

该方法可用于使用功能块 FB_SQLDatabase 的已打开的数据库连接，对功能块 FB_SQLCommand 的实例进行初始化。功能块 FB_SQLCommand 仅使用通过 CreateCmd 方法分配的数据库连接。使用相同的数据库连接可以对功能块 FB_SQLCommand 的多个实例进行初始化。

在同一个周期内完成功能块 FB_SQLCommand 的初始化。这意味着既不需要检查功能块的繁忙标志，也不需要检查 CreateCmd 方法的方法返回值。

语法

```
METHOD CreateCmd : BOOL
VAR_INPUT
    pSQLCommand: POINTER TO FB_SQLCommandEvt;
END_VAR
```

输入

Name	Type	描述
pSQLCommand	POINTER TO FB_SQLCommand	返回功能块 FB_SQLCommandEvt 的新实例。

返回值

Name	Type	描述
CreateCmd	BOOL	显示方法的状态。在方法执行完毕时，即使出现错误，也会返回 TRUE。

示例

```
VAR
    fbSqlDatabase : FB_SQLDatabaseEvt(sNetID := '', tTimeout := T#5S);
END_VAR
```

```
// create a command reference
IF fbSqlDatabase.CreateCmd(ADR(fbSqlCommand)) THEN
    IF fbSqlDatabase.bError THEN
        nState := 255;
    ELSE
        nState := nState+1;
    END_IF
END_IF
```

然后可以使用 FB_SQLCommandEvt [▶ 189] 来执行。

6.1.CreateSP

1.3.

2.3

该方法可用于使用功能块 FB_SQLDatabaseEvt 的已打开的数据库连接，对功能块 FB_SQLStoredProcedureEvt 的实例进行初始化。功能块 FB_SQLStoredProcedureEvt 仅使用通过 CreateCmd 方法分配的数据库连接。使用相同的数据库连接可以对功能块 FB_SQLStoredProcedureEvt 的多个实例进行初始化。

功能块 FB_SQLStoredProcedureEvt 的初始化可能需要几个周期。在可以使用功能块之前，必须检查功能块的繁忙标志或 CreateCmd 方法的方法返回值。

语法

```
METHOD CreateSP : BOOL
VAR_INPUT
    sProcedureName: T_MaxString;
    pParameterInfo: POINTER TO ARRAY [0..MAX_SPPARAMETER] OF ST_SQLSPParameter;
    cbParameterInfo: UDINT;
    pSQLProcedure: POINTER TO FB_SQLStoredProcedureEvt;
END_VAR
```

输入

Name	Type	描述
sProcedureName	T_MaxString	表示要执行的过程的名称。
pParameterInfo	POINTER TO ARRAY [0..MAX_SPPARAMETER] OF ST_SQLSPParameter	参数信息列表的指针地址。
cbParameterInfo	UDINT	表示参数信息列表的长度。
pSQLProcedure	POINTER TO FB_SQLStoredProcedureEvt	返回功能块 FB_SQLStoredProcedureEvt 的新实例。

返回值

Name	Type	描述
CreateSP	BOOL	显示方法的状态。在方法执行完毕时，即使出现错误，也会返回 TRUE。

示例

```
VAR
    fbSqlDatabase   : FB_SQLDatabaseEvt(sNetID := '', tTimeout := T#5S);
    ParaInfo        : ST_SQLSPParameter;
END_VAR
```

```
ParaInfo.sParameterName     := '@Customer_ID';
ParaInfo.eParameterType     := E_SPParameterType.Input;
ParaInfo.eParameterDataType := E_ColumnType.BigInt;
ParaInfo.nParameterSize     := 8;

IF fbSQLDatabase.CreateSP('dbo.SP_GetCustomerPositions', ADR(ParaInfo), SIZEOF(ParaInfo), ADR(fbSQLS
toredProcedure)) THEN
    IF fbSQLDatabase.bError THEN
        nState:=255;
    ELSE
        nState:= nState+1;
    END_IF
END_IF
```

随后，FB_SQLStoredProcedureEvt [▶ 194] 可以用于执行存储过程。

6.1.Disconnect

1.3.

2.4

该方法关闭该功能块实例打开的数据库连接。

语法

```
METHOD Disconnect : BOOL
```

返回值

Name	Type	描述
Disconnect	BOOL	显示方法的状态。在方法执行完毕时，即使出现错误，也会返回 TRUE。

示例

```
VAR
    fbSqlDatabase : FB_SQLDatabaseEvt(sNetID := '', tTimeout := T#5S);
END_VAR
```

```
// disconnect from database
IF fbSqlDatabase.Disconnect() THEN
    IF fbSqlDatabase.bError THEN
        nState := 255;
    ELSE
        nState := nState+1;
    END_IF
END_IF
```

6.1.1.3.3 FB_SQLCommandEvt

用于执行 SQL 命令的功能块。在使用它之前，必须使用功能块 FB_SQLDatabaseEvt 对其进行初始化。

语法

定义：

```
FUNCTION_BLOCK FB_SQLCommandEvt
VAR_INPUT
    sNetID: T_AmsNetID := '';
    tTimeout: TIME := T#5S;
END_VAR
VAR_OUTPUT
    bBusy: BOOL;
    bError: BOOL;
    ipTcResult: Tc3_EventLogger.I_TcMessage
END_VAR
```

输入

Name	Type	描述
sNetID	T_AmsNetID	ADS 命令指向的目标设备的 AMS 网络 ID。
tTimeout	TIME	表示函数被取消前的时间。

输出

Name	Type	描述
bBusy	BOOL	功能块的方法处于活动状态时即为 TRUE。
bError	BOOL	发生错误时即为 TRUE。
ipTcResult	Tc3_EventLogger.I_TcMessage [▶ 215]	TwinCAT 3 EventLogger 的消息接口，其中提供了有关返回值的详细信息。

属性

Name	Type	Access	描述
eTraceLevel	TcEventSeverity [▶ 216]	获取，设置	指定事件的重要性权重。只有权重高于该值的事件才会被发送到 TwinCAT 系统。

方法

Name	定义位置	描述
执行 [▶ 190]	本地	通过功能块 FB_SQLDatabaseEvt [▶ 185] 已打开的数据库连接，向数据库发送指定的 SQL 命令。
ExecuteDataReturn [▶ 191]	本地	通过功能块 FB_SQLDatabaseEvt [▶ 185] 已打开的数据库连接，向数据库发送指定的 SQL 命令。 可以传输功能块 FB_SQLResultEvt [▶ 191] 的实例，以读取返回的记录。

要求

开发环境	目标平台	要引入的 PLC 库
TwinCAT v3.1 版本 4022.20	PC 或 CX（x86）	Tc3_Database

6.1.执行
1.3.
3.1

该方法通过功能块 FB_SQLDatabase 已打开的数据库连接，向数据库发送指定的 SQL 命令。

语法

```
METHOD Execute : BOOL
VAR_INPUT
    pSQLCmd: POINTER TO BYTE;
    cbSQLCmd: UDINT;
END_VAR
```

输入

Name	Type	描述
pSQLCmd	POINTER TO BYTE	表示使用要执行的 SQL 命令的字符串变量的指针地址。
cbSQLCmd	UDINT	表示要执行的 SQL 命令的长度。

返回值

Name	Type	描述
执行	POINTER TO BYTE	显示方法的状态。在方法执行完毕时，即使出现错误，也会返回 TRUE。

示例

使用 FB_SQLDatabaseEvt.CreateCmd() [▶ 185] 创建的命令。

```
VAR
    fbSqlCommand : FB_SQLCommandEvt(sNetID := '', tTimeout := T#5S);
    tcMessage    : I_TcMessage;
END_VAR
```

```
// you can generate this with the SQL Query Editor
sCmd := 'INSERT INTO myTable_Double ( Timestamp, Name, Value) VALUES ( $'2018-01-31 14:59:27$', $'Te
mperature$', 21.3)';

// call sql command
IF fbSQLCommand.Execute(ADR(sCmd), SIZEOF(sCmd)) THEN
    IF fbSQLCommand.bError THEN
        tcMessage := fbSQLCommand.ipTcResult;
        nState := 255;
    ELSE
        nState := nState+1;
    END_IF
END_IF
```

6.1.ExecuteDataReturn
1.3.
3.2

该方法通过功能块 FB_SQLDatabase 已打开的数据库连接，向数据库发送指定的 SQL 命令。可以传输功能块 FB_SQLResult 的实例，以读取返回的记录。

语法

```
METHOD ExecuteDataReturn : BOOL
VAR_INPUT
    pSQLCmd: POINTER TO BYTE;
    cbSQLCmd: UDINT;
    pSQLDBResult: POINTER TO FB_SQLResult;
END_VAR
```

输入

Name	Type	描述
pSQLCmd	POINTER TO BYTE	表示使用要执行的 SQL 命令的字符串变量的指针地址。
cbSQLCmd	UDINT	表示要执行的 SQL 命令的长度。
pSQLDBResult	POINTER TO FB_SQLResult [▶ 191]	返回功能块 FB_SQLResult 的新实例。

返回值

Name	Type	描述
ExecuteDataReturn	POINTER TO BYTE	显示方法的状态。在方法执行完毕时，即使出现错误，也会返回 TRUE。

示例

使用 FB_SQLDatabaseEvt.CreateCmd() [▶ 185] 创建的命令。

```
VAR
    fbSqlCommand : FB_SQLCommandEvt(sNetID := '', tTimeout := T#5S);
    tcMessage    : I_TcMessage;
END_VAR
```

```
// you can generate this with the SQL Query Editor
sCmd := 'SELECT ID, Timestamp, Name, Value FROM myTable_Double';

// call sql command
IF fbSQLCommand.ExecuteDataReturn(ADR(sCmd), SIZEOF(sCmd), ADR(fbSqlResult)) THEN
    IF fbSQLCommand.bError THEN
        nState := 255;
    ELSE
         tcMessage := fbSQLCommand.ipTcResult;
        nState := nState+1;
    END_IF
END_IF
```

然后可以使用 FB_SQLResultEvt [▶ 191] 来读取数据。

6.1.1.3.4 FB_SQLResultEvt

该功能块可用于读取缓存记录。

语法

定义：

```
FUNCTION_BLOCK FB_SQLResultEvt
VAR_INPUT
    sNetID: T_AmsNetID := '';
    tTimeout: TIME := T#5S;
END_VAR
VAR_OUTPUT
    bBusy: BOOL;
    bError: BOOL;
    ipTcResult: Tc3_EventLogger.I_TcMessage
END_VAR
```

输入

Name	Type	描述
sNetID	T_AmsNetID	ADS 命令指向的目标设备的 AMS 网络 ID。
tTimeout	TIME	表示函数被取消前的时间。

输出

Name	Type	描述
bBusy	BOOL	功能块的方法处于活动状态时即为 TRUE。
bError	BOOL	发生错误时即为 TRUE。
ipTcResult	Tc3_EventLogger.I_TcMessage [▶ 215]	TwinCAT 3 EventLogger 的消息接口，其中提供了有关返回值的详细信息。

属性

Name	Type	Access	描述
eTraceLevel	TcEventSeverity [▶ 216]	获取，设置	指定事件的重要性权重。只有权重高于该值的事件才会被发送到 TwinCAT 系统。

方法

Name	定义位置	描述
读取 [▶ 192]	本地	从 TwinCAT 数据库服务器缓存的结果数据中读取指定数量的记录。
Release [▶ 193]	本地	释放 TwinCAT 数据库服务器缓冲的数据。

要求

开发环境	目标平台	要引入的 PLC 库
TwinCAT v3.1 版本 4022.20	PC 或 CX（x86）	Tc3_Database

6.1.读取
1.3.
4.1

该方法从 TwinCAT 数据库服务器缓存的结果数据中读取指定数量的记录。

语法

```
METHOD Read : BOOL
VAR_INPUT
    nStartIndex: UDINT := 0;
    nRecordCount: UDINT := 1;
    pData: POINTER TO BYTE;
    cbData: UDINT;
    bWithVerifying: BOOL := FALSE;
    bDataRelease: BOOL := TRUE;
END_VAR
```

输入

Name	Type	描述
nStartIndex	UDINT	表示要读取的第一个记录的索引。
nRecordCount	UDINT	表示要读取的记录数。
pData	POINTER TO BYTE	要写入记录的结构数组的地址。
cbData	UDINT	表示结构数组的大小，以字节为单位。
bWithVerifying	BOOL	将返回数据与 pData 结构数组进行比较，并在必要时进行调整。
bDataRelease	BOOL	释放缓存数据。

返回值

Name	Type	描述
读取	BOOL	显示方法的状态。在方法执行完毕时，即使出现错误，也会返回 TRUE。

示例

```
VAR
    fbSqlResult : FB_SQLResultEvt(sNetID:='', tTimeout := T#5S);
    aReadStruct : ARRAY[1..5] OF ST_StandardRecord;
END_VAR
```

```
// get values from internal tc db srv storage
IF fbSqlResult.Read(2, 3, ADR(aReadStruct), SIZEOF(aReadStruct), FALSE, TRUE) THEN
    IF fbSqlResult.bError THEN
        nState := 255;
    ELSE
        nState := nState+1;
    END_IF
END_IF
```

PLC 中的结果：

Expression	Type	Value
aReadStruct	ARRAY [1..5] OF ST...	
aReadStruct[1]	ST_StandardRecord	
nID	LINT	9
dtTimestamp	DATE_AND_TIME	DT#2018-1-31-15:4:59
sName	STRING(80)	'Temperature'
rValue	LREAL	21.3
aReadStruct[2]	ST_StandardRecord	
nID	LINT	10
dtTimestamp	DATE_AND_TIME	DT#2018-1-31-15:5:59
sName	STRING(80)	'Temperature'
rValue	LREAL	21.2

6.1.Release
1.3.
4.2

该方法可以用于释放 TwinCAT 数据库服务器缓存的数据。

语法

```
METHOD Release : BOOL
```

返回值

Name	Type	描述
Release	BOOL	显示方法的状态。在方法执行完毕时，即使出现错误，也会返回 TRUE。

6.1.1.3.5 FB_SQLStoredProcedureEvt

用于执行数据库的存储过程的功能块。在使用它之前，必须使用功能块 FB_SQLDatabaseEvt 对其进行初始化。

语法

定义：

```
FUNCTION_BLOCK FB_SQLStoredProcedureEvt
VAR_INPUT
    sNetID: T_AmsNetID := '';
    tTimeout: TIME := T#5S;
END_VAR
VAR_OUTPUT
    bBusy: BOOL;
    bError: BOOL;
    ipTcResult: Tc3_EventLogger.I_TcMessage
END_VAR
```

输入

Name	Type	描述
sNetID	T_AmsNetID	ADS 命令指向的目标设备的 AMS 网络 ID。
tTimeout	TIME	表示函数被取消前的时间。

输出

Name	Type	描述
bBusy	BOOL	功能块的方法处于活动状态时即为 TRUE。
bError	BOOL	发生错误时即为 TRUE。
ipTcResult	Tc3_EventLogger.I_TcMessage [▶ 215]	TwinCAT 3 EventLogger 的消息接口，其中提供了有关返回值的详细信息。

属性

Name	Type	Access	描述
eTraceLevel	TcEventSeverity [▶ 216]	获取，设置	指定事件的重要性权重。只有权重高于该值的事件才会被发送到 TwinCAT 系统。

方法

Name	定义位置	描述
执行 [▶ 195]	本地	通过功能块 FB_SQLDatabaseEvt [▶ 185] 已打开的数据库连接，向数据库发送指定的存储过程的调用。
ExecuteDataReturn [▶ 196]	本地	通过功能块 FB_SQLDatabaseEvt [▶ 185] 已打开的数据库连接，向数据库发送指定的存储过程的调用。 可以传输功能块 FB_SQLResultEvt [▶ 191] 的实例，以读取返回的记录。
Release [▶ 196]	本地	释放在初始化过程中传输的存储过程的参数信息。

要求

开发环境	目标平台	要引入的 PLC 库
TwinCAT v3.1 版本 4022.20	PC 或 CX（x86）	Tc3_Database

6.1.执行
1.3.
5.1

该方法通过功能块 FB_SQLDatabaseEvt 已打开的数据库连接，向数据库发送指定的存储过程的调用。

语法

```
METHOD Execute : BOOL
VAR_INPUT
    pParameterStrc: POINTER TO BYTE;
    cbParameterStrc: UDINT;
END_VAR
```

输入

Name	Type	描述
pParameterStrc	POINTER TO BYTE	传输到过程的参数结构的指针地址。
cbParameterStrc	UDINT	参数结构的长度。

返回值

Name	Type	描述
执行	BOOL	显示方法的状态。在方法执行完毕时，即使出现错误，也会返回 TRUE。

示例

使用先前用 FB_SQLDatabaseEvt.CreateSP() [▶ 185] 创建的存储过程。

```
VAR
    fbSQLStoredProcedure : FB_SQLStoredProcedureEvt(sNetID:='', tTimeout := T#5S);
    Customer_ID          : LINT;
    tcMessage            : I_TcMessage;
END_VAR
```

```
IF fbSQLStoredProcedure.Execute(pParameterStrc := ADR(Customer_ID) , cbParameterStrc:= SIZEOF(Custom
er_ID)) THEN
    IF fbSQLStoredProcedure.bError THEN
        tcMessage := fbSQLStoredProcedure.ipTcResult;
        nState := 255;
    ELSE
        nState := nState+1;
    END_IF
END_IF
```

6.1.ExecuteDataReturn

1.3.

5.2

该方法通过功能块 FB_SQLDatabase 已打开的数据库连接，向数据库发送指定的存储过程的调用。可以传输 FB_SQLResult 功能块的实例，以读取返回的记录。

语法

```
METHOD ExecuteDataReturn : BOOL
VAR_INPUT
    pParameterStrc: POINTER TO BYTE;
    cbParameterStrc: UDINT;
    pSQLDBResult: POINTER TO FB_SQLDBResultEvt;
END_VAR
```

输入

Name	Type	描述
pParameterStrc	POINTER TO BYTE	传输到过程的参数结构的指针地址。
cbParameterStrc	UDINT	参数结构的长度
pSQLDBResult	POINTER TO FB_SQLDBResultEvt	返回功能块 FB_SQLDBResultEvt 的新实例。

返回值

Name	Type	描述
读取	BOOL	显示方法的状态。在方法执行完毕时，即使出现错误，也会返回 TRUE。

示例

使用先前用 FB_SQLDatabaseEvt.CreateSP() [▶ 185] 创建的存储过程。

```
VAR
    fbSQLStoredProcedure : FB_SQLStoredProcedureEvt(sNetID:='', tTimeout := T#5S);
    Customer_ID          : LINT;
    tcMessage            : I_TcMessage;
END_VAR
```

```
IF fbSQLStoredProcedure.ExecuteDataReturn(pParameterStrc := ADR(Customer_ID), cbParameterStrc:= SIZE
OF(Customer_ID), pSQLDBResult := ADR(fbSqlResult)) THEN
    IF fbSQLStoredProcedure.bError THEN
        tcMessage := fbSQLStoredProcedure.ipTcResult;
        nState := 255;
    ELSE
        nState := nState+1;
    END_IF
END_IF
```

然后可以使用 FB_SQLResultEvt [▶ 191] 来读取数据。

6.1.Release

1.3.

5.3

该方法释放在初始化过程中传输的存储过程的参数信息。

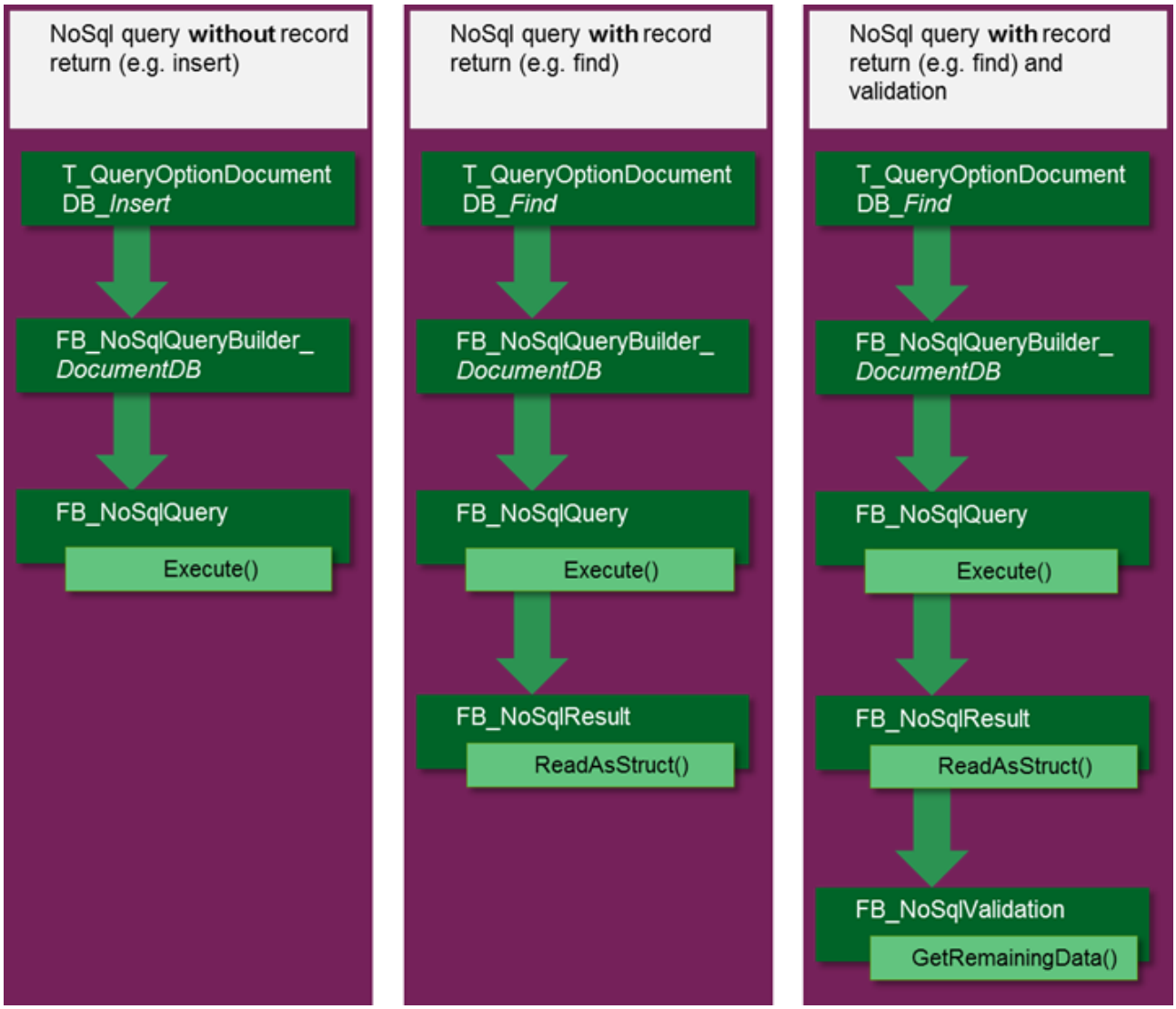
语法

```
METHOD Release : BOOL
```

返回值

Name	Type	描述
Release	BOOL	显示方法的状态。在方法执行完毕时，即使出现错误，也会返回 TRUE。

6.1.1.4 NoSQL 专家模式



6.1.1.4.1 查询构建器

为了支持尽可能多的 NoSQL 数据库，一些查询功能块为查询提供了不同的参数设置。然后，这些功能块将作为接口传递给 FB_NoSQLQuery 的方法。

6.1.FB_NoSQLQueryBuilder_DocumentDB

1.4.

1.1

用于定义数据库查询的功能块。通过 FB_NoSQLQueryEvt [▶ 200] 发送查询。无需调用构建方法。

语法

定义：

```
FUNCTION_BLOCK FB_NoSQLQueryBuilder_DocumentDB
VAR_INPUT
    eQueryType : E_DocumentDbQueryType;
    sCollectionName : T_MAXSTRING;
    pQueryOptions: POINTER TO BYTE;
    cbQueryOptions : UDINT;
END_VAR
VAR_OUTPUT
END_VAR
```

输入

Name	Type	描述
eQueryType	E_DocumentDbQueryType [▶ 224]	发送到数据库的查询的类型。
sCollectionName	T_ MAXSTRING	作为查询目标的集合的名称。
pQueryOptions	POINTER TO BYTE	指定查询选项 [▶ 224]的地址。
cbQueryOptionsr	UDINT	查询选项的长度。

方法

Name	定义位置	描述
Build [▶ 198]	本地	[可选] 该方法根据设置的参数为功能块 FB_NoSQLQueryEvt [▶ 200] 生成查询。

示例：

```
VAR
    fbNoSQLQueryBuilder_DocumentDB: FB_NoSQLQueryBuilder_DocumentDB;
    sFilter : T_MAXSTRING;
    stOptions : T_QueryOptionDocumentDB_Find;
END_VAR
```

```
// Set your settings before you run the query
stOptions.pFilter:= ADR(sFilter);
stOptions.cbFilter:= SIZEOF(sFilter);

fbNoSQLQueryBuilder_DocumentDB.eQueryType:=E_DocumentDbQueryType.Find;
fbNoSQLQueryBuilder_DocumentDB.sCollectionName:= 'MyCollectionName';
fbNoSQLQueryBuilder_DocumentDB.pQueryOptions:= ADR(stOptions);
fbNoSQLQueryBuilder_DocumentDB.cbQueryOptions:= SIZEOF(stOptions);
```

要求

开发环境	目标平台	要引入的 PLC 库
TwinCAT v3.1 版本 4022.20	PC 或 CX（x86）	Tc3_Database

6.1.Build

1.4.

1.1.

1

在发送查询之前，如果出现 FB_NoSQLQuery [▶ 200]Evt（使用 Excecute 或 ExecuteDataReturn），则会自动调用该方法。它根据 QueryBuilder 的指定参数创建一个 TwinCAT 3 数据库服务器特定的查询。

6.1.FB_NoSQLQueryBuilder_TimeSeriesDB
1.4.
1.2

用于定义时序数据库查询的功能块。通过 FB_NoSQLQueryEvt [▶ 200] 发送查询。无需调用构建方法。使用属性可以描述数据结构 [▶ 339]，以影响各个设置。

输入

Name	Type	描述
pQueryOptions	POINTER TO BYTE	指定查询选项 [▶ 227]的地址。
cbQueryOptions	UDINT	查询选项的长度。

语法

定义：

```
FUNCTION_BLOCK FB_NoSQLQueryBuilder_TimeSeriesDB
VAR_INPUT
    pQueryOptions : POINTER TO BYTE;
    cbQueryOptions : UDINT;
END_VAR
VAR_OUTPUT
END_VAR
```

方法

Name	定义位置	描述
Build [▶ 198]	本地	[可选] 该方法根据设置的参数为功能块 FB_NoSQLQueryEvt [▶ 200] 生成查询。

示例：

```
VAR
    fbNoSQLQueryBuilder_TimeSeriesDB : FB_NoSQLQueryBuilder_TimeSeriesDB;
    QueryOption_TSDB_Insert : T_QueryOptionTimeSeriesDB_Insert;
    fbNoSqlQueryEvt : FB_NoSQLQueryEvt(sNetID := '', tTimeout := T#15S);
    MyStructArray: ARRAY[1..1000] OF MyStruct;
END_VAR
```

```
CASE nState OF
    1: // init
        fbNoSQLQueryBuilder_TimeSeriesDB.pQueryOptions := ADR(QueryOption_TSDB_Insert);
        fbNoSQLQueryBuilder_TimeSeriesDB.cbQueryOptions := SIZEOF(QueryOption_TSDB_Insert);

        QueryOption_TSDB_Insert.sTableName := 'MeasurementName';
        QueryOption_TSDB_Insert.sDataType := 'MyStruct';
        QueryOption_TSDB_Insert.pSymbol := ADR(MyStructArray);
        QueryOption_TSDB_Insert.cbSymbol := SIZEOF(MyStructArray);
        QueryOption_TSDB_Insert.nDataCount := 1000; // ArrayLength
        QueryOption_TSDB_Insert.nStartTimestamp := F_GetSystemTime(); // get current twincat time
        QueryOption_TSDB_Insert.nCycleTime := 1000; // equivalent to 1 ms

    2: // write values
        IF fbNoSqlQueryEvt.Execute(dbid, fbNoSQLQueryBuilder_TimeSeriesDB) THEN
            IF fbNoSqlQueryEvt.bError THEN
                // do some error handling here
            ELSE
                // success
            END_IF
        END_IF
END_CASE
```

要求

开发环境	目标平台	要引入的 PLC 库
TwinCAT v3.1 版本 4022.20	PC 或 CX（x86）	Tc3_Database

6.1.Build
1.4.
1.2.
1

在发送查询之前，如果出现 FB_NoSQLQuery [▶ 200]Evt（使用 Excecute 或 ExecuteDataReturn），则会自动调用该方法。它根据 QueryBuilder 的指定参数创建一个 TwinCAT 3 数据库服务器特定的查询。

6.1.1.4.2 FB_NoSQLQueryEvt

用于执行 NoSQL 数据库查询的功能块。描述查询的 QueryBuilder 功能块可用作方法的输入参数。

语法

定义：

```
FUNCTION_BLOCK FB_NoSQLQueryEvt
VAR_INPUT
    sNetID: T_AmsNetID := '';
    tTimeout: TIME := T#5S;
END_VAR
VAR_OUTPUT
    bBusy: BOOL;
    bError: BOOL;
    ipTcResult: Tc3_EventLogger.I_TcMessage
END_VAR
```

输入

Name	Type	描述
sNetID	T_AmsNetID	ADS 命令指向的目标设备的 AMS 网络 ID。
tTimeout	TIME	表示函数被取消前的时间。

属性

Name	Type	Access	描述
eTraceLevel	TcEventSeverity [▶ 216]	获取，设置	指定事件的重要性权重。只有权重高于该值的事件才会被发送到 TwinCAT 系统。

输出

Name	Type	描述
bBusy	BOOL	功能块的方法处于活动状态时即为 TRUE。
bError	BOOL	发生错误时即为 TRUE。
ipTcResult	Tc3_EventLogger.I_TcMessage [▶ 215]	TwinCAT 3 EventLogger 的消息接口，其中提供了有关返回值的详细信息。

方法

Name	定义位置	描述
Execute [▶ 201]	本地	将 QueryBuilder 功能块创建的查询发送到数据库。
ExecuteDataReturn [▶ 202]	本地	将 QueryBuilder 功能块创建的查询发送到数据库。 可以传输 FB_NoSqlResultEvt [▶ 203] 功能块的实例，以读取返回的记录。

要求

开发环境	目标平台	要引入的 PLC 库
TwinCAT v3.1 版本 4022.20	PC 或 CX（x86）	Tc3_Database

6.1.执行
1.4.
2.1

该方法将查询发送到先前已使用 I_NoSQLQueryBuilder [▶ 197] 功能块进行设置的 NoSQL 数据库。

语法

```
METHOD Execute : BOOL
VAR_INPUT
    hDBID: UDINT;
    iNoSQLQueryBuilder: I_NoSQLQueryBuilder;
END_VAR
```

输入

Name	Type	描述
hDBID	UDINT	设置的数据库配置的 ID
iNoSSQLQuery Builder	I_NoSQLQueryBuild er	预先参数化的 QueryBuilder 功能块。根据数据库的不同，这会有所差异。

返回值

Name	Type	描述
执行	BOOL	显示方法的状态。在方法执行完毕时，即使出现错误，也会返回 TRUE。

使用 QueryBuilder 执行相应的查询。

```
VAR
    fbNoSQLQuery : FB_NoSQLQueryEvt(sNetID := '', tTimeout := T#5S);
    fbNoSQLQueryBuilder_DocumentDB: FB_NoSQLQueryBuilder_DocumentDB;
    InsertQueryOptions: T_QueryOptionDocumentDB_Insert;
    myDBID : UDINT := 1;
    sDocument : STRING(1000);
    TcMessage : I_TcMessage;
END_VAR
```

```
// set QueryInputs
fbNoSQLQueryBuilder_DocumentDB.eQueryType := E_DocumentDbQueryType.InsertOne;
fbNoSQLQueryBuilder_DocumentDB.pQueryOptions := ADR(InsertQueryOptions);
fbNoSQLQueryBuilder_DocumentDB.cbQueryOptions := SIZEOF(InsertQueryOptions);

// set insert parameter:
sDocument := '{Name : "MyValue", Value : 123.456}';
InsertQueryOptions.pDocuments:= ADR(sDocument);
InsertQueryOptions.cbDocuments:= SIZEOF(sDocument);

// call nosql command
IF fbNoSQLQuery.Execute(myDBID, fbNoSQLQueryBuilder_DocumentDB) THEN
    IF fbNoSQLQuery .bError THEN
        TcMessage := fbNoSQLQuery.ipTcResult
        nState := 255;
    ELSE
        nState := nState+1;
    END_IF
END_IF
```

首先，通过 FB_NoSQLQueryBuilder_DocumentDB [▶ 197] 功能块对 FB_NoSQLQueryEvt 功能块进行参数设置。根据查询类型的不同，有多种选项（例如 T_QueryOptionDocumentDB_Insert [▶ 226]）可以用于设置要插入的文档。

6.1.ExecuteDataReturn

1.4.

2.2

该方法对先前已使用 I_NoSQLQueryBuilder 功能块进行设置的 NoSQL 数据库执行查询。使用返回值填充传输的 FB_NoSQLResultEvt 类型的实例。

语法

```
METHOD ExecuteDataReturn : BOOL
VAR_INPUT
    hDBID : UDINT;
    iNoSSQLQueryBuilder: I_NoSQLQueryBuilder;
    pNoSQLResult: POINTER TO FB_NoSQLResultEvt;
END_VAR
```

输入

Name	Type	描述
hDBID	UDINT	设置的数据库配置的 ID
iNoSQLQueryBuilder	I_NoSQLQueryBuilder	预先配置的 QueryBuilder 功能块，用于定义要发送的查询。
pNoSQLResult	指向 FB_NoSSQLResultEvt 的指针	指定 FB_NoSQLResultEvt 功能块的地址，该功能块可以用于读取结果。

返回值

Name	Type	描述
ExecuteDataReturn	BOOL	显示方法的状态。在方法执行完毕时，即使出现错误，也会返回 TRUE。
nDataCount	UDINT	[可选] 返回的记录数

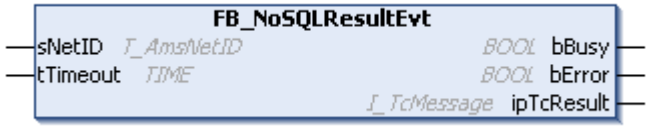
使用 QueryBuilder 执行相应的查询。

```
VAR
    fbNoSqlQuery : FB_NoSQLQueryEvt(sNetID := '', tTimeout := T#5S);
    fbNoSQLQueryBuilder_DocumentDB: FB_NoSQLQueryBuilder_DocumentDB
    FindQueryOptions : T_QueryOptionDocumentDB_Find;
    fbNoSqlResult : FB_NoSQLResultEvt(sNetID := '', tTimeout := T#5S);
    myDBID : UDINT := 1;
    sFilter : STRING(255);
    sSort: STRING(255);
    sProjection: STRING(255);
    TcMessage : I_TcMessage;
END_VAR
```

```
// set QueryInputs:
fbNoSQLQueryBuilder_DocumentDB.eQueryType := E_DocumentDbQueryType.Find;
fbNoSQLQueryBuilder_DocumentDB.pQueryOptions := ADR(FindQueryOptions);
fbNoSQLQueryBuilder_DocumentDB.cbQueryOptions := SIZEOF(FindQueryOptions);

//set Find Parameter ([optional] sort, projection):
sFilter := '{}'; // read all data from database
FindQueryOptions.pFilter:= ADR(sFilter);
FindQueryOptions.cbFilter:= SIZEOF(sFilter);

// call nosql query:
IF fbNoSqlQuery.ExecuteDataReturn(myDBID, fbNoSqlQuery, ADR(fbNoSqlResult)) THEN
    IF fbNoSqlQuery.bError THEN
        TcMessage := fbNoSqlQuery.ipTcResult;
        nState := 255;
    ELSE
        nState := nState+1;
    END_IF
END_IF
```

首先，通过 FB_NoSQLQueryBuilder_DocumentDB [▶ 197] 功能块对 FB_NoSQLQueryEvt 功能块进行参数设置。根据查询类型的不同，有多种选项（例如 T_QueryOptionDocumentDB_Find [▶ 224]）可以用于定义筛选器、排序或投影。

6.1.1.4.3 FB_NoSQLResultEvt



用于读取缓冲记录的功能块。

在调用 ExecuteDataReturn [▶ 200] 方法时，必须首先使用功能块 FB_NoSQLQueryEvt [▶ 202] 从数据库中检索记录。指定功能块 FB_NoSQLResultEvt 进行初始化。然后，可以将它们作为 PLC 结构或字符串读出。

语法

定义：

```
FUNCTION_BLOCK FB_SQLResultEvt
VAR_INPUT
    sNetID: T_AmsNetID := '';
    tTimeout: TIME := T#5S;
END_VAR
VAR_OUTPUT
    bBusy: BOOL;
    bError: BOOL;
    ipTcResult: Tc3_EventLogger.I_TcMessage
END_VAR
```

输入

Name	Type	描述
sNetID	T_AmsNetID	ADS 命令指向的目标设备的 AMS 网络 ID。
tTimeout	TIME	表示函数被取消前的时间。

输出

Name	Type	描述
bBusy	BOOL	功能块的方法处于活动状态时即为 TRUE。
bError	BOOL	发生错误时即为 TRUE。
ipTcResult	Tc3_EventLogger.I_TcMessage [▶ 215]	TwinCAT 3 EventLogger 的消息接口，其中提供了有关返回值的详细信息。

属性

Name	Type	描述
eTraceLevel	TcEventSeverity [▶ 216]	指定事件的重要性权重。只有权重高于该值的事件才会被发送到 TwinCAT 系统。
nDataCount	UDINT	表示调用功能块 FB_NoSQLQueryEvt.ExecuteDataReturn() [▶ 202] 后返回的可用记录数。

方法

Name	定义位置	描述
ReadAsString [▶ 204]	本地	从 TwinCAT 数据库服务器缓存的结果数据中以JSON形式读取指定数量的记录。
ReadAsStruct [▶ 205]	本地	从 TwinCAT 数据库服务器缓存的结果数据中读取指定数量的记录到指定的结构中。
Release [▶ 206]	本地	释放 TwinCAT 数据库服务器缓冲的数据。

要求

开发环境	目标平台	要引入的 PLC 库
TwinCAT v3.1 版本 4022.20	PC 或 CX（x86）	Tc3_Database

6.1.ReadAsString

1.4.

3.1

该方法从 TwinCAT 数据库服务器缓存的结果数据中读取指定数量的记录。指定一个字符串数组，数据将以JSON格式复制到该数组中。

语法

```
METHOD ReadAsString : BOOL
VAR_INPUT
    nStartIndex: UDINT := 0;
    nRecordCount: UDINT := 1;
    pData: POINTER TO BYTE;
    cbData: UDINT;
    nMaxDocumentSize : UDINT;
    bDataRelease: BOOL := TRUE;
END_VAR
```

输入

Name	Type	描述
nStartIndex	UDINT	表示要读取的第一个记录的索引。
nRecordCount	UDINT	表示要读取的记录数。
pData	POINTER TO BYTE	表示要写入记录的字符串数组的地址。
cbData	UDINT	表示字符串数组的大小，以字节为单位。
nMaxDocumentSize	UDINT	表示 pData 中单个 JSON 文档的最大大小。
bDataRelease	BOOL	释放缓存数据。

返回值

Name	Type	描述
ReadAsString	BOOL	显示方法的状态。在方法执行完毕时，即使出现错误，也会返回 TRUE。

示例：

```
VAR
    fbNoSqlResult : FB_NoSQLResultEvt(sNetID := '', tTimeout := T#5S);
    aRead_Json : ARRAY[0..2] OF STRING(1000);
    TcMessage : I_TcMessage;
END_VAR
```

```
IF fbNoSqlResult.ReadAsString(
    nStartIndex:= 0,
    nRecordCount:= 3,
    pData:= ADR(aRead_Json),
    cbData:= SIZEOF(aRead_Json),
    MaxDocumentSize:= SIZEOF(aRead_Json[0]),
```

```
    bDataRelease:= TRUE)
THEN
    IF fbNoSqlResult.bError THEN
        TcMessage := fbNoSqlResult.ipTcResult;
        nstate := 255;
    ELSE
        nstate := nstate+1;
    END_IF
END_IF
```

6.1.ReadAsStruct

1.4.

3.2

该方法从缓冲的结果数据中读取指定数量的记录。指定要写入数据的结构或结构数组。该结构的数据类型模式应尽可能与读取数据的数据类型模式一致。将变量名与记录的名称进行比较。通过验证可以检测到偏差，并采取应对措施。

如果需要在数据库和 PLC 中使用不同的名称，则可以使用属性 *“ElementName”* 在结构中描述名称，并使用从数据库分配的名称。

语法

```
METHOD ReadAsStruct: BOOL
VAR_INPUT
    nStartIndex: UDINT := 0;
    nRecordCount: UDINT := 1;
    pData: POINTER TO BYTE;
    cbData: UDINT;
    bValidate: BOOL := FALSE;
    pNoSQLValidation : POINTER TO FB_NoSQLValidationEvt;
    bDataRelease: BOOL := TRUE;
END_VAR
```

输入

Name	Type	描述
nStartIndex	UDINT	表示要读取的第一个记录的索引。
nRecordCount	UDINT	表示要读取的记录数。
pData	POINTER TO BYTE	要写入记录的结构数组的地址。
cbData	UDINT	表示结构数组的大小，以字节为单位。
bValidate	BOOL	将返回数据与 pData 结构数组进行比较，并在必要时进行调整。
pNoSQLValidation	POINTER TO FB_NoSQLValidationEvt	为验证调用而提供更多信息的功能块 FB_NoSQLValidationEvt 的地址。
bDataRelease	BOOL	释放缓存数据。

返回值

Name	Type	描述
ReadAsStruct	BOOL	显示方法的状态。在方法执行完毕时，即使出现错误，也会返回 TRUE。

示例：

```
VAR
    fbNoSQLResult: FB_NoSQLResultEvt(sNetID := '', tTimeout := T#5S);
    aRead : ARRAY[0..2] OF ST_MyDataStruct;
    fbNoSQLValidation : FB_NoSQLValidationEvt(sNetID := '', tTimeout := #5S);
END_VAR
```

```
IF fbNoSQLResult.ReadAsStruct(
    nStartIndex:= 0,
    nRecordCount:= 3,
    pData:= ADR(aRead),
    cbData:= SIZEOF(aRead),
    bValidate:= TRUE,
    pNoSQLValidation:= ADR(fbNoSQLValidation),
    bDataRelease:= TRUE)
THEN
    IF fbNoSQLResult.bError THEN
```

```
        TcMessage := fbNoSQLResult.ipTcResult;
        nstate := 255;
    ELSE
        nstate := nstate+1;
    END_IF
END_IF
```

6.1.Release

1.4.

3.3

该方法可以用于释放 TwinCAT 数据库服务器缓存的数据。

语法

```
METHOD Release : BOOL
```

返回值

Name	Type	描述
Release	BOOL	显示方法的状态。在方法执行完毕时，即使出现错误，也会返回 TRUE。

6.1.1.4.4 FB_NoSQLValidationEvt

用于读取使用 FB_NoSQLResultEvt [▶ 203] 读取数据时发生的验证事件和结果的功能块。该功能块通过 NoSQLResult 的 CreateValidation 方法进行初始化。它指的是 ReadAsStruct [▶ 205] 方法的最后一次调用。

语法

定义：

```
FUNCTION_BLOCK FB_NoSQLValidationEvt
VAR_INPUT
    sNetID: T_AmsNetID := '';
    tTimeout: TIME := T#5S;
END_VAR
VAR_OUTPUT
    bBusy: BOOL;
    bError: BOOL;
    ipTcResult: Tc3_EventLogger.I_TcResultEvent
END_VAR
```

输入

Name	Type	描述
sNetID	T_AmsNetID	ADS 命令指向的目标设备的 AMS 网络 ID。
tTimeout	TIME	表示函数被取消前的时间。

输出

Name	Type	描述
bBusy	BOOL	功能块的方法处于活动状态时即为 TRUE。
bError	BOOL	发生错误时即为 TRUE。
ipTcResult	Tc3_EventLogger.I_TcMessage [▶ 215]	TwinCAT 3 EventLogger 的消息接口，其中提供了有关返回值的详细信息。

属性

Name	Type	Access	描述
eTraceLevel	TcEventSeverity [▶ 216]	获取，设置	指定事件的重要性权重。只有权重高于该值的事件才会被发送到 TwinCAT 系统。

方法

Name	定义位置	描述
GetIssues [▶ 207]	本地	读取验证事件的列表作为字符串数组。
GetRemainingData [▶ 208]	本地	读取数据作为字符串，但无法将该字符串分配给 PLC 结构中的任何元素。
Release [▶ 208]	本地	释放 TwinCAT 数据库服务器中的缓冲数据。

要求

开发环境	目标平台	要引入的 PLC 库
TwinCAT v3.1 版本 4022.20	PC 或 CX（x86）	Tc3_Database

6.1.GetIssues
1.4.
4.1

将发生的验证事件读入 T_MAXSTRING 类型的数组。这包含有关剩余记录或未分配记录的信息，或者有关未填充 PLC 结构中的哪些元素的信息。

语法

```
METHOD GetIssues : BOOL
VAR_INPUT
    pData : POINTER TO BYTE;
    cbData: UDINT;
    bDataRelease : BOOL;
END_VAR
```

输入

Name	Type	描述
pData	POINTER TO BYTE	要写入记录的 T_MAXSTRING 类型的数组的地址。
cbData	UDINT	表示字符串数组的大小，以字节为单位。
bDataRelease	BOOL	释放缓存数据。

返回值

Name	Type	描述
ReadAsString	BOOL	显示方法的状态。在方法执行完毕时，即使出现错误，也会返回 TRUE。

示例：

```
VAR
    fbNoSqlValidation : FB_NoSQLValidation(sNetID := '', tTimeout := t#15S);
    aIssues : ARRAY[0..1000] OF T_MAXSTRING;
END_VAR
```

```
IF fbNoSqlValidation.GetIssues(
    pData:= ADR(aIssues),
    cbData:= SIZEOF(aIssues),
    bDataRelease:= TRUE)
THEN
    IF fbNoSqlValidation.bError THEN
```

```
        TcMessage := fbNoSqlValidation.ipTcResult;
        nstate := 255;
    ELSE
        nstate := nstate+1;
    END_IF
END_IF
```

6.1.GetRemainingData

1.4.

4.2

该方法可以用于读取验证后剩余的 JSON 格式的数据。例如，这包括无法分配给 PLC 结构体的记录。

语法

```
METHOD GetRemainingData : BOOL
VAR_INPUT
    pData : POINTER TO BYTE;
    cbData : UDINT;
    cbDocument : UDINT;
    bDataRelease : BOOL;
END_VAR
```

输入

Name	Type	描述
pData	POINTER TO BYTE	表示要写入记录的字符串数组的地址。
cbData	UDINT	表示字符串数组的大小，以字节为单位。
cbDocument	UDINT	指定数组中的字符串的长度。
bDataRelease	BOOL	释放缓存数据。

返回值

Name	Type	描述
GetRemainingData	BOOL	显示方法的状态。在方法执行完毕时，即使出现错误，也会返回 TRUE。

示例：

```
VAR CONSTANT
    cDocumentSize : UDINT := 1000;
END_VAR
VAR
    fbNoSqlValidation : FB_NoSQLValidation(sNetID := '', tTimeout := t#15S);
    aRemainingData : ARRAY[0..1000] OF STRING(cDocumentSize);
END_VAR
```

```
IF fbNoSqlValidation.GetRemainingData(
    pData:= ADR(aRemainingData),
    cbData:= SIZEOF(aRemainingData),
    cbDocument:= cDocumentSize,
    bDataRelease:= TRUE)
THEN
    IF fbNoSqlValidation.bError THEN
        TcMessage := fbNoSqlValidation.ipTcResult;
        nstate := 255;
    ELSE
        nstate := nstate+1;
    END_IF
END_IF
```

6.1.Release

1.4.

4.3

释放内存中的验证结果。

语法

```
METHOD Release : BOOL
```

返回值

Name	Type	描述
Release	BOOL	显示方法的状态。在方法执行完毕时，即使出现错误，也会返回 TRUE。

6.1.1.4.5 辅助功能块

如下，这些功能块提供了一些处理数据类型的有用功能。

6.1.FB_NoSQLObjectId_MongoDB
1.4.
5.1

用于从 MongoDB 解析 ObjectId 的功能块。在 PLC 中，它由数据类型 T_ObjectId_MongoDB [▶ 224] 描述。

语法

```
FUNCTION_BLOCK FB_NoSQLObjectId_MongoDB
VAR_INPUT
    ObjectId : T_ObjectId_MongoDB;
END_VAR
```

输入

Name	Type	描述
ObjectId	T_ObjectId_MongoDB	用于描述 ObjectId 的 12 字节数据类型。

属性

Name	Type	描述
eTraceLevel	TcEventSeverity	指定事件的重要性权重。只有权重高于该值的事件才会被发送到 TwinCAT 系统。
nId	UDINT	非唯一的连续编号
nMachineId	UDINT	机器标识
nProcessId	UINT	写入过程标识
tTimestamp	DATE_AND_TIME	记录的时间戳

方法

Name	定义位置	返回值	描述
ToString	本地	STRING(36)	返回带有类型名称的字符串形式的 ID。 示例： ,ObjectId(„5be15c11afa6ec72b107dafaf“) ‘
ValueOf	本地	STRING(24)	仅返回字符串形式的 ID。 示例：,5be15c11afa6ec72b107dafaf ‘

要求

开发环境	目标平台	要引入的 PLC 库
TwinCAT v3.1 版本 4022.20	PC 或 CX（x86）	Tc3_Database

6.1.1.5 支持 Tc3_Eventlogger

TwinCAT 3 数据库服务器支持 TwinCAT 3 EventLogger（TwinCAT 3 版本 4022.20）。这样可以通过接口读出功能块事件的详细信息。有关 EventLogger 的更多信息，请参见 TwinCAT 3 基础库。

TwinCAT 3 数据库服务器的所有功能块都支持 Tc3 EventLogger 的接口。接口 Tc3_Eventlogger.I_TcMessage [▶ 215] 可用作功能块的返回值。除返回值外，*eTraceLevel* 属性还可用于确定事件权重。

属性

Name	Type	Access	描述
eTraceLevel	TcEventSeverity [▶ 216]	获取，设置	指定事件的重要性权重。只有权重高于该值的事件才会被发送到 TwinCAT 系统。

示例：

例如，为功能块 FB_PLCDBWrite 指定以下权重：

```
fbPLCDBWriteEvt.eTraceLevel := TcEventSeverity.Warning;
```

现在，所有级别至少为警告的事件都会发送到这里。在这种情况下会忽略“信息”权重的事件。

Tc3_Database 功能块本身具有数据类型为 Tc3_Eventlogger.I_TcMessage 的输出 *ipTcResult*。该接口提供的所有功能都可以使用。

在本示例中，首先调用功能块。

```
1:  // Call Functionblock
    IF fbPLCDBWriteEvt.WriteStruct(
        hDBID:= 1,
        sTableName:= 'myTable_Struct',
        pRecord:= ADR(myRecord),
        cbRecord:= SIZEOF(myRecord),
        pColumnNames:= ADR(ColumnNames) ,
        cbColumnNames:= SIZEOF(ColumnNames))
    THEN
        IF fbPLCDBWriteEvt.bError THEN
            myTcMessage := fbPLCDBWriteEvt.ipTcResult
            nState := 255;
        ELSE
            nState := 0;
        END_IF
    END_IF
```

如果发生错误，我们现在想要在运行时环境中请求事件文本。为此，可以使用 *RequestEventText* 方法。使用 *nLangId =1031* 读取德语的错误代码。这是 Tc3_Eventlogger.I_TcMessage [▶ 215] 接口的众多功能之一。

```
255://Request EventText
    IF myTcMessage.RequestEventText(1031,
        ADR(MyEventString),
        SIZEOF(MyEventString))THEN
        nState := 0;
    END_IF
```

6.1.1.5.1 I_TcEventBase

此基本接口定义了事件的方法和属性。

方法

Name	描述
EqualsTo [▶ 211]	将事件与其他实例进行比较。
EqualsToEventClass [▶ 212]	将事件的事件类与其他事件类进行比较。
EqualsToEventEntryEx [▶ 213]	将事件的事件定义与其他事件定义进行比较。
GetJsonAttribute [▶ 213]	返回 Json 属性。
RequestEventClassName [▶ 214]	请求事件类的名称。
RequestEventText [▶ 214]	返回事件文本。

属性

Name	Type	Access	描述
eSeverity	TcEventSeverity [▶ 216]	Get	返回严重级别。
EventClass	GUID	Get	返回事件类的 GUID。
ipSourceInfo	I_TcSourceInfo	Get	返回指向源定义的指针。
nEventId	UDINT	Get	返回事件 ID。
stEventEntry	TcEventEntry	Get	返回事件定义。

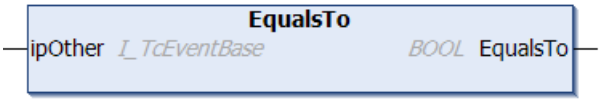
要求

开发环境	目标平台	要引入的 PLC 库
TwinCAT v3.1.4022.20	PC 或 CX（x64、x86、ARM）	Tc3_EventLogger

6.1.EqualsTo

1.5.

1.1



此方法执行与输入中指定的另一个事件的比较。

语法

```
METHOD EqualsTo : BOOL
VAR_INPUT
    ipOther : I_TcEventBase;
END_VAR
```

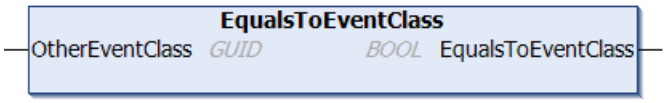
输入

Name	Type	描述
ipOther	I_TcEventBase	要比较的事件

返回值

Name	Type	描述
EqualsTo	BOOL	如果事件匹配，则返回 TRUE。

6.1.EqualsToEventClass
1.5.
1.2



此方法执行与输入中指定的另一个事件类的比较。

语法

```
METHOD EqualsToEventClass : BOOL
VAR_INPUT
    OtherEventClass : GUID
END_VAR
```

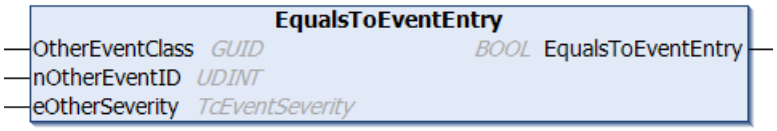
输入

Name	Type	描述
OtherEventClass	GUID	要比较的事件类。

返回值

Name	Type	描述
EqualsToEventClass	BOOL	如果事件类匹配，则返回 TRUE。

6.1.EqualsToEventEntry
1.5.
1.3



此方法执行与输入中指定的另一个事件的比较。

语法

```
METHOD EqualsToEventEntry : BOOL
VAR_INPUT
    OtherEventClass : GUID;
    nOtherEventID   : UDINT;
    eOtherSeverity  : TcEventSeverity;
END_VAR
```

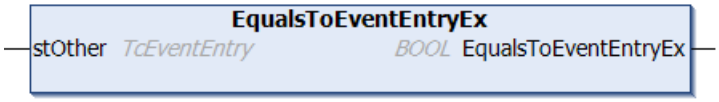
输入

Name	Type	描述
OtherEventClass	GUID	要比较的事件的事件类。
nOtherEventID	UDINT	要比较的事件的事件 ID。
eOtherSeverity	TcEventSeverity	要比较的事件的事件严重级别。

返回值

Name	Type	描述
EqualsToEventEntry	BOOL	如果事件匹配，则返回 TRUE。

6.1.EqualsToEventEntryEx
1.5.
1.4



此方法执行与输入中指定的另一个事件的比较。

语法

```
METHOD EqualsToEventEntryEx : BOOL
VAR_INPUT
    stOther : TcEventEntry;
END_VAR
```

输入

Name	Type	描述
stOther	TcEventEntry	要比较的事件。

返回值

Name	Type	描述
EqualsToEventEntryEx	BOOL	如果事件匹配，则返回 TRUE。

6.1.GetJsonAttribute
1.5.
1.5

此方法返回 Json 属性。

语法

```
METHOD GetJsonAttribute : HRESULT
VAR_INPUT
    sJsonAttribute : REFERENCE TO STRING;
    nJsonAttribute : UDINT;
END_VAR
```

输入

Name	Type	描述
sJsonAttribute	REFERENCE TO STRING	引用字符串类型的变量
nJsonAttribute	UDINT	字符串变量的长度

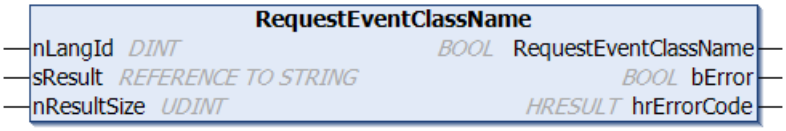
返回值

Name	Type	描述
GetJsonAttribute	HRESULT	如果方法调用成功，则返回 S_OK。 如果变量长度过小，则返回 ERROR_BAD_LENGTH。 否则，返回 HRESULT 作为错误代码。

6.1.RequestEventClassName

1.5.

1.6



此方法返回事件类的名称。

语法

```
METHOD RequestEventClassName : BOOL
VAR_INPUT
    nLangId     : DINT;
    sResult     : REFERENCE TO STRING;
    nResultSize : UDINT;
END_VAR
VAR_OUTPUT
    bError      : BOOL;
    hrErrorCode : HRESULT;
END_VAR
```

输入

Name	Type	描述
nLangId	DINT	指定语言 ID 英语（en-US） = 1033 德语（de-DE） = 1031 ……
sResult	REFERENCE TO STRING	引用字符串类型的变量
nResultSize	UDINT	字符串变量的大小，以字节为单位

返回值

Name	Type	描述
RequestEventClassName	BOOL	请求终止后立即返回 TRUE。如果异步请求仍处于激活状态，则返回 FALSE。必须调用此方法，直到返回值为 TRUE。

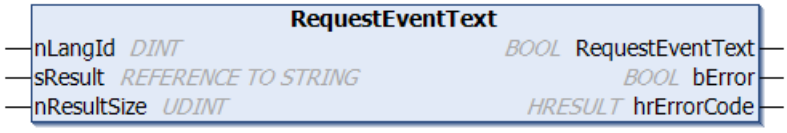
输出

Name	Type	描述
bError	BOOL	如果方法调用成功，则返回 FALSE。如果发生错误，则返回 TRUE。
hrErrorCode	HRESULT	如果方法调用成功，则返回 S_OK。发生错误时输出错误代码。

6.1.RequestEventText

1.5.

1.7



此方法返回事件文本。

语法

```
METHOD RequestEventText : BOOL
VAR_INPUT
    nLangId     : DINT;
    sResult     : REFERENCE TO STRING;
    nResultSize : UDINT;
END_VAR
VAR_OUTPUT
    bError      : BOOL;
    hrErrorCode : HRESULT;
END_VAR
```

输入

Name	Type	描述
nLangId	DINT	指定语言 ID 英语（en-US） = 1033 德语（de-DE） = 1031 ……
sResult	REFERENCE TO STRING	引用字符串类型的变量
nResultSize	UDINT	字符串变量的大小，以字节为单位

返回值

Name	Type	描述
RequestEventText	BOOL	请求终止后立即返回 TRUE。如果异步请求仍处于激活状态，则返回 FALSE。必须调用此方法，直到返回值为 TRUE。

输出

Name	Type	描述
bError	BOOL	如果方法调用成功，则返回 FALSE。如果发生错误，则返回 TRUE。
hrErrorCode	HRESULT	如果方法调用成功，则返回 S_OK。发生错误时输出错误代码。

6.1.1.5.2 I_TcMessage

此接口提供消息处理的方法和属性。

继承层次结构

I_TcEventBase [▶ 210]

I_TcMessage

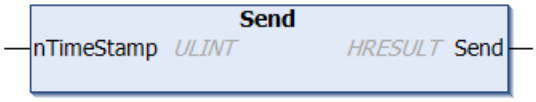
方法

Name	描述
发送 [▶ 216]	发送消息

要求

开发环境	目标平台	要包括的 PLC 库
TwinCAT v3.1.4022.20	PC 或 CX (x64、x86、ARM)	Tc3_EventLogger

6.1.发送
1.5.
2.1



此方法发送消息。

语法

```
METHOD Send : HRESULT
VAR_INPUT
    nTimeStamp: ULINT;
END_VAR
```

输入

Name	Type	描述
nTimeStamp	ULINT	0：已使用当前时间戳 > 0：外部时间戳以 100 纳秒为单位，自 1 月 1 日起，1601（UTC）。

返回值

Name	Type	描述
Send	FB_ HRESULT	如果方法调用成功，则返回 S_OK，否则返回 HRESULT 作为错误代码

6.1.1.5.3 数据类型

6.1.TcEventSeverity
1.5.
3.1

定义事件的严重级别。

语法

定义：

```
{attribute 'qualified_only'}
TYPE TcEventSeverity : (
    Verbose  := 0,
    Info     := 1,
    Warning  := 2,
    Error    := 3,
    Critical := 4);
END_TYPE
```

6.1.2 数据类型

6.1.2.1 配置

6.1.2.1.1 E_DatabaseType

语法

定义：

```
{attribute 'qualified_only'}
TYPE E_DatabaseType :
(
    MS_Compact_SQL := 0,
    MS_Access := 1,
    MS_SQL := 2,
    ASCII := 3,
    ODBC_MySQL := 4,
    ODBC_PostgreSQL := 5,
    ODBC_Oracle := 6,
    ODBC_DB2 := 7,
    ODBC_InterBase := 8,
    ODBC_Firebird := 9,
    XML := 10,
    OCI_Oracle := 11,
    NET_MySQL := 12,
    AzureSQL := 13,
    MS_Excel := 14,
    AS400ISeries := 15,
    OleDB_Database := 16,
    Odbc_Database := 17,
    SQLite:=18,
    ODP_Oracle := 19
);
END_TYPE
```

要求

开发环境	目标平台	要连接的 PLC 库
TwinCAT v3.1 Build 4020.10	PC 或 CX（x86）	Tc3_Database

6.1.2.1.2 E_DBAuthentication

语法

定义：

```
{attribute 'qualified_only'}
TYPE E_DBAuthentication:
(
    None:= 0,
    UserNamePassword := 1,
    x509Cert := 2,
    GSSAPI := 3,
    LDAP := 4
);
END_TYPE
```

要求

开发环境	目标平台	要连接的 PLC 库
TwinCAT v3.1 Build 4020.10	PC 或 CX（x86）	Tc3_Database

6.1.2.1.3 E_OdbcSubType

语法

定义：

```
{attribute 'qualified_only'}
TYPE E_OdbcSubType:
(
    Unknown:= 0,
    MySQL := 1,
    Oracle := 2,
    Postgre := 3,
    DB2 := 4
    Firebird := 5
);
END_TYPE
```

要求

开发环境	目标平台	要连接的 PLC 库
TwinCAT v3.1 Build 4020.10	PC 或 CX（x86）	Tc3_Database

6.1.2.1.4 E_TcDBSrvConfigType

语法

定义：

```
{attribute 'qualified_only'}
TYPE E_TcDBSrvConfigType :
(
    Database := 0,
    AutoLogGroup := 1,
    DBSrvSettings := 2,
    Symbol := 3,
    ADSDevice := 4,
    Table := 5,
    SymbolList := 6
);
END_TYPE
```

要求

开发环境	目标平台	要连接的 PLC 库
TwinCAT v3.1 Build 4020.10	PC 或 CX（x86）	Tc3_Database

6.1.2.1.5 ST_ConfigAutoLogGrp

该结构可用于功能块 FB_ConfigTcDBSrvEvt [▶ 152] 的读取方法。所有已配置的 AutoLog 组都以该结构的数组形式读入 PLC。

语法

定义：

```
TYPE ST_ConfigAutoLogGrp :
STRUCT
    sName: T_MaxString;
    hAutoLogGrpID: UDINT;
    hDBID: UDINT;
    sTableName: T_MaxString;
    stADSDev: ST_ADSDevice;
    eWriteMode: E_WriteMode;
    nCycleTime: TIME;
    nSymbolCount: UDINT;
END_STRUCT
END_TYPE
```

Paratemer

Name	Type	描述
sName	T_MaxString	组名称
hAutoLogGrpID	UDINT	声明的 AutoLog 组的 ID
hDBID	UDINT	指定的数据库的 ID
sTableName	T_MaxString	表格名称
stADSDev	ST_ADSDevice [▶ 233]	ADS 设备信息
eWriteMode	E_WriteMode [▶ 232]	写入模式
nCycleTime	TIME	周期时间
nSymbolCount	UDINT	符号数

要求

开发环境	目标平台	要连接的 PLC 库
TwinCAT v3.1 Build 4020.10	PC 或 CX（x86）	Tc3_Database

6.1.2.1.6 ST_ConfigDB

该结构可用于功能块 FB_ConfigTcDBSrvEvt [▶ 152] 的读取方法。所有已配置的数据库连接都以该结构的数组形式读入 PLC。

语法

定义：

```
TYPE ST_ConfigDB :
STRUCT
    sName: T_MaxString;
    hDBID: DINT;
    eDBType: E_DatabaseType;
    sServer: STRING(80);
    sDatabase: STRING(80);
    bTemp: BOOL;
END_STRUCT
END_TYPE
```

Name	Type	Paratemer
sName	T_MaxString	连接名称
hDBID	DINT	声明的数据库的 ID
eDBType	E_DatabaseType [▶ 216]	Database Type
sServer	STRING (80)	服务器名称
sDatabase	STRING (80)	Database name
bTemp	BOOL	如果只是临时存储连接，则为 TRUE。

要求

开发环境	目标平台	要连接的 PLC 库
TwinCAT v3.1 Build 4020.10	PC 或 CX（x86）	Tc3_Database

6.1.2.1.7 ST_ConnStringParameter

语法

定义：

```
TYPE ST_ConnStringParameter
STRUCT
    sName: T_MaxString;
    sValue: T_MaxString;
END_STRUCT
END_TYPE
```

要求

开发环境	目标平台	要连接的 PLC 库
TwinCAT v3.1 Build 4020.10	PC 或 CX（x86）	Tc3_Database

6.1.2.1.8 ConfigType

为支持的数据库类型提供了不同的数据库配置结构。

6.1.T_DBConfig_ASCII
2.1.
8.1

描述 ASCII 文件的数据库配置结构。

语法

定义：

```
TYPE T_DBConfig_ASCII
STRUCT
    sServer: T_MaxString;
END_STRUCT
END_TYPE
```

Paratemer

Name	Type	描述
sServer	T_MaxString	ASCII 文件的路径

要求

开发环境	目标平台	要连接的 PLC 库
TwinCAT v3.1 Build 4020.10	PC 或 CX（x86）	Tc3_Database

6.1.T_DBConfig_MSAccess
2.1.
8.2

描述 Microsoft Access 数据库的数据库配置结构。

语法

定义：

```
TYPE T_DBConfig_MSAccess
STRUCT
    sServer: T_MaxString;
    sProvider: T_MaxString;
END_STRUCT
END_TYPE
```

Paratemer

Name	Type	描述
sServer	T_MaxString	Access 数据库的路径
sProvider	T_MaxString	Access 2000 – Access 2003： “Microsoft.Jet.OLEDB.4.0” Access 2007： “Microsoft.ACE.OLEDB.12.0”

要求

开发环境	目标平台	要连接的 PLC 库
TwinCAT v3.1 Build 4020.10	PC 或 CX（x86）	Tc3_Database

6.1.T_DBConfig_MsCompactSQL
2.1.
8.3

描述 Microsoft Compact SQL 数据库的数据库配置结构。

语法

定义：

```
TYPE T_DBConfig_MsCompactSQL
STRUCT
    sServer: T_MaxString;
    sPassword: T_MaxString;
    bAuthentification: BOOL;
END_STRUCT
END_TYPE
```

Paratemer

Name	Type	描述
sServer	T_MaxString	Microsoft Compact SQL 文件（*.sdf）的路径
sPassword	T_MaxString	数据库的密码
bAuthentification	BOOL	如果数据库受密码保护，则为 TRUE。

要求

开发环境	目标平台	要连接的 PLC 库
TwinCAT v3.1 Build 4020.10	PC 或 CX（x86）	Tc3_Database

6.1.T_DBConfig_MsExcel
2.1.
8.4

描述 Microsoft Excel 文件的数据库配置结构。

语法

定义：

```
TYPE T_DBConfig_MsExcel
STRUCT
    sServer: T_MaxString;
    sProvider: T_MaxString;
END_STRUCT
END_TYPE
```

Paratemer

Name	Type	描述
sServer	T_MaxString	Microsoft Excel 文件的路径
sProvider	T_MaxString	“Microsoft.Jet.OLEDB.4.0”或“Microsoft.ACE.OLEDB.12.0”

要求

开发环境	目标平台	要连接的 PLC 库
TwinCAT v3.1 Build 4020.10	PC 或 CX（x86）	Tc3_Database

6.1.T_DBConfig_MsSQL
2.1.
8.5

描述 Microsoft SQL 数据库的数据库配置结构。

语法

定义：

```
TYPE T_DBConfig_MsSQL
STRUCT
    sServer: T_MaxString;
    sProvider: T_MaxString;
    sDatabase: T_MaxString;
    sUserID: T_MaxString;
    sPassword: T_MaxString;
    bAuthentification: BOOL;
END_STRUCT
END_TYPE
```

Paratemer

Name	Type	描述
sServer	T_MaxString	在这里输入您的 SQL 服务器的名称。 示例：“TESTSERVER\SQLEXPRESS”
sProvider	T_MaxString	“SQLOLEDB”或 SQL 本地客户端的提供程序，例如， “SQLNCLI10”
数据库	T_MaxString	在这里输入所需的数据库名称。
sUserID	T_MaxString	在这里输入用户名。
sPassword	T_MaxString	在这里输入用户密码。
bAuthentification	BOOL	将该变量设置为 TRUE，以激活基于用户 ID 和密码的身份验证。

要求

开发环境	目标平台	要连接的 PLC 库
TwinCAT v3.1 Build 4020.10	PC 或 CX（x86）	Tc3_Database

6.1.T_DBConfig_Odbc
2.1.
8.6

描述带有 ODBC 接口的数据库的数据库配置结构。

语法

定义：

```
TYPE T_DBConfig_NET_MySQL
STRUCT
    eOdbcSubType: E_OdbcSubType
    nParameterCount: UINT;
    arrParameter: ARRAY [0..MAX_CONFIGPARAMETER] OF ST_ConnStringParameter;
END_STRUCT
END_TYPE
```

Paratemer

Name	Type	描述
eOdbcSubType	E_OdbcSubType [▶ 217]	描述支持完整功能的 ODBC 数据库 [▶ 217]。
nParameterCount	UINT	连接字符串的参数的数量
arrParameter	ARRAY [0..MAX_CONFIGPARAMETER] OF ST_ConnStringParameter [▶ 219];	ST_ConnStringParameter [▶ 219] 类型的连接字符串的参数数组。

要求

开发环境	目标平台	要连接的 PLC 库
TwinCAT v3.1 Build 4020.10	PC 或 CX（x86）	Tc3_Database

6.1.T_DBConfig_SQLite
2.1.
8.7

描述 SQLite 数据库的数据库配置结构。

语法

定义：

```
TYPE T_DBConfig_SQLite
STRUCT
    sServer: T_MaxString;
    sPassword: T_MaxString;
    bAuthentification: BOOL;
END_STRUCT
END_TYPE
```

Paratemer

Name	Type	描述
sServer	T_MaxString	SQLite 文件的路径
sPassword	T_MaxString	数据库的密码
bAuthentification	BOOL	如果数据库受密码保护，则为 TRUE。

要求

开发环境	目标平台	要连接的 PLC 库
TwinCAT v3.1 Build 4020.10	PC 或 CX（x86）	Tc3_Database

6.1.T_DBConfig_XML
2.1.
8.8

描述自定义数据库格式的 XML 文件的数据库配置结构。

语法

定义：

```
TYPE T_DBConfig_XML
STRUCT
    sServer: T_MaxString;
    sSchema: T_MaxString;
    sDatabase: T_MaxString;
END_STRUCT
END_TYPE
```

Paratemer

Name	Type	描述
sServer	T_MaxString	XML 文件的名称和路径
sSchema	T_MaxString	XSD 文件的名称和路径
sDatabase	T_MaxString	描述数据库的名称。为该数据库类型自动创建 XML、XSD 和 XSL 文件。

要求

开发环境	目标平台	要连接的 PLC 库
TwinCAT v3.1 Build 4020.10	PC 或 CX（x86）	Tc3_Database

6.1.2.2 NoSQL

6.1.2.2.1 E_NoSQLDatabaseType

语法

```
{attribute 'qualified_only'}
TYPE E_NoSQLDatabaseType :
(
    UnknownType := 0,
    DocumentDB := 1,
    Binary := 2,
```

```
    TimeSeriesDB := 3
);
END_TYPE
```

要求

开发环境	目标平台	要连接的 PLC 库
TwinCAT v3.1 Build 4020.10	PC 或 CX（x86）	Tc3_Database

6.1.2.2.2 E_DocumentDBQueryType

语法

```
{attribute 'qualified_only'}
TYPE E_DocumentDbQueryType :
(
    InsertOne := 1,
    InsertMany := 2,
    UpdateOne := 3,
    UpdateMany := 4 ,
    Find := 5,
    Aggregation := 6,
    Delete := 7,
    DeleteMany := 8,
    CreateCollection := 9,
    DropCollection := 10
);
END_TYPE
```

要求

开发环境	目标平台	要连接的 PLC 库
TwinCAT v3.1 Build 4020.10	PC 或 CX（x86）	Tc3_Database

6.1.2.2.3 T_ObjectId_MongoDB

映射数据库特定“ObjectId”类型的数据类型。它由 12 个具有不同含义的字节组成：

表 1: 字节级别的 ObjectId

1	2	3	4	5	6	7	8	9	10	11	12
Timestamp				MachineId			ProcessId		Id		

功能块 FB_NoSQLObjecId_MongoDB [▶ 209] 可用于解析各个元素。

语法

```
TYPE T_ObjectId_MongoDB : ARRAY[0..11] OF BYTE;
END_TYPE
```

要求

开发环境	目标平台	要连接的 PLC 库
TwinCAT v3.1 Build 4020.10	PC 或 CX（x86）	Tc3_Database

6.1.2.2.4 QueryOptions

6.1.DocumentDB
2.2.
4.1
6.1.T_QueryOptionDocumentDB_Find
2.2.
4.1.
1

语法

定义：

```
TYPE T_QueryOptionDocumentDB_Find
STRUCT
    pFilter: POINTER TO BYTE;
    cbFilter: UDINT;
    pSort: POINTER TO BYTE;
    cbSort: UDINT;
    pProjection: POINTER TO BYTE;
    cbProjection: UDINT;
END_STRUCT
END_TYPE
```

Paratemer

Name	Type	描述
pFilter	POINTER TO BYTE	指定要基于其搜索集合的搜索筛选器的地址。
cbFilter	UDINT	搜索筛选器的长度。
pSort	POINTER TO BYTE	指定要基于其对集合进行排序的排序地址。
cbSort	UDINT	排序的长度
pProjection	POINTER TO BYTE	指定显示的地址以及如何显示收集数据。
cbProjection	UDINT	显示的长度。

要求

开发环境	目标平台	要引入的 PLC 库
TwinCAT v3.1 版本 4022.20	PC 或 CX（x86）	Tc3_Database

6.1.T_QueryOptionDocumentDB_Aggregate
2.2.
4.1.
2

语法

定义：

```
TYPE T_QueryOptionDocumentDB_Aggregate
STRUCT
    pPipeStages: POINTER TO BYTE;
    cbPipeStages: UDINT;
END_STRUCT
END_TYPE
```

Paratemer

Name	Type	描述
pPipeStages	POINTER TO BYTE	指定阶段文档的地址。其中可以定义几个阶段。
cbPipeStages	UDINT	阶段文档的长度。

要求

开发环境	目标平台	要引入的 PLC 库
TwinCAT v3.1 版本 4022.20	PC 或 CX（x86）	Tc3_Database

6.1.T_QueryOptionDocumentDB_Insert
2.2.
4.1.
3

语法

定义：

```
TYPE T_QueryOptionDocumentDB_Insert
STRUCT
    pDocuments: POINTER TO BYTE;
    cbDocuments: UDINT;
END_STRUCT
END_TYPE
```

Paratemer

Name	Type	描述
pDocuments	POINTER TO BYTE	指定要添加到集合中的一个或多个文档的地址。
cbDocuments	UDINT	文档的长度。

要求

开发环境	目标平台	要引入的 PLC 库
TwinCAT v3.1 版本 4022.20	PC 或 CX（x86）	Tc3_Database

6.1.T_QueryOptionDocumentDB_Update
2.2.
4.1.
4

语法

定义：

```
TYPE T_QueryOptionDocumentDB_Update
STRUCT
    pFilter: POINTER TO BYTE;
    cbFilter: UDINT;
    pDocuments: POINTER TO BYTE;
    cbDocuments: UDINT;
END_STRUCT
END_TYPE
```

Paratemer

Name	Type	描述
pFilter	POINTER TO BYTE	指定要基于其搜索集合的搜索筛选器的地址。
cbFilter	UDINT	搜索筛选器的长度。
pDocuments	POINTER TO BYTE	指定要将其值传输到集合的文档的地址。
cbDocuments	UDINT	文档的长度

要求

开发环境	目标平台	要引入的 PLC 库
TwinCAT v3.1 版本 4022.20	PC 或 CX（x86）	Tc3_Database

6.1.T_QueryOptionDocumentDB_Delete
2.2.
4.1.
5

语法

定义：

```
TYPE T_QueryOptionDocumentDB_Delete
STRUCT
    pFilter: POINTER TO BYTE;
    cbFilter: UDINT;
END_STRUCT
END_TYPE
```

Paratemer

Name	Type	描述
pFilter	POINTER TO BYTE	指定要基于其搜索集合的搜索筛选器的地址。
cbFilter	UDINT	搜索筛选器的长度。

要求

开发环境	目标平台	要引入的 PLC 库
TwinCAT v3.1 版本 4022.20	PC 或 CX（x86）	Tc3_Database

6.1.TimeSeriesDB
2.2.
4.2
6.1.T_QueryOptionTimeSeriesDB_Insert
2.2.
4.2.
1

该结构可用于将数据写入时间序列数据库。可以将结构数组指定为符号。每个单独的数组元素都会被映射为数据库中具有唯一时间戳的一行。通过指定开始时间戳和数据集之间的时间周期可以生成各个系列的时间戳。

语法

定义：

```
TYPE T_QueryOptionTimeSeriesDB_Insert
STRUCT
    pSymbol         : PVOID;
    cbSymbol        : UDINT;
    sDataType       : STRING;
    nDataCount      : UDINT := 1;
    sTableName      : T_MaxString;
    nCycleTime      : UDINT := 100000;
    nStartTimestamp : ULINT := 0;
END_STRUCT
END_TYPE
```

Paratemer

Name	Type	描述
pSymbol	POINTER TO BYTE	指向要写入数据库的符号的指针
cbSymbol	UDINT	指定的符号指针的长度
sDataType	STRING	指定的符号的（简单）类型名称
nDataCount	UDINT	传输的数组符号中的数据集数量
sTableName	T_MaxString	表格或等效表格的名称
nCycleTime	UDINT	循环数据集的时间间隔
nStartTimestamp	ULINT	TwinCAT 时间中首个数据集的时间戳：自 1601 年 1 月 1 日以来以 100 纳秒为单位的时间间隔数。 进一步了解：读取当前 TwinCAT 时间

要求

开发环境	目标平台	要引入的 PLC 库
TwinCAT v3.1 版本 4022.20	PC 或 CX（x86）	Tc3_Database

6.1.T_QueryOptionTimeSeriesDB_Query
2.2.
4.2.
2

该结构可用于从时间序列数据库中读取数据。查询以字符串的形式传递，然后可以传递给 FB_NoSQLQueryBuilder_TimeSeriesDB [▶ 199]。

语法

定义：

```
TYPE T_QueryOptionDocumentDB_Find
STRUCT
    pQuery: POINTER TO BYTE;
    cbQuery: UDINT;
END_STRUCT
END_TYPE
```

Paratemer

Name	Type	描述
pQuery	POINTER TO BYTE	指定查询（字符串）的指针
cbQuery	UDINT	指定查询指针的长度

要求

开发环境	目标平台	要引入的 PLC 库
TwinCAT v3.1 版本 4022.20	PC 或 CX（x86）	Tc3_Database

6.1.2.2.5 E_TimeSeriesDbQueryType

语法

```
{attribute 'qualified_only'}
TYPE E_ TimeSeriesDbQueryType:
(
    Insert := 1,
    Query := 2
);
END_TYPE
```

要求

开发环境	目标平台	要连接的 PLC 库
TwinCAT v3.1 Build 4020.10	PC 或 CX（x86）	Tc3_Database

6.1.2.3 SQL

6.1.2.3.1 E_ColumnType

语法

定义：

```
{attribute 'qualified_only'}
TYPE E_ColumnType :
(
    BigInt := 0,
    Integer := 1,
    SmallInt := 2,
    TinyInt := 3,
    BIT_ := 4,
    Money := 5,
    Float := 6,
    REAL_ := 7,
    DateTime := 8,
    NText := 9,
    NChar := 10,
    Image := 11,
    NVarChar := 12,
    Binary := 13,
    VarBinary := 14
);
END_TYPE
```

要求

开发环境	目标平台	要连接的 PLC 库
TwinCAT v3.1 Build 4020.10	PC 或 CX（x86）	Tc3_Database

6.1.2.3.2 E_SPParameterType

语法

定义：

```
{attribute 'qualified_only'}
TYPE E_SPParameterType :
(
    Input := 0,
    Output := 1,
    InputOutput := 2,
    ReturnValue := 3,
    OracleCursor := 4
);
END_TYPE
```

要求

开发环境	目标平台	要连接的 PLC 库
TwinCAT v3.1 Build 4020.10	PC 或 CX（x86）	Tc3_Database

6.1.2.3.3 E_VerifyResult

语法

定义：

```
{attribute 'qualified_only'}
TYPE E_VerifyResult:
(
    check_None:= 0,
    check_OK := 1,
    check_Error:= 2,
    check_TypeWarning:= 3,
    check_TypeError := 4,
    check_DataLengthWarning := 5,
    check_DataLengthError := 6
);
END_TYPE
```

Values

Name	描述
check_None	PLC 结构未经验证。
check_OK	PLC 结构与数据库表结构之间无差异
check_TypeWarning	就符号而言，PLC 和数据库数据类型之间的数据类型不同，例如 UINT <> INT
check_TypeError	就符号而言，PLC 结构和数据库表结构之间的数据类型不同
check_DataLengthWarning	PLC 结构和数据库表结构之间的长度不同。尽管数据可能会丢失，但是记录然可以映射记录。
check_DataLengthError	PLC 结构和数据库表结构之间的长度不同。无法映射记录。

6.1.2.3.4 ST_SQLSPParameter

功能块 FB_SQLStoredProcedureEvt [▶ 194] 需要该结构来描述要执行的过程的不同参数。

语法

定义：

```
TYPE ST_SQLSPParameter :
STRUCT
    eParameterType: E_SPParameterType;
    eParameterDataType: E_ColumnType;
    nParameterSize: UDINT;
    sParameterName: STRING(50);
END_STRUCT
END_TYPE
```

名称	类型	描述
eParameterType	E_SPParameterType [▶ 229]	参数类型（INPUT、OUTPUT……）
eParameterDataType	E_ColumnType [▶ 229]	参数类型
nParameterSize	UDINT	参数长度
sParameterName	STRING (50)	参数名称

要求

开发环境	目标平台	要连接的 PLC 库
TwinCAT v3.1 Build 4020.10	PC 或 CX（x86）	Tc3_Database

6.1.2.4 PLC

6.1.2.4.1 E_ADSRdWrtMode

语法

定义：

```
{attribute 'qualified_only'}
TYPE E_ADSRdWrtMode :
(
```

```
    bySymbolName := 1,
    IGroup_IOffset := 2
);
END_TYPE
```

要求

开发环境	目标平台	要连接的 PLC 库
TwinCAT v3.1 Build 4020.10	PC 或 CX（x86）	Tc3_Database

6.1.2.4.2 E_ErrorType

语法

定义：

```
{attribute 'qualified_only'}
TYPE E_ErrorType :
(
    noError := 0,
    InternalError,
    DataBaseError,
    ADSError
);
END_TYPE
```

要求

开发环境	目标平台	要连接的 PLC 库
TwinCAT v3.1 Build 4020.10	PC 或 CX（x86）	Tc3_Database

6.1.2.4.3 E_ExpParameterType

语法

定义：

```
{attribute 'qualified_only'}
TYPE E_ExpParameterType :
(
    NULL := 0,
    Boolean := 1,
    Byte_ := 2,
    Int16 := 3,
    Int32 := 4,
    Int64 := 5,
    UInt16 := 6,
    UInt32 := 7,
    UInt64 := 8,
    DateTime := 9,
    Float32 := 10,
    Double64 := 11,
    STRING_ := 12,
    ByteArray := 13,
    Struct_ := 14,
    XMLTAGName := 15
);
END_TYPE
```

要求

开发环境	目标平台	要连接的 PLC 库
TwinCAT v3.1 Build 4020.10	PC 或 CX（x86）	Tc3_Database

6.1.2.4.4 E_OrderColumn

语法

定义：

```
{attribute 'qualified_only'}
TYPE E_OrderColumn :
(
    ID := 0,
    Timestamp := 1,
    Name := 2,
    Value := 3
);
END_TYPE
```

要求

开发环境	目标平台	要连接的 PLC 库
TwinCAT v3.1 Build 4020.10	PC 或 CX（x86）	Tc3_Database

6.1.2.4.5 E_OrderType

语法

定义：

```
{attribute 'qualified_only'}
TYPE E_OrderType :
(
    ASC := 0,
    DESC := 1
);
END_TYPE
```

要求

开发环境	目标平台	要连接的 PLC 库
TwinCAT v3.1 Build 4020.10	PC 或 CX（x86）	Tc3_Database

6.1.2.4.6 E_PLCDataType

语法

定义：

```
{attribute 'qualified_only'}
TYPE E_PLCDataType :
(
    eType_BOOL := 0,
    eType_BYTE := 1,
    eType_SINT := 2,
    eType_INT := 3,
    eType_DINT := 4,
    eType_UINT := 5,
    eType_UDINT := 6,
    eType_WORD := 7,
    eType_DWORD := 8,
    eType_LREAL := 9,
    eType_REAL := 10,
    eType_LINT := 11,
    eType_ULINT := 12,
    eType_BigType := 13
);
END_TYPE
```

要求

开发环境	目标平台	要连接的 PLC 库
TwinCAT v3.1 Build 4020.10	PC 或 CX（x86）	Tc3_Database

6.1.2.4.7 E_WriteMode

语法

定义：

```
{attribute 'qualified_only'}
TYPE E_WriteMode :
(
    eADS_TO_DB_Update := 0,
    eADS_TO_DB_Append := 1,
    eADS_TO_DB_RingBuff_Time := 2,
    eADS_TO_DB_RingBuff_Count := 3
);
END_TYPE
```

要求

开发环境	目标平台	要连接的 PLC 库
TwinCAT v3.1 Build 4020.10	PC 或 CX（x86）	Tc3_Database

6.1.2.4.8 ST_ADSDevice

描述必须为功能块 FB_PLCDBWriteEvt [▶ 173] 的方法指定的 ADS 设备。

语法

定义：

```
TYPE ST_ADSDevice :
STRUCT
    sDevNetID: T_AmsNetId;
    nDevPort: T_AmsPort;
    eADSRdWrtMode: E_ADSRdWrtMode;
    tTimeout: TIME;
END_STRUCT
END_TYPE
```

Paratemer

Name	Type	描述
sDevNetID		ADS 设备的 NetID
nDevPort		AMS 端口
eADSRdWrtMode	E_ADSRdWrtMode [▶ 230]	连接模式 IGroup_IOffset / bySymbol
tTimeout	TIME	ADS 连接超时

要求

开发环境	目标平台	要连接的 PLC 库
TwinCAT v3.1 Build 4020.10	PC 或 CX（x86）	Tc3_Database

6.1.2.4.9 ST_AutoLogGrpStatus

提供有关各个 AutoLog 组的信息。

语法

定义：

```
TYPE ST_AutoLogGrpStatus :
STRUCT
    hAutoLogGrpID: UDINT;
    nCycleCount: UDINT;
    hrErrorCode: HRESULT;
    eErrorType: E_ErrorType;
    bError: BOOL;
END_STRUCT
END_TYPE
```

Paratemer

Paratemer

Name	Type	描述
hAutoLogGrpID	UDINT	声明的 AutoLog 组的 ID
nCycleCount	UDINT	执行的周期数
hrErrorCode	HRESULT	HRESULT 错误代码
eErrorType	E_ErrorType [▶ 231]	错误类型
bError	BOOL	如果发生错误，则为 TRUE。

要求

开发环境	目标平台	要连接的 PLC 库
TwinCAT v3.1 Build 4020.10	PC 或 CX（x86）	Tc3_Database

6.1.2.4.10 ST_ColumnInfo

语法

定义：

```
TYPE ST_ColumnInfo :
STRUCT
    sName: STRING(50);
    sProperty: STRING;
    nLength: UDINT;
    eType: E_ColumnType;
END_STRUCT
END_TYPE
```

Paratemer

Name	Type	描述
sName	STRING (50)	列的名称
sProperty	string	附加列属性的字符串
nLength	UDINT	最大长度（适用于字符串和字节流）
eType	E_ColumnType [▶ 229]	Column type

要求

开发环境	目标平台	要连接的 PLC 库
TwinCAT v3.1 Build 4020.10	PC 或 CX（x86）	Tc3_Database

6.1.2.4.11 ST_ExpParameter

功能块 FB_PLCCmd [▶ 177] 需要该结构，用于在 SQL 命令中提供不同参数（占位符）的描述。

语法

定义：

```
TYPE ST_ExpParameter:
STRUCT
    sParaName : T_MaxString;
    nParaSize : UDINT;
    eParaType : E_ExpParameterType;
END_STRUCT
END_TYPE
```

Paratemer

Name	Type	描述
sParaName	T_MaxString	参数（占位符）的名称
nParaSize	UDINT	参数值的长度
eParaType	E_ExpParameterType [▶ 231]	参数的数据类型

要求

开发环境	目标平台	要连接的 PLC 库
TwinCAT v3.1 Build 4020.10	PC 或 CX（x86）	Tc3_Database

6.1.2.4.12 ST_StandardRecord

如果您想要使用 TwinCAT 数据库服务器的标准表结构工作，则可以在 PLC 中使用该结构。

该结构不能用于 Microsoft Access 数据库，因为该数据库类型不支持 64 位整数数据类型。在这种情况下，应该使用 ST_StandardRecord_MSAccess [▶ 235] 结构。

语法

定义：

```
TYPE ST_StandardRecord :
STRUCT
    nID: LINT;
    dtTimestamp: DT;
    sName: STRING(80);
    rValue: LREAL;
END_STRUCT
END_TYPE
```

要求

开发环境	目标平台	要连接的 PLC 库
TwinCAT v3.1 Build 4020.10	PC 或 CX（x86）	Tc3_Database

6.1.2.4.13 ST_StandardRecord_MSAccess

如果您想要使用 TwinCAT 数据库服务器的标准表结构工作，则可以在 PLC 中使用该结构。该结构专门用于 Microsoft Access 数据库，因为该数据库类型不支持 64 位整数数据类型。

语法

定义：

```
TYPE ST_StandardRecord_MSAccess:
STRUCT
    nID: DINT;
    dtTimestamp: DT;
    sName: STRING(80);
    rValue: LREAL;
END_STRUCT
END_TYPE
```

要求

开发环境	目标平台	要连接的 PLC 库
TwinCAT v3.1 Build 4020.10	PC 或 CX（x86）	Tc3_Database

6.1.2.4.14 ST_Symbol

描述了 ADS 符号，该符号必须为功能块 FB_PLCDBWriteEvt [▶ 173] 的方法指定。

语法

定义：

```
TYPE ST_Symbol :
STRUCT
    sSymbolName: T_MaxString;
    sDBSymbolName: T_MaxString;
    nIGroup: UDINT;
    nIOffset: UDINT;
    nBitSize: UDINT;
    eDataType: E_PLCDataType;
END_STRUCT
END_TYPE
```

Paratemer

Name	Type	描述
sSymbolName	T_MaxString	符号名称
sDBSymbolName	T_MaxString	要写入数据库的名称。
nIGroup	UDINT	索引组（仅适用于 ADSRdWrtMode “eADSMode_IGroup_IOffset”）
nIOffset	UDINT	索引偏移（仅适用于 ADSRdWrtMode “eADSMode_IGroup_IOffset”）
nBitSize	UDINT	长度，以位为单位（仅适用于 ADSRdWrtMode “eADSMode_IGroup_IOffset”）
eDataType	E_PLCDataType [▶ 232]	数据类型（仅适用于 ADSRdWrtMode “eADSMode_IGroup_IOffset”）

要求

开发环境	目标平台	要连接的 PLC 库
TwinCAT v3.1 Build 4020.10	PC 或 CX（x86）	Tc3_Database

6.1.3 全局常量

6.1.3.1 常量

VAR_GLOBAL CONSTANT GVL

```
AMSPORT_DBSRV : UINT := 21372;
MAX_DBCONNECTIONS : UDINT := 255;
MAX_DBCOLUMNS : UDINT := 255;
MAX_SPPARAMETER : UDINT := 255;
MAX_CONFIGURATIONS : UDINT := 255;
MAX_RECORDS : UDINT := 255;
```

要求

开发环境	目标平台	要连接的 PLC 库
TwinCAT v3.1 Build 4020.10	PC 或 CX（x86）	Tc3_Database

6.1.4 不再支持

6.1.4.1 配置模式

6.1.4.1.1 FB_ConfigTcDBSrv

用于创建、读取和删除 TwinCAT 数据库服务器的配置条目的功能块。

语法

定义：

```
FUNCTION_BLOCK FB_ConfigTcDBSrv
VAR_INPUT
    sNetID: T_AmsNetID := '';
    tTimeout: TIME := T#5S;
END_VAR
VAR_OUTPUT
    bBusy: BOOL;
    bError: BOOL;
    ipTcResultEvent: Tc3_EventLogger.I_TcResultEvent;
END_VAR
```

输入

Name	Type	描述
sNetID	T_AmsNetID	ADS 命令指向的目标设备的 AMS 网络 ID。
tTimeout	TIME	表示函数被取消前的时间。

输出

Name	Type	描述
bBusy	BOOL	功能块的方法处于活动状态时即为 TRUE。
bError	BOOL	发生错误时即为 TRUE。
ipTcResultEvent	Tc3_EventLogger.I_TcResultEvent	结果界面，提供有关返回值的详细信息。

方法

Name	定义位置	描述
创建 [▶ 153]	本地	在 XML 配置文件中为 TwinCAT 数据库服务器创建新条目
读取 [▶ 154]	本地	读取 TwinCAT 数据库服务器的当前配置
删除 [▶ 155]	本地	从 TwinCAT 数据库服务器的配置中删除数据库和 AutoLog 组

要求

开发环境	目标平台	要连接的 PLC 库
TwinCAT v3.1 Build 4020.10	PC 或 CX（x86）	Tc3_Database

6.1.创建
4.1.
1.1

该方法可在 XML 配置文件中为 TwinCAT 数据库服务器创建新条目。TwinCAT 数据库服务器可以选择临时使用新条目。在这种情况下，不会将任何数据写入 XML 文件。

语法

```
METHOD Create : BOOL
VAR_INPUT
    pTcDBSrvConfig: POINTER TO BYTE;
    cbTcDBSrvConfig: UDINT;
    bTemporary: BOOL := TRUE;
    pConfigID: POINTER TO UDINT;
END_VAR
```

输入

Name	Type	描述
pTcDBSrvConfig	POINTER TO BYTE	要创建的配置结构的指针。
cbTcDBSrvConfig	UDINT	配置结构的长度
bTemporary	BOOL	指定是否要将配置存储在 XML 文件中。
pConfigID	POINTER TO UDINT	返回配置 ID（hDBID 或 hAutoLogGrpID）的指针

目前不支持创建 AutoLog 组。

返回值

Name	Type	描述
创建	BOOL	显示方法的状态。在方法执行完毕时，即使出现错误，也会返回 TRUE。

示例

```
VAR
    fbConfigTcDBSrv : FB_ConfigTcDBSrv(sNetId := '', tTimeout:=T#5S);
    myConfigHandle  : INT;
    // Any other ConfigType can be used here
    stConfigDB      : T_DBConfig_MsCompactSQL;
END_VAR
```

```
stConfigDB.bAuthentification := FALSE;
stConfigDB.sServer := 'C:\Recipes.sdf';

IF fbConfigTcDBSrv.Create(
    pTcDBSrvConfig:= ADR(stConfigDB),
    cbTcDBSrvConfig:= SIZEOF(stConfigDB),
    bTemporary:= TRUE,
    pConfigID:= ADR(myConfigHandle))
THEN
    IF fbSQLStoredProcedure.bError THEN
        nState := 255;
    ELSE
        nState := 0;
    END_IF
END_IF
```

6.1.读取

4.1.

1.2

该方法可以用于读取 TwinCAT 数据库服务器的当前配置。任何可能被包含的临时配置都会对其进行相应的标记。

语法

```
METHOD Read : BOOL
VAR_INPUT
    pDBConfig: POINTER TO ARRAY [1..MAX_CONFIGURATIONS] OF ST_ConfigDB;
    cbDBConfig: UDINT;
    pAutoLogGrpConfig: POINTER TO ARRAY[1..MAX_CONFIGURATIONS] OF
ST_ConfigAutoLogGrp;
    cbAutoLogGrpConfig: UDINT;
```

```
    pDBCount: POINTER TO UDINT;
    pAutoLogGrpCount: POINTER TO UDINT;
END_VAR
```

输入

Name	Type	描述
pDBConfig	POINTER TO ARRAY [1..MAX_CONFIGURATIONS] OF ST_ConfigDB [▶ 219]	用于写入数据库配置的数组的指针地址。
cbDBConfig	UDINT	数据库配置数组的长度
pAutoLogGrpConfig	POINTER TO ARRAY[1..MAX_CONFIGURATIONS] OF ST_ConfigAutoLogGrp [▶ 218]	用于写入 AutoLogGrp 配置的数组的指针地址。
cbAutoLogGrpConfig	UDINT	AutoLogGrp 配置数组的长度
pDBCount	POINTER TO UDINT	用于存储数据库配置数量的指针地址。
pAutoLogGrpCount	POINTER TO UDINT	用于存储 AutoLogGrp 配置数量的指针地址。

返回值

Name	Type	描述
读取	BOOL	显示方法的状态。在方法执行完毕时，即使出现错误，也会返回 TRUE。

示例

```
VAR
    fbConfigTcDBSrv    : FB_ConfigTcDBSrv(sNetId := '', tTimeout:=T#5S);
    aDBConfig          : ARRAY[0..MAX_CONFIGURATIONS] OF ST_ConfigDB;
    aAutoGrpConfig     : ARRAY[0..MAX_CONFIGURATIONS] OF ST_ConfigAutoLogGrp;
    nDbCount           : UDINT;
    nAutoGrpCount      : UDINT;
END_VAR
```

```
IF fbConfigTcDBSrv.Read(
    pDBConfig := ADR(aDBConfig),
    cbDBConfig := SIZEOF(aDBConfig),
    pAutologGrpConfig := ADR(aAutoGrpConfig),
    cbAutoLogGrpConfig := SIZEOF(aAutoGrpConfig),
    pDBCount := ADR(nDbCount),
    pAutoLogGrpCount := ADR(nAutoGrpCount))
 THEN
    IF fbConfigTcDBSrv.bError THEN
        nState := 255;
    ELSE
        nState := 0;
    END_IF
END_IF
```

6.1.删除

4.1.

1.3

该方法可以用于从 TwinCAT 数据库服务器的配置中删除数据库和 AutoLog 组。

语法

```
METHOD Delete : BOOL
VAR_INPUT
    eTcDBSrvConfigType: E_TcDBSrvConfigType;
    hConfigID: UDINT;
END_VAR
```

输入

Name	Type	描述
eTcDBSrvConfigType	E_TcDBSrvConfigType	要删除的配置的类型（数据库/AutoLog 组）
hConfigID	UDINT	要删除的配置的 ID（hDBID 或 hAutoLogGrpID）

返回值

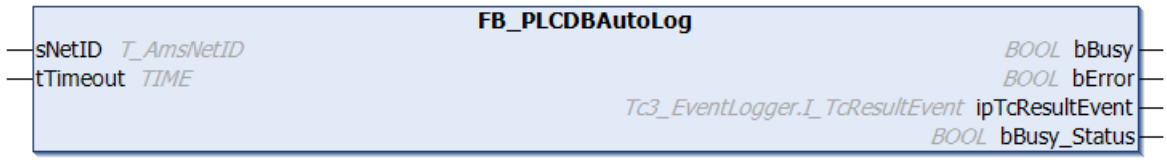
Name	Type	描述
删除	BOOL	显示方法的状态。在方法执行完毕时，即使出现错误，也会返回 TRUE。

示例

```
VAR
    fbConfigTcDBSrv : FB_ConfigTcDBSrv(sNetId := '', tTimeout:=T#5S);
    myConfigHandle  : INT;
END_VAR
```

```
IF fbConfigTcDBSrv.Delete(
    eTcDBSrvConfigType := E_TcDBSrvConfigType.Database,
    hConfigID := myConfigHandle) THEN
IF fbConfigTcDBSrv.bError THEN
        nState := 255;
    ELSE
        nState := 0;
    END_IF
END_IF
```

6.1.4.1.2 FB_PLCDBAutoLog



功能块具有 4 种方法，用于启动和停止已定义的 AutoLog 组，以及读取相应的组状态。

语法

定义：

```
FUNCTION_BLOCK FB_PLCDBAutoLog
VAR_INPUT
    sNetID: T_AmsNetID := '';
    tTimeout: TIME := T#5S;
END_VAR
VAR_OUTPUT
    bBusy: BOOL;
    bError: BOOL;
    ipTcResultEvent: Tc3_EventLogger.I_TcResultEvent;
    bBusy_Status: BOOL;
END_VAR
```

输入

名称	类型	描述
sNetID	T_AmsNetID	ADS 命令指向的目标设备的 AMS 网络 ID。
tTimeout	TIME	表示函数被取消前的时间。

输出

名称	类型	描述
bBusy	BOOL	除Status方法外，功能块的方法处于活动状态时为 TRUE。
bError	BOOL	发生错误时即为 TRUE。
ipTcResultEvent	Tc3_EventLogger.I_TcResultEvent	结果界面，提供有关返回值的详细信息。
bBusy_Status	BOOL	Status方法可以独立于功能块的其他 3 种方法执行，因此该方法有自己独立的繁忙标志。Status 方法处于活动状态时为 TRUE。

方法

名称	定义位置	描述
RunOnce [▶ 157]	本地	执行一次 AutoLog 组
启动 [▶ 157]	本地	使用相应配置的 AutoLog 组启动 AutoLog 模式
Status [▶ 157]	本地	查询 AutoLog 组的状态。
停止 [▶ 158]	本地	停止 AutoLog 模式

要求

开发环境	目标平台	要连接的 PLC 库
TwinCAT v3.1 Build 4020.10	PC 或 CX（x86）	Tc3_Database

6.1.RunOnce
4.1.
2.1

该方法可以用于执行一次 AutoLog 组，例如，基于控制器中的一个事件。

语法

```
METHOD RunOnce : BOOL
VAR_INPUT
    hAutoLogGrpID: UDINT;
    bAll: BOOL;
END_VAR
```

输入

名称	类型	描述
hAutoLogGrpID	UDINT	要执行一次的 AutoLog 组的 ID。
bAll	BOOL	如果为 TRUE，则所有 AutoLog 组都执行一次。

返回值

名称	类型	描述
RunOnce	BOOL	显示方法的状态。在方法执行完毕时，即使出现错误，也会返回 TRUE。

示例

```
VAR
    fbPLCDBAutoLog    : FB_PLCDBAutoLog (sNetID:='', tTimeout := T#5S);
END_VAR
```

```
IF fbPLCDBAutoLog.RunOnce(hAutologGrpID := 1, bAll := FALSE) THEN
    ; // ...
END_IF
```

6.1.启动
4.1.
2.2

该方法使用相应配置的 AutoLog 组启动 AutoLog 模式。

语法

```
METHOD Start : BOOL
```

返回值

名称	类型	描述
启动	BOOL	显示方法的状态。在方法执行完毕时，即使出现错误，也会返回 TRUE。

示例

```
VAR
    fbPLCDBAutoLog    : FB_PLCDBAutoLog (sNetID:='', tTimeout := T#5S);
END_VAR
```

```
IF fbPLCDBAutoLog.Start() THEN
    ; // ...
END_IF
```

6.1.Status
4.1.
2.3

该方法可以用于查询 AutoLog 组的状态。由于该方法可以独立于功能块的其他方法调用，因此在功能块主体中为该方法提供了一个单独的繁忙标志：bBusy_Status。

语法

```
METHOD Status : BOOL
VAR_INPUT
    tCheckCycle: TIME;
    pError: POINTER TO BOOL;
    pAutoLogGrpStatus: POINTER TO ARRAY [1..MAX_CONFIGURATIONS] OF ST_AutoLogGrpStatus;
    cbAutoLogGrpStatus: UDINT;
END_VAR
```

输入

名称	类型	描述
tCheckCycle	TIME	更新状态数组的间隔时间。
pError	POINTER TO BOOL	如果在 AutoLog 模式下发生错误，则为 TRUE。
pAutoLogStatus	指向 ARRAY [1..MAX_CONFIGURATIONS] OF ST_AutoLogGrpStatus 的指针	包含所有组的状态数组的地址。
cbAutoLogStatus	UDINT	状态数组的长度

返回值

名称	类型	描述
Status	BOOL	显示方法的状态。在方法执行完毕时，即使出现错误，也会返回 TRUE。

示例

```
VAR
    fbPLCDBAutoLog    : FB_PLCDBAutoLog(sNetID:='', tTimeout := T#5S);
    bError            : BOOL;
    aAutologGrpStatus : ARRAY[0..MAX_CONFIGURATIONS] OF ST_AutoLogGrpStatus;
END_VAR
```

```
IF fbPLCDBAutoLog.Status(tCheckCycle := T#30S, ADR(bError), ADR(aAutologGrpStatus), SIZEOF(aAutologG
rpStatus)) THEN
    ; // ...
END_IF
```

6.1.停止

4.1.

2.4

该方法可停止 AutoLog 模式。

语法

```
METHOD Stop : BOOL
```

返回值

名称	类型	描述
停止	BOOL	显示方法的状态。在方法执行完毕时，即使出现错误，也会返回 TRUE。

示例

```
VAR
    fbPLCDBAutoLog    : FB_PLCDBAutoLog (sNetID:='', tTimeout := T#5S);
END_VAR
```

```
IF fbPLCDBAutoLog.Stop() THEN
    ; // ...
END_IF
```

6.1.4.2 PLC 专家模式

6.1.4.2.1 FB_ConfigTcDBSrv

用于创建、读取和删除 TwinCAT 数据库服务器的配置条目的功能块。

语法

定义：

```
FUNCTION_BLOCK FB_ConfigTcDBSrv
VAR_INPUT
    sNetID: T_AmsNetID := '';
    tTimeout: TIME := T#5S;
END_VAR
VAR_OUTPUT
    bBusy: BOOL;
    bError: BOOL;
    ipTcResultEvent: Tc3_EventLogger.I_TcResultEvent;
END_VAR
```

输入

Name	Type	描述
sNetID	T_AmsNetID	ADS 命令指向的目标设备的 AMS 网络 ID。
tTimeout	TIME	表示函数被取消前的时间。

输出

Name	Type	描述
bBusy	BOOL	功能块的方法处于活动状态时即为 TRUE。
bError	BOOL	发生错误时即为 TRUE。
ipTcResultEvent	Tc3_EventLogger.I_TcResultEvent	结果界面，提供有关返回值的详细信息。

方法

Name	定义位置	描述
创建 [▶ 244]	本地	在 XML 配置文件中为 TwinCAT 数据库服务器创建新条目
读取 [▶ 245]	本地	读取 TwinCAT 数据库服务器的当前配置
删除 [▶ 246]	本地	从 TwinCAT 数据库服务器的配置中删除数据库和 AutoLog 组

要求

开发环境	目标平台	要连接的 PLC 库
TwinCAT v3.1 Build 4020.10	PC 或 CX（x86）	Tc3_Database

6.1.创建

4.2.

1.1

该方法可在 XML 配置文件中为 TwinCAT 数据库服务器创建新条目。TwinCAT 数据库服务器可以选择临时使用新条目。在这种情况下，不会将任何数据写入 XML 文件。

语法

```
METHOD Create : BOOL
VAR_INPUT
    pTcDBSrvConfig: POINTER TO BYTE;
    cbTcDBSrvConfig: UDINT;
    bTemporary: BOOL := TRUE;
    pConfigID: POINTER TO UDINT;
END_VAR
```

输入

Name	Type	描述
pTcDBSrvConfig	POINTER TO BYTE	要创建的配置结构的指针。
cbTcDBSrvConfig	UDINT	配置结构的长度
bTemporary	BOOL	指定是否要将配置存储在 XML 文件中。
pConfigID	POINTER TO UDINT	返回配置 ID（hDBID 或 hAutoLogGrpID）的指针

目前不支持创建 AutoLog 组。

返回值

Name	Type	描述
创建	BOOL	显示方法的状态。在方法执行完毕时，即使出现错误，也会返回 TRUE。

示例

```
VAR
    fbConfigTcDBSrv : FB_ConfigTcDBSrv(sNetId := '', tTimeout:=T#5S);
    myConfigHandle  : INT;
```

```
    // Any other ConfigType can be used here
    stConfigDB      : T_DBConfig_MsCompactSQL;
END_VAR
```

```
stConfigDB.bAuthentification := FALSE;
stConfigDB.sServer := 'C:\Recipes.sdf';

IF fbConfigTcDBSrv.Create(
    pTcDBSrvConfig:= ADR(stConfigDB),
    cbTcDBSrvConfig:= SIZEOF(stConfigDB),
    bTemporary:= TRUE,
    pConfigID:= ADR(myConfigHandle))
THEN
    IF fbSQLStoredProcedure.bError THEN
        nState := 255;
    ELSE
        nState := 0;
    END_IF
END_IF
```

6.1.读取

4.2.

1.2

该方法可以用于读取 TwinCAT 数据库服务器的当前配置。任何可能被包含的临时配置都会对其进行相应的标记。

语法

```
METHOD Read : BOOL
VAR_INPUT
    pDBConfig: POINTER TO ARRAY [1..MAX_CONFIGURATIONS] OF ST_ConfigDB;
    cbDBConfig: UDINT;
    pAutoLogGrpConfig: POINTER TO ARRAY[1..MAX_CONFIGURATIONS] OF
ST_ConfigAutoLogGrp;
    cbAutoLogGrpConfig: UDINT;
    pDBCount: POINTER TO UDINT;
    pAutoLogGrpCount: POINTER TO UDINT;
END_VAR
```

输入

Name	Type	描述
pDBConfig	POINTER TO ARRAY [1..MAX_CONFIGURATIONS] OF ST_ConfigDB [▶ 219]	用于写入数据库配置的数组的指针地址。
cbDBConfig	UDINT	数据库配置数组的长度
pAutoLogGrpConfig	POINTER TO ARRAY[1..MAX_CONFIGURATIONS] OF ST_ConfigAutoLogGrp [▶ 218]	用于写入 AutoLogGrp 配置的数组的指针地址。
cbAutoLogGrpConfig	UDINT	AutoLogGrp 配置数组的长度
pDBCount	POINTER TO UDINT	用于存储数据库配置数量的指针地址。
pAutoLogGrpCount	POINTER TO UDINT	用于存储 AutoLogGrp 配置数量的指针地址。

返回值

Name	Type	描述
读取	BOOL	显示方法的状态。在方法执行完毕时，即使出现错误，也会返回 TRUE。

示例

```
VAR
    fbConfigTcDBSrv   : FB_ConfigTcDBSrv(sNetId := '', tTimeout:=T#5S);
    aDBConfig         : ARRAY[0..MAX_CONFIGURATIONS] OF ST_ConfigDB;
    aAutoGrpConfig    : ARRAY[0..MAX_CONFIGURATIONS] OF ST_ConfigAutoLogGrp;
    nDbCount          : UDINT;
    nAutoGrpCount     : UDINT;
END_VAR
```

```
IF fbConfigTcDBSrv.Read(
    pDBConfig := ADR(aDBConfig),
    cbDBConfig := SIZEOF(aDBConfig),
    pAutologGrpConfig := ADR(aAutoGrpConfig),
    cbAutoLogGrpConfig := SIZEOF(aAutoGrpConfig),
    pDBCount := ADR(nDbCount),
    pAutoLogGrpCount := ADR(nAutoGrpCount))
 THEN
    IF fbConfigTcDBSrv.bError THEN
        nState := 255;
    ELSE
        nState := 0;
    END_IF
END_IF
```

6.1.删除

4.2.

1.3

该方法可以用于从 TwinCAT 数据库服务器的配置中删除数据库和 AutoLog 组。

语法

```
METHOD Delete : BOOL
VAR_INPUT
    eTcDBSrvConfigType: E_TcDBSrvConfigType;
    hConfigID: UDINT;
END_VAR
```

输入

Name	Type	描述
eTcDBSrvConfigType	E_TcDBSrvConfigType	要删除的配置的类型（数据库/AutoLog 组）
hConfigID	UDINT	要删除的配置的 ID（hDBID 或 hAutoLogGrpID）

返回值

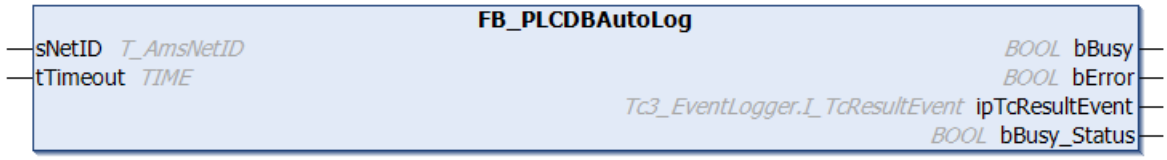
Name	Type	描述
删除	BOOL	显示方法的状态。在方法执行完毕时，即使出现错误，也会返回 TRUE。

示例

```
VAR
    fbConfigTcDBSrv : FB_ConfigTcDBSrv(sNetId := '', tTimeout:=T#5S);
    myConfigHandle  : INT;
END_VAR
```

```
IF fbConfigTcDBSrv.Delete(
    eTcDBSrvConfigType := E_TcDBSrvConfigType.Database,
    hConfigID := myConfigHandle) THEN
IF fbConfigTcDBSrv.bError THEN
        nState := 255;
    ELSE
        nState := 0;
    END_IF
END_IF
```

6.1.4.2.2 FB_PLCDBAutoLog



功能块具有 4 种方法，用于启动和停止已定义的 AutoLog 组，以及读取相应的组状态。

语法

定义：

```
FUNCTION_BLOCK FB_PLCDBAutoLog
VAR_INPUT
    sNetID: T_AmsNetID := '';
    tTimeout: TIME := T#5S;
END_VAR
VAR_OUTPUT
    bBusy: BOOL;
    bError: BOOL;
    ipTcResultEvent: Tc3_EventLogger.I_TcResultEvent;
    bBusy_Status: BOOL;
END_VAR
```

输入

名称	类型	描述
sNetID	T_AmsNetID	ADS 命令指向的目标设备的 AMS 网络 ID。
tTimeout	TIME	表示函数被取消前的时间。

输出

名称	类型	描述
bBusy	BOOL	除Status方法外，功能块的方法处于活动状态时为 TRUE。
bError	BOOL	发生错误时即为 TRUE。
ipTcResultEvent	Tc3_EventLogger.I_TcResultEvent	结果界面，提供有关返回值的详细信息。
bBusy_Status	BOOL	Status方法可以独立于功能块的其他 3 种方法执行，因此该方法有自己独立的繁忙标志。Status方法处于活动状态时为 TRUE。

方法

名称	定义位置	描述
RunOnce [▶ 157]	本地	执行一次 AutoLog 组
启动 [▶ 157]	本地	使用相应配置的 AutoLog 组启动 AutoLog 模式
Status [▶ 157]	本地	查询 AutoLog 组的状态。
停止 [▶ 158]	本地	停止 AutoLog 模式

要求

开发环境	目标平台	要连接的 PLC 库
TwinCAT v3.1 Build 4020.10	PC 或 CX（x86）	Tc3_Database

6.1.RunOnce

4.2.

2.1

该方法可以用于执行一次 AutoLog 组，例如，基于控制器中的一个事件。

语法

```
METHOD RunOnce : BOOL
VAR_INPUT
    hAutoLogGrpID: UDINT;
    bAll: BOOL;
END_VAR
```

输入

名称	类型	描述
hAutoLogGrpID	UDINT	要执行一次的 AutoLog 组的 ID。
bAll	BOOL	如果为 TRUE，则所有 AutoLog 组都执行一次。

返回值

名称	类型	描述
RunOnce	BOOL	显示方法的状态。在方法执行完毕时，即使出现错误，也会返回 TRUE。

示例

```
VAR
    fbPLCDBAutoLog    : FB_PLCDBAutoLog (sNetID:='', tTimeout := T#5S);
END_VAR
```

```
IF fbPLCDBAutoLog.RunOnce(hAutologGrpID := 1, bAll := FALSE) THEN
    ; // ...
END_IF
```

6.1.启动

4.2.

2.2

该方法使用相应配置的 AutoLog 组启动 AutoLog 模式。

语法

```
METHOD Start : BOOL
```

返回值

名称	类型	描述
启动	BOOL	显示方法的状态。在方法执行完毕时，即使出现错误，也会返回 TRUE。

示例

```
VAR
    fbPLCDBAutoLog    : FB_PLCDBAutoLog (sNetID:='', tTimeout := T#5S);
END_VAR
```

```
IF fbPLCDBAutoLog.Start() THEN
    ; // ...
END_IF
```

6.1.Status

4.2.

2.3

该方法可以用于查询 AutoLog 组的状态。由于该方法可以独立于功能块的其他方法调用，因此在功能块主体中为该方法提供了一个单独的繁忙标志：bBusy_Status。

语法

```
METHOD Status : BOOL
VAR_INPUT
    tCheckCycle: TIME;
    pError: POINTER TO BOOL;
    pAutoLogGrpStatus: POINTER TO ARRAY [1..MAX_CONFIGURATIONS] OF ST_AutoLogGrpStatus;
    cbAutoLogGrpStatus: UDINT;
END_VAR
```

输入

名称	类型	描述
tCheckCycle	TIME	更新状态数组的间隔时间。
pError	POINTER TO BOOL	如果在 AutoLog 模式下发生错误，则为 TRUE。
pAutoLogStatus	指向 ARRAY [1..MAX_CONFIGURATIONS] OF ST_AutoLogGrpStatus 的指针	包含所有组的状态数组的地址。
cbAutoLogStatus	UDINT	状态数组的长度

返回值

名称	类型	描述
Status	BOOL	显示方法的状态。在方法执行完毕时，即使出现错误，也会返回 TRUE。

示例

```
VAR
    fbPLCDBAutoLog    : FB_PLCDBAutoLog(sNetID:='', tTimeout := T#5S);
    bError            : BOOL;
    aAutologGrpStatus : ARRAY[0..MAX_CONFIGURATIONS] OF ST_AutoLogGrpStatus;
END_VAR
```

```
IF fbPLCDBAutoLog.Status(tCheckCycle := T#30S, ADR(bError), ADR(aAutologGrpStatus), SIZEOF(aAutologGrpStatus)) THEN
    ; // ...
END_IF
```

6.1.停止

4.2.

2.4

该方法可停止 AutoLog 模式。

语法

```
METHOD Stop : BOOL
```

返回值

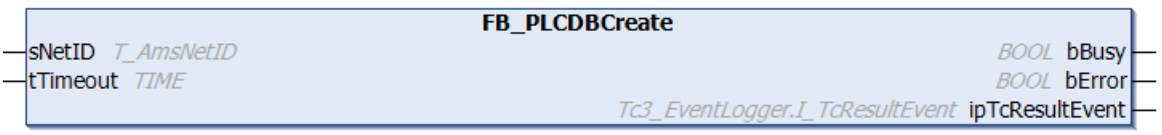
名称	类型	描述
停止	BOOL	显示方法的状态。在方法执行完毕时，即使出现错误，也会返回 TRUE。

示例

```
VAR
    fbPLCDBAutoLog    : FB_PLCDBAutoLog (sNetID:='', tTimeout := T#5S);
END_VAR
```

```
IF fbPLCDBAutoLog.Stop() THEN
    ; // ...
END_IF
```

6.1.4.2.3 FB_PLCDBCreate



具有 2 种方法的功能块。一种可以用于在 PLC 中指定的数据库服务器上从 PLC 创建数据库的方法。另一种方法可用于在指定的数据库中生成一个新的表格。

语法

定义：

```
FUNCTION_BLOCK FB_PLCDBCreate
VAR_INPUT
    sNetID: T_AmsNetID := '';
    tTimeout: TIME := T#5S;
END_VAR
VAR_OUTPUT
    bBusy: BOOL;
    bError: BOOL;
    ipTcResultEvent: Tc3_EventLogger.I_TcResultEvent
END_VAR
```

输入

Name	Type	描述
sNetID	T_AmsNetID	ADS 命令指向的目标设备的 AMS 网络 ID。
tTimeout	TIME	表示函数被取消前的时间。

输出

Name	Type	描述
bBusy	BOOL	功能块的方法处于活动状态时即为 TRUE。
bError	BOOL	发生错误时即为 TRUE。
ipTcResultEvent	Tc3_EventLogger.I_TcResultEvent	结果界面，提供有关返回值的详细信息。

方法

Name	定义位置	描述
数据库 [▶ 250]	本地	创建一个新的数据库
表格 [▶ 251]	本地	通过 PLC 中一个包含 x 个元素（即 x 列）的数组来定义表结构，并创建一个新表。

要求

开发环境	目标平台	要连接的 PLC 库
TwinCAT v3.1 Build 4020.10	PC 或 CX（x86）	Tc3_Database

6.1.数据库

4.2.

3.1

该方法可创建一个新的数据库。您可以指定已创建的数据库是否也应该用于 TwinCAT 数据库服务器的配置。

语法

```
METHOD Database : BOOL
VAR_INPUT
    pDatabaseConfig: POINTER TO BYTE;
    cbDatabaseConfig: UDINT;
    bCreateXMLConfig: BOOL;
    pDBID: POINTER TO UDINT;
END_VAR
```

输入

Name	Type	描述
pDatabaseConfig	POINTER TO BYTE	数据库配置结构 [▶ 219]的地址
cbDatabaseConfig	UDINT	数据库配置结构的长度
bCreateXMLConfig	BOOL	指定是否将新创建的数据库作为新的配置条目录入到 XML 文件。
pDBID	UDINT	如果/当创建了新的配置条目时，则返回 hDBID。

返回值

Name	Type	描述
数据库	BOOL	显示方法的状态。在方法执行完毕时，即使出现错误，也会返回 TRUE。

示例

```
VAR
    fbPLCDBCreate : FB_PLCDBCreateEvt(sNetID := '', tTimeout := T#5S);
    stConfigDB    : T_DBConfig_MsCompactSQL;
    hDBID         : UDINT;
    tcMessage     : I_TcMessage;
END_VAR
```

```
stConfigDB.bAuthentification := FALSE;
stConfigDB.sServer := 'C:\Test.sdf';

IF fbPLCDBCreate.Database(
    pDatabaseConfig:= ADR(stConfigDB),
    cbDatabaseConfig := SIZEOF(stConfigDB),
    bCreateXMLConfig := TRUE,
    pDBID := ADR(hDBID))
THEN
    IF fbPLCDBCreate.bError THEN
        tcMessage := fbPLCDBCreate.ipTcResult;
        nState := 255;
    ELSE
        nState := 0;
    END_IF
END_IF
```

6.1.表格

4.2.

3.2

通过 PLC 中一个包含 x 个元素（即 x 列）的数组来定义表结构，并创建一个新表。

语法

```
METHOD Table : BOOL
VAR_INPUT
    hDBID : UDINT;
    sTableName : T_MaxString;
    pTableCfg : POINTER TO ARRAY[0..MAX_DBCOLUMNS] OF ST_ColumnInfo;
    cbTableCfg : UDINT;
END_VAR
```

输入

Name	Type	描述
hDBID	UDINT	表示要使用的数据库的 ID。
sTableName	MaxString	要创建的表格的名称。
pTableCfg	指向 ARRAY[0..MAX_DBCOLUMNS] OF ST_ColumnInfo [▶ 234] 的指针	表示表结构数组的指针地址。将各个列写入该数组。
cbTableCfg	UDINT	表示配置列信息的数组的长度。

返回值

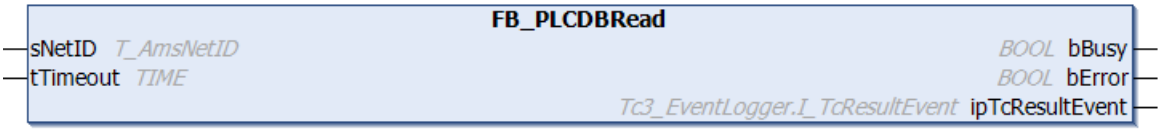
Name	Type	描述
表格	BOOL	显示方法的状态。在方法执行完毕时，即使出现错误，也会返回 TRUE。

示例

```
VAR
    fbPLCDBCreate : FB_PLCDBCreateEvt(sNetID := '', tTimeout := T#5S);
    ColumnInfo    : ARRAY [0..14] OF ST_ColumnInfo;
    tcMessage     : I_TcMessage;
END_VAR
```

```
ColumnInfo[0].sName := 'colBigInt';     ColumnInfo[0].eType := E_ColumnType.BigInt;     ColumnInfo[0
].nLength := 8;     ColumnInfo[0].sProperty := 'IDENTITY(1,1)';
ColumnInfo[1].sName := 'colInteger';    ColumnInfo[1].eType := E_ColumnType.Integer;    ColumnInfo[1
].nLength := 4;
ColumnInfo[2].sName := 'colSmallInt';   ColumnInfo[2].eType := E_ColumnType.SmallInt;   ColumnInfo[2
].nLength := 2;
ColumnInfo[3].sName := 'colTinyInt';    ColumnInfo[3].eType := E_ColumnType.TinyInt;    ColumnInfo[3
].nLength := 1;
ColumnInfo[4].sName := 'colBit';        ColumnInfo[4].eType := E_ColumnType.BIT_;       ColumnInfo[4
].nLength := 1;
ColumnInfo[5].sName := 'colMoney';      ColumnInfo[5].eType := E_ColumnType.Money;      ColumnInfo[5
].nLength := 8;
ColumnInfo[6].sName := 'colFloat';      ColumnInfo[6].eType := E_ColumnType.Float;      ColumnInfo[6
].nLength := 8;
ColumnInfo[7].sName := 'colReal';       ColumnInfo[7].eType := E_ColumnType.REAL_;      ColumnInfo[7
].nLength := 4;
ColumnInfo[8].sName := 'colDateTime';   ColumnInfo[8].eType := E_ColumnType.DateTime;   ColumnInfo[8
].nLength := 4;
ColumnInfo[9].sName := 'colNText';      ColumnInfo[9].eType := E_ColumnType.NText;      ColumnInfo[9
].nLength := 256;
ColumnInfo[10].sName := 'colNChar';     ColumnInfo[10].eType := E_ColumnType.NChar;     ColumnInfo[1
0].nLength := 10;
ColumnInfo[11].sName := 'colImage';     ColumnInfo[11].eType := E_ColumnType.Image;     ColumnInfo[1
1].nLength := 256;
ColumnInfo[12].sName := 'colNVarChar';  ColumnInfo[12].eType := E_ColumnType.NVarChar;  ColumnInfo[1
2].nLength := 50;
ColumnInfo[13].sName := 'colBinary';    ColumnInfo[13].eType := E_ColumnType.Binary;    ColumnInfo[1
3].nLength := 30;
ColumnInfo[14].sName := 'colVarBinary'; ColumnInfo[14].eType := E_ColumnType.VarBinary; ColumnInfo[1
4].nLength := 20;

IF fbPLCDBCreate.Table(
    hDBID:= 1,
    sTableName:= 'myNewTable',
    pTableCfg:= ADR(ColumnInfo),
    cbTableCfg:= SIZEOF(ColumnInfo))
THEN
    IF fbPLCDBCreate.bError THEN
        TcMessage:= fbPLCDBCreate.ipTcResult;
        nState := 255;
    ELSE
        nState := 0;
    END_IF
END_IF
```

6.1.4.2.4 FB_PLCDBRead



用于从数据库读取记录的功能块。

语法

```
FUNCTION_BLOCK FB_PLCDBRead
VAR_INPUT
    sNetID: T_AmsNetID := '';
    tTimeout: TIME := T#5S;
END_VAR
VAR_OUTPUT
    bBusy: BOOL;
    bError: BOOL;
    ipTcResultEvent: Tc3_EventLogger.I_TcResultEvent
END_VAR
```

输入

Name	Type	描述
sNetID	T_AmsNetID	ADS 命令指向的目标设备的 AMS 网络 ID。
tTimeout	TIME	表示函数被取消前的时间。

输出

Name	Type	描述
bBusy	BOOL	功能块的方法处于活动状态时即为 TRUE。
bError	BOOL	发生错误时即为 TRUE。
ipTcResultEvent	Tc3_EventLogger.I_TcResultEvent	结果界面，提供有关返回值的详细信息。

Methods

Name	定义位置	描述
读取 [▶ 253]	本地	从采用倍福指定的标准表结构的数据库表中读取指定数量的记录。
ReadStruct [▶ 254]	本地	从采用任何表结构的数据库表中读取指定数量的记录。

要求

开发环境	目标平台	要连接的 PLC 库
TwinCAT v3.1 Build 4020.10	PC 或 CX（x86）	Tc3_Database

6.1.读取
4.2.
4.1

该方法从采用倍福指定的标准表结构的数据库表中读取指定数量的记录。（例如，标准表结构可用于 AutoLog 模式和 FB_DBWrite 功能块）。

语法

```
METHOD Read : BOOL
VAR_INPUT
    hDBID: UDINT;
    sTableName: T_MaxString;
    sDBSymbolName: T_MaxString;
    eOrderBy: E_OrderColumn := E_OrderColumn.eColumnID;
    eOrderType: E_OrderType := E_OrderType.eOrder_ASC;
    nStartIndex: UDINT;
    nRecordCount: UDINT;
    pData: POINTER TO ST_StandardRecord;
    cbData: UDINT;
END_VAR
```

输入

Name	Type	描述
hDBID	UDINT	表示要使用的数据库的 ID。
sTableName	T_MaxString	要读取的表格的名称。
sDBSymbolName	T_MaxString	要从标准表结构中读取的符号名称。
eOrderBy	E_OrderColumn.eColumnID	排序列（ID、时间戳、名称或值）
eOrderType	E_OrderType.eOrder_ASC	排序方向（ASC 或 DESC）
nStartIndex	UDINT	表示要读取的第一个记录的索引。
nRecordCount	UDINT	表示要读取的记录数。
pData	POINTER TO ST_StandardRecord	要写入记录的结构数组的地址。
cbData	UDINT	表示结构数组的大小，以字节为单位。

返回值

Name	Type	描述
读取	BOOL	显示方法的状态。在方法执行完毕时，即使出现错误，也会返回 TRUE。

示例

```
VAR
    fbPLCDBRead    : FB_PLCDBReadEvt (sNetID := '', tTimeout := T#5S);
    ReadStruct     : ST_StandardRecord;
    tcMessage      : I_TcMessage;
END_VAR
```

```
IF fbPLCDBRead.Read(
    hDBID:= 1,
    sTableName:= 'MyTable_WithLReal',
    sDBSymbolName:= 'MyValue',
    eOrderBy:= E_OrderColumn.ID,
    eOrderType:= E_OrderType.DESC,
    nStartIndex:= 0,
    nRecordCount:= 1,
    pData:= ADR(ReadStruct),
    cbData:= SIZEOF(ReadStruct))
THEN
    IF fbPLCDBRead.bError THEN
        tcMessage := fbPLCDBRead.ipTcResult;
        nState := 255;
    ELSE
        nState := 0;
    END_IF
END_IF
```

PLC 中的结果：

Expression	Type	Value
ReadStruct	ST_StandardRecord	
nID	LINT	2
dtTimestamp	DATE_AND_TIME	DT#2018-2-1-16:8:8
sName	STRING(80)	'MyValue'
rValue	LREAL	15.9

6.1.ReadStruct
4.2.
4.2

此方法可从具有任意表结构的数据库表中读取指定数量的记录。

语法

```
METHOD ReadStruct : BOOL
VAR_INPUT
    hDBID: UDINT;
    sTableName: T_MaxString;
```

```
    pColumnNames: POINTER TO ARRAY [0..MAX_DBCOLUMNS] OF STRING(50);
    cbColumnNames: UDINT;
    sOrderByColumn: STRING(50);
    eOrderType: E_OrderType := E_OrderType.eOrder_ASC
    nStartIndex: UDINT;
    nRecordCount: UDINT;
    pData: POINTER TO BYTE;
    cbData: UDINT;
END_VAR
```

输入

Name	Type	描述
hDBID	UDINT	表示要使用的数据库的 ID。
sTableName	T_MaxString	要读取的表格的名称。
pColumnNames	POINTER TO ARRAY [0..MAX_DB COLUMNS] OF STRING(50)	包含待读取列名的数组的地址。
cbColumnNames	UDINT	列名数组的长度
sOrderByColumn	STRING(50)	指定排序列的名称。
eOrderType	E_OrderType	排序方向（ASC 或 DESC）
nStartIndex	UDINT	表示要读取的第一个记录的索引。
nRecordCount	UDINT	表示要读取的记录数。
pData	POINTER TO BYTE	要写入记录的结构数组的地址。
cbData	UDINT	表示结构数组的大小，以字节为单位。

返回值

Name	Type	描述
ReadStruct	BOOL	显示方法的状态。在方法执行完毕时，即使出现错误，也会返回 TRUE。

示例

```
VAR
    fbPLCDBRead    : FB_PLCDBReadEvt (sNetID := '', tTimeout := T#5S);
    myCustomStruct : ST_Record;
    tcMessage      : I_TcMessage;
END_VAR
```

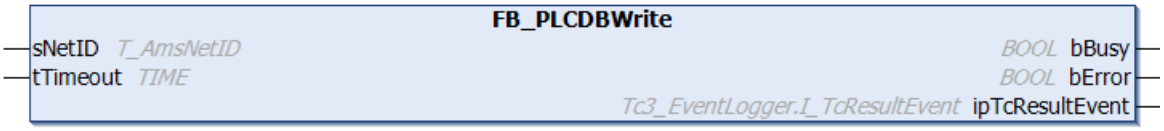
```
TYPE ST_Record :
STRUCT
    nID        : LINT;
    dtTimestamp: DATE_AND_TIME;
    sName      : STRING;
    nSensor1   : LREAL;
    nSensor2   : LREAL;
END_STRUCT
END_TYPE
```

```
// set columnnames
ColumnNames[0] := 'ID';
ColumnNames[1] := 'Timestamp';
ColumnNames[2] := 'Name';
ColumnNames[3] := 'Sensor1';
ColumnNames[4] := 'Sensor2';

IF fbPLCDBRead.ReadStruct(
    hDBID:= 1,
    sTableName:= 'MyTable_Struct',
    pColumnNames:= ADR(ColumnNames),
    cbColumnNames:= SIZEOF(ColumnNames),
    sOrderByColumn:= ColumnNames[0],
    eOrderType:= E_OrderType.DESC,
    nStartIndex:= 0,
    nRecordCount:= 1,
    pData:= ADR(myCustomStruct),
    cbData:= SIZEOF(myCustomStruct))
THEN
    IF fbPLCDBRead.bError THEN
        tcMessage:= fbPLCDBRead.ipTcResult;
        nState := 255;
    ELSE
        nState := 0;
    END_IF
END_IF
```

PLC 中的结果：

Expression	Type	Value
myCustomStruct	ST_Record	
nID	LINT	1
dtTimestamp	DATE_AND_TIME	DT#2018-2-1-15:17:54
sName	STRING	'MyStructVal'
nSensor1	LREAL	12.34
nSensor2	LREAL	102.5

6.1.4.2.5 FB_PLCDBWrite



用于将记录写入数据库的功能块。

语法

定义：

```
FUNCTION_BLOCK FB_PLCDBWrite
VAR_INPUT
    sNetID: T_AmsNetID := '';
    tTimeout: TIME := T#5S;
END_VAR
VAR_OUTPUT
    bBusy: BOOL;
    bError: BOOL;
    ipTcResultEvent: Tc3_EventLogger.I_TcResultEvent
END_VAR
```

输入

Name	Type	描述
sNetID	T_AmsNetID	ADS 命令指向的目标设备的 AMS 网络 ID。
tTimeout	TIME	表示函数被取消前的时间。

输出

Name	Type	描述
bBusy	BOOL	功能块的方法处于活动状态时即为 TRUE。
bError	BOOL	发生错误时即为 TRUE。
ipTcResultEvent	Tc3_EventLogger.I_TcResultEvent	结果界面，提供有关返回值的详细信息。

方法

Name	定义位置	描述
Write [▶ 257]	本地	在倍福指定的标准表结构中创建一条记录。
WriteBySymbol [▶ 258]	本地	读取指定的 ADS 符号的值，并将其保存在倍福指定的标准表结构中。
WriteStruct [▶ 259]	本地	创建具有任何表结构的记录

要求

开发环境	目标平台	要连接的 PLC 库
TwinCAT v3.1 Build 4020.10	PC 或 CX（x86）	Tc3_Database

6.1.Write
4.2.
5.1

该方法在倍福指定的标准表结构中创建记录。

语法

```
METHOD Write : BOOL
VAR_INPUT
    hDBID: UDINT;
    sTableName: T_MaxString;
    pValue: POINTER TO BYTE;
    cbValue: UDINT;
    sDBSymbolName: T_MaxString;
    eDBWriteMode: E_WriteMode := E_WriteMode.eADS_TO_DB_Append;
    nRingBuffParameter: UDINT;
END_VAR
```

输入

Name	Type	描述
hDBID	UDINT	表示要使用的数据库的 ID。
sTableName	T_MaxString	要读取的表格的名称。
pValue	POINTER TO BYTE	在标准表结构中要记录的变量的地址。
cbValue	UDINT	要记录的变量的长度。
sDBSymbolName	T_MaxString	在表格中记录的名称。
eDBWriteMode	E_WriteMode	表示写入模式。（添加、更新、环形缓冲区）
nRingBuffParameter	UDINT	“环形缓冲区”写入模式的附加参数。

返回值

Name	Type	描述
Write	BOOL	显示方法的状态。在方法执行完毕时，即使出现错误，也会返回 TRUE。

示例

本示例显示了如何使用 FB_PLCDBWriteEvt.Write 方法：

```
VAR
    fbPLCDBWrite    : FB_PLCDBWriteEvt(sNetID := '', tTimeout := T#5S);
    myValue         : LREAL := 43.23;
    tcMessage       : I_TcMessage;
END_VAR
```

```
IF fbPLCDBWrite.Write(
    hDBID:= 1,
    sTableName:= 'myTable_WithLReal',
    pValue:= ADR(myValue),
    cbValue:= SIZEOF(myValue),
    sDBSymbolName:= 'MyValue',
    eDBWriteMode:= E_WriteMode.eADS_TO_DB_RingBuff_Count,
    nRingBuffParameter:= 3)
THEN
    IF fbPLCDBWrite.bError THEN
        tcMessage := fbPLCDBWrite.ipTcResult;
        nState := 255;
    ELSE
        nState := 0;
    END_IF
END_IF
```

数据库中的结果：

ID	Timestamp	Name	值
27	已丢弃		
28	'2018-01-30 14:04:19'	'MyValue'	41.23
29	'2018-01-30 14:04:29'	'MyValue'	42.23
30	'2018-01-30 14:04:39'	'MyValue'	43.23

使用环形缓冲区选项时，在任何一个时间在数据库中只能存在 3 个此名称的条目。较旧的条目会被删除。

6.1.WriteBySymbol
4.2.
5.2

该方法读取指定的 ADS 符号的值，并将其保存在倍福指定的标准表结构中。此外，还可以读取其他 ADS 设备中的 ADS 符号。

语法

```
METHOD WriteBySymbol : BOOL
VAR_INPUT
    hDBID: UDINT;
    sTableName: T_MaxString;
    stADSDevice: ST_ADSDevice;
    stSymbol: ST_Symbol;
    eDBWriteMode: E_WriteMode := E_WriteMode.eADS_TO_DB_Append;
    nRingBuffParameter: UDINT;
END_VAR
```

输入

Name	Type	描述
hDBID	UDINT	表示要使用的数据库的 ID。
sTableName	T_MaxString	要读取的表格的名称。
stADSDevice	ST_ADSDevice	要在标准表结构中记录符号的 ADS 设备。
stSymbol	ST_Symbol	要写入的变量的符号名称
eDBWriteMode	E_WriteMode	表示写入模式。（添加、更新、环形缓冲区）
nRingBuffParameter	UDINT	“环形缓冲区”写入模式的额外参数

返回值

Name	Type	描述
WriteBySymbol	BOOL	显示方法的状态。在方法执行完毕时，即使出现错误，也会返回 TRUE。

示例

本示例展示了如何使用 FB_PLCDBWriteEvt.WriteBySymbol 方法：

```
VAR
    fbPLCDBWrite    :  FB_PLCDBWriteEvt(sNetID := '', tTimeout := T#5S);
    myValue         :  LREAL := 43.23;
    myAdsDevice     :  ST_ADSDevice;
    mySymbol        :  ST_Symbol;
    tcMessage       :  I_TcMessage;
END_VAR
```

```
// Set ADSDevice Information
myAdsDevice.sDevNetID     := '127.0.0.1.1.1';
myAdsDevice.nDevPort      := 851;
myAdsDevice.eADSRdWrtMode := E_ADSRdWrtMode.bySymbolName;
myAdsDevice.tTimeout      := T#5S;

// Set Symbol Information
mySymbol.eDataType        := E_PLCDataType.eType_LREAL;
mySymbol.sDBSymbolName    := 'MySymbol';
mySymbol.sSymbolName      := 'MAIN.myValue';
mySymbol.nBitSize         := 8;

// Call Functionblock
IF fbPLCDBWrite.WriteBySymbol(
    hDBID:= 1,
```

```
    sTableName:= 'myTable_WithLReal',
    stADSDevice:= myAdsDevice,
    stSymbol:= mySymbol,
    eDBWriteMode:= E_WriteMode.eADS_TO_DB_Append,
    nRingBuffParameter:= 1)
THEN
    IF fbPLCDBWrite.bError THEN
        tcMessage := fbPLCDBWrite.ipTcResult;
        nState := 255;
    ELSE
        nState := 0;
    END_IF
END_IF
```

数据库中的结果：

ID	Timestamp	Name	值
28	'2018-01-30 14:04:19'	'MyValue'	41.23
29	'2018-01-30 14:04:29'	'MyValue'	42.23
30	'2018-01-30 14:04:39'	'MyValue'	43.23
31	'2018-01-30 14:06:12’	‘MySymbol’	86.2

6.1.WriteStruct
4.2.
5.3

该方法创建具有可自由选择的表结构的记录。

语法

```
METHOD WriteStruct : BOOL
VAR_INPUT
    hDBID: UDINT;
    sTableName: T_MaxString;
    pRecord: POINTER TO BYTE;
    cbRecord: UDINT;
    pColumnNames: POINTER TO ARRAY [0..MAX_DBCOLUMNS] OF STRING(50);
    cbColumnNames: UDINT;
END_VAR
```

输入

Name	Type	描述
hDBID	UDINT	表示要使用的数据库的 ID。
sTableName	T_MaxString	要读取的表格的名称。
pRecord	POINTER TO BYTE	在可自由选择的表结构中要记录的结构体的地址。
cbRecord	UDINT	要写入的结构的长度
pColumnNames	POINTER TO ARRAY [0 ..MAX_DBCOLUMNS] OF STRING(50)	包含要填写的列名的数组的地址。
cbColumnNames	UDINT	列名数组的长度

返回值

Name	Type	描述
WriteStruct	BOOL	显示方法的状态。在方法执行完毕时，即使出现错误，也会返回 TRUE。

示例

本示例显示了如何使用方法 FB_PLCDBWriteEvt.WriteStruct：

```
VAR
    fbPLCDBWrite    :  FB_PLCDBWriteEvt(sNetID := '', tTimeout := T#5S);
    myRecord        :  ST_Record;
    ColumnNames     :  ARRAY[0..4] OF STRING(50);

    systime         :  GETSYSTEMTIME;
```

```
    currentTime     :  T_FILETIME;
    tcMessage       :  I_TcMessage;
END_VAR
```

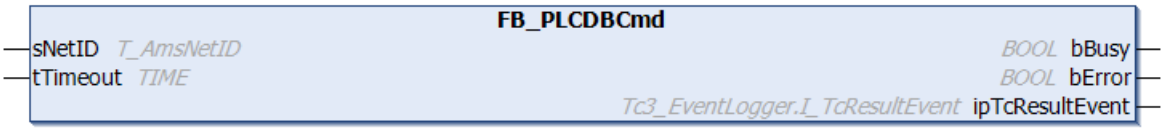
```
TYPE ST_Record :
STRUCT
    nID         : LINT;
    dtTimestamp: DATE_AND_TIME;
    sName       : STRING;
    nSensor1    : LREAL;
    nSensor2    : LREAL;
END_STRUCT
END_TYPE
```

```
// set Values
systime(timeLoDw => currentTime.dwLowDateTime, timeHiDW => currentTime.dwHighDateTime );
myRecord.dtTimestamp := FILETIME_TO_DT(currentTime);
myRecord.sName       := 'MyStructVal';
myRecord.nSensor1    := 12.34;
myRecord.nSensor2    := 102.5;

// set columnnames
ColumnNames[0] := 'ID';
ColumnNames[1] := 'Timestamp';
ColumnNames[2] := 'Name';
ColumnNames[3] := 'Sensor1';
ColumnNames[4] := 'Sensor2';

// Call Functionblock
IF fbPLCDBWrite.WriteStruct(
    hDBID:= 1,
    sTableName:= 'myTable_Struct',
    pRecord:= ADR(myRecord),
    cbRecord:= SIZEOF(myRecord),
    pColumnNames:= ADR(ColumnNames) ,
    cbColumnNames:= SIZEOF(ColumnNames))
THEN
    IF fbPLCDBWrite.bError THEN
        tcMessage := fbPLCDBWrite.ipTcResult;
        nState := 255;
    ELSE
        nState := 0;
    END_IF
END_IF
```

数据库中的结果：

ID	Timestamp	Name	Sensor1	Sensor2
5	'2018-01-30 15:23:26'	‘MyStructVal’	12.34	102.5

6.1.4.2.6 FB_PLCDBCmd



具有 2 种方法的功能块。用户可以定义并传输自己的 SQL 命令。SQL 命令中的占位符可以与 PLC 中的结构相关联，从而反映表结构。数据库服务器最终将结构体的当前数据输入 SQL 命令。

语法

定义：

```
FUNCTION_BLOCK FB_PLCDBCmd
VAR_INPUT
    sNetID: T_AmsNetID := '';
    tTimeout: TIME := T#5S;
END_VAR
VAR_OUTPUT
    bBusy: BOOL;
    bError: BOOL;
    ipTcResultEvent: Tc3_EventLogger.I_TcResultEvent
END_VAR
```

输入

Name	Type	描述
sNetID	T_AmsNetID	ADS 命令指向的目标设备的 AMS 网络 ID。
tTimeout	TIME	表示函数被取消前的时间。

输出

Name	Type	描述
bBusy	BOOL	功能块的方法处于活动状态时即为 TRUE。
bError	BOOL	发生错误时即为 TRUE。
ipTcResultEvent	Tc3_EventLogger.I_TcResultEvent	结果界面，提供有关返回值的详细信息。

Methods

Name	定义位置	描述
执行 [▶ 261]	本地	向数据库发送任何 SQL 命令。 无法读取返回的记录。
ExecuteDataReturn [▶ 262]	本地	向数据库发送任何 SQL 命令。 可以读取指定数量的记录。

要求

开发环境	目标平台	要连接的 PLC 库
TwinCAT v3.1 Build 4020.10	PC 或 CX（x86）	Tc3_Database

6.1.执行

4.2.

6.1

该方法可以用于向数据库发送 SQL 命令。每次调用时都会打开数据库连接，然后再次关闭。在命令中可以定义占位符，然后，在执行之前，TwinCAT 数据库服务器会使用相应的值替换这些占位符。无法读取返回的记录。

语法

```
METHOD Execute : BOOL
VAR_INPUT
    hDBID: UDINT;
    pExpression: POINTER TO BYTE;
    cbExpression: UDINT;
    pData: POINTER TO BYTE;
    cbData: UDINT;
    pParameter: POINTER TO ARRAY[0..MAX_DBCOLUMNS] OF ST_ExpParameter;
    cbParameter: UDINT;
END_VAR
```

示例

```
VAR
    fbPLCDBCmd    : FB_PLCDBCmdEvt(sNetID := '', tTimeout := T#5S);
    sCmd          : STRING (1000);
    myStruct      : ST_DataAll;
    aPara         : ARRAY[0..14] OF ST_ExpParameter;
    tcMessage     : I_TcMessage;
END_VAR
```

```
TYPE ST_DataAll :
STRUCT
    colBigInt: LINT;
    colInteger: DINT;
    colSmallInt: INT;
    colTinyInt: BYTE;
    colBit: BOOL;
    colMoney: LREAL;
    colFloat: LREAL;
    colReal: REAL;
```

```
    colDateTime: DT;
    colNText: STRING(255);
    colNChar: STRING(10);
    colImage: ARRAY[0..255] OF BYTE;
    colNVarChar: STRING(50);
    colBinary: ARRAY[0..29] OF BYTE;
    colVarBinary: ARRAY[0..19] OF BYTE;
END_STRUCT
END_TYPE
```

```
// set Parameter configuration
aPara[0].sParaName := 'colBigInt';    aPara[0].eParaType :=
E_ExpParameterType.Int64;      aPara[0].nParaSize := 8;
aPara[1].sParaName := 'colInteger';   aPara[1].eParaType :=
E_ExpParameterType.Int32;      aPara[1].nParaSize := 4;
aPara[2].sParaName := 'colSmallInt';  aPara[2].eParaType :=
E_ExpParameterType.Int16;      aPara[2].nParaSize := 2;
aPara[3].sParaName := 'colTinyInt';   aPara[3].eParaType :=
E_ExpParameterType.Byte_;      aPara[3].nParaSize := 1;
aPara[4].sParaName := 'colBit';       aPara[4].eParaType :=
E_ExpParameterType.Boolean;    aPara[4].nParaSize := 1;
aPara[5].sParaName := 'colMoney';     aPara[5].eParaType :=
E_ExpParameterType.Double64;   aPara[5].nParaSize := 8;
aPara[6].sParaName := 'colFloat';     aPara[6].eParaType :=
E_ExpParameterType.Double64;   aPara[6].nParaSize := 8;
aPara[7].sParaName := 'colReal';      aPara[7].eParaType :=
E_ExpParameterType.Float32;    aPara[7].nParaSize := 4;
aPara[8].sParaName := 'colDateTime';  aPara[8].eParaType :=
E_ExpParameterType.DateTime;   aPara[8].nParaSize := 4;
aPara[9].sParaName := 'colNText';     aPara[9].eParaType :=
E_ExpParameterType.STRING_;    aPara[9].nParaSize := 256;
aPara[10].sParaName:= 'colNChar';     aPara[10].eParaType :=
E_ExpParameterType.STRING_;   aPara[10].nParaSize := 10;
aPara[11].sParaName:= 'colImage';     aPara[11].eParaType :=
E_ExpParameterType.ByteArray; aPara[11].nParaSize := 256;
aPara[12].sParaName:= 'colNVarChar';  aPara[12].eParaType :=
E_ExpParameterType.STRING_;   aPara[12].nParaSize := 50;
aPara[13].sParaName:= 'colBinary';    aPara[13].eParaType :=
E_ExpParameterType.ByteArray; aPara[13].nParaSize := 30;
aPara[14].sParaName:= 'colVarBinary'; aPara[14].eParaType :=
E_ExpParameterType.ByteArray; aPara[14].nParaSize := 20;

// set command
sCmd := 'INSERT INTO MyTableName (colInteger, colSmallInt, colTinyInt, colBit, colMoney, colFloat,
colReal, colDateTime, colNText, colNChar, colImage, colNVarChar, colBinary, colVarBinary) VALUES
({colInteger}, {colSmallInt}, {colTinyInt}, {colBit}, {colMoney}, {colFloat}, {colReal},
{colDateTime}, {colNText}, {colNChar}, {colImage}, {colNVarChar}, {colBinary}, {colVarBinary})';

// call functionblock
IF fbPLCDBCmd.Execute(
    hDBID:= 1,
    pExpression:= ADR(sCmd),
    cbExpression:= SIZEOF(sCmd),
    pData:= ADR(myStruct),
    cbData:= SIZEOF(myStruct),
    pParameter:= ADR(aPara),
    cbParameter:= SIZEOF(aPara))
THEN
    IF fbPLCDBCmd.bError THEN
        tcMessage := fbPLCDBCmd.ipTcResult;
        nState := 255;
    ELSE
        nState := 0;
    END_IF
END_IF
```

6.1.ExecuteDataReturn

4.2.

6.2

该方法可以用于向数据库发送 SQL 命令。每次调用时都会打开数据库连接，然后再次关闭。在命令中可以定义占位符，然后，在执行之前，TwinCAT 数据库服务器会使用相应的值替换这些占位符。可以读取指定数量的记录。

语法

```
METHOD ExecuteDataReturn : BOOL
VAR_INPUT
    hDBID: UDINT;
    pExpression: POINTER TO BYTE;
    cbExpression: UDINT;
    pData: POINTER TO BYTE;
    cbData: UDINT;
    pParameter: POINTER TO ARRAY[0..MAX_DBCOLUMNS] OF ST_ExpParameter;
    cbParameter: UDINT;
    nStartIndex: UDINT;
    nRecordCount: UDINT;
    pReturnData: POINTER TO BYTE;
```

```
    cbReturnData: UDINT;
    pRecords: POINTER TO UDINT;
END_VAR
```

输入

Name	Type	描述
hDBID	UDINT	表示要使用的数据库的 ID。
pExpression	POINTER TO BYTE	使用 SQL 命令的字符串变量的地址。
cbExpression	UDINT	使用 SQL 命令的字符串变量的长度。
pData	POINTER TO BYTE	带有参数值的结构的地址
cbData	UDINT	带有参数值的结构的长度
pParameter	POINTER TO ARRAY[0..MAX_DBCOLUMNS] OF ST_ExpParameter	带有参数信息的结构体数组的地址。
cbParameter	UDINT	带有参数信息的结构体数组的长度。
nStartIndex	UDINT	表示要读取的第一个记录的索引。
nRecordCount	UDINT	表示要读取的记录数。
pReturnData	POINTER TO BYTE	要写入记录的结构数组的地址。
cbReturnData	UDINT	表示结构数组的大小，以字节为单位。
pRecords	POINTER TO BYTE	读取的记录数。

返回值

Name	Type	描述
ExecuteDataReturn	BOOL	显示方法的状态。在方法执行完毕时，即使出现错误，也会返回 TRUE。

对命令进行参数设置

在 SQL 命令中的大括号内指定各个参数的列名。
示例：‚SELECT * FROM MyHouse_Temperatures WHERE Room = {SelectedRoom}‘ 。
因此，在结构 ST_ExpParameter 中必须指定 SelectedRoom 作为参数名称。

一些数据库不支持 SQL 语句的参数设置。（TOP/LIMIT/ROWNUM/...）通常不支持可参数化的表格名称。

示例

```
VAR
    fbPLCDBCmd       : FB_PLCDBCmdEvt(sNetID := '', tTimeout := T#5S);
    sCmd             : STRING (1000);
    stPara           : ST_ExpParameter;
    RecordAmt        : ULINT := 3;
    ReturnDataStruct : ARRAY [0..9] OF ST_DataAll;
    nRecords         : UDINT;
    tcMessage        : I_TcMessage;
END_VAR
```

```
// set Parameter configuration
stPara.eParaType := E_ExpParameterType.Int64;
stPara.nParaSize := 8;
stPara.sParaName := 'RecordAmt';

// set command with placeholder
sCmd := 'SELECT TOP ({RecordAmt}) * FROM MyTableName';

// call functionblock
IF fbPLCDBCmd.ExecuteDataReturn(
    hDBID:= 1,
    pExpression:= ADR(sCmd),
    cbExpression:= SIZEOF(sCmd),
    pData:= ADR(RecordAmt),
    cbData:= SIZEOF(RecordAmt),
    pParameter:= ADR(stPara),
    cbParameter:= SIZEOF(stPara),
    nStartIndex:= 0,
    nRecordCount:= 10,
    pReturnData:= ADR(ReturnDataStruct),
```

```
    cbReturnData:= SIZEOF(ReturnDataStruct),
    pRecords:= ADR(nRecords))
THEN
    IF fbPLCDBCmd.bError THEN
        tcMessage := fbPLCDBCmd.ipTcResult;
        nState := 255;
    ELSE
        nState := 0;
    END_IF
END_IF
```

6.1.4.3 SQL 专家模式

6.1.4.3.1 FB_ConfigTcDBSrv

用于创建、读取和删除 TwinCAT 数据库服务器的配置条目的功能块。

语法

定义：

```
FUNCTION_BLOCK FB_ConfigTcDBSrv
VAR_INPUT
    sNetID: T_AmsNetID := '';
    tTimeout: TIME := T#5S;
END_VAR
VAR_OUTPUT
    bBusy: BOOL;
    bError: BOOL;
    ipTcResultEvent: Tc3_EventLogger.I_TcResultEvent;
END_VAR
```

输入

Name	Type	描述
sNetID	T_AmsNetID	ADS 命令指向的目标设备的 AMS 网络 ID。
tTimeout	TIME	表示函数被取消前的时间。

输出

Name	Type	描述
bBusy	BOOL	功能块的方法处于活动状态时即为 TRUE。
bError	BOOL	发生错误时即为 TRUE。
ipTcResultEvent	Tc3_EventLogger.I_TcResultEvent	结果界面，提供有关返回值的详细信息。

方法

Name	定义位置	描述
创建 [▶ 153]	本地	在 XML 配置文件中为 TwinCAT 数据库服务器创建新条目
读取 [▶ 154]	本地	读取 TwinCAT 数据库服务器的当前配置
删除 [▶ 155]	本地	从 TwinCAT 数据库服务器的配置中删除数据库和 AutoLog 组

要求

开发环境	目标平台	要连接的 PLC 库
TwinCAT v3.1 Build 4020.10	PC 或 CX（x86）	Tc3_Database

6.1.创建
4.3.
1.1

该方法可在 XML 配置文件中为 TwinCAT 数据库服务器创建新条目。TwinCAT 数据库服务器可以选择临时使用新条目。在这种情况下，不会将任何数据写入 XML 文件。

语法

```
METHOD Create : BOOL
VAR_INPUT
    pTcDBSrvConfig: POINTER TO BYTE;
    cbTcDBSrvConfig: UDINT;
    bTemporary: BOOL := TRUE;
    pConfigID: POINTER TO UDINT;
END_VAR
```

输入

Name	Type	描述
pTcDBSrvConfig	POINTER TO BYTE	要创建的配置结构的指针。
cbTcDBSrvConfig	UDINT	配置结构的长度
bTemporary	BOOL	指定是否要将配置存储在 XML 文件中。
pConfigID	POINTER TO UDINT	返回配置 ID（hDBID 或 hAutoLogGrpID）的指针

目前不支持创建 AutoLog 组。

返回值

Name	Type	描述
创建	BOOL	显示方法的状态。在方法执行完毕时，即使出现错误，也会返回 TRUE。

示例

```
VAR
    fbConfigTcDBSrv : FB_ConfigTcDBSrv(sNetId := '', tTimeout:=T#5S);
    myConfigHandle  : INT;
    // Any other ConfigType can be used here
    stConfigDB      : T_DBConfig_MsCompactSQL;
END_VAR
```

```
stConfigDB.bAuthentification := FALSE;
stConfigDB.sServer := 'C:\Recipes.sdf';

IF fbConfigTcDBSrv.Create(
    pTcDBSrvConfig:= ADR(stConfigDB),
    cbTcDBSrvConfig:= SIZEOF(stConfigDB),
    bTemporary:= TRUE,
    pConfigID:= ADR(myConfigHandle))
THEN
    IF fbSQLStoredProcedure.bError THEN
        nState := 255;
    ELSE
        nState := 0;
    END_IF
END_IF
```

6.1.读取

4.3.

1.2

该方法可以用于读取 TwinCAT 数据库服务器的当前配置。任何可能被包含的临时配置都会对其进行相应的标记。

语法

```
METHOD Read : BOOL
VAR_INPUT
    pDBConfig: POINTER TO ARRAY [1..MAX_CONFIGURATIONS] OF ST_ConfigDB;
    cbDBConfig: UDINT;
    pAutoLogGrpConfig: POINTER TO ARRAY[1..MAX_CONFIGURATIONS] OF
ST_ConfigAutoLogGrp;
    cbAutoLogGrpConfig: UDINT;
    pDBCount: POINTER TO UDINT;
    pAutoLogGrpCount: POINTER TO UDINT;
END_VAR
```

输入

Name	Type	描述
pDBConfig	POINTER TO ARRAY [1..MAX_CONFIGURATIONS] OF ST_ConfigDB [▶ 219]	用于写入数据库配置的数组的指针地址。
cbDBConfig	UDINT	数据库配置数组的长度
pAutoLogGrpConfig	POINTER TO ARRAY[1..MAX_CONFIGURATIONS] OF ST_ConfigAutoLogGrp [▶ 218]	用于写入 AutoLogGrp 配置的数组的指针地址。
cbAutoLogGrpConfig	UDINT	AutoLogGrp 配置数组的长度
pDBCount	POINTER TO UDINT	用于存储数据库配置数量的指针地址。
pAutoLogGrpCount	POINTER TO UDINT	用于存储 AutoLogGrp 配置数量的指针地址。

返回值

Name	Type	描述
读取	BOOL	显示方法的状态。在方法执行完毕时，即使出现错误，也会返回 TRUE。

示例

```
VAR
    fbConfigTcDBSrv    : FB_ConfigTcDBSrv(sNetId := '', tTimeout:=T#5S);
    aDBConfig          : ARRAY[0..MAX_CONFIGURATIONS] OF ST_ConfigDB;
    aAutoGrpConfig     : ARRAY[0..MAX_CONFIGURATIONS] OF ST_ConfigAutoLogGrp;
    nDbCount           : UDINT;
    nAutoGrpCount      : UDINT;
END_VAR
```

```
IF fbConfigTcDBSrv.Read(
    pDBConfig := ADR(aDBConfig),
    cbDBConfig := SIZEOF(aDBConfig),
    pAutologGrpConfig := ADR(aAutoGrpConfig),
    cbAutoLogGrpConfig := SIZEOF(aAutoGrpConfig),
    pDBCount := ADR(nDbCount),
    pAutoLogGrpCount := ADR(nAutoGrpCount))
 THEN
    IF fbConfigTcDBSrv.bError THEN
        nState := 255;
    ELSE
        nState := 0;
    END_IF
END_IF
```

6.1.删除

4.3.

1.3

该方法可以用于从 TwinCAT 数据库服务器的配置中删除数据库和 AutoLog 组。

语法

```
METHOD Delete : BOOL
VAR_INPUT
    eTcDBSrvConfigType: E_TcDBSrvConfigType;
    hConfigID: UDINT;
END_VAR
```

输入

Name	Type	描述
eTcDBSrvConfigType	E_TcDBSrvConfigType	要删除的配置的类型（数据库/AutoLog 组）
hConfigID	UDINT	要删除的配置的 ID（hDBID 或 hAutoLogGrpID）

返回值

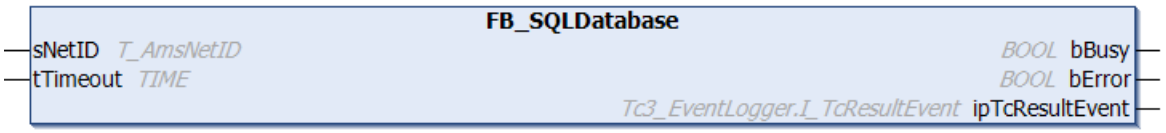
Name	Type	描述
删除	BOOL	显示方法的状态。在方法执行完毕时，即使出现错误，也会返回 TRUE。

示例

```
VAR
    fbConfigTcDBSrv : FB_ConfigTcDBSrv(sNetId := '', tTimeout:=T#5S);
    myConfigHandle  : INT;
END_VAR
```

```
IF fbConfigTcDBSrv.Delete(
    eTcDBSrvConfigType := E_TcDBSrvConfigType.Database,
    hConfigID := myConfigHandle) THEN
IF fbConfigTcDBSrv.bError THEN
        nState := 255;
    ELSE
        nState := 0;
    END_IF
END_IF
```

6.1.4.3.2 FB_SQLDatabase



用于打开、关闭和管理数据库连接的功能块。

语法

定义：

```
FUNCTION_BLOCK FB_SQLDatabase
VAR_INPUT
    sNetID: T_AmsNetID := '';
    tTimeout: TIME := T#5S;
END_VAR
VAR_OUTPUT
    bBusy: BOOL;
    bError: BOOL;
    ipTcResultEvent: Tc3_EventLogger.I_TcResultEvent
END_VAR
```

输入

Name	Type	描述
sNetID	T_AmsNetID	ADS 命令指向的目标设备的 AMS 网络 ID。
tTimeout	TIME	表示函数被取消前的时间。

输出

Name	Type	描述
bBusy	BOOL	功能块的方法处于活动状态时即为 TRUE。
bError	BOOL	发生错误时即为 TRUE。
ipTcResultEvent	Tc3_EventLogger.I_TcResultEvent	结果界面，提供有关返回值的详细信息。

方法

Name	定义位置	描述
连接 [▶ 268]	本地	打开与已声明数据库的连接。
CreateCmd [▶ 269]	本地	使用功能块 FB_SQLDatabase 的已打开的数据库连接，对功能块 FB_SQLCommand 的实例进行初始化。
CreateSP [▶ 270]	本地	使用功能块 FB_SQLDatabase 的已打开的数据库连接，对功能块 FB_SQLStoredProcedure 的实例进行初始化。
Disconnect [▶ 270]	本地	关闭该功能块实例打开的数据库连接。

要求

开发环境	目标平台	要连接的 PLC 库
TwinCAT v3.1 Build 4020.10	PC 或 CX（x86）	Tc3_Database

6.1.连接
4.3.
2.1

该方法打开与已声明数据库的连接。

语法

```
METHOD Connect : BOOL
VAR_INPUT
    hDBID: UDINT := 1;
END_VAR
```

输入

Name	Type	描述
hDBID	UDINT	表示要使用的数据库的 ID。

返回值

Name	Type	描述
连接	BOOL	显示方法的状态。在方法执行完毕时，即使出现错误，也会返回 TRUE。

示例

```
VAR
    fbSqlDatabase : FB_SQLDatabaseEvt(sNetID := '', tTimeout := T#5S);
END_VAR
```

```
// open connection
IF fbSqlDatabase.Connect(1) THEN
    IF fbSqlDatabase.bError THEN
        nState := 255;
    ELSE
        nState := nState+1;
    END_IF
END_IF
```

6.1.CreateCmd

4.3.

2.2

该方法可用于使用功能块 FB_SQLDatabase 的已打开的数据库连接，对功能块 FB_SQLCommand 的实例进行初始化。功能块 FB_SQLCommand 仅使用通过 CreateCmd 方法分配的数据库连接。使用相同的数据库连接可以对功能块 FB_SQLCommand 的多个实例进行初始化。

在同一个周期内完成功能块 FB_SQLCommand 的初始化。这意味着既不需要检查功能块的繁忙标志，也不需要检查 CreateCmd 方法的方法返回值。

语法

```
METHOD CreateCmd : BOOL
VAR_INPUT
    pSQLCommand: POINTER TO FB_SQLCommand;
END_VAR
```

输入

Name	Type	描述
pSQLCommand	POINTER TO FB_SQLCommand	返回功能块 FB_SQLCommand 的新实例。

返回值

Name	Type	描述
CreateCmd	BOOL	显示方法的状态。在方法执行完毕时，即使出现错误，也会返回 TRUE。

示例

```
VAR
    fbSqlDatabase : FB_SQLDatabaseEvt(sNetID := '', tTimeout := T#5S);
END_VAR
```

```
// create a command reference
IF fbSqlDatabase.CreateCmd(ADR(fbSqlCommand)) THEN
    IF fbSqlDatabase.bError THEN
```

```
        nState := 255;
    ELSE
        nState := nState+1;
    END_IF
END_IF
```

然后可以使用 FB_SQLCommandEvt [▶ 189] 来执行。

6.1.CreateSP

4.3.

2.3

该方法可用于使用功能块 FB_SQLDatabase 的已打开的数据库连接，对功能块 FB_SQLStoredProcedure 的实例进行初始化。功能块 FB_SQLStoredProcedure 仅使用通过 CreateCmd 方法分配的数据库连接。使用相同的数据库连接可以对功能块 FB_SQLStoredProcedure 的多个实例进行初始化。

功能块 FB_SQLStoredProcedure 的初始化可能需要几个周期。在可以使用功能块之前，必须检查功能块的繁忙标志或 CreateCmd 方法的方法返回值。

语法

```
METHOD CreateSP : BOOL
VAR_INPUT
    sProcedureName: T_MaxString;
    pParameterInfo: POINTER TO ARRAY [0..MAX_SPPARAMETER] OF ST_SQLSPParameter;
    cbParameterInfo: UDINT;
    pSQLProcedure: POINTER TO FB_SQLStoredProcedure;
END_VAR
```

输入

Name	Type	描述
sProcedureName	T_MaxString	表示要执行的过程的名称。
pParameterInfo	POINTER TO ARRAY [0..MAX_SPPARAMETER] OF ST_SQLSPParameter	参数信息列表的指针地址。
cbParameterInfo	UDINT	表示参数信息列表的长度。
pSQLProcedure	POINTER TO FB_SQLStoredProcedure	返回功能块 FB_SQLStoredProcedure 的新实例。

返回值

Name	Type	描述
CreateSP	BOOL	显示方法的状态。在方法执行完毕时，即使出现错误，也会返回 TRUE。

示例

```
VAR
    fbSqlDatabase   : FB_SQLDatabaseEvt(sNetID := '', tTimeout := T#5S);
    ParaInfo        : ST_SQLSPParameter;
END_VAR
```

```
ParaInfo.sParameterName     := '@Customer_ID';
ParaInfo.eParameterType     := E_SPParameterType.Input;
ParaInfo.eParameterDataType := E_ColumnType.BigInt;
ParaInfo.nParameterSize     := 8;

IF fbSQLDatabase.CreateSP('dbo.SP_GetCustomerPositions', ADR(ParaInfo), SIZEOF(ParaInfo), ADR(fbSQLS
toredProcedure)) THEN
    IF fbSQLDatabase.bError THEN
        nState:=255;
    ELSE
        nState:= nState+1;
    END_IF
END_IF
```

随后，FB_SQLStoredProcedureEvt [▶ 194] 可以用于执行存储过程。

6.1.Disconnect

4.3.

2.4

该方法关闭该功能块实例打开的数据库连接。

语法

```
METHOD Disconnect : BOOL
```

返回值

Name	Type	描述
Disconnect	BOOL	显示方法的状态。在方法执行完毕时，即使出现错误，也会返回 TRUE。

示例

```
VAR
    fbSqlDatabase : FB_SQLDatabaseEvt(sNetID := '', tTimeout := T#5S);
END_VAR
```

```
// disconnect from database
IF fbSqlDatabase.Disconnect() THEN
    IF fbSqlDatabase.bError THEN
        nState := 255;
    ELSE
        nState := nState+1;
    END_IF
END_IF
```

6.1.4.3.3 FB_SQLCommand

用于执行 SQL 命令的功能块。在使用它之前，必须使用功能块 FB_SQLDatabase 对其进行初始化。

语法

定义：

```
FUNCTION_BLOCK FB_SQLCommand
VAR_INPUT
    sNetID: T_AmsNetID := '';
    tTimeout: TIME := T#5S;
END_VAR
VAR_OUTPUT
    bBusy: BOOL;
    bError: BOOL;
    ipTcResultEvent: Tc3_EventLogger.I_TcResultEvent
END_VAR
```

输入

Name	Type	描述
sNetID	T_AmsNetID	ADS 命令指向的目标设备的 AMS 网络 ID。
tTimeout	TIME	表示函数被取消前的时间。

输出

Name	Type	描述
bBusy	BOOL	功能块的方法处于活动状态时即为 TRUE。
bError	BOOL	发生错误时即为 TRUE。
ipTcResultEvent	Tc3_EventLogger.I_TcResultEvent	结果界面，提供有关返回值的详细信息。

方法

Name	定义位置	描述
执行 [▶ 272]	本地	通过功能块 FB_SQLDatabase 已打开的数据库连接，向数据库发送指定的 SQL 命令。
ExecuteDataReturn [▶ 273]	本地	通过功能块 FB_SQLDatabase 已打开的数据库连接，向数据库发送指定的 SQL 命令。 可以传输功能块 FB_SQLResult 的实例，以读取返回的记录。

要求

开发环境	目标平台	要连接的 PLC 库
TwinCAT v3.1 Build 4020.10	PC 或 CX（x86）	Tc3_Database

6.1.执行
4.3.
3.1

该方法通过功能块 FB_SQLDatabase 已打开的数据库连接，向数据库发送指定的 SQL 命令。

语法

```
METHOD Execute : BOOL
VAR_INPUT
    pSQLCmd: POINTER TO BYTE;
    cbSQLCmd: UDINT;
END_VAR
```

输入

Name	Type	描述
pSQLCmd	POINTER TO BYTE	表示使用要执行的 SQL 命令的字符串变量的指针地址。
cbSQLCmd	UDINT	表示要执行的 SQL 命令的长度。

返回值

Name	Type	描述
执行	POINTER TO BYTE	显示方法的状态。在方法执行完毕时，即使出现错误，也会返回 TRUE。

示例

使用 FB_SQLDatabaseEvt.CreateCmd() [▶ 185] 创建的命令。

```
VAR
    fbSqlCommand : FB_SQLCommandEvt(sNetID := '', tTimeout := T#5S);
    tcMessage    : I_TcMessage;
END_VAR
```

```
// you can generate this with the SQL Query Editor
sCmd := 'INSERT INTO myTable_Double ( Timestamp, Name, Value) VALUES ( $'2018-01-31 14:59:27$', $'Te
mperature$', 21.3)';

// call sql command
IF fbSQLCommand.Execute(ADR(sCmd), SIZEOF(sCmd)) THEN
    IF fbSQLCommand.bError THEN
        tcMessage := fbSQLCommand.ipTcResult;
        nState := 255;
    ELSE
        nState := nState+1;
    END_IF
END_IF
```

6.1.ExecuteDataReturn
4.3.
3.2

该方法通过功能块 FB_SQLDatabase 已打开的数据库连接，向数据库发送指定的 SQL 命令。可以传输功能块 FB_SQLResult 的实例，以读取返回的记录。

语法

```
METHOD ExecuteDataReturn : BOOL
VAR_INPUT
    pSQLCmd: POINTER TO BYTE;
    cbSQLCmd: UDINT;
    pSQLDBResult: POINTER TO FB_SQLResult;
END_VAR
```

输入

Name	Type	描述
pSQLCmd	POINTER TO BYTE	表示使用要执行的 SQL 命令的字符串变量的指针地址。
cbSQLCmd	UDINT	表示要执行的 SQL 命令的长度。
pSQLDBResult	POINTER TO FB_SQLResult [▶ 273]	返回功能块 FB_SQLResult 的新实例。

返回值

Name	Type	描述
ExecuteDataReturn	POINTER TO BYTE	显示方法的状态。在方法执行完毕时，即使出现错误，也会返回 TRUE。

示例

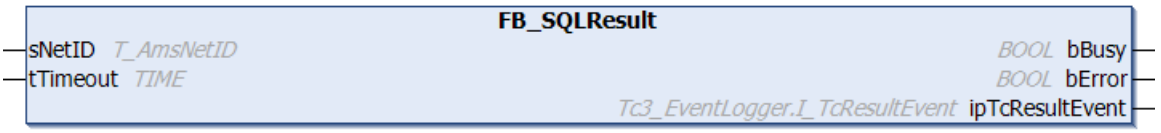
使用 FB_SQLDatabaseEvt.CreateCmd() [▶ 185] 创建的命令。

```
VAR
    fbSqlCommand : FB_SQLCommandEvt(sNetID := '', tTimeout := T#5S);
    tcMessage    : I_TcMessage;
END_VAR
```

```
// you can generate this with the SQL Query Editor
sCmd := 'SELECT ID, Timestamp, Name, Value FROM myTable_Double';

// call sql command
IF fbSQLCommand.ExecuteDataReturn(ADR(sCmd), SIZEOF(sCmd), ADR(fbSqlResult)) THEN
    IF fbSQLCommand.bError THEN
        nState := 255;
    ELSE
         tcMessage := fbSQLCommand.ipTcResult;
        nState := nState+1;
    END_IF
END_IF
```

然后可以使用 FB_SQLResultEvt [▶ 191] 来读取数据。

6.1.4.3.4 FB_SQLResult



该功能块可用于读取缓存记录。

语法

定义：

```
FUNCTION_BLOCK FB_SQLResult
VAR_INPUT
    sNetID: T_AmsNetID := '';
    tTimeout: TIME := T#5S;
END_VAR
VAR_OUTPUT
    bBusy: BOOL;
    bError: BOOL;
    ipTcResultEvent: Tc3_EventLogger.I_TcResultEvent
END_VAR
```

输入

Name	Type	描述
sNetID	T_AmsNetID	ADS 命令指向的目标设备的 AMS 网络 ID。
tTimeout	TIME	表示函数被取消前的时间。

输出

Name	Type	描述
bBusy	BOOL	功能块的方法处于活动状态时即为 TRUE。
bError	BOOL	发生错误时即为 TRUE。
ipTcResultEvent	Tc3_EventLogger.I_TcResultEvent	结果界面，提供有关返回值的详细信息。

方法

Name	定义位置	描述
读取 [▶ 274]	本地	从 TwinCAT 数据库服务器缓存的结果数据中读取指定数量的记录。
Release [▶ 275]	本地	释放 TwinCAT 数据库服务器缓冲的数据。

要求

开发环境	目标平台	要连接的 PLC 库
TwinCAT v3.1 Build 4020.10	PC 或 CX（x86）	Tc3_Database

6.1.读取
4.3.
4.1

该方法从 TwinCAT 数据库服务器缓存的结果数据中读取指定数量的记录。

语法

```
METHOD Read : BOOL
VAR_INPUT
    nStartIndex: UDINT := 0;
    nRecordCount: UDINT := 1;
    pData: POINTER TO BYTE;
    cbData: UDINT;
    bWithVerifying: BOOL := FALSE;
    bDataRelease: BOOL := TRUE;
END_VAR
```

输入

Name	Type	描述
nStartIndex	UDINT	表示要读取的第一个记录的索引。
nRecordCount	UDINT	表示要读取的记录数。
pData	POINTER TO BYTE	要写入记录的结构数组的地址。
cbData	UDINT	表示结构数组的大小，以字节为单位。
bWithVerifying	BOOL	将返回数据与 pData 结构数组进行比较，并在必要时进行调整。
bDataRelease	BOOL	释放缓存数据。

返回值

Name	Type	描述
读取	BOOL	显示方法的状态。在方法执行完毕时，即使出现错误，也会返回 TRUE。

示例

```
VAR
    fbSqlResult : FB_SQLResultEvt(sNetID:='', tTimeout := T#5S);
    aReadStruct : ARRAY[1..5] OF ST_StandardRecord;
END_VAR
```

```
// get values from internal tc db srv storage
IF fbSqlResult.Read(2, 3, ADR(aReadStruct), SIZEOF(aReadStruct), FALSE, TRUE) THEN
    IF fbSqlResult.bError THEN
        nState := 255;
    ELSE
        nState := nState+1;
    END_IF
END_IF
```

PLC 中的结果：

Expression	Type	Value
aReadStruct	ARRAY [1..5] OF ST...	
aReadStruct[1]	ST_StandardRecord	
nID	LINT	9
dtTimestamp	DATE_AND_TIME	DT#2018-1-31-15:4:59
sName	STRING(80)	'Temperature'
rValue	LREAL	21.3
aReadStruct[2]	ST_StandardRecord	
nID	LINT	10
dtTimestamp	DATE_AND_TIME	DT#2018-1-31-15:5:59
sName	STRING(80)	'Temperature'
rValue	LREAL	21.2

6.1.Release
4.3.
4.2

该方法可以用于释放 TwinCAT 数据库服务器缓存的数据。

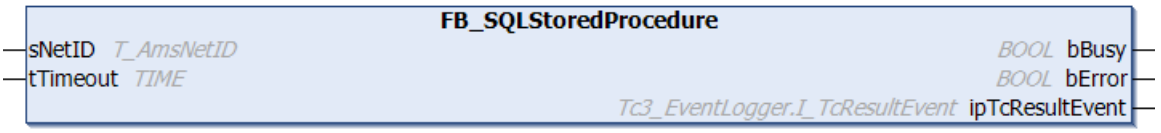
语法

```
METHOD Release : BOOL
```

返回值

Name	Type	描述
Release	BOOL	显示方法的状态。在方法执行完毕时，即使出现错误，也会返回 TRUE。

6.1.4.3.5 FB_SQLStoredProcedure



用于执行数据库的存储过程的功能块。在使用它之前，必须使用功能块“FB_SQLDatabase”对其进行初始化。

语法

定义：

```
FUNCTION_BLOCK FB_SQLStoredProcedure
VAR_INPUT
    sNetID: T_AmsNetID := '';
    tTimeout: TIME := T#5S;
END_VAR
VAR_OUTPUT
    bBusy: BOOL;
    bError: BOOL;
    ipTcResultEvent: Tc3_EventLogger.I_TcResultEvent
END_VAR
```

输入

Name	Type	描述
sNetID	T_AmsNetID	ADS 命令指向的目标设备的 AMS 网络 ID。
tTimeout	TIME	表示函数被取消前的时间。

输出

Name	Type	描述
bBusy	BOOL	功能块的方法处于活动状态时即为 TRUE。
bError	BOOL	发生错误时即为 TRUE。
ipTcResultEvent	Tc3_EventLogger.I_TcResultEvent	结果界面，提供有关返回值的详细信息。

方法

Name	定义位置	描述
执行 [▶ 277]	本地	通过功能块 FB_SQLDatabase 已打开的数据库连接，向数据库发送指定的存储过程的调用。
ExecuteDataReturn [▶ 277]	本地	通过功能块 FB_SQLDatabase 已打开的数据库连接，向数据库发送指定的存储过程的调用。 可以传输 FB_SQLResult 功能块的实例，以读取返回的记录。
Release [▶ 278]	本地	释放在初始化过程中传输的存储过程的参数信息。

要求

开发环境	目标平台	要连接的 PLC 库
TwinCAT v3.1 Build 4020.10	PC 或 CX（x86）	Tc3_Database

6.1.执行
4.3.
5.1

该方法通过功能块 FB_SQLDatabase 已打开的数据库连接，向数据库发送指定的存储过程的调用。

语法

```
METHOD Execute : BOOL
VAR_INPUT
    pParameterStrc: POINTER TO BYTE;
    cbParameterStrc: UDINT;
END_VAR
```

输入

Name	Type	描述
pParameterStrc	POINTER TO BYTE	传输到过程的参数结构的指针地址。
cbParameterStrc	UDINT	参数结构的长度

返回值

Name	Type	描述
执行	BOOL	显示方法的状态。在方法执行完毕时，即使出现错误，也会返回 TRUE。

示例

使用先前用 FB_SQLDatabaseEvt.CreateSP() [▶ 185] 创建的存储过程。

```
VAR
    fbSQLStoredProcedure : FB_SQLStoredProcedureEvt(sNetID:='', tTimeout := T#5S);
    Customer_ID          : LINT;
    tcMessage            : I_TcMessage;
END_VAR
```

```
IF fbSQLStoredProcedure.Execute(pParameterStrc := ADR(Customer_ID) , cbParameterStrc:= SIZEOF(Custom
er_ID)) THEN
    IF fbSQLStoredProcedure.bError THEN
        tcMessage := fbSQLStoredProcedure.ipTcResult;
        nState := 255;
    ELSE
        nState := nState+1;
    END_IF
END_IF
```

6.1.ExecuteDataReturn
4.3.
5.2

该方法通过功能块 FB_SQLDatabase 已打开的数据库连接，向数据库发送指定的存储过程的调用。可以传输 FB_SQLResult 功能块的实例，以读取返回的记录。

语法

```
METHOD ExecuteDataReturn : BOOL
VAR_INPUT
    pParameterStrc: POINTER TO BYTE;
    cbParameterStrc: UDINT;
    pSQLDBResult: POINTER TO FB_SQLDBResult;
END_VAR
```

输入

Name	Type	描述
pParameterStrc	POINTER TO BYTE	传输到过程的参数结构的指针地址。
cbParameterStrc	UDINT	参数结构的长度
pSQLDBResult	POINTER TO FB_SQLDBResult	返回功能块 FB_SQLDBResult 的新实例。

返回值

Name	Type	描述
读取	BOOL	显示方法的状态。在方法执行完毕时，即使出现错误，也会返回 TRUE。

示例

使用先前用 FB_SQLDatabaseEvt.CreateSP() [▶ 185] 创建的存储过程。

```
VAR
    fbSQLStoredProcedure : FB_SQLStoredProcedureEvt(sNetID:='', tTimeout := T#5S);
    Customer_ID          : LINT;
    tcMessage            : I_TcMessage;
END_VAR
```

```
IF fbSQLStoredProcedure.ExecuteDataReturn(pParameterStrc := ADR(Customer_ID), cbParameterStrc:= SIZE
OF(Customer_ID), pSQLDBResult := ADR(fbSqlResult)) THEN
    IF fbSQLStoredProcedure.bError THEN
        tcMessage := fbSQLStoredProcedure.ipTcResult;
        nState := 255;
    ELSE
        nState := nState+1;
    END_IF
END_IF
```

然后可以使用 FB_SQLResultEvt [▶ 191] 来读取数据。

6.1.Release
4.3.
5.3

该方法释放在初始化过程中传输的存储过程的参数信息。

语法

```
METHOD Release : BOOL
```

返回值

Name	Type	描述
Release	BOOL	显示方法的状态。在方法执行完毕时，即使出现错误，也会返回 TRUE。

6.2 Tc2_Database

概述

Tc2_Database 库包含用于控制和配置 TwinCAT 3 数据库服务器的功能块。

功能块

Name	描述
FB_GetStateTcDatabase [▶ 280]	检索状态信息。
FB_DBConnectionAdd [▶ 282]	将数据库连接添加到 XML 配置文件中。
FB_DBAuthentificationAdd [▶ 300]	将相应数据库连接的身份验证信息添加到 XML 配置文件中。
FB_DBOdbcConnectionAdd [▶ 283]	将 ODBC 数据库连接添加到 XML 配置文件中。
FB_AdsDeviceConnectionAdd [▶ 284]	将 ADS 设备添加到 XML 配置文件中。
FB_DBReloadConfig [▶ 281]	重新加载 XML 配置文件
FB_GetDBXMLConfig [▶ 285]	从 XML 配置文件中读取所有数据库配置。
FB_GetAdsDevXMLConfig [▶ 286]	从 XML 配置文件中读取所有 ADS 设备配置。
FB_DBConnectionOpen [▶ 287]	打开与数据库的连接。
FB_DBConnectionClose [▶ 287]	关闭与数据库的连接。
FB_DBCreate [▶ 288]	创建一个新的数据库
FB_DBTableCreate [▶ 289]	创建具有任何所需表结构的表格
FB_DBRead [▶ 291]	从数据库中读取一个值
FB_DBWrite [▶ 292]	将一个带有时间戳的变量值写入数据库
FB_DBCyclicRdWrt [▶ 290]	开始或停止记录/写入变量
FB_DBRecordSelect [▶ 302]	从表格中读取一个数据记录
FB_DBRecordSelect_EX [▶ 304]	从表格中读取一个数据记录（命令长度小于 10,000 个字符）
FB_DBRecordArraySelect [▶ 296]	从表格中读取多个记录。
FB_DBRecordInsert [▶ 301]	创建一个新的数据记录。
FB_DBRecordInsert_EX [▶ 295]	创建一个新的数据记录。（命令长度小于 10,000 个字符）
FB_DBRecordDelete [▶ 294]	从表格中删除一个记录。
FB_DBStoredProcedures [▶ 298]	执行存储过程。
FB_DBStoredProceduresRecordReturn [▶ 305]	执行存储过程并返回一个记录。
FB_DBStoredProceduresRecordArray [▶ 299]	执行存储过程并返回多个记录。

数据类型

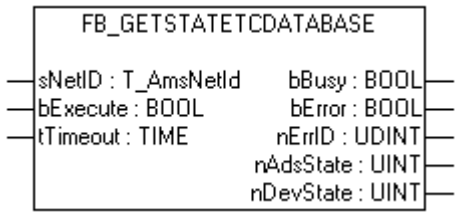
Name
ST_DBColumnCfg [▶ 306]
ST_DBXMLCfg [▶ 306]
ST_ADSDevXMLCfg [▶ 307]
ST_DBSQLError [▶ 307]
ST_DBParameter [▶ 307]
E_DbColumnTypes [▶ 308]
E_DBTypes [▶ 308]
E_DBValueType [▶ 309]
E_DBWriteModes [▶ 309]
E_DBParameterTypes [▶ 309]

要求

开发环境	目标系统类型	要连接的 PLC 库
TwinCAT v3.0.0	PC 或 CX（x86）	Tc2_Database

6.2.1 功能块

6.2.1.1 FB_GetStateTcDatabase



该功能块允许获取 TwinCAT 数据库服务器的当前状态。

VAR_INPUT

```
VAR_INPUT
    sNetID      : T_AmsNetID;
    bExecute    : BOOL;
    tTimeout    : TIME;
END_VAR
```

sNetID：包含 ADS 命令指向的目标设备的 AMS 网络 ID 的字符串。

bExecute：命令在上升沿执行。

tTimeout：表示超时时间。

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy      : BOOL;
    bError     : BOOL;
    nErrID     : UDINT;
    nAdsState  : UINT;
    nDevState  : UINT;
END_VAR
```

bBusy：命令正在由 ADS 传输。只要 bBusy 仍为 TRUE，就不会接受任何新命令。

bError：一旦发生错误，则变为 TRUE。

nErrID：如果设置了 bError 输出，则返回 ADS 错误代码。

nAdsState：包含 ADS 目标设备的状态识别代码。在这里返回的代码针对所有 ADS 服务器而指定：

- ADSSTATE_INVALID =0;
- ADSSTATE_IDLE =1;
- ADSSTATE_RESET =2;
- ADSSTATE_INIT =3;
- ADSSTATE_START =4;
- ADSSTATE_RUN =5;
- ADSSTATE_STOP =6;
- ADSSTATE_SAVECFG =7;
- ADSSTATE_LOADCFG =8;
- ADSSTATE_POWERFAILURE =9;
- ADSSTATE_POWERGOOD =10;
- ADSSTATE_ERROR =11;

nDevState：包含 ADS 目标设备的特定状态识别代码。在这里返回的代码是特定于 ADS 设备的补充信息。

- 1 = TwinCAT 数据库服务器已启动
- 2 = 循环读取或写入已开始

要求

开发环境	目标系统类型	要连接的 PLC 库
TwinCAT v3.0.0	PC 或 CX（x86）	Tc2_Database

6.2.1.2 FB_DBReloadConfig

通过 FB_DBReloadConfig 功能块可以重新加载 XML 配置文件。
如果对 XML 配置文件进行修改，则必须借助 FB_DBReloadConfig 将修改情况通知数据库服务器。

VAR_INPUT

```
VAR_INPUT
    sNetID   : T_AmsNetId;
    bExecute : BOOL;
    tTimeout : TIME;
END_VAR
```

sNetID：包含 ADS 命令指向的目标设备的 AMS 网络 ID 的字符串。

bExecute：命令在上升沿执行。

tTimeout：规定执行 ADS 命令时不得超过的超时长度。

VAR_OUTPUT

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

bBusy：命令正在由 ADS 传输。只要 bBusy 仍为 TRUE，就不会接受任何新命令。

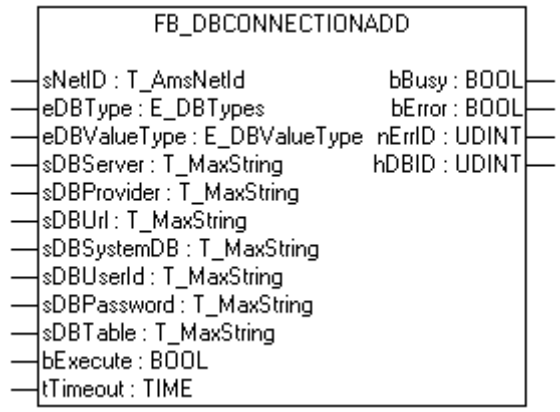
bError：一旦发生错误，则变为 TRUE。

nErrID：如果设置了 bError 输出，则返回 ADS 错误代码。

要求

开发环境	目标系统类型	要连接的 PLC 库
TwinCAT v3.0.0	PC 或 CX（x86）	Tc2_Database

6.2.1.3 FB_DBConnectionAdd



FB_DBConnectionAdd 功能块允许将额外的数据库连接添加到 XML 配置文件中。

VAR_INPUT

```
VAR_INPUT
    sNetID       :T_AmsNetId;
    eDBType      :E_DBTypes;
    eDBValueType :E_DBValueType;
    sDBServer    :T_MaxString;
    sDBProvider  :T_MaxString;
    sDBUrl       :T_MaxString;
    sDBSystemDB  :T_MaxString;
    sDBUserId    :T_MaxString;
    sDBPassword  :T_MaxString;
    sDBTable     :T_MaxString;
    bExecute     :BOOL;
    tTimeout     :TIME;
END_VAR
```

sNetID：包含 ADS 命令指向的目标设备的 AMS 网络 ID 的字符串。

eDBType：表示数据库的类型，例如“移动服务器”。

eDBValueType：表示数值存储或将要存储值的形式。

sDBServer：提供服务器的名称：可选。

sDBProvider：给出数据库的提供商：可选。

sDBUrl：提供数据库的路径。

sSystemDB：仅适用于 Access 数据库。表示 MDW 文件的路径。

sUserId：表示登录用户名。

sPassword：表示密码。

sDBTable：提供要将写入值的表格的名称。

bExecute：命令在上升沿执行。

tTimeout：表示函数被取消前的时间。

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy   : BOOL;
    bError  : BOOL;
    bErrID  : UDINT;
    hDBID   : UDINT;
END_VAR
```

bBusy：命令正在由 ADS 传输。只要 bBusy 仍为 TRUE，就不会接受任何新命令。

bError：一旦发生错误，则变为 TRUE。

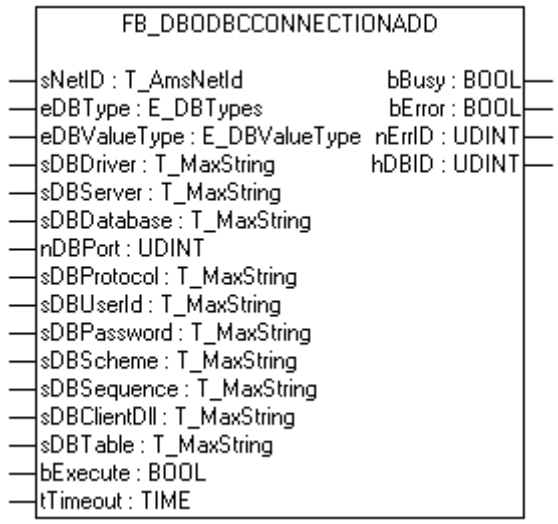
nErrID：如果设置了 bError 输出，则返回 ADS 错误代码。

hDBID：返回数据库的 ID。

要求

开发环境	目标系统类型	要连接的 PLC 库
TwinCAT v3.0.0	PC 或 CX（x86）	Tc2_Database

6.2.1.4 FB_DBOdbcConnectionAdd



功能块 FB_DBOdbcConnectionAdd 可以用于将更多的 ODBC 数据库连接添加到 XML 配置文件中。

VAR_INPUT

```
VAR_INPUT
    sNetID       :T_AmsNetId;
    eDBType      :E_DBTypes;
    eDBValueType :E_DBValueType;
    sDBDriver    :T_MaxString;
    sDBServer    :T_MaxString;
    sDBDatabase  :T_MaxString;
    nDBPort      :UDINT;
    sDBProtocol  :T_MaxString;
    sDBUserId    :T_MaxString;
    sDBPassword  :T_MaxString;
    sDBScheme    :T_MaxString;
    sDBSequence  :T_MaxString;
    sDBClientDll :T_MaxString;
    sDBTable     :T_MaxString;
    bExecute     :BOOL;
    tTimeout     :TIME;
END_VAR
```

sNetID：包含 ADS 命令指向的目标设备的 AMS 网络 ID 的字符串。

eDBType：表示数据库的类型，例如“Mobile Server”。

eDBValueType：表示存储或将要存储值的形式。

sDBDriver：表示要使用的 ODBC 驱动程序的名称。

sDBServer：表示服务器的名称。

sDBDatabase：表示数据库的名称。

nDBPort：表示 ODBC 连接的端口。

sDBProtocol：表示要使用的协议（TCPIP）。

sDBUserId：表示用户名。

sDBPassword：表示要使用的密码。

sDBScheme：表示要使用的数据库模式。

sDBSequence：表示 Oracle 数据库的序列名称。

sDBClientDll：包含 fbclient.dll 的路径。（仅适用于 Firebird/Interbase 数据库）

sDBTable：提供要将写入值的表格的名称。

bExecute：命令在上升沿执行。

tTimeout：表示函数被取消前的时间。

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy  : BOOL;
    bError : BOOL;
    bErrID : UDINT;
    hDBID  : UDINT;
END_VAR
```

bBusy：命令正在由 ADS 传输。只要 bBusy 仍为 TRUE，就不会接受任何新命令。

bError：一旦发生错误，则变为 TRUE。

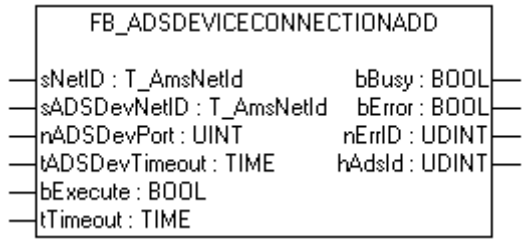
nErrID：如果设置了 bError 输出，则返回 ADS 错误代码。

hDBID：返回数据库的 ID。

要求

开发环境	目标系统类型	要连接的 PLC 库
TwinCAT v3.0.0	PC 或 CX（x86）	Tc2_Database

6.2.1.5 FB_AdsDeviceConnectionAdd



功能块 FB_AdsDeviceConnectionAdd 允许将额外的 Ads 设备连接添加到 XML 配置文件中。

VAR_INPUT

```
VAR_INPUT
    sNetID          : T_AmsNetID;
    sADSDevNetID    : T_AmsNetID;
    nADSDevPort     : UINT;
    tADSDevTimeout  : TIME;
```

```
    bExecute        : BOOL;
    tTimeout        : TIME;
END_VAR
```

sNetID：包含 ADS 命令指向的目标设备的 AMS 网络 ID 的字符串。

sADSDevNetID：包含 ADS 设备的 AMS 网络 ID 的字符串。

nADSDevPort：表示 ADS 设备的端口。

tAdsDevTimeout：表示 ADS 设备的超时时间。

bExecute：命令在上升沿执行。

tTimeout：表示超时的持续时间。

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy       : BOOL;
    bError      : BOOL;
    nErrID      : UDINT;
    hAdsId      : UDINT;
END_VAR
```

bBusy：命令正在由 ADS 传输。只要 bBusy 仍为 TRUE，就不会接受任何新命令。

bError：一旦发生错误，则变为 TRUE。

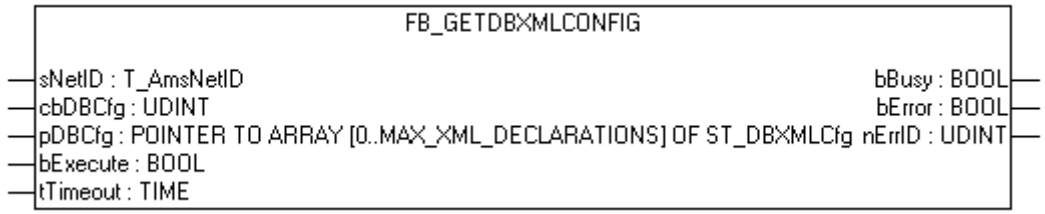
nErrID：如果设置了 bError 输出，则返回 ADS 错误代码。

hAdsId：返回 ADS 设备的 ID。

要求

开发环境	目标系统类型	要连接的 PLC 库
TwinCAT v3.0.0	PC 或 CX（x86）	Tc2_Database

6.2.1.6 FB_GetDBXMLConfig



通过该功能块 FB_GetDBXMLConfig，从 XML 配置文件中可以读出所有已声明的数据库。

VAR_INPUT

```
VAR_INPUT
    sNetID    : T_AmsNetId;
    cbDBCfg   : UDINT;
    pDBCfg    : POINTER TO ARRAY [0.. MAX_XML_DECLARATIONS] OF ST_DBXMLCfg
    bExecute  : BOOL;
    tTimeout  : TIME;
END_VAR
```

sNetID：包含 ADS 命令指向的目标设备的 AMS 网络 ID 的字符串。

cbDBCfg：表示要将配置写入的数组的长度。

pDBCfg：表示要将配置写入的数组的指针地址。

bExecute：命令在上升沿执行。

tTimeout：表示函数被取消前的时间。

VAR_OUTPUT

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

bBusy：命令正在由 ADS 传输。只要 bBusy 仍为 TRUE，就不会接受任何新命令。

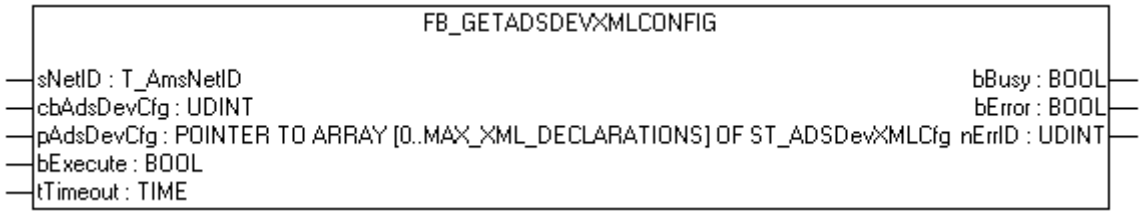
bError：一旦发生错误，则变为 TRUE。

nErrID：如果设置了 bError 输出，则返回 ADS 错误代码。

要求

开发环境	目标系统类型	要连接的 PLC 库
TwinCAT v3.0.0	PC 或 CX（x86）	Tc2_Database

6.2.1.7 FB_GetAdsDevXMLConfig



通过该功能块 FB_GetAdsDevXMLConfig，从 XML 配置文件中可以读出所有已声明的 ADS 设备。

VAR_INPUT

```
VAR_INPUT
    sNetID      : T_AmsNetId;
    cbAdsDevCfg : UDINT;
    pAdsDevCfg  : POINTER TO ARRAY [0.. MAX_XML_DECLARATIONS] OF ST_ADSDevXMLCfg
    bExecute    : BOOL;
    tTimeout    : TIME;
END_VAR
```

sNetID：包含 ADS 命令指向的目标设备的 AMS 网络 ID 的字符串。

cbAdsDevCfg：表示要将配置写入的数组的长度。

pAdsDevCfg：表示要将配置写入的数组的指针地址。

bExecute：命令在上升沿执行。

tTimeout：表示函数被取消前的时间。

VAR_OUTPUT

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

bBusy：命令正在由 ADS 传输。只要 bBusy 仍为 TRUE，就不会接受任何新命令。

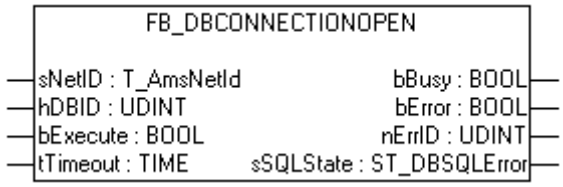
bError：一旦发生错误，则变为 TRUE。

nErrID：如果设置了 bError 输出，则返回 ADS 错误代码。

要求

开发环境	目标系统类型	要连接的 PLC 库
TwinCAT v3.0.0	PC 或 CX（x86）	Tc2_Database

6.2.1.8 FB_DBConnectionOpen



您可以使用该功能块 FB_DBConnectionOpen 打开与数据库的连接。这可以提高功能块 FB_DBWrite、FB_DBRead、FB_DBRecordInsert 和 FB_FBRecordSelect 的读取和写入访问速度。

VAR_INPUT

```
VAR_INPUT
    sNetID   : T_AmsNetId;
    hDBID    : DINT;
    bExecute: BOOL;
    tTimeout: TIME;
END_VAR
```

sNetID：包含 ADS 命令指向的目标设备的 AMS 网络 ID 的字符串。

hDBID：表示要使用的数据库的 ID。

bExecute：命令在上升沿执行。

tTimeout：表示函数被取消前的时间。

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy    : BOOL;
    bError   : BOOL;
    nErrID   : UDINT;
    sSQLState: ST_DBSQLError;
END_VAR
```

bBusy：命令正在由 ADS 传输。只要 bBusy 仍为 TRUE，就不会接受任何新命令。

bError：一旦发生错误，则变为 TRUE。

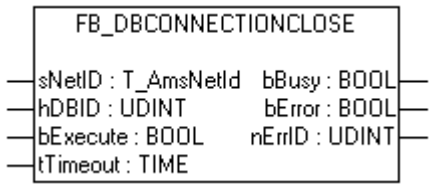
nErrID：如果设置了 bError 输出，则返回 ADS 错误代码或 TcDatabaseSrv_Error_Codes [▶ 378]。

sSQLState：返回相应数据库类型的 SQL 错误代码 [▶ 307]

要求

开发环境	目标系统类型	要连接的 PLC 库
TwinCAT v3.0.0	PC 或 CX（x86）	Tc2_Database

6.2.1.9 FB_DBConnectionClose



功能块 FB_DBConnectionClose 可以用于与数据库建立连接。如果之前已打开与数据库的连接，则必须再次关闭。

VAR_INPUT

```
VAR_INPUT
    sNetID   : T_AmsNetId;
    hDBID    : DINT;
    bExecute : BOOL;
    tTimeout : TIME;
END_VAR
```

sNetID：包含 ADS 命令指向的目标设备的 AMS 网络 ID 的字符串。

hDBID：表示要使用的数据库的 ID。

bExecute：命令在上升沿执行。

tTimeout：表示函数被取消前的时间。

VAR_OUTPUT

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

bBusy：命令正在由 ADS 传输。只要 bBusy 仍为 TRUE，就不会接受任何新命令。

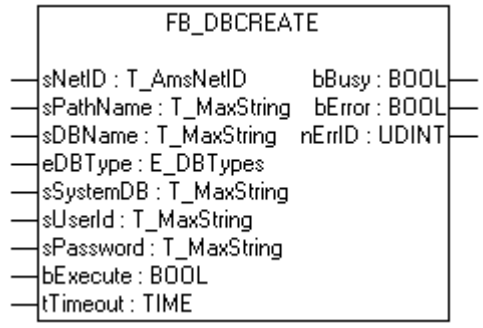
bError：一旦发生错误，则变为 TRUE。

nErrID：如果设置了 bError 输出，则返回 ADS 错误代码。

要求

开发环境	目标系统类型	要连接的 PLC 库
TwinCAT v3.0.0	PC 或 CX（x86）	Tc2_Database

6.2.1.10 FB_DBCreate



FB_DBCreate 功能块允许创建数据库。

使用该功能块可以创建以下数据库类型：MS SQL 数据库、MS SQL Compact 数据库、MS Access 数据库和 XML 数据库

使用功能块 FB_DBCreate 可以（但并非必须）创建 ASCII 文件。如果它们不存在，则会在首次写入访问期间自动创建。只需在 XML 配置文件中声明它们即可。

无法创建 DB2、Oracle、MySQL、PostgreSQL、InterBase 和 Firebird 数据库。此外，也无法覆盖现有的数据库。在这种情况下，功能块 FB_DBCreate 将返回错误。

VAR_INPUT

```
VAR_INPUT
    sNetID      : T_AmsNetID;
    sPathName   : T_MaxString;
    sDBName     : T_MaxString;
    eDBType     : E_DBTypes;
    sSystemDB   : T_MaxString;
    sUserID     : T_MaxString;
    sPassword   : T_MaxString;
    bExecute    : BOOL;
    tTimeout    : TIME;
END_VAR
```

sNetID：包含 ADS 命令指向的目标设备的 AMS 网络 ID 的字符串。

sPathName：提供数据库的路径。

sDBName：提供要创建的数据库的名称。

eDBType：提供要创建的数据库的类型。

sSystemDB：仅适用于 Access 数据库。包含 MDW 文件的路径。

sUserID：相应注册的用户名

sPassword：相应的密码

bExecute：命令在上升沿执行。

tTimeout：表示超时的持续时间。

VAR_OUTPUT

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

bBusy：命令正在由 ADS 传输。只要 bBusy 仍为 TRUE，就不会接受任何新命令。

bError：一旦发生错误，则变为 TRUE。

nErrID：如果设置了 bError 输出，则返回 ADS 错误代码。

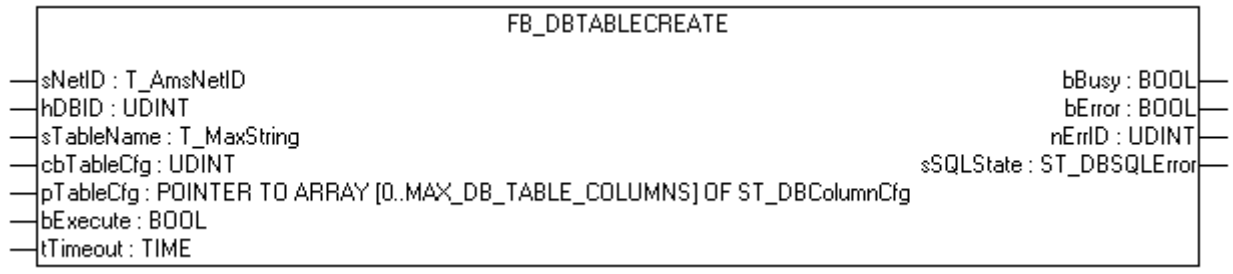
TwinCAT 数据库服务器

如果 TwinCAT 数据库服务器要使用新创建的数据库，则必须借助功能块 FB_DBConnectionADD 将连接数据写入 XML 配置文件。

要求

开发环境	目标系统类型	要连接的 PLC 库
TwinCAT v3.0.0	PC 或 CX（x86）	Tc2_Database

6.2.1.11 FB_DBTableCreate



FB_DBTableCreate 功能块允许在数据库中创建具有任何所需表结构的表格。

VAR_INPUT

```
VAR_INPUT
    sNetID      : T_AmsNetID;
    hDBID       : UDINT;
    sTableName  : T_MaxString;
    cbTableCfg  : UDINT;
    pTableCfg   : POINTER TO ARRAY[0..MAX_DB_TABLE_COLUMNS] OF ST_DBColumnCfg;
    bExecute    : BOOL;
    tTimeout    : TIME;
END_VAR
```

sNetID：包含 ADS 命令指向的目标设备的 AMS 网络 ID 的字符串。

hDBID：要使用的数据库的 ID。

sTableName：提供表格的名称。

cbTableCfg：返回配置列的数组的长度。

pTableCfg：提供表结构数组的指针地址。将各个列写入该数组。

bExecute：命令在上升沿执行。

tTimeout：表示超时的持续时间。

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy     : BOOL;
    bError    : BOOL;
    nErrID    : UDINT;
    sSQLState: ST_DBSQLError;
END_VAR
```

bBusy：命令正在由 ADS 传输。只要 bBusy 仍为 TRUE，就不会接受任何新命令。

bError：一旦发生错误，则变为 TRUE。

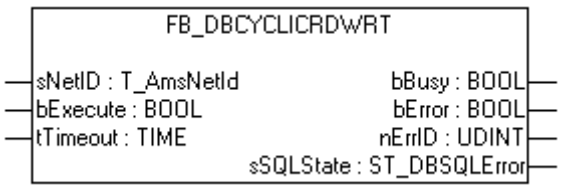
nErrID：如果设置了 bError 输出，则返回 ADS 错误代码或 TcDatabaseSrv_Error_Codes [▶ 378]。

sSQLState：返回相应数据库类型的 SQL 错误代码 [▶ 307]

要求

开发环境	目标系统类型	要连接的 PLC 库
TwinCAT v3.0.0	PC 或 CX（x86）	Tc2_Database

6.2.1.12 FB_DBCyclicRdWrt



FB_DBCyclicRdWrt 功能块可以用于启动或停止对变量的循环日志记录/写入。

VAR_INPUT

```
VAR_INPUT
    sNetID   : T_AmsNetId;
    bExecute: BOOL;
    tTimeout: TIME;
END_VAR
```

sNetID：包含 ADS 命令指向的目标设备的 AMS 网络 ID 的字符串。

bExecute：读取/写入周期以上升沿开始，并以下降沿停止。

tTimeout：规定执行 ADS 命令时不得超过的超时长度。

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy    : BOOL;
    bError   : BOOL;
    nErrID   : UDINT;
    sSQLState: ST_DBSQLError;
END_VAR
```

bBusy：命令正在由 ADS 传输。只要 bBusy 仍为 TRUE，就不会接受任何新命令。

bError：一旦发生错误，则变为 TRUE。

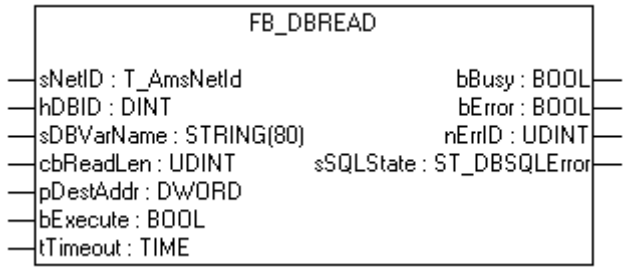
nErrID：如果设置了 bError 输出，则返回 ADS 错误代码或 TcDatabaseSrv_Error_Codes [▶ 378]。

sSQLState：返回相应数据库类型的 SQL 错误代码 [▶ 307]

要求

开发环境	目标系统类型	要连接的 PLC 库
TwinCAT v3.0.0	PC 或 CX（x86）	Tc2_Database

6.2.1.13 FB_DBRead



FB_DBRead 允许从数据库中读取数值。

VAR_INPUT

```
VAR_INPUT
    sNetID    : T_AmsNetId;
    hDBID     : DINT;
    sDBVarName: STRING(80);
    cbReadLen : UDINT;
    pDestAddr : POINTER TO BYTE;
    bExecute  : BOOL;
    tTimeout  : TIME;
END_VAR
```

sNetID：包含 ADS 命令指向的目标设备的 AMS 网络 ID 的字符串。

hDBID：表示要使用的数据库的 ID。

sDBVarName：提供要读取的变量的名称。

cbReadLen：表示要读取的缓冲区的长度。

pDestAddr：包含要接收已读取数据的缓冲区的地址。

bExecute：命令在上升沿执行。

tTimeout：表示函数被取消前的时间。

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy    : BOOL;
    bError   : BOOL;
```

```
    nErrID   : UDINT;
    sSQLState: ST_DBSQLError;
END_VAR
```

bBusy：命令正在由 ADS 传输。只要 bBusy 仍为 TRUE，就不会接受任何新命令。

bError：一旦发生错误，则变为 TRUE。

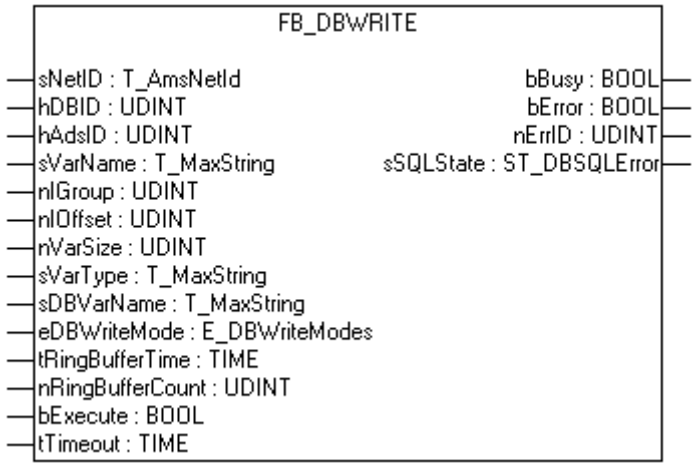
nErrID：如果设置了 bError 输出，则返回 ADS 错误代码或 TcDatabaseSrv_Error_Codes [▶ 378]。

sSQLState：返回相应数据库类型的 SQL 错误代码 [▶ 307]

要求

开发环境	目标系统类型	要连接的 PLC 库
TwinCAT v3.0.0	PC 或 CX（x86）	Tc2_Database

6.2.1.14 FB_DBWrite



FB_DBWrite 功能块可以用于将各个变量的值写入数据库。
表结构必须包含“时间戳”、“名称”和“值”列（请参见“SQL Compact 数据库” [▶ 125]）。为了能够使用该功能块，在 XML 配置文件中必须声明要用于写入访问的数据库和要用于读取变量的 ADS 设备。

VAR_INPUT

```
VAR_INPUT
    sNetID          : T_AmsNetID;
    hDBID           : UDINT;
    hAdsID          : UDINT;
    sVarName        : T_MaxString;
    nIGroup         : UDINT;
    nIOffset        : UDINT;
    nVarSize        : UDINT;
    sVarType        : T_MaxString;
    sDBVarName      : T_MaxString;
    eDBWriteMode    : E_DBWriteModes;
    tRingBufferTime : TIME;
    nRingBufferCount: UDINT;
    bExecute        : BOOL;
    tTimeout        : TIME;
END_VAR
```

sNetID：包含 ADS 命令指向的目标设备的 AMS 网络 ID 的字符串。

hDBID：要使用的数据库的 ID。

hAdsID：要使用的 ADS 设备的 ID。

sVarName：提供变量的名称。

nIGroup：变量的索引组（可选，仅限在 BC9000 上）。

nIOffset：变量的索引偏移（可选，仅限在 BC9000 上）。

nVarSize：变量的大小，以字节为单位（可选，仅限在 BC9000 上）。

sVarType：变量的数据类型（可选，仅限在 BC9000 上）。

可能的变量数据类型：“BOOL” / “LREAL” / “REAL” / “INT16” / “DINT” / “USINT” / “BYTE” / “UDINT” / “DWORD” / “UINT16” / “WORD” / “SINT”
sDBVarName：要在数据库中使用的变量名称。

eDBWriteMode：表示是要在新记录中添加值，还是要更新现有记录。

tRingBufferTime：表示表格中的记录的最大年龄（仅适用于 Ringbuffer_WriteMode）。

nRingBufferCount：表示表格中的记录的最大数量（仅适用于 Ringbuffer_WriteMode）。

bExecute：命令在上升沿执行。

tTimeout：表示函数被取消前的时间。

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy    : BOOL;
    bError   : BOOL;
    nErrID   : UDINT;
    sSQLState: ST_DBSQLError;
END_VAR
```

bBusy：命令正在由 ADS 传输。只要 bBusy 仍为 TRUE，就不会接受任何新命令。

bError：一旦发生错误，则变为 TRUE。

nErrID：如果设置了 bError 输出，则返回 ADS 错误代码或 TcDatabaseSrv_Error_Codes [▶ 378]。

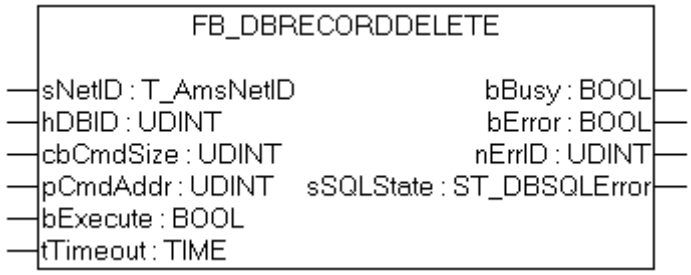
sSQLState：返回相应数据库类型的 SQL 错误代码 [▶ 307]

ADS 设备的日志值（BC9000 除外）	BC9000 的日志值
FB_DBWrite1(sNetID:= , hDBID:= 1, hAdsID:= 1, sVarName:= 'MAIN.TestVar', sDBVarName:= 'DBTestVar', eDBWriteMode:= eDBWriteMode_Append, bExecute:= TRUE, tTimeout:= T#15s, bBusy=> busy, bError=> err, nErrID=> errid, sSQLState=> sqlstate);	FB_DBWrite1(sNetID:= , hDBID:= 1, hAdsID:= 1, sVarName:= 'MAIN.TestVar', nIGroup:= 16448, nIOffset:= 0, nVarSize:= 16, sVarType:= 'REAL', sDBVarName:= 'DBTestVar', eDBWriteMode:= eDBWriteMode_Append, bExecute:= TRUE, tTimeout:= T#15s, bBusy=> busy, bError=> err, nErrID=> errid, sSQLState=> sqlstate);

要求

开发环境	目标系统类型	要连接的 PLC 库
TwinCAT v3.0.0	PC 或 CX（x86）	Tc2_Database

6.2.1.15 FB_DBRecordDelete



功能块 FB_DBRecordDelete 可以用于删除数据库中的单个记录。该功能块可以用于执行多达 10,000 个字符的 SQL DELETE 命令。
要使用该功能块，必须在 XML 配置文件中声明要删除记录的数据库。

VAR_INPUT

```
VAR_INPUT
    sNetID    : T_AmsNetId;
    hDBID     : UDINT;
    cbCmdSize: UDINT;
    pCmdAddr  : POINTER TO BYTE;
    bExecute  : BOOL;
    tTimeout  : TIME;
END_VAR
```

sNetID：包含 ADS 命令指向的目标设备的 AMS 网络 ID 的字符串。

hDBID：表示要使用的数据库的 ID。

cbCmdSize：表示 INSERT 命令的长度。

pCmdAddr：执行 INSERT 命令的指针。

bExecute：命令在上升沿执行。

tTimeout：表示函数被取消前的时间。

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy     : BOOL;
    bError    : BOOL;
    nErrID    : UDINT;
    sSQLState: ST_DBSQLError;
END_VAR
```

bBusy：命令正在由 ADS 传输。只要 bBusy 仍为 TRUE，就不会接受任何新命令。

bError：一旦发生错误，则变为 TRUE。

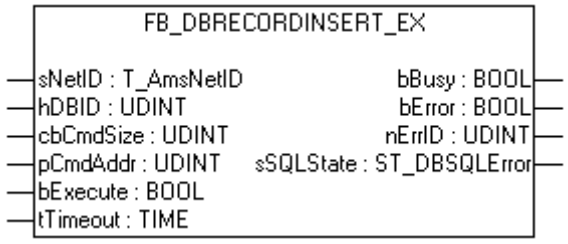
nErrID：如果设置了 bError 输出，则返回 ADS 错误代码或 TcDatabaseSrv_Error_Codes [▶ 378]。

sSQLState：返回相应数据库类型的 SQL 错误代码 [▶ 307]

要求

开发环境	目标系统类型	要连接的 PLC 库
TwinCAT v3.0.0	PC 或 CX（x86）	Tc2_Database

6.2.1.16 FB_DBRecordInsert_EX



功能块 FB_DBRecordInsert_EX 可以用于将任何结构的单个记录写入数据库。该功能块可以用于执行多达 10,000 个字符的 SQL INSERT 命令。
要使用该功能块，必须在 XML 配置文件中声明要向其写入记录的数据库。

VAR_INPUT

```
VAR_INPUT
    sNetID   : T_AmsNetId;
    hDBID    : UDINT;
    cbCmdSize: UDINT;
    pCmdAddr : POINTER TO BYTE;
    bExecute : BOOL;
    tTimeout : TIME;
END_VAR
```

sNetID：包含 ADS 命令指向的目标设备的 AMS 网络 ID 的字符串。

hDBID：表示要使用的数据库的 ID。

cbCmdSize：表示 INSERT 命令的长度。

pCmdAddr：执行 INSERT 命令的指针

bExecute：命令在上升沿执行。

tTimeout：表示函数被取消前的时间。

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy    : BOOL;
    bError   : BOOL;
    nErrID   : UDINT;
    sSQLState: ST_DBSQLError;
END_VAR
```

bBusy：命令正在由 ADS 传输。只要 bBusy 仍为 TRUE，就不会接受任何新命令。

bError：一旦发生错误，则变为 TRUE。

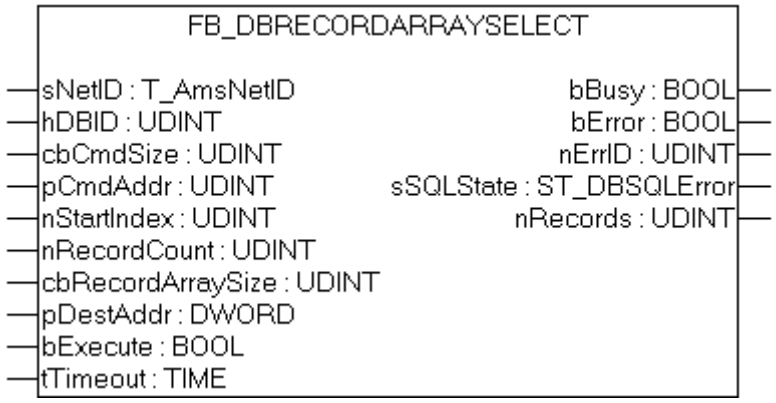
nErrID：如果设置了 bError 输出，则返回 ADS 错误代码或 TcDatabaseSrv_Error_Codes [▶ 378]。

sSQLState：返回相应数据库类型的 SQL 错误代码 [▶ 307]

要求

开发环境	目标系统类型	要连接的 PLC 库
TwinCAT v3.0.0	PC 或 CX（x86）	Tc2_Database

6.2.1.17 FB_DBRecordArraySelect



功能块 FB_DBRecordArraySelect 可以用于从数据库中读取任何结构的多个记录。该功能块可以用于执行多达 10,000 个字符的 SQL SELECT 命令。
该功能块与 ASCII 文件不兼容。

VAR_INPUT

```
VAR_INPUT
    sNetID           : T_AmsNetID;
    hDBID            : UDINT;
    cbCmdSize        : UDINT;
    pCmdAddr         : UDINT;
    nStartIndex      : UDINT;
    nRecordCount     : UDINT;
    cbRecordArraySize: UDINT;
    pDestAddr        : POINTER TO BYTE;
    bExecute         : BOOL;
    tTimeout         : TIME;
END_VAR
```

sNetID：包含 ADS 命令指向的目标设备的 AMS 网络 ID 的字符串。

hDBID：表示要使用的数据库的 ID。

cbCmdSize：表示要执行的 SELECT 命令的长度。

pCmdSize：表示使用要执行的 SQL 命令的字符串变量的指针地址。

nStartIndex：表示要读取的第一个记录的索引。

nRecordCount：表示要读取的记录数量。

cbRecordArraySize：表示结构数组的大小，以字节为单位。

pDestAddr：表示要写入记录的结构数组的地址。

bExecute：命令在上升沿执行。

tTimeout：表示函数被取消前的时间。

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy    : BOOL;
    bError   : BOOL;
    nErrID   : UDINT;
    sSQLState: ST_DBSQLError;
    nRecords : UDINT;
END_VAR
```

ST_DBSQLError [▶ 307]

bBusy：命令正在由 ADS 传输。只要 bBusy 仍为 TRUE，就不会接受任何新命令。

bError：一旦发生错误，则变为 TRUE。

nErrID：如果设置了 bError 输出，则返回 ADS 错误代码或 TcDatabaseSrv_Error_Codes [▶ 378]。

sSQLState：返回相应数据库类型的 SQL 错误代码

nRecords：返回数据记录的数量。

ST 中的示例

由于要从中读取记录的表格具有如下结构，因此必须创建一个具有类似结构的 PLC 结构。

表格：

ColumnName	Data type
ID	Bigint
Timestamp	DateTime
Name	nvarchar(80)
值	float

结构：

```
TYPE ST_Record:
STRUCT
    ID        : T_ULARGE_INTEGER;
    Timestamp: DT;
    Name      : STRING(80);
    VALUE     : LREAL;
END_STRUCT
END_TYPE
```

必须集成 TcUtilities.lib 库才能使用数据类型 T_ULARGE_INTEGER。

对于 ARM 处理器，由于字节对齐，数据类型必须以不同的方式排列，并且必须添加“虚字节”。

```
TYPE ST_Record :
STRUCT
    ID        : T_ULARGE_INTEGER;
    Timestamp: DT;
    Value     : LREAL;
    Name      : STRING(80);
    Dummy     : BYTE;
END_STRUCT
END_TYPE
```

```
PROGRAM MAIN
VAR
    FB_DBRecordArraySelect1 : FB_DBRecordArraySelect;
    cmd          : T_Maxstring := 'SELECT * FROM myTable';
    (* Unter ARM*)
    (*cmd        : T_Maxstring := 'SELECT ID,Timestamp,Value,Name FROM myTable'*)
    (*----------*)
    recordArray : ARRAY [1..5] OF ST_Record;
    busy         : BOOL;
    err          : BOOL;
    errid        : UDINT;
    sqlstate     : ST_DBSQLError;
    recAnz       : UDINT;
END_VAR
```

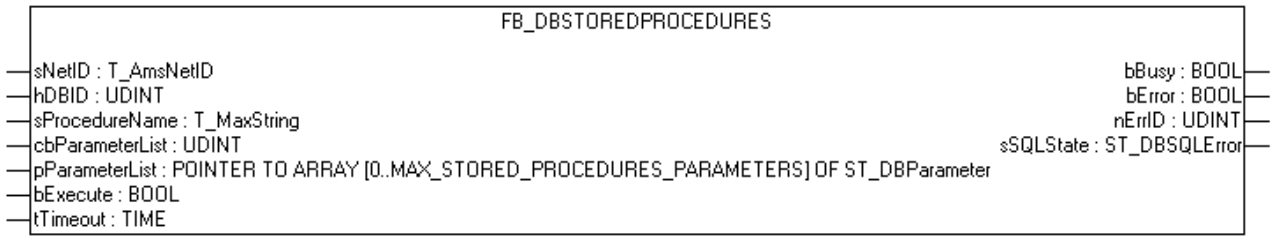
PLC 程序

```
FB_DBRecordArraySelect1(
    sNetID:= ,
    hDBID:= 1,
    cbCmdSize:= SIZEOF(cmd),
    pCmdAddr:= ADR(cmd),
    nStartIndex:= 0,
    nRecordCount:= 5,
    cbRecordArraySize:= SIZEOF(recordArray),
    pDestAddr:= ADR(recordArray),
    bExecute:= TRUE,
    tTimeout:= T#15s,
    bBusy=> busy,
    bError=> err,
    nErrID=> errid,
    sSQLState=> sqlstate,
    nRecords=> recAnz);
```

要求

开发环境	目标系统类型	要连接的 PLC 库
TwinCAT v3.0.0	PC 或 CX（x86）	Tc2_Database

6.2.1.18 FB_DBStoredProcedures



功能块 FB_DBStoredProcedures 可以用于调用存储过程。它们可以在流程中包含在存储过程中使用的参数。

VAR_INPUT

```
VAR_INPUT
 sNetID          : T_AmsNetID     :='';
 hDBID           : UDINT          :=1;
 sProcedureName  : T_MaxString    :='';
 cbParameterList: UDINT;
 pParameterList  : POINTER TO ARRAY[0..MAX_STORED_PROCEDURES_PARAMETERS] OF ST_DBParameter;
 bExecute        : BOOL;
 tTimeout        : TIME           := T#15s;
END_VAR
```

sNetID：包含 ADS 命令指向的目标设备的 AMS 网络 ID 的字符串。

hDBID：表示要使用的数据库的 ID。

sProcedureName：表示要执行的过程的名称

cbParameterList：表示参数列表的长度。

pParameterList：包含参数列表的地址

bExecute：命令在上升沿执行。

tTimeout：表示函数被取消前的时间。

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy    : BOOL;
    bError   : BOOL;
    nErrID   : UDINT;
    sSQLState: ST_DBSQLError;
END_VAR
```

bBusy：命令正在由 ADS 传输。只要 bBusy 仍为 TRUE，就不会接受任何新命令。

bError：一旦发生错误，则变为 TRUE。

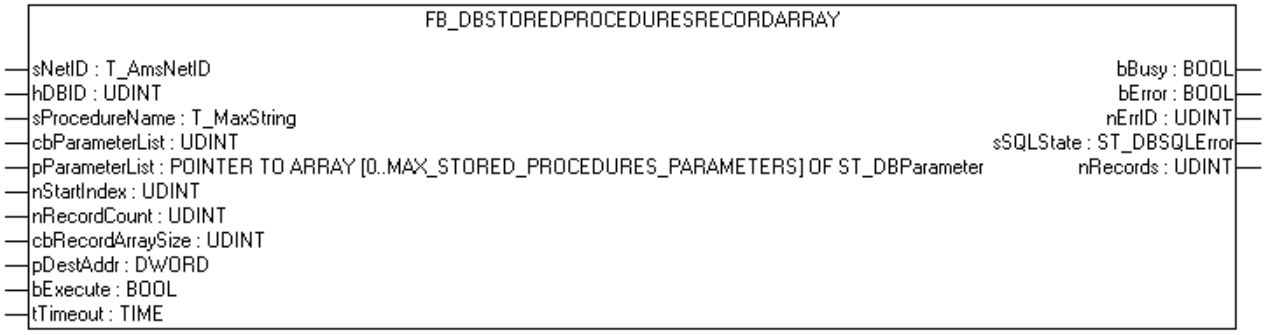
nErrID：如果设置了 bError 输出，则返回 ADS 错误代码或 TcDatabaseSrv_Error_Codes [▶ 378]。

sSQLState：返回相应数据库类型的 SQL 错误代码 [▶ 307]

要求

开发环境	目标系统类型	要连接的 PLC 库
TwinCAT v3.0.0	PC 或 CX（x86）	Tc2_Database

6.2.1.19 FB_DBStoredProceduresRecordArray



功能块 FB_DBStoredProceduresRecordArray 可以用于调用返回记录的存储过程。与 FB_DBStoredProceduresRecordReturn 功能块相比，该功能块可以用于通过一次调用返回多个记录。它们可以在流程中包含在存储过程中使用的参数。

VAR_INPUT

```
VAR_INPUT
 sNetID           : T_AmsNetID           :='';
 hDBID            : UDINT                :=1;
 sProcedureName   : T_MaxString          :='';
 cbParameterList  : UDINT;
 pParameterList   : POINTER TO ARRAY[0..MAX_STORED_PROCEDURES_PARAMETERS] OF ST_DBParameter;
 nStartIndex      : UDINT;
 nRecordCount     : UDINT
 cbRecordArraySize: UDINT;
 pDesAddr         : POINTER TO BYTE;
 bExecute         : BOOL;
 tTimeout         : TIME                  := T#15s;
END_VAR
```

sNetID：包含 ADS 命令指向的目标设备的 AMS 网络 ID 的字符串。

hDBID：表示要使用的数据库的 ID。

sProcedureName：表示要执行的过程的名称。

cbParameterList：表示参数列表的长度。

pParameterList：包含参数列表的地址

nStartIndex：表示要读取的第一个记录的索引。

nRecordCount：表示要读取的记录数量。

cbRecordArraySize：表示结构数组的大小，以字节为单位。

pDestAddr：表示要写入记录的结构数组的地址。

bExecute：命令在上升沿执行。

tTimeout：表示函数被取消前的时间。

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy    : BOOL;
    bError   : BOOL;
    nErrID   : UDINT;
    sSQLState: ST_DBSQLError;
    nRecords : UDINT;
END_VAR
```

bBusy：命令正在由 ADS 传输。只要“bBusy”仍为 TRUE，就不会接受任何新命令。

bError：一旦发生错误，则变为 TRUE。

nErrID：如果设置了 bError 输出，则返回 ADS 错误代码或 TcDatabaseSrv_Error_Codes [▶ 378]。

sSQLState：返回相应数据库类型的 SQL 错误代码 [▶ 307]

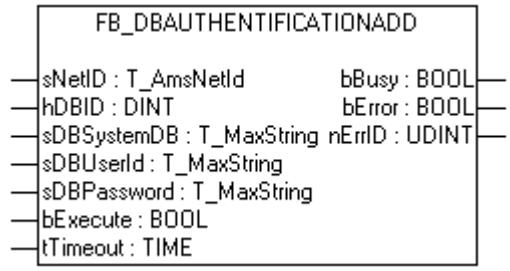
nRecords：返回数据记录的数量。

要求

开发环境	目标系统类型	要连接的 PLC 库
TwinCAT v3.0.0	PC 或 CX（x86）	Tc2_Database

6.2.1.20 不再支持

6.2.1.20.1 FB_DBAuthentificationAdd



功能块 FB_DBAuthentificationAdd 允许已声明的数据库连接的身份验证信息添加到 XML 配置文件中添加或对其进行更改。

VAR_INPUT

```
VAR_INPUT
    sNetID      : T_AmsNetID;
    hDBID       : DINT;
    sDBSystemDB: T_MaxString;
    sDBUserId   : T_MaxString;
    sDBPassword: T_MaxString;
    bExecute    : BOOL;
    tTimeout    : TIME;
END_VAR
```

sNetID：包含 ADS 命令指向的目标设备的 AMS 网络 ID 的字符串。

hDBID：是要使用的数据库的 ID。

sSystemDB：仅适用于 Access 数据库。表示 MDW 文件的路径。

sUserId：表示登录用户名。

sPassword：表示密码。

bExecute：命令在上升沿执行。

tTimeout：表示超时的持续时间。

VAR_OUTPUT

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

bBusy：命令正在由 ADS 传输。只要“bBusy”仍为 TRUE，就不会接受任何新命令。

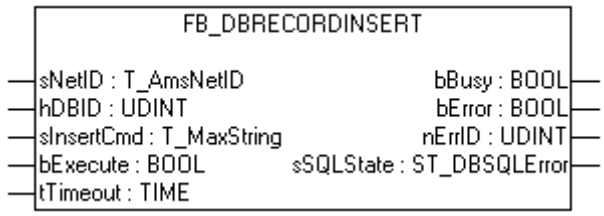
bError：一旦发生错误，则变为 TRUE。

nErrID：如果设置了 bError 输出，则返回 ADS 错误代码。

要求

开发环境	目标系统类型	要连接的 PLC 库
TwinCAT v3.0.0	PC 或 CX（x86）	Tc2_Database

6.2.1.20.2 FB_DBRecordInsert



功能块 FB_DBRecordInsert 可以用于将任何结构的单个记录写入数据库。要使用该功能块，必须在 XML 配置文件中声明要向其写入记录的数据库。

VAR_INPUT

```
VAR_INPUT
    sNetID     : T_AmsNetId;
    hDBID      : UDINT;
    sInsertCmd: T_MaxString;
    bExecute   : BOOL;
    tTimeout   : TIME;
END_VAR
```

sNetID：包含 ADS 命令指向的目标设备的 AMS 网络 ID 的字符串。

hDBID：表示要使用的数据库的 ID。

sInsertCmd：表示要执行的 INSERT 命令。

bExecute：命令在上升沿执行。

tTimeout：表示函数被取消前的时间。

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy     : BOOL;
    bError    : BOOL;
    nErrID    : UDINT;
    sSQLState: ST_DBSQLError;
END_VAR
```

bBusy：命令正在由 ADS 传输。只要 bBusy 仍为 TRUE，就不会接受任何新命令。

bError：一旦发生错误，则变为 TRUE。

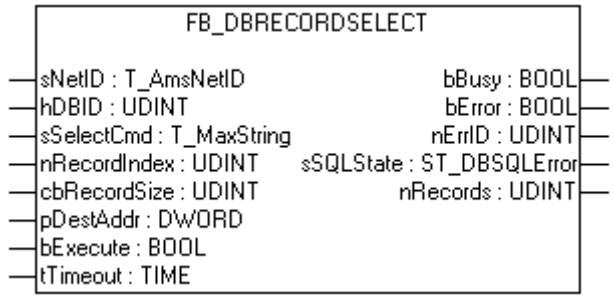
nErrID：如果设置了 bError 输出，则返回 ADS 错误代码或 TcDatabaseSrv_Error_Codes [▶ 378]。

sSQLState：返回相应数据库类型的 SQL 错误代码 [▶ 307]

要求

开发环境	目标系统类型	要连接的 PLC 库
TwinCAT v3.0.0	PC 或 CX（x86）	Tc2_Database

6.2.1.20.3 FB_DBRecordSelect



FB_DBRecordSelect 允许从数据库中读取单个数据记录。
该功能块与 ASCII 文件不兼容。

VAR_INPUT

```
VAR_INPUT
    sNetID      : T_AmsNetID;
    hDBID       : UDINT;
    sSelectCmd  : T_MaxString;
    nRecordIndex: UDINT;
    cbRecordSize: UDINT;
    pDestAddr   : DWORD;
    bExecute    : BOOL;
    tTimeout    : TIME;
END_VAR
```

sNetID：包含 ADS 命令指向的目标设备的 AMS 网络 ID 的字符串。

hDBID：表示要使用的数据库的 ID。

sSelectCmd：表示要执行的 SELECT 命令。

nRecordIndex：提供要读取的数据记录的索引。

cbRecordSize：提供数据记录的大小，以字节为单位。

pDestAddr：表示要写入记录的结构的地址。

bExecute：命令在上升沿执行。

tTimeout：表示函数被取消前的时间。

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy    : BOOL;
    bError   : BOOL;
    nErrID   : UDINT;
    sSQLState: ST_DBSQLError;
    nRecords : UDINT;
END_VAR
```

bBusy：命令正在由 ADS 传输。只要 bBusy 仍为 TRUE，就不会接受任何新命令。

bError：一旦发生错误，则变为 TRUE。

nErrID：如果设置了 bError 输出，则返回 ADS 错误代码或 TcDatabaseSrv_Error_Codes [▶ 378]。

sSQLState：返回相应数据库类型的 SQL 错误代码 [▶ 307]

nRecords：返回数据记录的数量。

ST 中的示例：

由于要从中读取记录的表格具有如下结构，因此必须创建一个具有类似结构的 PLC 结构。

表格：

ColumnName	Data type
ID	Bigint
Timestamp	DateTime
Name	Ntext
值	Float

结构：

```
TYPE ST_Record :
STRUCT
    ID        : T_ULARGE_INTEGER;
    Timestamp: DT;
    Name      : STRING;
    VALUE     : LREAL;
END_STRUCT
END_TYPE
```

必须集成 TcUtilities.lib 库才能使用数据类型 T_ULARGE_INTEGER。

对于 ARM 处理器，由于字节对齐，数据类型必须以不同的方式排列，并且必须添加“虚 BYTE”。

```
TYPE ST_Record :
STRUCT
    ID        : T_ULARGE_INTEGER;
    Timestamp: DT;
    Value     : LREAL;
    Name      : STRING;
    Dummy     : BYTE;
END_STRUCT
END_TYPE
```

```
PROGRAM MAIN
VAR
    FB_DBRecordSelect1: FB_DBRecordSelect;
    cmd                : T_Maxstring := 'SELECT * FROM myTable';
    (* Unter ARM*)
    (*cmd              : T_Maxstring := 'SELECT ID,Timestamp,Value,Name FROM myTable'*)
    (*----------*)
    record             : ST_Record;
    busy               : BOOL;
    err                : BOOL;
    errid              : UDINT;
    recAnz             : DINT;
END_VAR
```

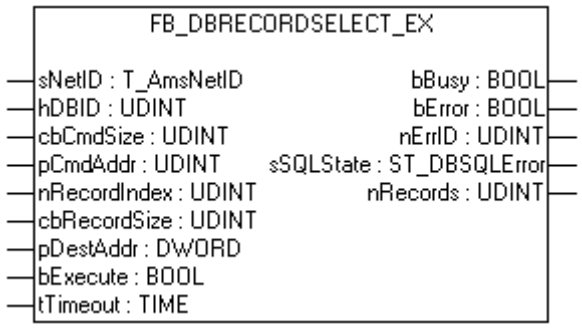
PLC 程序

```
FB_DBRecordSelect1(
    sNetID      := ,
    hDBID       := 2,
    sSelectCmd  := cmd,
    nRecordIndex:= 0,
    cbRecordSize:= SIZEOF(record),
    pDestAddr   := ADR(record),
    bExecute    := TRUE,
    tTimeout    := T#15s,
    bBusy       => busy,
    bError      => err,
    nErrID      => errid,
    nRecords    => recAnz);
```

要求

开发环境	目标系统类型	要连接的 PLC 库
TwinCAT v3.0.0	PC 或 CX（x86）	Tc2_Database

6.2.1.20.4 FB_DBRecordSelect_EX



功能块 FB_DBRecordSelect_EX 可以用于从数据库中读取任何结构的单个记录。该功能块可以用于执行多达 10,000 个字符的 SQL SELECT 命令。
该功能块与 ASCII 文件不兼容。

VAR_INPUT

```
VAR_INPUT
    sNetID      : T_AmsNetID;
    Hdbid       : UDINT;
    cbCmdSize   : UDINT;
    pCmdAddr    : UDINT;
    nRecordIndex: UDINT;
    cbRecordSize: UDINT;
    pDestAddr   : POINTER TO BYTE;
    bExecute    : BOOL;
    tTimeout    : TIME;
END_VAR
```

sNetID：包含 ADS 命令指向的目标设备的 AMS 网络 ID 的字符串。

hDBID：表示要使用的数据库的 ID。

cbCmdSize：表示要执行的 SELECT 命令的长度。

pCmdSize：表示使用要执行的 SQL 命令的字符串变量的指针地址。

nRecordIndex：提供要读取的数据记录的索引。

cbRecordSize：提供数据记录的大小，以字节为单位。

pDestAddr：表示要写入记录的结构的地址。

bExecute：命令在上升沿执行。

tTimeout：表示函数被取消前的时间。

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy    : BOOL;
    bError   : BOOL;
    nErrID   : UDINT;
    sSQLState: ST_DBSQLError;
    nRecords : UDINT;
END_VAR
```

bBusy：命令正在由 ADS 传输。只要 bBusy 仍为 TRUE，就不会接受任何新命令。

bError：一旦发生错误，则变为 TRUE。

nErrID：如果设置了 bError 输出，则返回 ADS 错误代码或 TcDatabaseSrv_Error_Codes [▶ 378]。

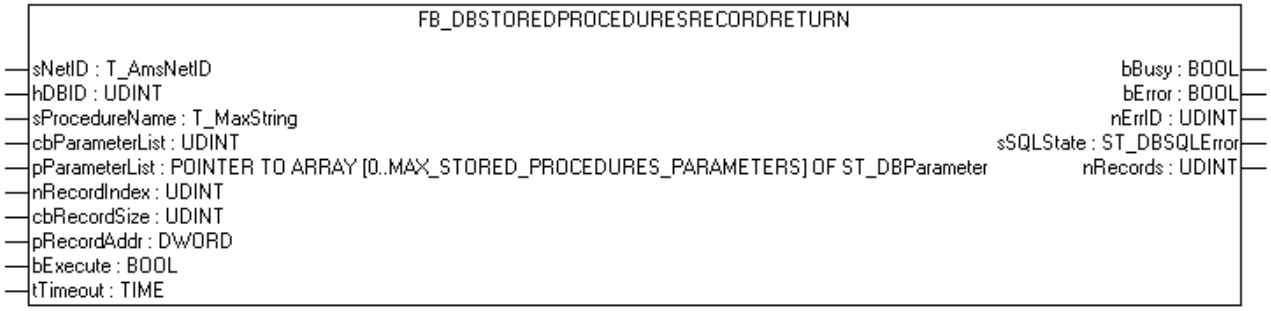
sSQLState：返回相应数据库类型的 SQL 错误代码 [▶ 307]

nRecords：返回数据记录的数量。

要求

开发环境	目标系统类型	要连接的 PLC 库
TwinCAT v3.0.0	PC 或 CX（x86）	Tc2_Database

6.2.1.20.5 FB_DBStoredProceduresRecordReturn



功能块 FB_DBStoredProceduresRecordReturn 可以用于调用返回记录的存储过程。它们可以在流程中包含在存储过程中使用的参数。

VAR_INPUT

```
VAR_INPUT
 sNetID          : T_AmsNetID     :='';
 hDBID           : UDINT          :=1;
 sProcedureName  : T_MaxString    :='';
 cbParameterList: UDINT;
 pParameterList  : POINTER TO ARRAY[0..MAX_STORED_PROCEDURES_PARAMETERS] OF ST_DBParameter;
 nRecordIndex    : UDINT;
 cbRecordSize    : UDINT;
 pRecordAddr     : POINTER TO BYTE;
 bExecute        : BOOL;
 tTimeout        : TIME           := T#15s;
END_VAR
```

sNetID：包含 ADS 命令指向的目标设备的 AMS 网络 ID 的字符串。

hDBID：表示要使用的数据库的 ID。

sProcedureName：表示要执行的过程的名称。

cbParameterList：表示参数列表的长度。

pParameterList：包含参数列表的地址

nRecordIndex：提供要读取的数据记录的索引。

cbRecordSize：提供数据记录的大小，以字节为单位。

pRecordAddr：表示要写入记录的结构的地址。

bExecute：命令在上升沿执行。

tTimeout：表示函数被取消前的时间。

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy    : BOOL;
    bError   : BOOL;
    nErrID   : UDINT;
    sSQLState: ST_DBSQLError;
    nRecords : UDINT;
END_VAR
```

bBusy：命令正在由 ADS 传输。只要 bBusy 仍为 TRUE，就不会接受任何新命令。

bError：一旦发生错误，则变为 TRUE。

nErrID：如果设置了 bError 输出，则返回 ADS 错误代码或 TcDatabaseSrv_Error_Codes [▶ 378]。

sSQLState：返回相应数据库类型的 SQL 错误代码 [▶ 307]

nRecords：返回数据记录的数量。

要求

开发环境	目标系统类型	要连接的 PLC 库
TwinCAT v3.0.0	PC 或 CX（x86）	Tc2_Database

6.2.2 数据类型

6.2.2.1 ST_DBColumnCfg

VAR_INPUT

```
TYPE ST_DBColumnCfg :
STRUCT
    sColumnName    : STRING(59);
    sColumnProperty: STRING(59);
    eColumnType    : E_DbColumnTypes;
END_STRUCT
END_TYPE
```

sColumnName：包含要创建的列的名称。

sColumnProperty：包含特定列属性。

eColumnType：提供列的类型。

要求

开发环境	目标平台	要连接的 PLC 库
TwinCAT v3.0.0	PC 或 CX（x86）	Tc2_Database

6.2.2.2 ST_DBXMLCfg

VAR_INPUT

```
TYPE ST_DBXMLCfg :
STRUCT
    sDBName : STRING;
    sDBTable: STRING;
    nDBID   : DINT;
    eDBType : E_DBTypes;
END_STRUCT
END_TYPE
```

sDBName：包含数据库的名称。

sDBTable：包含表格的名称。

nDBID：返回数据库的 ID。

eDBType：提供数据库的类型。

要求

开发环境	目标平台	要连接的 PLC 库
TwinCAT v3.0.0	PC 或 CX（x86）	Tc2_Database

6.2.2.3 ST_ADSDevXMLCfg

VAR_INPUT

```
TYPE ST_ADSDevXMLCfg :
STRUCT
    sAdsDevNetID  : T_AmsNetID;
    tAdsDevTimeout: TIME;
    nAdsDevID     : DINT;
    nAdsDevPort   : UINT;
END_STRUCT
END_TYPE
```

sAdsDevNetID：包含 ADS 设备的 AMS 网络 ID 的字符串。

tAdsDevTimeout：表示 ADS 设备的超时时间。

nAdsDevID：返回 ADS 设备的 ID。

nAdsDevPort：表示 ADS 设备的端口。

要求

开发环境	目标平台	要连接的 PLC 库
TwinCAT v3.0.0	PC 或 CX（x86）	Tc2_Database

6.2.2.4 ST_DBSQLError

VAR_INPUT

```
TYPE ST_DBSQLError :
STRUCT
    sSQLState    : STRING(5);
    nSQLErrorCode: DINT;
END_STRUCT
END_TYPE
```

sSQLState：包含 5 个字符的错误代码，该代码基于 SQL ANSI 标准。

nSQLErrorCode：返回数据库特定的错误代码。

如果没有发生错误，则该结构包含以下值：
sSQLState := **'00000'**;
nSQLErrorCode := **0**;

要求

开发环境	目标平台	要连接的 PLC 库
TwinCAT v3.0.0	PC 或 CX（x86）	Tc2_Database

6.2.2.5 ST_DBParameter

VAR_INPUT

```
TYPE ST_DBParameter :
STRUCT
    sParameterName    : STRING(59);
    cbParameterValue  : UDINT;
    pParameterValue   : UDINT;
    eParameterDataType: E_DBColumnTypes;
    eParameterType    : E_DBParameterTypes;
END_STRUCT
END_TYPE
```

sParameterName：表示参数的名称。

cbParameterValue：包含要使用的变量的大小，以字节为单位。

pParameterValue：包含要使用的变量的地址。

eParameterDataType：表示参数的数据类型（E_DBColumnTypes [▶ 308]）。

eParameterType：表示参数类型（E_DBParameterTypes [▶ 309]）。

声明示例

变量声明

```
PROGRAM MAIN
VAR
    paraList: ARRAY [0..2] OF ST_DBParameter;
    p1: DINT := 3;
    p2: LREAL;
    p3: STRING;
END_VAR
```

PLC 程序

```
paraList[0].sParameterName    := 'p1';
paraList[0].eParameterDataType:= eDBColumn_Integer;
paraList[0].eParameterType    := eDBParameter_Input;
paraList[0].cbParameterValue  := SIZEOF(p1);
paraList[0].pParameterValue   := ADR(p1);

paraList[1].sParameterName    := 'p2';
paraList[1].eParameterDataType:= eDBColumn_Float;
paraList[1].eParameterType    := eDBParameter_Output;
paraList[1].cbParameterValue  := SIZEOF(p2);
paraList[1].pParameterValue   := ADR(p1);

paraList[2].sParameterName    := 'p3';
paraList[2].eParameterDataType:= eDBColumn_NText;
paraList[2].eParameterType    := eDBParameter_Output;
paraList[2].cbParameterValue  := SIZEOF(p3);
paraList[2].pParameterValue   := ADR(p3);
```

要求

开发环境	目标平台	要连接的 PLC 库
TwinCAT v3.0.0	PC 或 CX（x86）	Tc2_Database

6.2.2.6 E_DbColumnTypes

```
TYPE E_DbColumnTypes :
(
    eDBColumn_BigInt    :=0,
    eDBColumn_Integer   :=1,
    eDBColumn_SmallInt  :=2,
    eDBColumn_TinyInt   :=3,
    eDBColumn_Bit       :=4,
    eDBColumn_Money     :=5,
    eDBColumn_Float     :=6,
    eDBColumn_Real      :=7,
    eDBColumn_DateTime  :=8,
    eDBColumn_NText     :=9,
    eDBColumn_NChar     :=10,
    eDBColumn_Image     :=11,
    eDBColumn_NVarChar  :=12,
    eDBColumn_Binary    :=13,
    eDBColumn_VarBinary :=14
);
END_TYPE
```

要求

开发环境	目标平台	要连接的 PLC 库
TwinCAT v3.0.0	PC 或 CX（x86）	Tc2_Database

6.2.2.7 E_DBTypes

```
TYPE E_DBTypes :
(
    eDBType_Mobile_Server  := 0,
    eDBType_Access         := 1,
    eDBType_Sequal_Server  := 2,
    eDBType_ASCII          := 3,
    eDBType_ODBC_MySQL     := 4,
    eDBType_ODBC_PostgreSQL:= 5,
    eDBType_ODBC_Oracle    := 6,
    eDBType_ODBC_DB2       := 7,
```

```
    eDBType_ODBC_InterBase := 8,
    eDBType_ODBC_Firebird  := 9,
    eDBType_XML            := 10,
    eDBType_OCI_Oracle     := 11
);
END_TYPE
```

要求

开发环境	目标平台	要连接的 PLC 库
TwinCAT v3.0.0	PC 或 CX（x86）	Tc2_Database

6.2.2.8 E_DBValueType

```
TYPE E_DBValueType :
(
    eDBValue_Double:= 0,
    eDBValue_Bytes := 1
);
END_TYPE
```

要求

开发环境	目标平台	要连接的 PLC 库
TwinCAT v3.0.0	PC 或 CX（x86）	Tc2_Database

6.2.2.9 E_DBWriteModes

```
TYPE E_DBWriteModes :
(
    eDBWriteMode_Update          := 0,
    eDBWriteMode_Append          := 1,
    eDBWriteMode_RingBuffer_Time := 2,
    eDBWriteMode_RingBuffer_Count:= 3
);
END_TYPE
```

要求

开发环境	目标平台	要连接的 PLC 库
TwinCAT v3.0.0	PC 或 CX（x86）	Tc2_Database

6.2.2.10 E_DBParameterTypes

```
TYPE E_DBParameterTypes :
(
    eDBParameter_Input       := 0,
    eDBParameter_Output      := 1,
    eDBParameter_InputOutput := 2,
    eDBParameter_ReturnValue := 3,
    eDBParameter_OracleCursor:= 4
);
END_TYPE
```

要求

开发环境	目标平台	要连接的 PLC 库
TwinCAT v3.0.0	PC 或 CX（x86）	Tc2_Database

6.2.3 全局常量

6.2.3.1 常量

```
VAR_GLOBAL_CONSTANT

    AMSPORT_DATABASESRV                         : UINT         := 21372;

    DBADS_IGR_RELOADXML                         : UDINT         :=16#100;

    DBADS_IGR_GETSTATE                          : UDINT         :=16#200;
```

```
    DBADS_IGR_DBCONNOPEN                            : UDINT         :=16#300;
    DBADS_IGR_DBCONNCLOSE                           : UDINT         :=16#301;
    DBADS_IGR_ADSDEVCONNOPEN                        : UDINT         :=16#302;
    DBADS_IGR_ADSDEVCONNCLOSE                       : UDINT         :=16#303;

    DBADS_IGR_DBSTOREDPROCEDURES                    : UDINT         :=16#400;
    DBADS_IGR_DBSTOREDPROCEDURES_RETURNRECORD       : UDINT         :=16#401;
    DBADS_IGR_DBSTOREDPROCEDURES_RETURNRECORDARRAY: UDINT           :=16#402;

    DBADS_IGR_START                                 : UDINT         :=16#10000;
    DBADS_IGR_STOP                                  : UDINT         :=16#20000;

    DBADS_IGR_DBCONNADD                             : UDINT         :=16#30000;
    DBADS_IGR_ADSDEVCONNADD                         : UDINT         :=16#30001;
    DBADS_IGR_ODBC_DBCONNADD                        : UDINT         :=16#30010;

    DBADS_IGR_GETDBXMLCONFIG                        : UDINT         :=16#30101;
    DBADS_IGR_GETADSDEVXMLCONFIG                    : UDINT         :=16#30102;

    DBADS_IGR_DBWRITE                               : UDINT         :=16#40000;
    DBADS_IGR_DBREAD                                : UDINT         :=16#50000;

    DBADS_IGR_DBTABLECREATE                         : UDINT         :=16#60000;
    DBADS_IGR_DBCREATE                              : UDINT         :=16#70000;

    DBADS_IGR_DBRECORDSELECT                        : UDINT         :=16#80001;
    DBADS_IGR_DBRECORDINSERT                        : UDINT         :=16#80002;
    DBADS_IGR_DBRECORDDELETE                        : UDINT         :=16#80003;

    DBADS_IGR_DBAUTHENTIFICATIONADD                 : UDINT         :=16#90000;

    MAX_DB_TABLE_COLUMNS                            : UDINT         := 255;
    MAX_XML_DECLARATIONS                            : UDINT         := 255;
    MAX_STORED_PROCEDURES_PARAMETERS                : UDINT         := 255;

END_VAR
```

要求

开发环境	目标平台	要连接的 PLC 库
TwinCAT v3.0.0	PC 或 CX（x86）	Tc2_Database

6.2.3.2 • 库版本

所有库都有特定版本。例如，版本会在 PLC 功能库中显示。全局常量包含有关库版本的信息：

Global_Version

```
VAR_GLOBAL CONSTANT
stLibVersion_TC3_Database_Server : ST_LibVersion;
END_VAR
```

使用功能 F_CmpLibVersion（在 Tc2_System 库中定义）检查您是否正在使用正确版本。

您可能从 TwinCAT 2 中了解到的用于比较库版本的所有其他选项都已过时！

7 示例

7.1 Tc3_Database

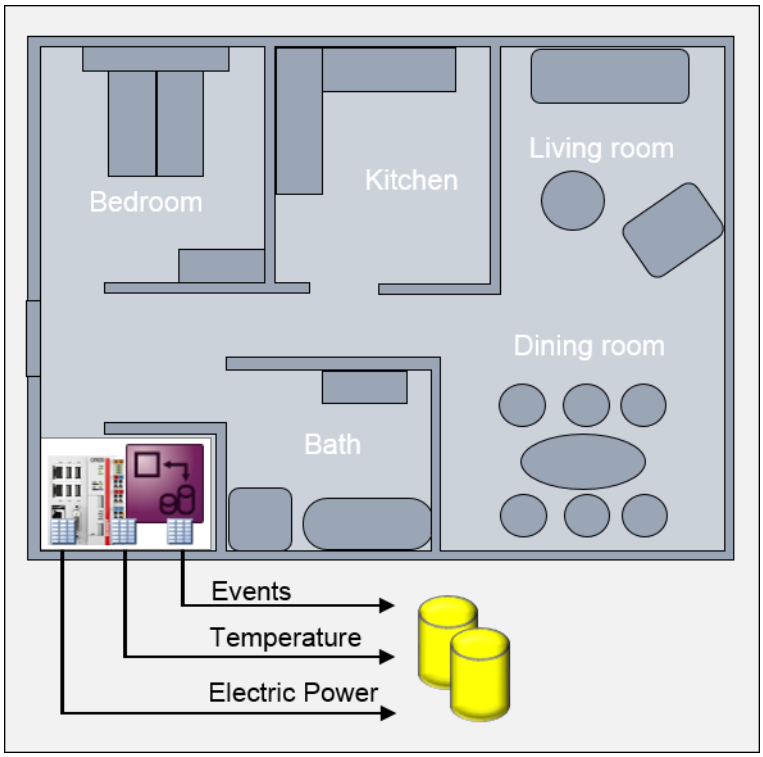
下一页面介绍了将 TwinCAT 数据库服务器与 TwinCAT 3 库一起使用的示例和技巧。

7.1.1 情景示例

每个示例都被分配到一个场景中，可根据特定的用例进行调整。此外，还提供了有关示例与哪些数据库兼容的信息。

7.1.1.1 家居自动化

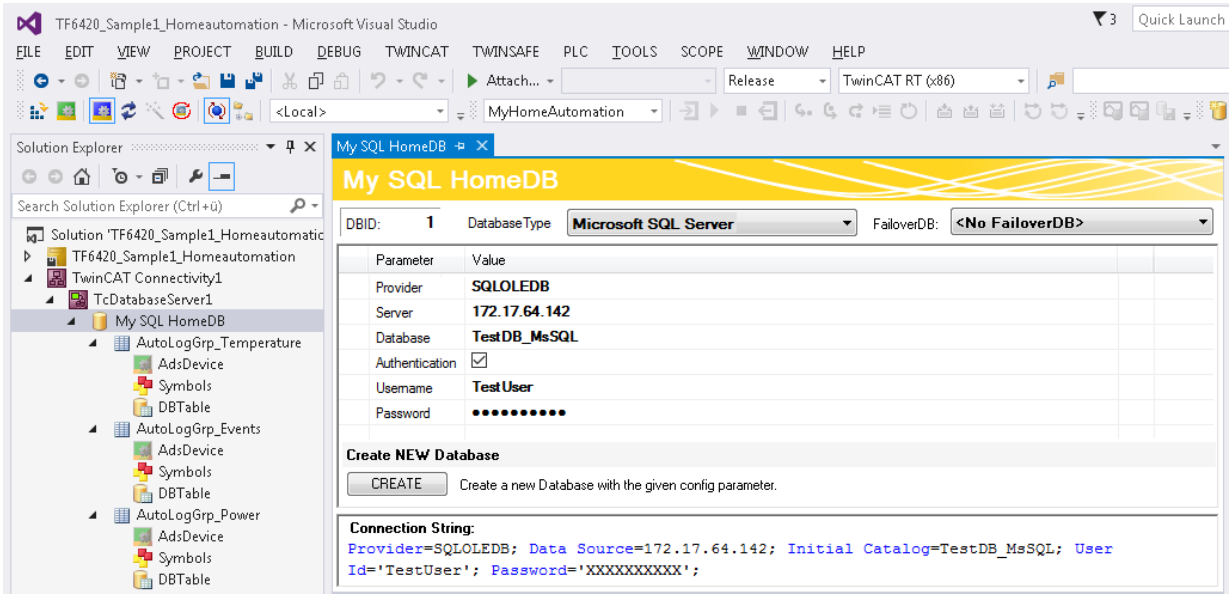
本情景示例说明了楼宇自动化的配置模式。符号被写入 MySQL 数据库的 3 个不同的 AutoLog 组，无需额外编程，即完全基于配置。在数据库中以 5 分钟为间隔记录室温。以 1 分钟为间隔保存能源数据。可存储诸如“灯亮”、“灯灭”等事件。



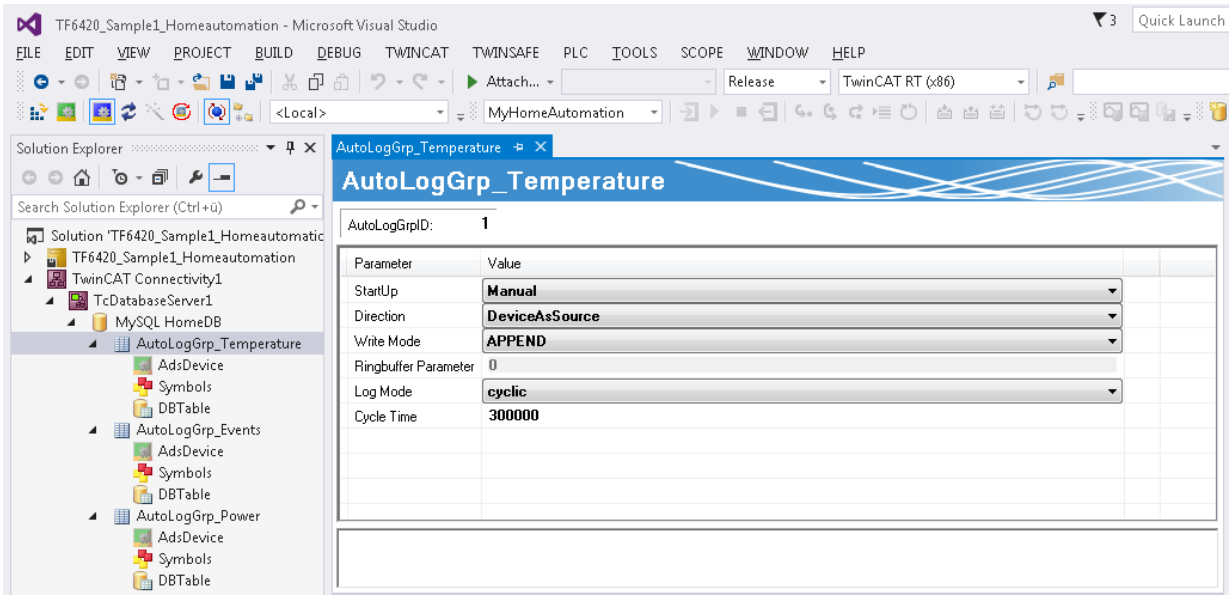
类别	配置模式
使用的数据库	MySQL
兼容的数据库	可用于所有支持的数据库类型
使用的 PLC 功能块	无
使用的 PLC 库	Tc3_Database、Tc2_BABasic
下载	https://infosys.beckhoff.com/content/1033/TF6420_TC3_Database_Server/Resources/3495185419.zip

✓ TwinCAT 数据库服务器项目已创建。

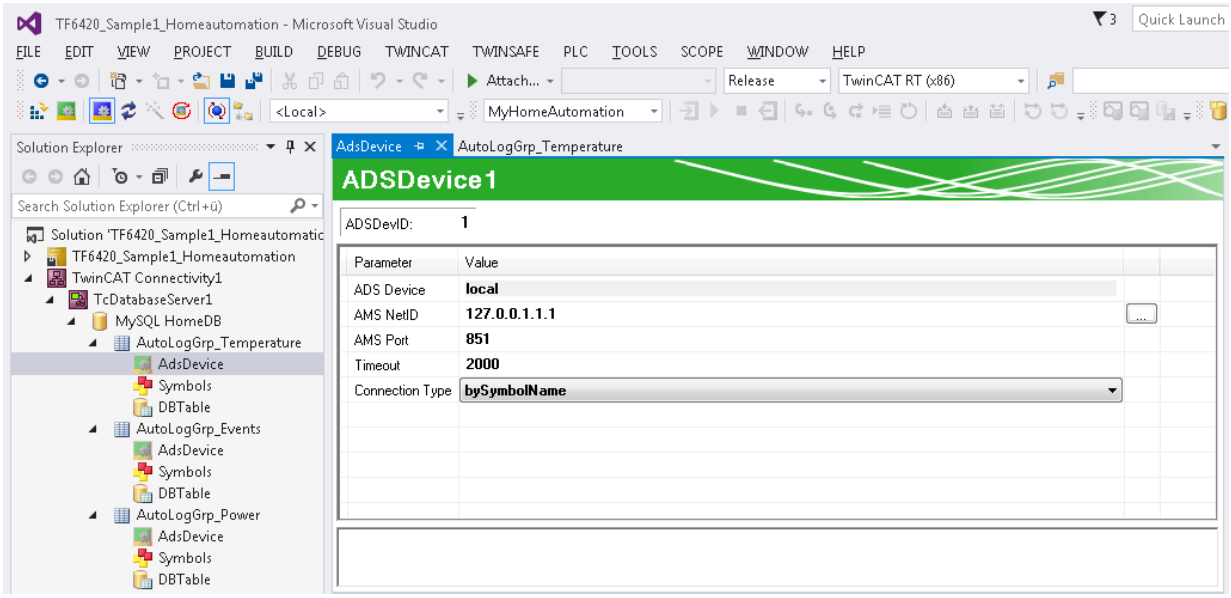
1. 添加并配置数据库连接。



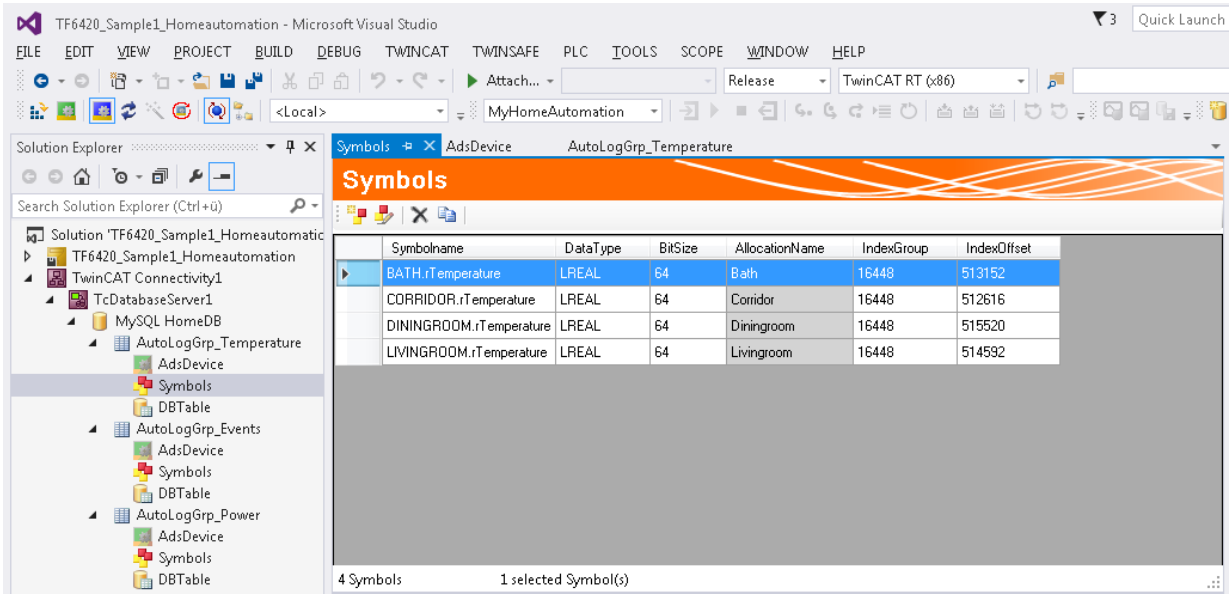
2. 由于要在 3 个不同的表格中进行日志记录，因此要将 3 个 AutoLog 组添加到在数据库连接中。首先，配置温度 AutoLog 组。CycleTime 被设置为 300000 ms，以匹配 5 分钟的数据库日志记录间隔。



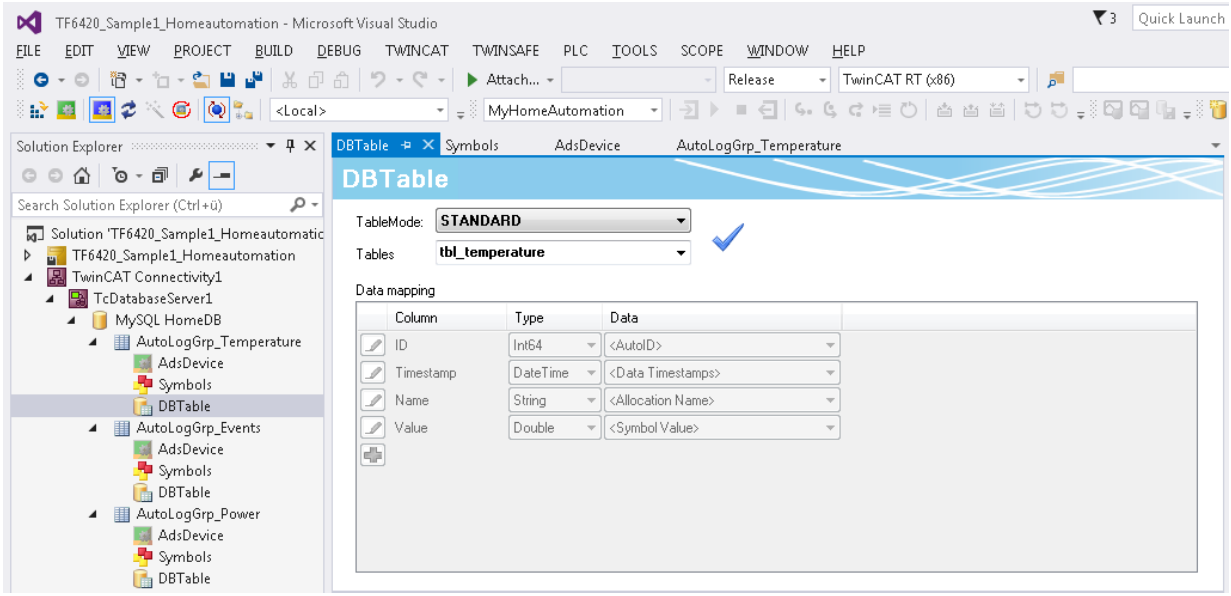
3. 然后，设置要从中读取 ADS 符号的 ADS 设备。



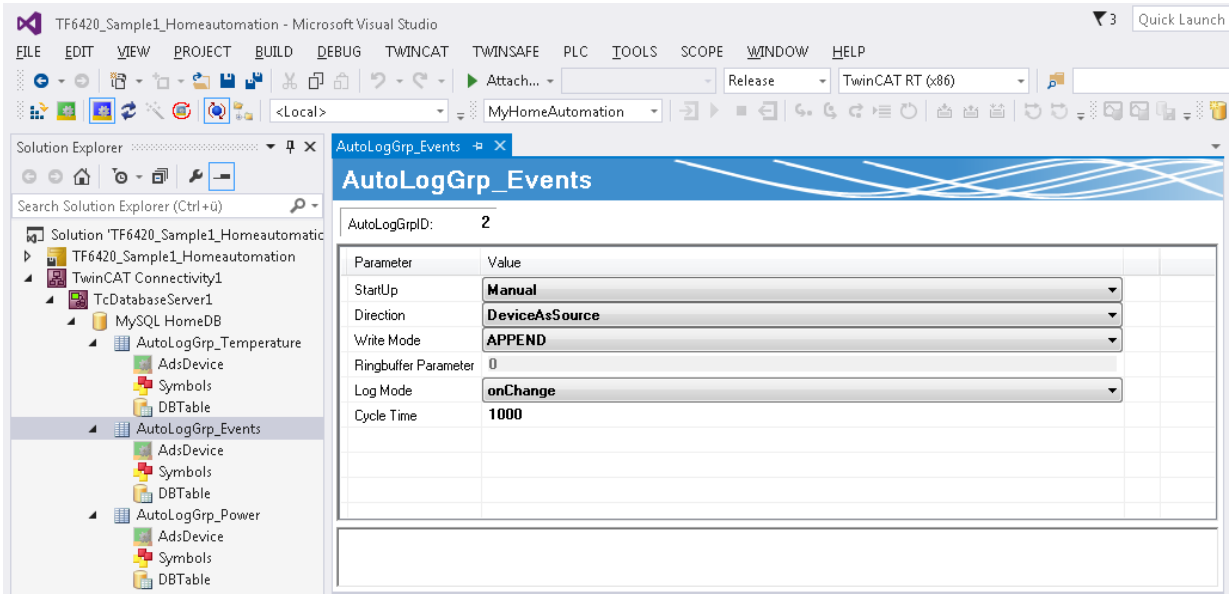
4. 通过目标浏览器创建符号。



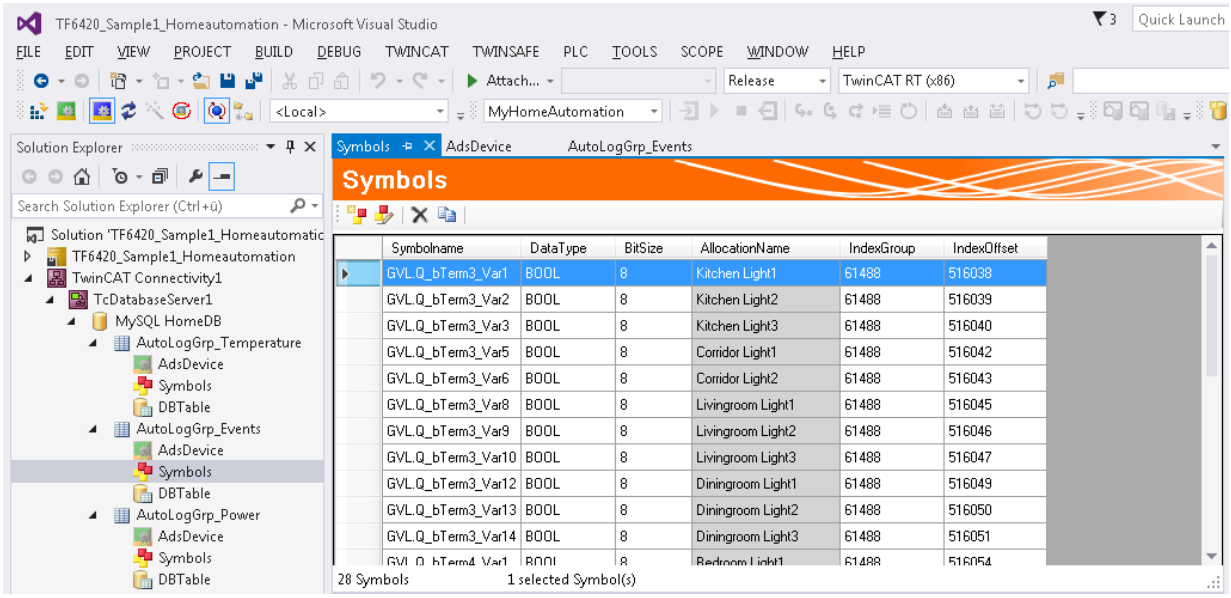
5. 现在，您可以选择要写入温度值的表格。AutoLog 组支持 2 个 TableModes [▶ 39]。标准表结构可用于温度值。勾号表示选定的表格具有正确的结构。



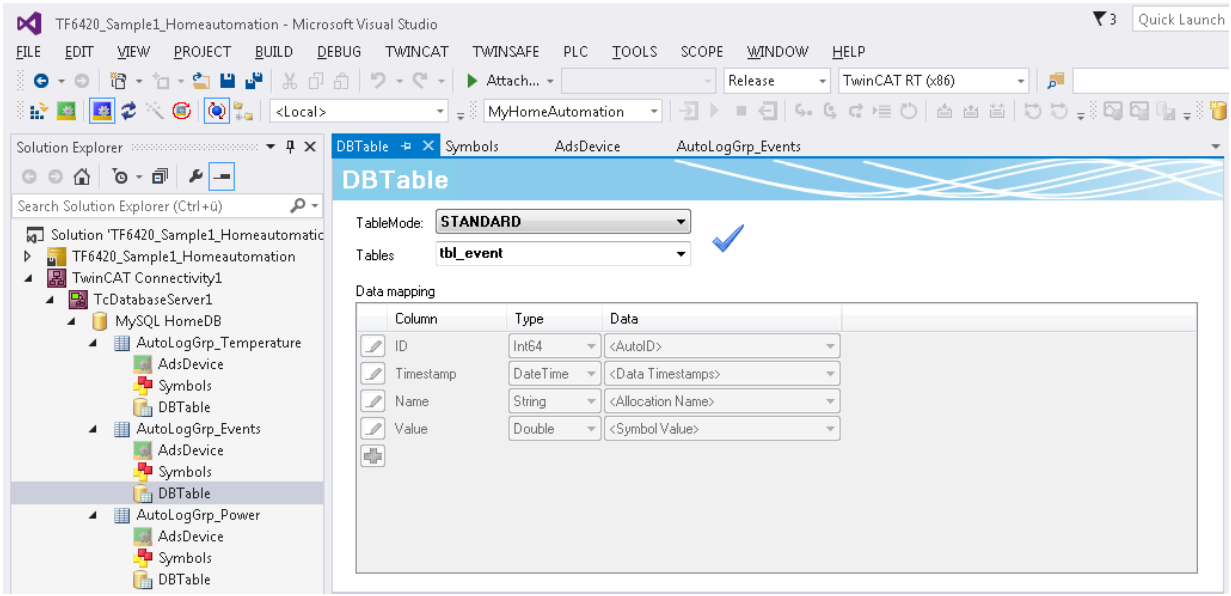
6. 然后，配置事件的 AutoLog 组。组的 LogMode 被设置为“onChange”，循环时间为 1000 ms。这意味着以 1 秒为间隔检查符号，但只有当值发生变化时才会创建表格条目。



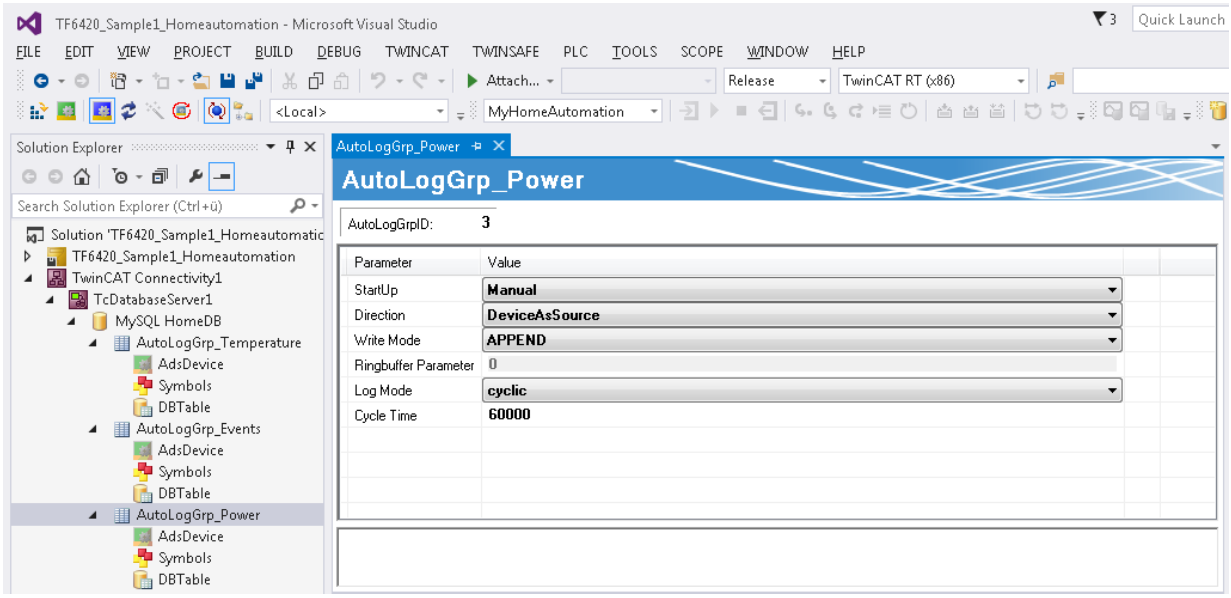
7. 然后，按照步骤 3 和 4 所述，选择 ADS 设备和符号。



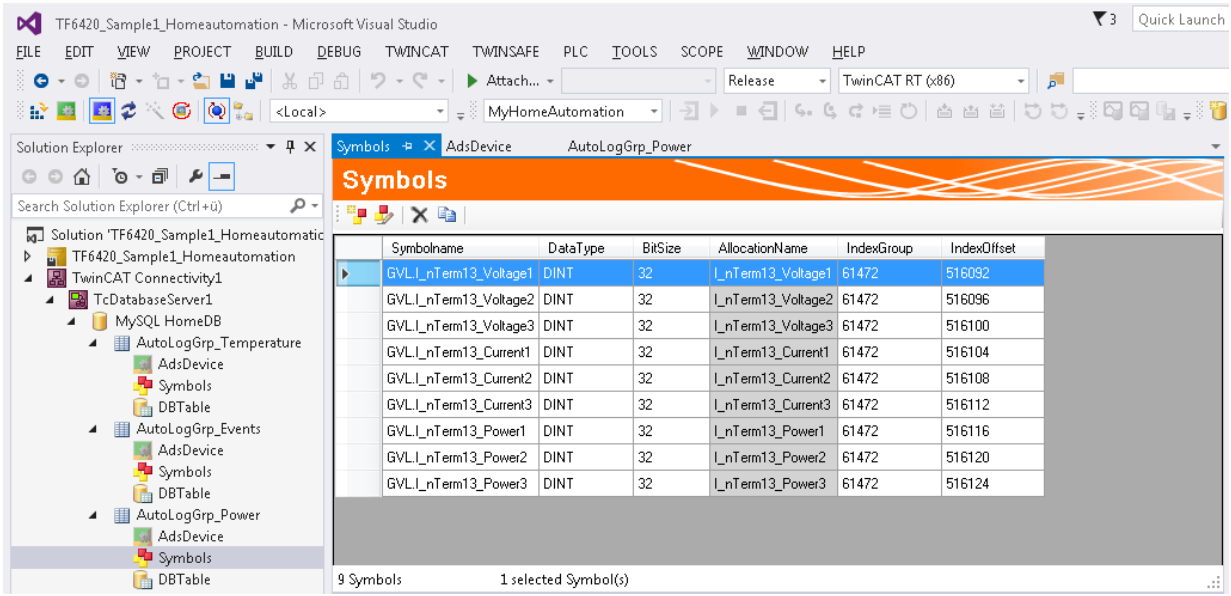
8. 事件也存储在标准表结构中。



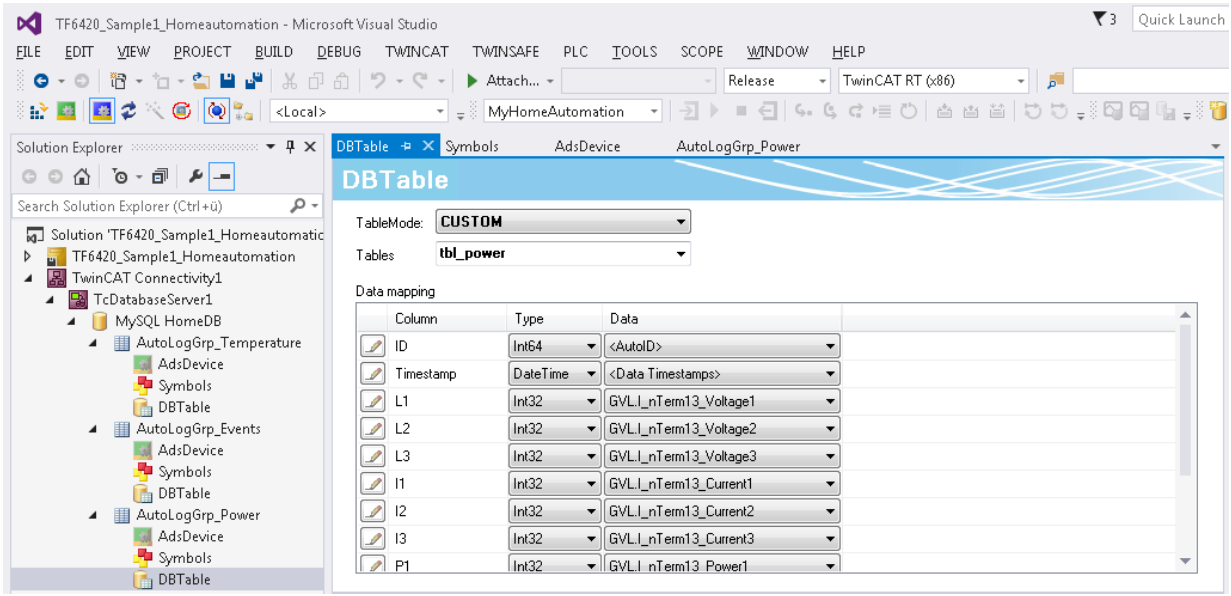
9. 最后，以 1 分钟为间隔保存能源数据。



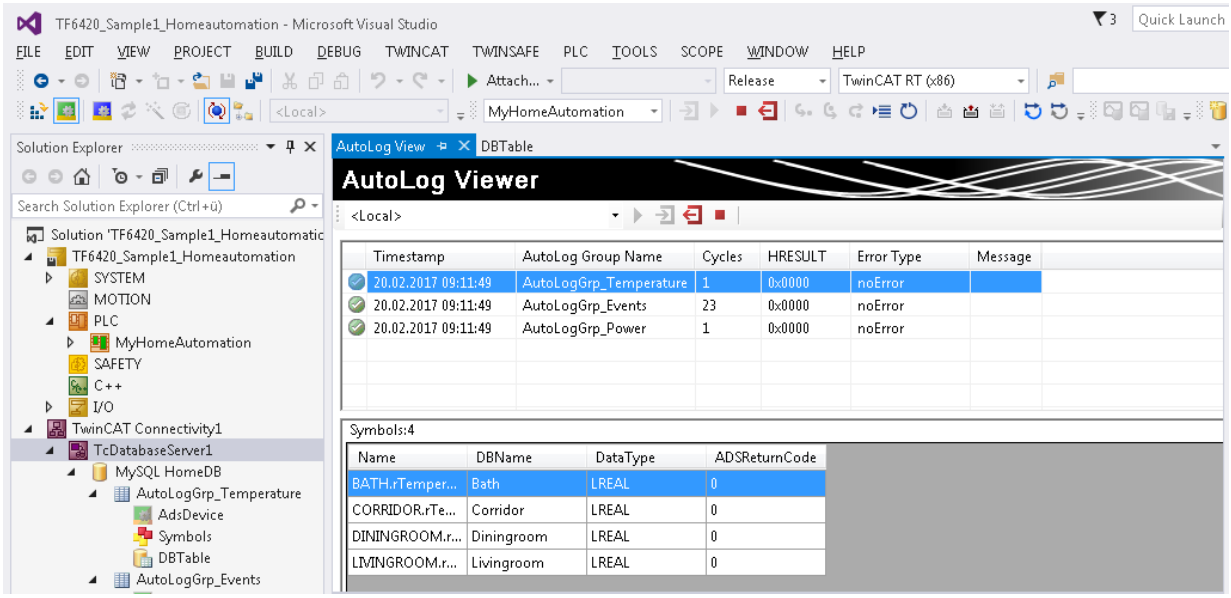
10. 对于能源数据，还可以选择 ADS 设备和符号。



11. 能源数据应存储在自定义表结构中，并为各列分配不同的符号。

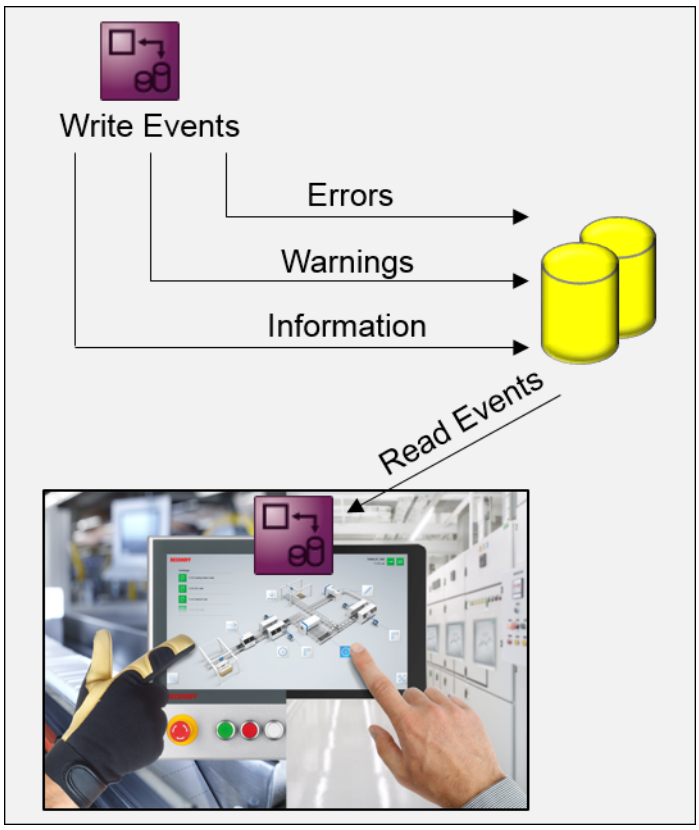


⇨ 现在可以开始日志记录，并使用 AutoLog 查看器 [▶ 47]进行监控。



7.1.1.2 消息记录器

本情景示例说明了 PLC 中消息记录器的 PLC 专家模式。在示例程序中，TwinCAT 数据库服务器的功能块可用于创建一个功能块，该功能块提供了用于生成和读取消息的各种方法。存储消息的数据库由 PLC 创建。在 MAIN 程序中实现了已创建功能块的示例应用。每 7 天创建一个新的数据库文件。可以发送 3 种不同的消息。此外，还可以调用最后一个消息或特定间隔内的所有消息。



类别	PLC 专家模式
使用的数据库	MS Compact [▶ 125]
兼容的数据库	只需稍加修改，即可用于所有支持的数据库类型
使用的 PLC 功能块	FB_PLCDBCreateEvt [▶ 166]、FB_PLCDBCmdEvt [▶ 177]、FB_PLCDBWriteEvt [▶ 173]、FB_PLCDBReadEvt [▶ 169]
使用的 PLC 库	Tc3_Database、Tc3_Eventlogger
下载	https://infosys.beckhoff.com/content/1033/TF6420_TC3_Database_Server/Resources/3495187979.zip

FB_ErrorLogger

CreateErrorLogDB（方法）

CreateErrorLogDB 方法可创建 MS Compact 数据库文件和存储消息的表格。

```
METHOD CreateErrorLogDB : BOOL
VAR_INPUT
    sDBName : T_MaxString;
END_VAR
```

```
CASE nState_CreateDB OF
    0:
        stDBConfig.sServer := CONCAT(CONCAT(sDBPath, sDBName), '.sdf');
        stDBConfig.bAuthentification := FALSE;

        nState_CreateDB := 1;
    1:
        IF fbPLCDBCreate.Database(pDatabaseConfig:= ADR(stDBConfig),
            cbDatabaseConfig:= SIZEOF(stDBConfig),
            bCreateXMLConfig:= FALSE, pDBID:= 0) THEN
            ipResultEvt := fbPLCDBCreate.ipTcResult;
            IF fbPLCDBCreate.bError THEN
                nState_CreateDB := 100;
            ELSE
                nState_CreateDB := 2;
            END_IF
        END_IF
    2:
        IF fbConfigTcDBSrv.Create(pTcDBSrvConfig:= ADR(stDBConfig),
            cbTcDBSrvConfig:= SIZEOF(stDBConfig), bTemporary:= TRUE,
```

```
            pConfigID:= ADR(nDBID) ) THEN
            ipResultEvt := fbConfigTcDBSrv.ipTcResult;
            IF fbConfigTcDBSrv.bError THEN
                nState_CreateDB := 100;
            ELSE
                nState_CreateDB := 3;
            END_IF
        END_IF
    3:
        arrTableColumns[0].sName := 'ID';
        arrTableColumns[0].eType := E_ColumnType.BigInt;
        arrTableColumns[0].nLength := 8;
        arrTableColumns[0].sProperty := 'IDENTITY(1,1)';

        arrTableColumns[1].sName := 'Timestamp';
        arrTableColumns[1].eType := E_ColumnType.DateTime;
        arrTableColumns[1].nLength := 4;

        arrTableColumns[2].sName := 'Severity';
        arrTableColumns[2].eType := E_ColumnType.NVarChar;
        arrTableColumns[2].nLength := 10;

        arrTableColumns[3].sName := 'ErrorCode';
        arrTableColumns[3].eType := E_ColumnType.Integer;
        arrTableColumns[3].nLength := 4;

        arrTableColumns[4].sName := 'Message';
        arrTableColumns[4].eType := E_ColumnType.NVarChar;
        arrTableColumns[4].nLength := 255;

        IF fbPLCDBCreate.Table(hDBID:= nDBID, sTableName:= sTableName,
            pTableCfg:= ADR(arrTableColumns),
            cbTableCfg:= SIZEOF(arrTableColumns)) THEN
            ipResultEvt := fbPLCDBCreate.ipTcResultEvent;
            nState_CreateDB := 100;
        END_IF
    100:
        IF _SetResultInfo(1033) THEN
            IF NOT bError THEN
                _bHasCreated := TRUE;
            END_IF

            nState_CreateDB := 0;
        END_IF
END_CASE

CreateErrorLogDB := nState_CreateDB = 0;
```

AddErrorEntry（方法）

AddErrorEntry 方法可用于将不同的消息写入数据库。

```
METHOD AddErrorEntry : BOOL
VAR_INPUT
    tTimestamp : DT;
    eSeverity : E_Severity;
    nErrCode : UDINT;
    sMessage : T_MaxString;
END_VAR
```

```
CASE nState_AddEntry OF
    0:
        ipResultEvt := fbPLCDBWrite.ipTcResult;

        stError.tTimestamp := tTimestamp;
        CASE eSeverity OF
            TcEventSeverity.Info:
                stError.sSeverity := 'Info';
            TcEventSeverity.Warning:
                stError.sSeverity := 'Warning';
            TcEventSeverity.Verbose:
                stError.sSeverity := 'Verbose';
            TcEventSeverity.Critical:
                stError.sSeverity := 'Critical';
            TcEventSeverity.Error:
                stError.sSeverity := 'Error';
        END_CASE
         stError.nErrCode := nErrCode;
         stError.sMsg := sMessage;

        arrColumns[0] := 'Timestamp';
        arrColumns[1] := 'ErrorCode';
        arrColumns[2] := 'Severity';
        arrColumns[3] := 'Message';

         nState_AddEntry := 1;
    1:
        IF fbPLCDBWrite.WriteStruct(
            hDBID:= nDBID,
            sTableName:= sTableName,
            pRecord:= ADR(stError),
            cbRecord:= SIZEOF(stError),
            pColumnNames:= ADR(arrColumns),
```

```
            cbColumnNames:= SIZEOF(arrColumns)) THEN

            nState_AddEntry := 100;
        END_IF
    100:
        IF _SetResultInfo(1033) THEN
            nState_AddEntry := 0;
        END_IF
END_CASE

AddErrorEntry := nState_AddEntry = 0;
```

ReadLastError（方法）

ReadLastError 方法可用于从数据库中读取最新的（最后的）条目。

```
METHOD ReadLastError : BOOL
VAR_OUTPUT
    tTimestamp : DT;
    sSeverity : STRING(10);
    nErrCode : UDINT;
    sMessage : T_MaxString;
END_VAR
```

```
CASE nState_ReadLastEntry OF
    0:
        ipResultEvt := fbPLCDBRead.ipTcResult;

        arrColumns[0] := 'Timestamp';
        arrColumns[1] := 'ErrorCode';
        arrColumns[2] := 'Severity';
        arrColumns[3] := 'Message';

         nState_ReadLastEntry := 1;
    1:
        IF fbPLCDBRead.ReadStruct(
            hDBID:= nDBID,
            sTableName:= sTableName,
            pColumnNames:= ADR(arrColumns),
            cbColumnNames:= SIZEOF(arrColumns),
            sOrderByColumn:= 'ID',
            eOrderType:= E_OrderType.DESC,
            nStartIndex:= 0,
            nRecordCount:= 1,
            pData:= ADR(stReadData),
            cbData:= SIZEOF(stReadData)) THEN

            nState_ReadLastEntry := 100;
        END_IF
    100:
        IF _SetResultInfo(1033) THEN
            IF NOT fbPLCDBRead.bError THEN
                tTimestamp := stReadData.tTimestamp;
                sSeverity := stReadData.sSeverity;
                nErrCode := stReadData.nErrCode;
                sMessage := stReadData.sMsg;
            END_IF

            nState_ReadLastEntry := 0;
        END_IF
END_CASE

ReadLastError := nState_ReadLastEntry = 0;
```

GetErrorTimerange（方法）

GetErrorTimerange 方法可用于读取特定间隔内的所有消息。

```
METHOD GetErrorTimerange : BOOL
VAR_INPUT
    tStartTimestamp : DT;
    tEndTimestamp : DT;
    nStartIndex : UDINT;
END_VAR
VAR_OUTPUT
    nErrorCount: UDINT;
    arrErrors : ARRAY [0..10] OF ST_ErrorEntry;
END_VAR
```

```
CASE nState_ErrorTimerange OF
    0:
        ipResultEvt := fbPLCDBRead.ipTcResult;

        stSearchData.dtStartTimestamp := tStartTimestamp;
        stSearchData.dtEndTimestamp := tEndTimestamp;

        sCmd := 'SELECT Timestamp, ErrorCode, Severity, Message FROM
         tbl_Errors WHERE Timestamp >= {start} AND Timestamp <= {end}';

        arrParameter[0].sParaName := 'start';
        arrParameter[0].eParaType := E_ExpParameterType.DateTime;
```

```
        arrParameter[0].nParaSize := 4;

        arrParameter[1].sParaName := 'end';
        arrParameter[1].eParaType := E_ExpParameterType.DateTime;
        arrParameter[1].nParaSize := 4;

        nState_ErrorTimerange := 1;
    1:
        IF fbPLCDBCmd.ExecuteDataReturn(
            hDBID:= nDBID,
            pExpression:= ADR(sCmd),
            cbExpression:= SIZEOF(sCmd),
            pData:= ADR(stSearchData),
            cbData:= SIZEOF(stSearchData),
            pParameter:= ADR(arrParameter),
            cbParameter:= SIZEOF(arrParameter),
            nStartIndex:= nStartIndex,
            nRecordCount:= 10,
            pReturnData:= ADR(arrErrs),
            cbReturnData:= SIZEOF(arrErrs),
            pRecords:= ADR(nErrCount)) THEN

            nState_ErrorTimerange := 100;
        END_IF
    100:
        IF _SetResultInfo(1033) THEN
            nErrorCount := nErrCount;
            arrErrors := arrErrs;

            nState_ErrorTimerange := 0;
        END_IF
END_CASE

GetErrorTimerange := nState_ErrorTimerange = 0;
```

_SetResultInfo（私有方法）

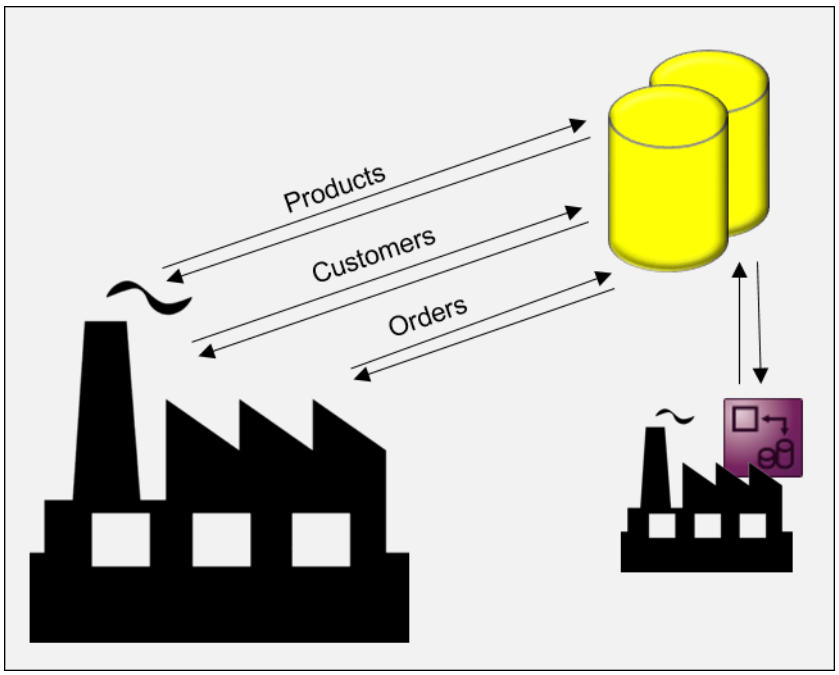
I_Message 消息接口由 TwinCAT EventLogger 在私有 _SetResultInfo 方法中进行评估。

```
METHOD _SetResultInfo : BOOL
VAR_INPUT
    nLangId : INT := 1033;
END_VAR
```

```
_SetResultInfo := FALSE;

CASE nState_SetResInfo OF
    0:
        IF ipResultEvt.RequestEventText(nLangId, EventText, SIZEOF(EventText)) THEN
            nState_SetResInfo := 1;
        END_IF
    1:
        IF ipResultEvt.RequestEventClassName(nLangId, EventClassName, SIZEOF(EventClassName)) THEN
            EventSourcePath := ipResultEvt.ipSourceInfo.sName;
            EventId := ipResultEvt.nEventId;

            bError := (ipResultEvt.eSeverity = TcEventSeverity.Error) OR
                      (ipResultEvt.eSeverity = TcEventSeverity.Critical);

            nState_SetResInfo:=0;
            _SetResultInfo := TRUE;
        END_IF
END_CASE
```

7.1.1.3 生产寄存器

本情景示例说明了使用 SQL 专家模式处理存储过程的情况。从 PLC 建立与数据库的连接。存储过程可用于从多个表格中读取产品位置。操作采用可视化方式。



类别	SQL 专家模式
使用的数据库	MS SQL [▶ 123]
兼容的数据库	MS SQL、MySQL、Oracle
使用的 PLC 功能块	FB_SQLDatabaseEvt [▶ 185]、 FB_SQLStoredProcedureEvt [▶ 194]、 FB_SQLResultEvt [▶ 191]
使用的 PLC 库	Tc3_Database、Tc3_Eventlogger
下载	https://infosys.beckhoff.com/content/1033/TF6420_TC3_Database_Server/Resources/3495190539.zip

在 MAIN 程序中实现了所谓的状态机处理，通过状态机可以控制不同的 SQL 功能块。由于功能块的方法不再具有执行标志，因此用户必须确保在下一个循环中不再调用该方法，以避免重复该过程。通过状态机可以很容易地确保这一点。

```
PROGRAM MAIN
VAR
    bCONNECT: BOOL;
    bEXECUTE: BOOL;
    bREAD : BOOL;
    bDISCONNECT: BOOL;

    R_TRIG1: R_TRIG;
    R_TRIG2: R_TRIG;
    R_TRIG3: R_TRIG;
    R_TRIG4: R_TRIG;

    nState: INT;
    nState_Connect: INT;
    nState_Disconnect: INT;

    bConn: BOOL;
    bSP: BOOL;
    bResult: BOOL;
    bData: BOOL;

    nDBID: UDINT := 1;

    fbSQLDatabase: FB_SQLDatabaseEvt(sNetID:='', tTimeout:=T#10S);
    fbSQLStoredProcedure: FB_SQLStoredProcedureEvt(
                        sNetID:='', tTimeout:=T#10S);
    fbSQLResult: FB_SQLResultEvt(sNetID:='', tTimeout:=T#10S);

    arrParameter: ARRAY [0..0] OF ST_SQLSPParameter;

    nCustomerID: DINT := 12345;

    nRecordStartIndex: UDINT;
    stRecordArr: ARRAY [1..20] OF ST_Record;
    nRecs: UDINT;
```

```
    ipResultEvt : Tc3_Eventlogger.I_TcMessage;
    bError : BOOL;
    nEventID: UDINT;
    sEventClass : STRING(255);
    sEventMsg : STRING(255);
END_VAR
```

```
R_TRIG1(CLK:=bCONNECT);
IF R_TRIG1.Q AND nState = 0 THEN
    nState := 1;
END_IF

R_TRIG2(CLK:=bEXECUTE);
IF R_TRIG2.Q AND nState = 0 THEN
    nState := 2;
END_IF

R_TRIG3(CLK:=bREAD);
IF R_TRIG3.Q AND nState = 0 THEN
    nState := 3;
END_IF

R_TRIG4(CLK:=bDISCONNECT);
IF R_TRIG4.Q THEN
    nState := 4;
END_IF
```

```
CASE nState OF
0:(*Idle*)
    ;
1: // Connect to database and create stored procedure instance
    CASE nState_Connect OF
        0:
            IF fbSQLDatabase.Connect(hDBID:= nDBID) THEN
                ipResultEvt := fbSQLDatabase.ipTcResult;
                bConn := NOT fbSQLDatabase.bError;
                IF bConn THEN
                    nState_Connect := 1;
                ELSE
                    nState:=200;
                END_IF
            END_IF
        1:
            arrParameter[0].sParameterName := '@Customer_ID';
            arrParameter[0].eParameterDataType :=
                          Tc3_Database.E_ColumnType.Integer;
            arrParameter[0].eParameterType := E_SPParameterType.Input;
            arrParameter[0].nParameterSize := SIZEOF(nCustomerID);

            IF fbSQLDatabase.CreateSP('SP_GetAddressByCustomerID',
                        ADR(arrParameter), SIZEOF(arrParameter),
                       ADR(fbSQLStoredProcedure)) THEN
                ipResultEvt:= fbSQLDatabase.ipTcResult;
                bSP := NOT fbSQLDatabase.bError;
                nState_Connect:=0;
                nState := 200;
            END_IF
    END_CASE
2: // Execute stored procedure
    IF fbSQLStoredProcedure.ExecuteDataReturn(
                         pParameterStrc:= ADR(nCustomerID),
                         cbParameterStrc:= SIZEOF(nCustomerID),
                         pSQLDBResult:= ADR(fbSQLResult)) THEN
        ipResultEvt:= fbSQLStoredProcedure.ipTcResult;
        MEMSET(ADR(stRecordArr),0,SIZEOF(stRecordArr));
        bResult := NOT fbSQLStoredProcedure.bError;
        nState := 200;
    END_IF
3:    // Read customer positions
    IF fbSQLResult.Read(nRecordStartIndex, 20, ADR(stRecordArr),
                    SIZEOF(stRecordArr), TRUE, FALSE) THEN
        ipResultEvt:= fbSQLResult.ipTcResult;
        bData := NOT fbSQLStoredProcedure.bError;
        nRecs := fbSQLResult.nDataCount;
        nState := 200;
    END_IF
4:// Disconnect all
    CASE nState_Disconnect OF
        0:
            IF bData THEN
                IF fbSQLResult.Release() THEN
                    nState_Disconnect := 1;
                END_IF
            ELSE
                nState_Disconnect := 1;
            END_IF
        1:
            IF bSP THEN
                IF fbSQLStoredProcedure.Release() THEN
                    nState_Disconnect := 2;
                END_IF
            ELSE
                nState_Disconnect := 2;
            END_IF
```

```
        2:
            IF bConn THEN
                IF fbSQLDatabase.Disconnect() THEN
                    nState_Disconnect := 3;
                END_IF
            ELSE
                nState_Disconnect := 3;
            END_IF
        3:
            bData := FALSE;
            bSP := FALSE;
            bConn := FALSE;
            bResult := FALSE;
            sEventClass := "";
            sEventMsg := "";
            nEventID := 0;
            bError := FALSE;
            nState_Disconnect := 0;
            nState := 0;
    END_CASE
200:
    IF ipResultEvt.RequestEventText(1033, sEventMsg, SIZEOF(sEventMsg)) THEN
        nState := 201;
    END_IF
201:
    IF ipResultEvt.RequestEventClassName(1033, sEventClass, SIZEOF(sEventClass)) THEN

        nEventID := ipResultEvt.nEventId;

        bError := (ipResultEvt.eSeverity = TcEventSeverity.Error) OR
                  (ipResultEvt.eSeverity = TcEventSeverity.Critical);

        nState:=0;
    END_IF
END_CASE
```

各个过程步骤可在各个 PLC 状态下重现。提供布尔标志，以方便处理。

1. bConnect：建立与数据库的连接
2. bExecute：执行存储过程，并将结果加载到缓存中
3. bRead：将结果传输至 PLC
4. bDisconnect：关闭连接

如果连续执行这些步骤，则会使用数据库中的值填充数组 stRecordArr：

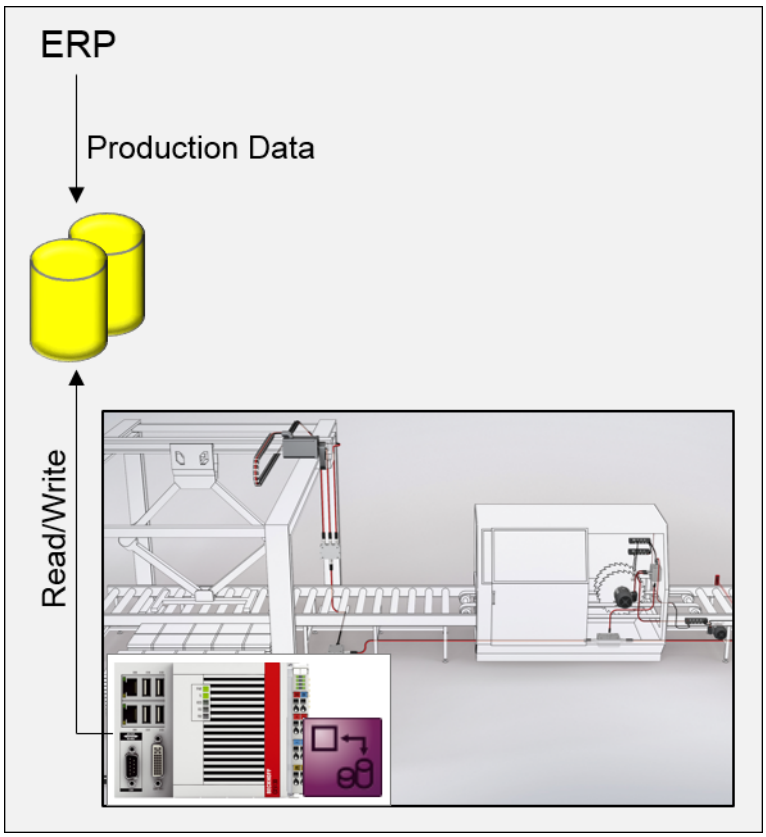
MAIN [Online] | MsSQL Customer DB | Sql Query Editor

TF6420_Sample3_CustomerProducts.CustomerProducts.MAIN

Expression	Type	Value	Prepared value
bCONNECT	BOOL	FALSE	TRUE
bEXECUTE	BOOL	FALSE	
bREAD	BOOL	FALSE	
bDISCONNECT	BOOL	FALSE	
stRecordArr	ARRAY [1..20] OF S...		
stRecordArr[1]	ST_Record		
nID	ULINT	12345	
sCustomer	STRING(50)	'Johann'	
sName	STRING(50)	'Groß'	
nProductNum	DINT	1	
sProductName	STRING(50)	'CX1000'	
sProductInfo	T_MaxString	'E'	
tTimestamp	DATE_AND_TIME	DT#2010-10-10-...	
stRecordArr[2]	ST_Record		
stRecordArr[3]	ST_Record		

7.1.1.4 生产配方

本情景示例说明了 TwinCAT 数据库服务器如何处理任何结构的 XML 文件。从 XML 文件中可读取用于构建产品的生产配方。从不同的文件中可读取相应的测试参数。此外，测试结果会被写入现有的 XML 文件。



类别	SQL 专家模式
使用的数据库	XML [▶ 132]（作为免费的 XML 文档）
兼容的数据库	XML
使用的 PLC 功能块	FB_PLCDBCmdEvt [▶ 177]
使用的 PLC 库	Tc3_Database、Tc3_Eventlogger
下载	https://infosys.beckhoff.com/content/1033/TF6420_TC3_Database_Server/Resources/3495193099.zip

配方 XML：

```
<ProductionConfig>
    <Config TypeNum="12345" Test="985-524">
        <rLength>65.85</rLength>
        <rWidth>30</rWidth>
        <rHeight>2.5</rHeight>
        <iQuantity>500</iQuantity>
        <iCounter>0</iCounter>
    </Config>
    <Config TypeNum="23456" Test="985-524">
        <rLength>15.85</rLength>
        <rWidth>300</rWidth>
        <rHeight>12.5</rHeight>
        <iQuantity>1200</iQuantity>
        <iCounter>0</iCounter>
    </Config>
    <Config TypeNum="34567" Test="125-594">
        <rLength>195.85</rLength>
        <rWidth>378</rWidth>
        <rHeight>145.5</rHeight>
        <iQuantity>10</iQuantity>
        <iCounter>0</iCounter>
    </Config>
</ProductionConfig>
```

测试 XML：

```
<ProductionConfig>
    <TestParameter>
        <Test Num="985-524">
            <MaxTemp>55.6</MaxTemp>
            <MinTemp>25.6</MinTemp>
            <MaxPSI>5500</MaxPSI>
        </Test>
        <Test Num="695-784">
            <MaxTemp>85.6</MaxTemp>
            <MinTemp>20.2</MinTemp>
            <MaxPSI>1300</MaxPSI>
        </Test>
        <Test Num="125-594">
            <MaxTemp>25.9</MaxTemp>
            <MinTemp>12.0</MinTemp>
            <MaxPSI>500</MaxPSI>
        </Test>
    </TestParameter>
    <Tests>
        <Test TestNum="985-524" TypeNum="12345" Timestamp="2016-09-24-06:00" Tester="Mustermann" Result="OK" />
    </Tests>
</ProductionConfig>
```

FB_ProductionConfigData

GetConfig（方法）

该方法可从 XML 文件中读取产品的生产配方。XPath 查询可以用于查找所需的配方。

```
METHOD GetConfig : BOOL
VAR_INPUT
    nTypeNum : DINT;
END_VAR
VAR_OUTPUT
    stConfig : ST_Config;
END_VAR
```

```
GetConfig:= FALSE;

arrPara[0].sParaName := 'rLength';
arrPara[0].eParaType := Tc3_Database.E_ExpParameterType.Float32;
arrPara[0].nParaSize := 4;

arrPara[1].sParaName := 'rWidth';
arrPara[1].eParaType := Tc3_Database.E_ExpParameterType.Float32;
arrPara[1].nParaSize := 4;

arrPara[2].sParaName := 'rHeight';
arrPara[2].eParaType := Tc3_Database.E_ExpParameterType.Float32;
```

```
arrPara[2].nParaSize := 4;

arrPara[3].sParaName := 'iQuantity';
arrPara[3].eParaType := Tc3_Database.E_ExpParameterType.Int32;
arrPara[3].nParaSize := 4;

arrPara[4].sParaName := 'iCounter';
arrPara[4].eParaType := Tc3_Database.E_ExpParameterType.Int32;
arrPara[4].nParaSize := 4;

sCmd := CONCAT(CONCAT('XPATH_SEL<SUBTAG>#ProductionConfig/Config[@TypeNum
                    = ', DINT_TO_STRING(nTypeNum)), ']');

CASE nState_GetConfig OF
0:
    IF fbPLCDBCmd.ExecuteDataReturn(
        hDBID:= 1,
        pExpression:= ADR(sCmd),
        cbExpression:= SIZEOF(sCmd),
        pData:= 0,
        cbData:= 0,
        pParameter:= ADR(arrPara),
        cbParameter:= SIZEOF(arrPara[0])*5,
        nStartIndex:= 0,
        nRecordCount:= 1,
        pReturnData:= ADR(_stConfig),
        cbReturnData:= SIZEOF(_stConfig),
        pRecords:= 0) THEN

        ipResultEvt := fbPLCDBCmd.ipTcResult;
        nState_GetConfig := 100;
    END_IF
100:
    IF _SetResultInfo(1033) THEN
        GetConfig := TRUE;

        stConfig := _stConfig;

        nState_GetConfig := 0;
    END_IF
END_CASE
```

GetTestParameter（方法）

该方法可读取产品特定的测试参数。

```
METHOD GetTestParameter : BOOL
VAR_INPUT
    nTypeNum : DINT;
END_VAR
VAR_OUTPUT
    sTestNum : STRING(8);
    stTestPara: ST_TestParameter;
END_VAR
```

```
GetTestParameter := FALSE;

CASE nState_GetTestPara OF
0:
    arrPara[0].sParaName := 'Test';
    arrPara[0].eParaType := Tc3_Database.E_ExpParameterType.STRING_;
    arrPara[0].nParaSize := 8;

    sCmd := CONCAT(CONCAT('XPATH_SEL<ATTR>#ProductionConfig/Config
                    [@TypeNum = ', DINT_TO_STRING(nTypeNum)), ']');

    IF fbPLCDBCmd.ExecuteDataReturn(
        hDBID:= 1,
        pExpression:= ADR(sCmd),
        cbExpression:= SIZEOF(sCmd),
        pData:= 0,
        cbData:= 0,
        pParameter:= ADR(arrPara),
        cbParameter:= SIZEOF(arrPara[0]),
        nStartIndex:= 0,
        nRecordCount:= 1,
        pReturnData:= ADR(_sTestNum),
        cbReturnData:= SIZEOF(_sTestNum),
        pRecords:= 0) THEN

        bError := fbPLCDBCmd.bError;
        sErrClass := fbPLCDBCmd.ipTcResultEvent.EventClassDisplayName;
        nErrID := fbPLCDBCmd.ipTcResultEvent.EventId;
        sErrText := fbPLCDBCmd.ipTcResultEvent.Text;

        IF fbPLCDBCmd .bError THEN
            ipResultEvt := fbPLCDBCmd.ipTcResult;
            nState_GetTestPara:= 100;
        ELSE
            nState_GetTestPara:= 1;
        END_IF
    END_IF
1:
```

```
    arrPara[0].sParaName := 'MaxTemp';
    arrPara[0].eParaType := Tc3_Database.E_ExpParameterType.Float32;
    arrPara[0].nParaSize := 4;

    arrPara[1].sParaName := 'MinTemp';
    arrPara[1].eParaType := Tc3_Database.E_ExpParameterType.Float32;
    arrPara[1].nParaSize := 4;

    arrPara[2].sParaName := 'MaxPSI';
    arrPara[2].eParaType := Tc3_Database.E_ExpParameterType.Int32;
    arrPara[2].nParaSize := 4;

    sCmd := CONCAT(CONCAT('XPATH_SEL<SUBTAG>#ProductionConfig/
                TestParameter/Test[@Num = $'', _sTestNum), '$']');

    IF fbPLCDBCmd.ExecuteDataReturn(
        hDBID:= 2,
        pExpression:= ADR(sCmd),
        cbExpression:= SIZEOF(sCmd),
        pData:= 0,
        cbData:= 0,
        pParameter:= ADR(arrPara),
        cbParameter:= SIZEOF(arrPara[0])*3,
        nStartIndex:= 0,
        nRecordCount:= 1,
        pReturnData:= ADR(_stTest),
        cbReturnData:= SIZEOF(_stTest),
        pRecords:= 0) THEN

        ipResultEvt := fbPLCDBCmd.ipTcResult;
        nState_GetTestPara:= 100;
100:
    IF _SetResultInfo(1033) THEN
        nState_GetTestPara := 0;

        stTestPara := _stTest;
        sTestNum := _sTestNum;

        GetTestParameter := TRUE;
    END_IF
END_CASE
```

AddTestEntry（方法）

该方法可将测试结果添加到测试 XML 文件中。

```
METHOD AddTestEntry : BOOL
VAR_INPUT
    sTestNum : STRING(8);
    nTypeNum : DINT;
    sTimestamp : STRING;
    sTester : STRING;
    sResult : STRING;
END_VAR
```

```
AddTestEntry := FALSE;

arrPara[0].sParaName := 'TestNum';
arrPara[0].eParaType := Tc3_Database.E_ExpParameterType.STRING_;
arrPara[0].nParaSize := 8;

arrPara[1].sParaName := 'TypeNum';
arrPara[1].eParaType := Tc3_Database.E_ExpParameterType.Int32;
arrPara[1].nParaSize := 4;

arrPara[2].sParaName := 'Timestamp';
arrPara[2].eParaType := Tc3_Database.E_ExpParameterType.STRING_;
arrPara[2].nParaSize := 81;

arrPara[3].sParaName := 'Tester';
arrPara[3].eParaType := Tc3_Database.E_ExpParameterType.STRING_;
arrPara[3].nParaSize := 81;

arrPara[4].sParaName := 'Result';
arrPara[4].eParaType := Tc3_Database.E_ExpParameterType.STRING_;
arrPara[4].nParaSize := 81;

arrPara[5].sParaName := 'Test';
arrPara[5].eParaType := Tc3_Database.E_ExpParameterType.XMLTAGName;
arrPara[5].nParaSize := 0;

sCmd := 'XPATH_ADD<ATTR>#ProductionConfig/Tests';

stTest.sTestNum := sTestNum;
stTest.nTypeNum := nTypeNum;
stTest.sTimestamp := sTimestamp;
stTest.sTester := sTester;
stTest.sResult := sResult;

CASE nState_AddEntry OF
0:
    IF fbPLCDBCmd.Execute(
        hDBID:= 2,
```

```
        pExpression:= ADR(sCmd),
        cbExpression:= SIZEOF(sCmd),
        pData:= ADR(stTest),
        cbData:= SIZEOF(stTest),
        pParameter:= ADR(arrPara),
        cbParameter:= SIZEOF(arrPara)) THEN

        ipResultEvt := fbPLCDBCmd.ipTcResult;
        nState_AddEntry:= 100;
    END_IF
100:
    IF _SetResultInfo(1033) THEN
        nState_AddEntry:= 0;
        AddTestEntry:= TRUE;
    END_IF
END_CASE
```

_SetResultInfo（私有方法）

I_Message 消息接口由 TwinCAT EventLogger 在私有 _SetResultInfo 方法中进行评估。

```
METHOD _SetResultInfo : BOOL
VAR_INPUT
    nLangId : INT := 1033;
END_VAR
```

```
_SetResultInfo := FALSE;

CASE nState_SetResInfo OF
    0:
        IF ipResultEvt.RequestEventText(nLangId, EventText, SIZEOF(EventText)) THEN
            nState_SetResInfo := 1;
        END_IF
    1:
        IF ipResultEvt.RequestEventClassName(nLangId, EventClassName, SIZEOF(EventClassName)) THEN
            EventId := ipResultEvt.nEventId;

            bError := (ipResultEvt.eSeverity = TcEventSeverity.Error) OR
                      (ipResultEvt.eSeverity = TcEventSeverity.Critical);

            nState_SetResInfo:=0;
            _SetResultInfo := TRUE;
        END_IF
END_CASE
```

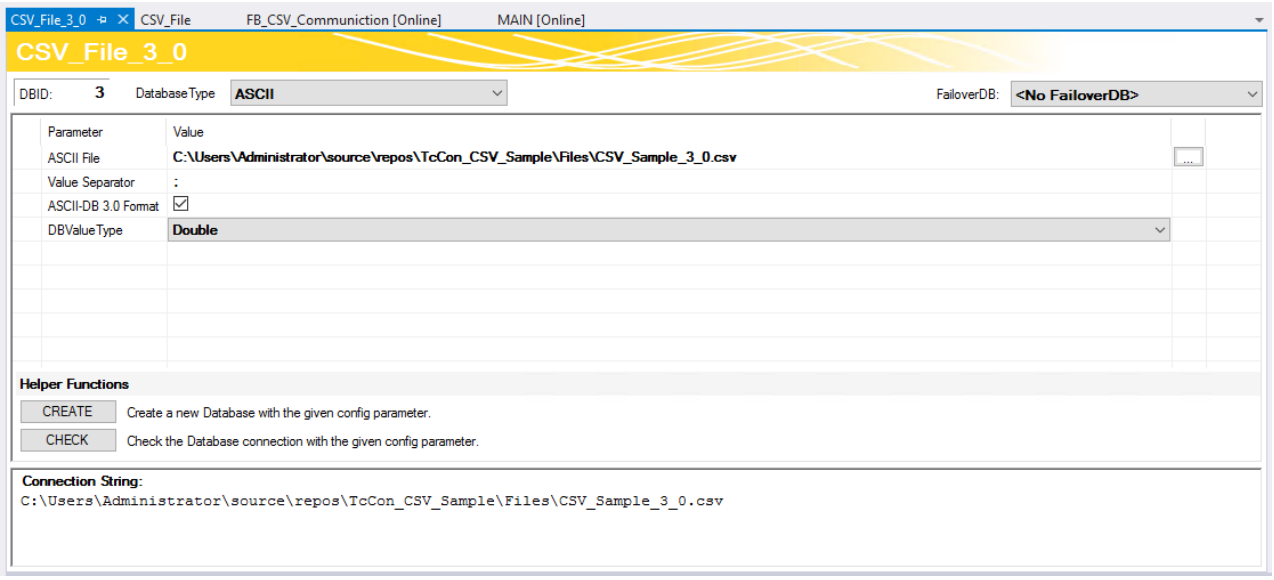
7.1.2 最佳实践

使用 TwinCAT 数据库服务器的技巧说明了各个功能块和应用程序在性能、灵活性和复杂性方面的优势。

7.1.2.1 写入 CSV 文件

TwinCAT 3 数据库服务器支持 CSV 文件格式。将内容写入文件或从文件中读取内容有不同的方法，每种方法都各有利弊。在这里将对其中 2 种方法进行更加详细的解释。

选择 ASCII 数据库。在文件路径下可以指定 .csv 文件格式。ASCII-DB 3.0 格式标志表示 ASCII/CSV 文件的格式。如果检查了格式，则使用 SAX 过程。在该设置下，对文件进行写入访问（特别是使用 FB_PLCDBCmdEvt 功能块）对于大文件也非常有效。如果未检查格式，则使用 DOM 过程，该过程特别适合读取文件。数据以结构化形式存储在 RAM 中。因此，对于较小的文件（小于 1 MB），建议使用该方法。然而，由于结构化存储，该方法具有一些优势。CSV 文件可以用作使用存储的表结构的 SQL 数据库。使用 SQL 查询编辑器执行此操作。通过“Create”（创建）按钮可以直接创建该文件。



将您的配置加载到 TwinCAT 数据库服务器目标系统上。

表 2: ASCII 格式兼容性

功能块	表结构	ASCII 3.0 格式	标准 ASCII
FB_PLCDBWriteEvt.Write	标准	✓	✓
FB_PLCDBWriteEvt.WriteStruct*	任意	✗	✓
FB_PLCDBReadEvt.Read	标准	✓	✓
FB_:PLCDBReadEvt.ReadStruct*	任意	✗	✓
FB_PLCDBCmdEvt.Execute*	任意	✓	✗
FB_SQLCommandEvt	任意	✗	✓

标有 * 的项目可用于以下示例

高性能写入 CSV 文件

写入 CSV 文件的最有效方法是基于功能块 FB_PLCDBCmdEvt。为此，CSV 文件的链接必须被设置为 ASCII-DB 3.0 格式。DBValueType 在这里不相关。不必预先定义表结构。

示例：

以下面的结构为例：

```
TYPE ST_CSVDataStruct :
STRUCT
    ID: LINT;
    Timestamp: DT;
    Name: STRING(80);
    Velocity: LREAL;
    Temperature: LREAL;
END_STRUCT
END_TYPE
```

功能块的初始化如下：

```
VAR
    InputData: ST_CSVDataStruct;
    fbPLCDBCmd: FB_PLCDBCmd (sNetID:= '', tTimeout := T#30S);

    sCmd : T_MaxString := '{ID};{Timestamp};{Name};{Velocity};{Temperature}';

    para : ARRAY [0..4] OF ST_ExpParameter :=[
            (eParaType:= E_ExpParameterType.Int64, nParaSize := 8, sParaName := 'ID'),
            (eParaType:= E_ExpParameterType.DateTime, nParaSize := 4, sParaName := 'Timestamp'),
```

```
            (eParaType:= E_ExpParameterType.STRING_, nParaSize := 81, sParaName := 'Name'),
            (eParaType:= E_ExpParameterType.Double64, nParaSize := 8, sParaName := 'Velocity'),
            (eParaType:= E_ExpParameterType.Double64, nParaSize := 8, sParaName := 'Temperature')];
END_VAR
```

在命令中的大括号内指定各个参数。通过初始化分配有关类型、字节长度和名称的信息。名称可用于识别命令中的参数，并在将其写入文件时用 PLC 中的值替换。

功能块的 PLC 源代码中的调用包含一个调用：

```
IF fbPLCDBCmd.Execute(
    hDBID:= 3,
    pExpression:= ADR(sCmd),
    cbExpression:= SIZEOF(sCmd),
    pData:= ADR(InputData),
    cbData:= SIZEOF(InputData) ,
    pParameter:= ADR(para),
    cbParameter:=SIZEOF(para))
THEN
    ;//Place for errorhandling or reactions;
END_IF
```

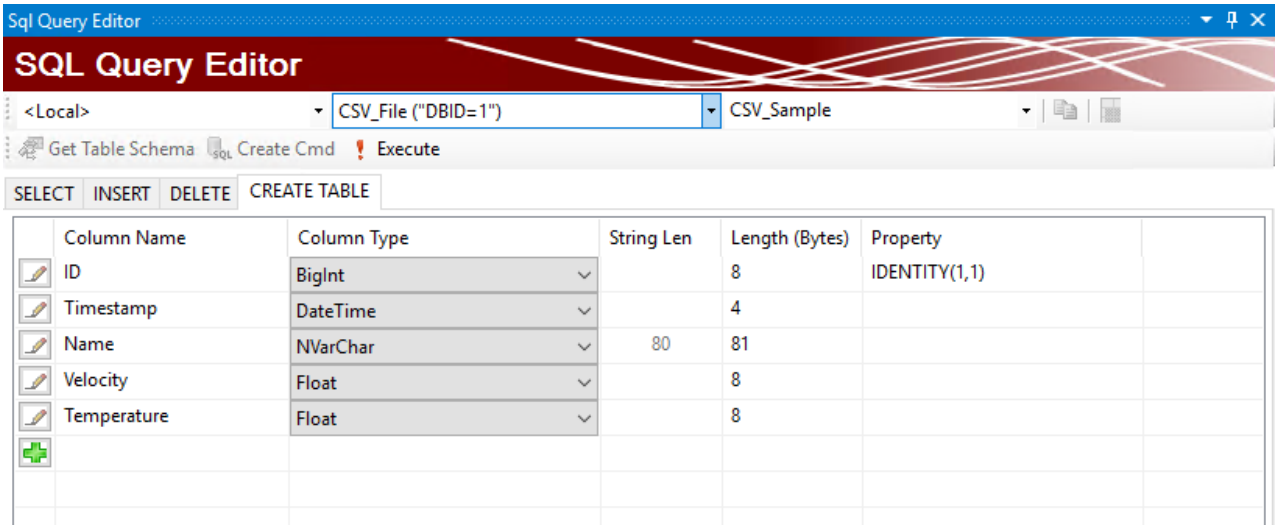
```
// Result: 16160;19-10-2018 12:27:38;Water Turbine;35.2238040741592;62.6461585412374
```

hDBID 取决于其配置，可以从数据库链接中获取。*pData*（或 *cbData*）可以是单个结构的地址，也可以是其结构数组的地址。这可以进一步提高性能。

CSV 文件的结构化写入和读取

并非所有功能块都可以使用 ASCII 格式 3.0。TwinCAT 数据库服务器的一些功能需要预先配置的表结构。但是，这不能以 ASCII 格式 3.0 进行存储。在本示例中，固定结构可用于在任何结构中使用 PLCDBWriteEvt 和 PLCDBReadEvt 功能块写入和读取数据。

以下面的结构为例：



也可以在“Select”（选择）选项卡下导出 PLC：

```
TYPE ST_CSVDataStruct :
STRUCT
    ID: LINT;
    Timestamp: DT;
    Name: STRING(80);
    Velocity: LREAL;
    Temperature: LREAL;
END_STRUCT
END_TYPE
```

各个 PLC 功能块的 Write/ReadStruct 方法可用于任何表结构：

```
VAR
    fbPLCDBWrite: FB_PLCDBWrite(sNetID:= '', tTimeout := T#30S);
    fbPLCDBRead : FB_PLCDBRead(sNetID:= '', tTimeout := T#30S);
    ColumnNames : ARRAY [0..4] OF STRING(50) := ['ID','Timestamp','Name','Velocity','Temperature'];
    Data: ST_CSVDataStruct;
    ReadData: ARRAY[0..4] OF ST_CSVDataStruct;
END_VAR
```

```
IF fbPLCDBWrite.WriteStruct(
    hDBID:= hDBID,
    sTableName:= 'CSV_Sample',
```

```
    pRecord:= ADR(Data),
    cbRecord:= SIZEOF(Data),
    pColumnNames:= ADR(ColumnNames),
    cbColumnNames:= SIZEOF(ColumnNames) )
THEN
    ;//Place for errorhandling or reactions
END_IF

IF fbPLCDBRead.ReadStruct(
    hDBID:= hDBID,
    sTableName:= 'CSV_Sample',
    pColumnNames:= ADR(ColumnNames),
    cbColumnNames:= SIZEOF(ColumnNames) ,
    sOrderByColumn:= 'ID',
    eOrderType := E_OrderType.ASC,
    nStartIndex:= 0,
    nRecordCount:= 5,
    pData:= ADR(ReadData),
    cbData:=SIZEOF(ReadData))
THEN
    ;//Place for errorhandling or reactions
END_IF
```

WriteStruct(...) 方法可将*数据*结构写入数据库。根据 *ColumnNames* 比较 PLC 和 CSV 文件的结构。

ReadStruct(...) 方法可从 CSV 文件中读取一定数量 *（nRecordCount）* 的记录。这些内容可以根据选定的列进行排序。*ReadData* 目标数组的大小应该足以接收所有检索到的数据。

附录

在这里可以下载这两个示例的示例配置以及简单示例程序的完整代码：https://infosys.beckhoff.com/content/1033/TF6420_TC3_Database_Server/Resources/5778536715.zip。为了说明该过程，程序会生成值并重复将它们发送到 CSV。上面使用的设置存储在一个单独的功能块中，该功能块以不同方式与 2 种 CSV 格式通信。

7.1.2.2 使用数据缓冲区进行快速数据记录

为了以毫秒为间隔将数据记录到数据库中，必须首先对数据进行整合，然后再通过 TwinCAT 数据库服务器将其传输到数据库中。这些数据缓冲区的大小可以根据要求而变化。在示例中，将 100 个数据样本组合在缓冲区中，然后通过 TwinCAT 数据库服务器传输它们。为了避免在写入过程中出现间隙，必须创建多个缓冲区来组合数据样本。在示例中，使用二维数组总共创建了 20 个缓冲区。

数据样本

定义：

```
TYPE ST_Data :
STRUCT
    Timestamp        : LINT;
    fAM              : LREAL;
    fPeak            : LREAL;
    fPulse           : LREAL;
    fSawtooth        : LREAL;
    fSine            : LREAL;
    fSquare          : LREAL;
    fStairs          : LREAL;
    fTriangular      : LREAL;
END_STRUCT
END_TYPE
```

每个周期都会填充数据缓冲区的一个元素。在示例中，这种情况每隔 10 ms 发生一次。因此，缓冲区包含周期为 1 s 的数据。如果缓冲区已填充 100 个元素，则另一个数组表示现在可以通过功能块 FB_PLCDBCmdEvt 传输这 100 个元素。为此，可以将整个缓冲区传输到功能块中。然后，将每个单独的元素从 TwinCAT 数据库服务器传输到数据库。本示例也可以使用其他功能块来实现。请注意，并非所有功能块都支持数组。

从功能块 FB_Record_tbl_Signals 中提取

（“状态机” => 状态：记录）

```
…
    2://Recording
        bRecording := TRUE;

        //Fill buffer
        stData[nWriteBufferIndex, nWriteIndex].Timestamp := nTimestamp;
        stData[nWriteBufferIndex, nWriteIndex].fAM := fAM;
        stData[nWriteBufferIndex, nWriteIndex].fPeak := fPeak;
        stData[nWriteBufferIndex, nWriteIndex].fPulse := fPulse;
```

```
        stData[nWriteBufferIndex, nWriteIndex].fSawtooth := fSawtooth;
        stData[nWriteBufferIndex, nWriteIndex].fSine := fSine;
        stData[nWriteBufferIndex, nWriteIndex].fSquare := fSquare;
        stData[nWriteBufferIndex, nWriteIndex].fStairs := fStairs;
        stData[nWriteBufferIndex, nWriteIndex].fTriangular := fTriangular;

        //Set buffer index
        nWriteIndex := nWriteIndex + 1;
        IF nWriteIndex = 100 THEN
            nWriteIndex := 0;
            aWriteSQL[nWriteBufferIndex]:= TRUE;
            nWriteBufferIndex := nWriteBufferIndex + 1;

            IF nWriteBufferIndex = 20 THEN
                nWriteBufferIndex := 0;
            END_IF

            IF aWriteSQL[nWriteBufferIndex] THEN
                nState := 255;
                RETURN;
            END_IF
        END_IF

        //Write buffer element (100 samples) to database
        IF aWriteSQL[nSQLIndex] THEN
            IF fbPLCDBCmd.Execute(nDBID, ADR(sCmd), SIZEOF(sCmd),
                                  ADR(stData[nSQLIndex,0]), SIZEOF(stData[nSQLIndex,0]) * 100,
                                  ADR(aPara), SIZEOF(aPara)) THEN
                IF fbPLCDBCmd.bError THEN
                    nState := 255;
                ELSE
                    nRecords := nRecords + 100;

                    aWriteSQL[nSQLIndex] := FALSE;

                    nSQLIndex := nSQLIndex + 1;
                    IF nSQLIndex = 20 THEN
                        nSQLIndex := 0;
                    END_IF

                    IF NOT bRecord THEN
                        bRecording := FALSE;
                        nState := 0;
                    END_IF
                END_IF
            END_IF
        END_IF
….
```

附录：

在本最佳实践示例中，一个函数发生器模块可用于生成各种信号，这些信号可以被记录到数据库中。尽管 INSERT 命令的语法通常是有效的，但是已使用 MS SQL 数据库进行了专门测试。下面附加的 ZIP 文件包含 Tnzip 格式的完整程序代码。

下载：https://infosys.beckhoff.com/content/1033/TF6420_TC3_Database_Server/Resources/6263666699.zip

7.1.2.3 NoSQL

本文档介绍了 NoSQL 数据库的处理。

使用的数据库：MongoDB
使用的数据库类型：DocumentDB

数据写入

DocumentDB 类型的数据库可以存储任何结构的 JSON 文档中。因此，在 *DocumentDB* 中可以映射 PLC 的任何结构。使用 FB_JSONDataType 可以自动创建该文档，或者使用字符串块可以汇编该文档。确保文档变量足够大。如果您想要同时写入多个文档，您可以在 JSON 数组中传输它们。

在准备过程中会定义 QueryOptions [▶ 224]。为此，指定了相关的集合和查询类型。每个查询类型都有自己的结构。结构 T_QueryOptionDocumentDB_Insert [▶ 226] 可用于写入文档。

```
VAR
    fbNoSQLQueryBuilder_DocumentDB: FB_NoSQLQueryBuilder_DocumentDB;
    InsertQueryOptions: T_QueryOptionDocumentDB_Insert;
    sDocument : STRING(2000);
END_VAR
```

```
InsertQueryOptions.pDocuments:= ADR(sDocument);
InsertQueryOptions.cbDocuments:= SIZEOF(sDocument);
fbNoSQLQueryBuilder_DocumentDB.eQueryType := E_DocumentDbQueryType.InsertOne;
fbNoSQLQueryBuilder_DocumentDB.sCollectionName := 'myCollection';
fbNoSQLQueryBuilder_DocumentDB.pQueryOptions := ADR(InsertQueryOptions);
fbNoSQLQueryBuilder_DocumentDB.cbQueryOptions := SIZEOF(InsertQueryOptions);
```

功能块 FB_NoSQLQueryEvt [▶ 200] 可用于将文档写入数据库。Execute() [▶ 201] 方法可将传输的文档写入数据库。该执行相对于 PLC 是异步的，可能需要几个周期。布尔返回值可指示功能块何时完成其过程：

```
VAR
    fbNoSQLQuery: FB_NoSQLQueryEvt(sNetID := '', tTimeout := TIME#15S0MS);
    fbJsonDataType: FB_JsonReadWriteDatatype;
END_VAR
```

```
CASE eState OF
    …
    eMyDbState.Write:
```

```
        // set the document yourself as json format (Example)
        sDocument := '{"myBool" : true,
                       "Name" : "Some Name Value",
                       "Value": 2.3,
                       "Value2":3,
                       "Child":{"Name":"Single Child",
                                "Value":1,
                                "myBool":true,
                                "arr":[12.0,13.0,14.0,15.0],
                                "myBool2" : true},
                       "Children":[
                                {"Name":"Child1"
                                ,"Value": 1,
                                "myBool" : true,
                                "arr":[12.1,13.1,14.1,15.1],
                                "myBool2" : true},
                                {"Name":"Child2",
                                "Value":2,
                                "myBool" : true,
                                "arr":[12.2,13.2,14.2,15.2],
                                "myBool2" : true},
                                {"Name":"Child3",
                                "Value":1,
                                "myBool" : true,
                                "arr":[12.3,13.3,14.3,15.3],
                                "myBool2" : true}]
                       }';
```

```
        IF fbNoSQLQuery.Execute(1, myQueryBuilder) THEN
            IF fbNoSQLQuery.bError THEN
                InfoResult := fbNoSQLQuery.ipTcResult;
                eState:= eMyDbState.Error;
            ELSE
                eState:= eMyDbState.Idle;
            END_IF
        END_IF
    …
END_CASE
```

数据库可识别存储各个变量所使用的数据类型。不过，与 *MongoDB* 一样，可以明确指定数据类型。如果要将时间戳明确保存为数据类型，则必须在 JSON 文档中对其进行定义：

```
sDocument := '{…"myTimestamp": ISODate("2019-02-01T14:46:06.0000000"), …}';
```

不仅可以通过 TwinCAT 3 库的字符串格式化功能块对字符串进行格式化，还可以通过 JSON 文档的辅助功能块对其进行格式化，例如，Tc3_JsonXml 中的 FB_JsonReadWriteDatatype。

```
// set the document by JsonDataType
sTypeName := fbJsonDataType.GetDatatypeNameByAddress(SIZEOF(anyValue[1]), ADR(anyValue[1]));
sDocument := fbJsonDataType.GetJsonStringFromSymbol(sTypeName, SIZEOF(anyValue [1]), ADR(anyValue
[1]));
```

读取数据

对于每个文档，基于文档的数据库中的数据模式可能有所不同。相比之下，PLC 遵循固定的过程映像。数据可能与过程映像不一致。

有 2 种不同的方法可以读取数据库中的数据：查找查询和聚合方法。这两种方法都从数据库返回结果，不过，聚合提供了更多选项，可将数据转换为适当的形式或执行多种操作，例如，直接计算平均值。

在准备过程中会定义 QueryOptions [▶ 224]。为此，指定了相关的集合和查询类型。每个查询类型都有自己的结构。结构 T_QueryOptionDocumentDB_Aggregation [▶ 225] 可用于聚合文档。

```
VAR
    fbNoSQLQueryBuilder_DocumentDB: FB_NoSQLQueryBuilder_DocumentDB;
    AggregationQueryOptions: T_QueryOptionDocumentDB_Aggregate;
    sPipeStages: STRING(1000);
END_VAR
```

```
AggregationQueryOptions.pPipeStages := ADR(sPipeStages);
AggregationQueryOptions.cbPipeStages := SIZEOF(sPipeStages);
fbNoSQLQueryBuilder_DocumentDB.eQueryType := E_DocumentDbQueryType.Aggregation;
fbNoSQLQueryBuilder_DocumentDB.sCollectionName := 'myCollection';
fbNoSQLQueryBuilder_DocumentDB.pQueryOptions := ADR(AggregationQueryOptions);
fbNoSQLQueryBuilder_DocumentDB.cbQueryOptions := SIZEOF(AggregationQueryOptions);
```

FB_NoSQLQueryEvt [▶ 200] 可用于发送聚合查询。ExecuteDataReturn() [▶ 202] 方法可以用于传输参数，并将返回的数据放入传输的内存引用中。该执行相对于 PLC 是异步的，需要几个周期。布尔返回值可指示功能块何时完成其过程：

```
VAR
    fbNoSQLQuery: FB_NoSQLQueryEvt(sNetID := '', tTimeout := TIME#15S0MS);
    fbNoSQLResult: FB_NoSQLResultEvt(sNetID := '', tTimeout := TIME#15S0MS);
END_VAR
```

```
CASE eState OF
    …
    eMyDbState.Aggregation:
        sPipeStages :='{$$match :{}}';
        IF fbNoSQLQuery.ExecuteDataReturn(1, myQueryBuilder, pNoSqlResult:= ADR(fbNoSQLResult),
nDocumentLength=> nDocumentLength)) THEN
            IF fbNoSQLQuery.bError THEN
                InfoResult := fbNoSQLQuery.ipTcResult;
                eState:= eMyDbState.Error;
            ELSE
                eState:= eMyDbState.Idle;
            END_IF
        END_IF
    …
END_CASE
```

sPipeStages 的语法取决于数据库类型。它将返回所有记录。其他选项（带有虚构记录）包括：

运算符	描述
{$$match : {Place : “NorthEast” }}	元素“Place”的值为“NorthEast”的所有记录。
{$$project : { myValue : { $arrayElemAt : ["$WindPlantData.RotorSensor", 2]} } }	将数组元素位置 2 中的所有 RotorSensor 数据作为“myValue”返回。
{$$project : {RotorAvg : {$avg: "$WindPlantData.RotorSensor"} } }	以“RotorAvg”的形式返回数据数组“RotorSensor”的平均值。

从各数据库提供程序中获取运算符的完整文档。

现在可以在功能块 FB_NoSQLResultEvt [▶ 203] 中找到对返回数据的引用。现在可以将它们作为字符串中的 JSON 文档或结构进行读取。现在将数据直接读入具有合适结构的数组。您可以使用数据库服务器的 SQL 查询编辑器直接生成一个与记录相匹配的结构。除了数组之外，还可以存储单个结构的地址，以便在仅检索一条记录时使用。

```
VAR
    fbNoSQLResult: FB_NoSQLResultEvt(sNetID := '', tTimeout := TIME#15S0MS);
    aRdStruct : ARRAY [0..9] OF ST_MyCustomStruct;
    fbNoSqlValidation : FB_NoSQLValidationEvt(sNetID := '', tTimeout := TIME#15S0MS);
END_VAR
```

```
CASE eState OF
    …
    eMyDbState.ReadStruct:
        IF fbnoSQLDBResult.ReadAsStruct(0, 4, ADR(aRdStruct), SIZEOF(aRdStruct), bValidate := TRUE,
ADR(fbNoSqlValidation), bDataRelease:= TRUE) THEN
            IF fbnoSQLDBResult.bError THEN
                InfoResult := fbnoSQLDBResult.ipTcResult;
                eState:= eMyDbState.Error;
            ELSE
                eState:= eMyDbState.Idle;
            END_IF
        END_IF
    …
END_CASE
```

TwinCAT 数据库服务器在分配记录或结构时会考虑记录中元素的名称和变量的名称。如果两者不同，则可在 PLC 中使用属性“ElementName”：

```
TYPE ST_WindFarmData :
STRUCT
    {attribute 'ElementName' := '_id'}
    ID: T_ObjectId_MongoDB;
    {attribute 'ElementName' := 'Timestamp'}
    LastTime: DT;
```

```
    {attribute 'ElementName' := 'WindPlantData'}
    Data: ST_WindFarmData_WindPlantData;
END_STRUCT
END_TYPE
```

在本示例中，“ElementName”指定了数据库文档中数据的名称。起始索引和记录数可以用于确定该调用将返回哪些记录。为了避免可能的重复，请注意这些选项可能已经由“PipeplineStages”的运算符执行。

数据验证

如果在 FB_NoSQLResult [▶ 203] 的 PLC 中记录与结构存在冲突，则可以通过 FB_NoSQLValidationEvt [▶ 206] 将它们读出。冲突的例子包括记录丢失或过剩，或者数据类型问题。方法 GetIssues() [▶ 207] 可以用于将所有冲突读取为字符串数组。通过 GetRemainingData() [▶ 208] 可以将在 PLC 结构中未找到的剩余数据读取为 JSON 格式的字符串数组。如果有必要，可以将它们单独读出到正确的结构中或者通过 TwinCAT JSON 库进行解析。

```
VAR
    fbNoSqlValidation : FB_NoSQLValidationEvt(sNetID := '', tTimeout := TIME#15S0MS);
    aIssues : ARRAY[0..99] OF STRING(512);
    aRemaining : ARRAY [0..9] OF STRING(1000);
END_VAR
```

```
CASE eState OF
    …
    eMyDbState.ValidationIssues:
        IF fbValidation.GetIssues(ADR(aIssues), SIZEOF(aIssues), FALSE) THEN
            IF fbValidation.bError THEN
                InfoResult := fbValidation.ipTcResult;
                eState:= eMyDbState.Error;
            ELSE
                eState:= eMyDbState.Idle;
            END_IF

    eMyDbState.ValidationRemaining:
        IF fbValidation.GetRemainingData(ADR(aRemaining), SIZEOF(aRemaining), SIZEOF(aRemaining[1]),
bDataRelease:= FALSE)THEN
            IF fbValidation.bError THEN
                InfoResult := fbValidation.ipTcResult;
                eState:= eMyDbState.Error;
            ELSE
                eState:= eMyDbState.Idle;
            END_IF
    …
END_CASE
```

下载：https://infosys.beckhoff.com/content/1033/TF6420_TC3_Database_Server/Resources/13743807627.zip

7.1.2.4 PostgreSql 例程

除了函数（PostgreSQL 例程）外，从 PostgreSQL 11 开始，数据库还支持存储过程，以实现服务器端编程。这两个例程类型具有不同的属性和功能：

表 3: 存储过程和函数的比较

	存储过程	函数
OUT 参数	+	-
返回值	-	+
可用于查询	-	+
支持交易	+	-

这些属性需要与 TwinCAT 数据库服务器建立不同的接口。与其他支持的数据库一样，使用 FB_SQLStoredProcedureEvt [▶ 194] 功能块可执行存储过程。函数可以集成在 SQL 命令中，通过 FB_PLCDBCmdEvt [▶ 177] 或 FB_SQLCommandEvt [▶ 189] 进行调用。

PostgreSQL 使用“RefCursor”来返回例程的数据集。TwinCAT 数据库服务器可自动评估这些“RefCursor”，并返回其中引用的数据集。无法解析多个“RefCursor”。

调用存储过程

通过 FB_SQLStoredProcedureEvt 可以调用过程。

示例（SQL）

创建过程的 SQL 脚本：

```
CREATE OR REPLACE PROCEDURE "public"."SP_getLastData"(
    INOUT result_data refcursor)
LANGUAGE 'plpgsql'
AS $BODY$
BEGIN
    open result_data for SELECT * FROM "myTable_Double" LIMIT 10;
END
$BODY$;
```

如果过程定义了一个（或多个）“RefCursor”作为输出参数，则会自动解析该参数（或第一个参数），并将生成的数据集存储到 FB_SQLResultEvt [▶ 191] 的缓冲区中。TwinCAT 数据库服务器将数据类型“RefCursor”视为字符串。

示例（ST 中的 TwinCAT 3）

```
VAR
    fbSqlDatabase   : FB_SQLDatabaseEvt(sNetID := '', tTimeout := T#5S);
    ParaInfo        : ST_SQLSPParameter;
END_VAR
```

```
ParaInfo.sParameterName     := '@result_data';
ParaInfo.eParameterType     := E_SPParameterType.InputOutput;
ParaInfo.eParameterDataType := E_ColumnType.RefCursor; // 19
ParaInfo.nParameterSize     := 81;

IF fbSQLDatabase.CreateSP('"public"."SP_getLastData"', ADR(ParaInfo), SIZEOF(ParaInfo), ADR(fbSQLSto
redProcedure)) THEN
    IF fbSQLDatabase.bError THEN
        nState:=255;
    ELSE
        nState:= nState+1;
    END_IF
END_IF
```

FB_SQLStoredProcedureEvt [▶ 194] 使用先前与 FB_SQLDatabaseEvt.CreateSP() [▶ 185] 链接的存储过程

```
VAR
    fbSQLStoredProcedure : FB_SQLStoredProcedureEvt(sNetID:='', tTimeout := T#5S);
    sRefCursor            : STRING;
    tcMessage            : I_TcMessage;
END_VAR
```

```
IF fbSQLStoredProcedure.ExecuteDataReturn(pParameterStrc := ADR(sRefCursor),
cbParameterStrc:= SIZEOF(sRefCursor), pSQLDBResult := ADR(fbSqlResult)) THEN
    IF fbSQLStoredProcedure.bError THEN
        tcMessage := fbSQLStoredProcedure.ipTcResult;
        nState := 255;
    ELSE
        nState := nState+1;
    END_IF
END_IF
```

然后可以使用 FB_SQLResultEvt [▶ 191] 来读取数据。

调用函数

在 SQL 命令中可以调用函数。

示例（SQL）

创建函数的 SQL 脚本：

```
CREATE OR REPLACE FUNCTION "public"."F_getLastData"()
    RETURNS refcursor
    LANGUAGE 'plpgsql'
AS $BODY$
DECLARE result_data refcursor;
BEGIN
    open result_data for SELECT * FROM "myTable_Double" ORDER BY "ID" DESC LIMIT 10;
    return result_data;
END;$BODY$;
```

下面的 SQL 命令可用于调用函数：

```
SELECT "public"."F_getLastData"();
```

调用本身会返回一个“RefCursor”。这由 TwinCAT 数据库服务器自动解析。

示例（ST 中的 TwinCAT 3）

FB_SQLCommandEvt [▶ 189] 可使用 FB_SQLDatabaseEvt.CreateCmd() [▶ 185] 创建的命令。

```
VAR
    fbSqlCommand : FB_SQLCommandEvt(sNetID := '', tTimeout := T#5S);
    tcMessage    : I_TcMessage;
END_VAR
```

```
sCmd := 'SELECT "public"."getLastData"();';

// call sql command
IF fbSQLCommand.ExecuteDataReturn(ADR(sCmd), SIZEOF(sCmd), ADR(fbSqlResult)) THEN
    IF fbSQLCommand.bError THEN
        tcMessage := fbSQLCommand.ipTcResult;
        nState := 255; // error state
    ELSE
        nState := nState+1;
    END_IF
END_IF
```

然后可以使用 FB_SQLResultEvt [▶ 191] 来读取数据集。

建议在状态机中使用该程序代码。

下载：https://infosys.beckhoff.com/content/1033/TF6420_TC3_Database_Server/Resources/13743810955.zip

7.1.2.5 循环数据和时间序列数据库

本文档介绍了时间序列的处理以及如何在时间序列数据库中存储循环数据。

使用的数据库：InfluxDB
使用的数据库类型：TimeSeriesDB

简介

定期或循环写入数据是控制技术中的常见应用。数据应以较高的时间精度记录。由于数据库通信不具备实时性，因此将定期测量的数据存储在缓冲区中十分有用。为此，可以使用数据结构的数组。然后将收集的数据发送到 TwinCAT 数据库服务器，在那里可以不受时间限制地进行处理，然后再存储在数据库中。

时间

在数据库中存储的每个数据集都分配有一个时间戳。这些时间戳与标签列一起构成了唯一的 ID。如果 2 个数据集具有相同的 ID（相同的时间戳和标签值），则较新的数据集将覆盖旧的数据集。

示例：

	时间	locationname（标签）	温度（字段）	风速（字段）
1	1581675200630326200	费尔	11.5	6.3
2	1581675200630327200	费尔	10.3	5.2
3	1581675200630328200	费尔	9.8	2.8
4	1581675200630328200	汉堡	14.2	14.9
4	1581675200630328200	汉堡	15.6	8.9
…	…	…	…	…

由于 ID 相同，因此 4 号数据集将被新的数据集覆盖。

数据库中的时间戳默认被保存为 UNIX 时间戳。除用户创建的插入命令外，在 TwinCAT 3 数据库服务器的功能块中可以接收时间戳并将其转换为 TwinCAT 时间（自 1601 年 1 月 1 日以来的 100 ns 步数）。对于插入命令，不会转换时间。

数据库配置

在数据库配置中应选择 InfluxDB。插入所需数据库的连接参数。如果尚无可用的数据库，则您可以点击“Create”（创建）按钮来创建数据库。注意您的防火墙设置或端口授权。**无需创建表格，因为它会在首次 InfluxDB 访问期间自动创建。**InfluxDB 没有固定的表格模式。稍后还可以扩展或添加列。

写入循环数据

本示例展示了如何以最小的工作量将符号从 PLC 写入时间序列数据库。

声明：

```
    State: E_DbLogState;
    bWriting: BOOL; // Set this bool fla to write the data once into the InfluxDB

    dbid: UDINT := 1; // Handle to the configured database

    QueryOption_TSDB_Insert : T_QueryOptionTimeSeriesDB_Insert; // defines detailed Queryparameter
    fbNoSQLQueryBuilder_TimeSeriesDB : FB_NoSQLQueryBuilder_TimeSeriesDB; // defines database type
specific api
    fbNoSqlQueryEvt : FB_NoSQLQueryEvt(sNetID := '', tTimeout := T#15S); // functionblock to execute
queries

    // databuffer for 1 second with 10 ms time delta
    windTurbineData: ARRAY[1..100] OF WindTurbineData;

    // error handling helper values
    TcResult: Tc3_Database.I_TcMessage;
    bError: BOOL;
    sErrorMessage: STRING(255);

    i: INT;
    rand : DRAND;
    nrand: LREAL;
```

声明数据源结构：

在该结构中，属性“TagName”和“FieldName”可用于将数据字段声明为标签或字段。默认情况下会将它们声明为字段。如果您希望表格中的列名与 PLC 中的符号名称不同，也可以使用这些属性。

为了在数据分析过程中捕获未设置的数据，可以使用值范围之外的默认值在分析过程中检测此类数据。

```
TYPE WindTurbineData :
STRUCT

    {attribute 'TagName' := 'ID'}
    WindTurbineID : STRING(255);

    {attribute 'FieldName' := 'Power'}
    Power : LREAL := -1; // [0) [kW]

    {attribute 'FieldName' := 'Wind Speed'}
    WindSpeed : LREAL := -1; // [0) [m/s]

    {attribute 'FieldName' := 'Wind Direction'}
    WindDirection : LREAL := -1; // [0,360][°]

END_STRUCT
END_TYPE
```

（WindTurbineData.tcDUT）

状态机的 ENUM 声明：

```
TYPE E_DbLogState :
(
    idle := 0,
    init,
    writing,
    error
);
END_TYPE
```

生成示例数据：

```
        FOR i := 1 TO 100 BY 1 DO
             rand();
             nrand := rand.Num;
            windTurbineData[i].WindDirection := 240.328 + nrand;
```

```
            windTurbineData[i].WindSpeed := 7.3292 + nrand;
            windTurbineData[i].Power := 1133.1975 + nrand;
            windTurbineData[i].WindTurbineID := 'Wind Turbine Verl 13';
        END_FOR

CASE State OF

    E_DbLogState.idle:
        IF bWriting THEN
            bWriting := FALSE;
            State := E_DbLogState.init;
        END_IF
```

准备调用：

在这种情况下会将数组“windTurbineData”写入数据库的“WindMeasurement”表格中。因此，从过程映像中可以直接读取数据。指定数据类型，以读取内存中的地址。通过指定开始时间和时间间隔，可以自动生成数据集的时间。数据必须正确存储在数组中。例如，每个 PLC 周期可以使用数组中的一个数据集。为该过程创建一个 PLC 任务十分有用。在本示例中，循环时间为 10 ms。

```
    E_DbLogState.init:

        fbNoSQLQueryBuilder_TimeSeriesDB.pQueryOptions := ADR(QueryOption_TSDB_Insert);
        fbNoSQLQueryBuilder_TimeSeriesDB.cbQueryOptions := SIZEOF(QueryOption_TSDB_Insert);

        QueryOption_TSDB_Insert.sTableName := 'WindMeasurement';
        QueryOption_TSDB_Insert.sDataType := 'WindTurbineData';
        QueryOption_TSDB_Insert.pSymbol := ADR(windTurbineData);
        QueryOption_TSDB_Insert.cbSymbol := SIZEOF(windTurbineData);
        QueryOption_TSDB_Insert.nDataCount := 100;
        QueryOption_TSDB_Insert.nStartTimestamp := F_GetSystemTime();
        QueryOption_TSDB_Insert.nCycleTime := 10000; // (in 100 ns)
        State := E_DbLogState.writing;
```

写入数据：

该调用可将数据写入具有相应数据库 ID 的配置数据库。这可能需要几个周期，因为这是一个异步过程。如有必要，必须使用多个存储数组，以确保无缝记录数据，不留任何间隙。

```
    E_DbLogState.writing:

        IF fbNoSqlQueryEvt.Execute(dbid, fbNoSQLQueryBuilder_TimeSeriesDB) THEN
            IF fbNoSqlQueryEvt.bError THEN
                TcResult := fbNoSqlQueryEvt.ipTcResult;
                State := E_DbLogState.error;
            ELSE
                State := E_DbLogState.idle;
            END_IF
        END_IF
```

错误处理：

使用 Tc3 Eventlogger 进行错误处理

```
    E_DbLogState.error:

        IF TcResult.RequestEventText(1033, sErrorMessage, SIZEOF(sErrorMessage)) THEN
            TcResult.Send(F_GetSystemTime());
            State := E_DbLogState.idle;
            bError := TRUE;
        END_IF


END_CASE
```

下载：https://infosys.beckhoff.com/content/1033/TF6420_TC3_Database_Server/Resources/13743809291.zip

7.2 Tc2_Database

TwinCAT 数据库服务器的所有示例应用程序都整合在一个解决方案中。在这里可以从一个中央位置下载该解决方案：https://infosys.beckhoff.com/content/1033/TF6420_TC3_Database_Server/Resources/3494041099.zip

除了用于 TwinCAT 3 解决方案的 tszip 文件外，该 zip 文件还包含所有必需的基于文件的数据库。如果 zip 文件中的“Samples”（示例）文件夹位于默认的安装文件夹中：*C:\TwinCAT\Functions\TF6420-Database-Server\Win32*，则无需对数据库服务器配置中的路径进行进一步编辑。使用基于非文件的数据库（例如 MS SQL）的示例必须通过配置器进行单独调整。

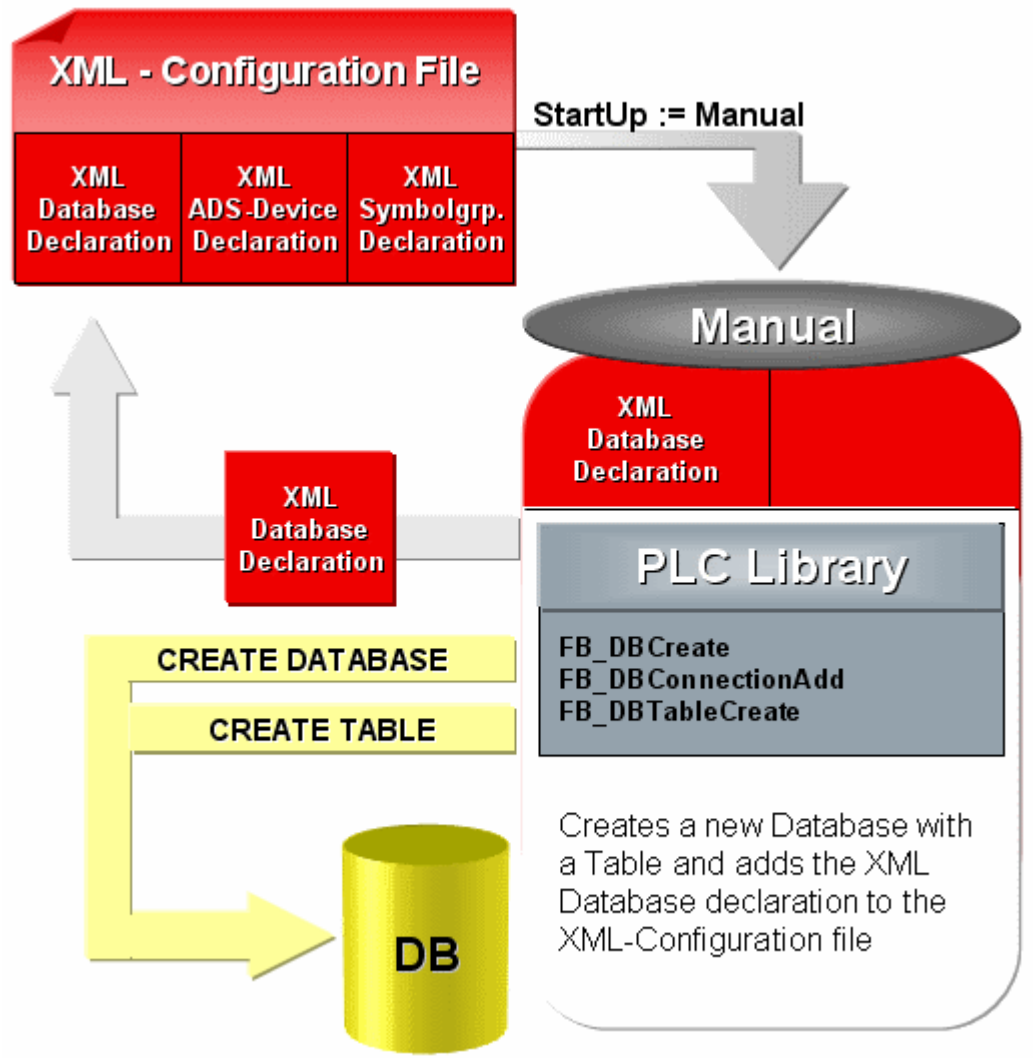
在单独的页面上详细地记录了各个示例：

SP 项目名称	描述
Create_DB_Sample [▶ 341]	从 PLC 创建数据库连接和表格
Cyclic_RdWrt_Sample [▶ 343]	从数据库循环读取/写入数据
Write_DB_Sample [▶ 345]	使用简单的 PLC 功能块将变量写入数据库，无需 SQL 命令
SQL_InsertSelect_Sample [▶ 348]	使用功能块 FB_DBRecordInsert/FB_DBRecordArraySelect 的示例
StoredProcedures_Sample [▶ 351]	使用 FB_DBStoredProceduresRecordArray 的存储过程
XML_DB_Sample [▶ 354]	使用 XML 文件作为数据库
XML_XPath_Sample [▶ 358]	没有模式的 XML XPath 示例
XML_XPath_Schema_Sample [▶ 360]	具有 XML 模式的 XML XPath 示例，可与 TwinCAT XML 服务器“读取”功能相媲美

7.2.1 创建 MS Access 数据库

本示例说明了如何从 PLC 创建数据库。此外，会添加一个表格，并在 XML 配置文件中声明已生成的数据库。

下载：https://infosys.beckhoff.com/content/1033/TF6420_TC3_Database_Server/Resources/3494041099.zip



使用的数据库类型	MS Access
兼容的数据库类型	MS SQL、MS Compact SQL、MS Access、XML
使用的功能块	FB_DBCreate、FB_DBConnectionAdd、FB_DBTableCreate
需要使用的库	Tc2_Database、Tc2_System、Tc2_Standard
下载文件列表	TcDBSrv_InfoSysSamples.tszip

在生成的数据库中添加一个名为“myTable”的表格，其结构如下：

ColumnName	Data type	属性
ID	Bigint	IDENTITY(1,1)
Timestamp	DateTime	
Name	Ntext	
值	Float	

该表结构由以下数组生成：

```
tablestrc: ARRAY [0..3] OF ST_DBColumnCfg :=
     [(sColumnName:='ID',sColumnProperty:='IDENTITY(1,1)',eColumnType:=EDBCOLUMN_BIGINT),
     (sColumnName:='Timestamp',eColumnType:=EDBCOLUMN_DATETIME),
     (sColumnName:='Name',eColumnType:=EDBCOLUMN_NTEXT),
     (sColumnName:='Value',eColumnType:=EDBCOLUMN_FLOAT)];
```

变量声明

```
PROGRAM MAIN
VAR
    R_TRIG1             : R_TRIG;
    bSTART              : BOOL;

    FB_FileDelete1      : FB_FileDelete;
    FB_DBCreate1        : FB_DBCreate;
    FB_DBConnectionAdd1: FB_DBConnectionAdd;
    FB_DBTableCreate1   : FB_DBTableCreate;

    bBusy_Delete        : BOOL;
    bBusy_CreateDB      : BOOL;
    bBusy_ConnAdd       : BOOL;
    bBusy_CreateTable   : BOOL;

    bErr                : BOOL;
    nErrid              : UDINT;

    nDBid               : UDINT;

    arrTablestrc        : ARRAY [0..3] OF ST_DBColumnCfg :=
     [(sColumnName:='ID',sColumnProperty:='IDENTITY(1,1)',eColumnType:=EDBCOLUMN_BIGINT),
     (sColumnName:='Timestamp',eColumnType:=EDBCOLUMN_DATETIME),
     (sColumnName:='Name',eColumnType:=EDBCOLUMN_NTEXT),
     (sColumnName:='Value',eColumnType:=EDBCOLUMN_FLOAT)];

    nState:BYTE := 0;

END_VAR
```

PLC 程序

```
CASE nState OF
    0:
        (*To start this sample you have to set a rising edge to the variable bSTART*)
        R_TRIG1(CLK:=bSTART);
        IF R_TRIG1.Q THEN
            nState   := 1;
            FB_FileDelete1(bExecute:=FALSE);
            FB_DBCreate1(bExecute:=FALSE);
            FB_DBConnectionAdd1(bExecute:=FALSE);
            FB_DBTableCreate1(bExecute:=FALSE);
            bSTART   := FALSE;
        END_IF
    1:
        (*It isn't possible to overwrite an existing database file.
         If the database file exist the FB_FileDelete block will delete the file*)
        FB_FileDelete1(
            sNetId   := ,
            sPathName:= 'C:\TwinCAT\TcDatabaseSrv\Samples\TestDB1000SPS.mdb',
            ePath    := PATH_GENERIC,
            bExecute := TRUE,
            tTimeout := T#5s,
            bBusy    => bBusy_Delete,
            bError   => ,
```

```
            nErrId    => );

        IF NOT bBusy_Delete THEN
            nState   := 2;
          END_IF

    2:
        (*The FB_DBCreate block will create the database file
         "C:\TwinCAT\TcDatabaseSrv\Samples\TestDB1000SPS.mdb"*)
        FB_DBCreate1(
            sNetID   := ,
            sPathName:= 'C:\TwinCAT\TcDatabaseSrv\Samples',
            sDBName  := 'TestDB1000SPS',
            eDBType  := eDBType_Access,
            bExecute := TRUE,
            tTimeout := T#15s,
            bBusy     => bBusy_CreateDB,
            bError    => bErr,
            nErrID    => nErrid);

        IF NOT bBusy_CreateDB AND NOT bErr THEN
            nState   := 3;
        END_IF
    3:
        (*The FB_DBConnectionAdd adds the connection information to the
         XML configuration file*)
        FB_DBConnectionAdd1(
            sNetID      := ,
            eDBType     := eDBType_Access,
            eDBValueType:= eDBValue_Double,
            sDBServer   := ,
            sDBProvider := 'Microsoft.Jet.OLEDB.4.0',
            sDBUrl      := 'C:\TwinCAT\TcDatabaseSrv\Samples\TestDB1000SPS.mdb',
            sDBTable    := 'myTable',
            bExecute    := TRUE,
            tTimeout    := T#15s,
            bBusy        => bBusy_ConnAdd,
            bError       => bErr,
            nErrID       => nErrid,
            hDBID        => nDBid);

        IF NOT bBusy_ConnAdd AND NOT bErr THEN
            nState      := 4;
        END_IF
    4:
        (*The FB_DBTableCreate create the table "myTable"*)
        FB_DBTableCreate1(
            sNetID      := ,
            hDBID       := nDBid,
            sTableName  := 'myTable',
            cbTableCfg  := SIZEOF(arrTablestrc),
            pTableCfg   := ADR(arrTablestrc),
            bExecute    := TRUE,
            tTimeout    := T#15s,
            bBusy        => bBusy_CreateTable,
            bError       => bErr,
            nErrID       => nErrid);

        IF NOT bBusy_CreateTable AND NOT bErr THEN
            nState := 0;
        END_IF
END_CASE
```

为了使用该示例，您只需将 ADS 设备（安装有 TwinCAT 数据库服务器）的 NetID 传输到 sNetID 参数即可。

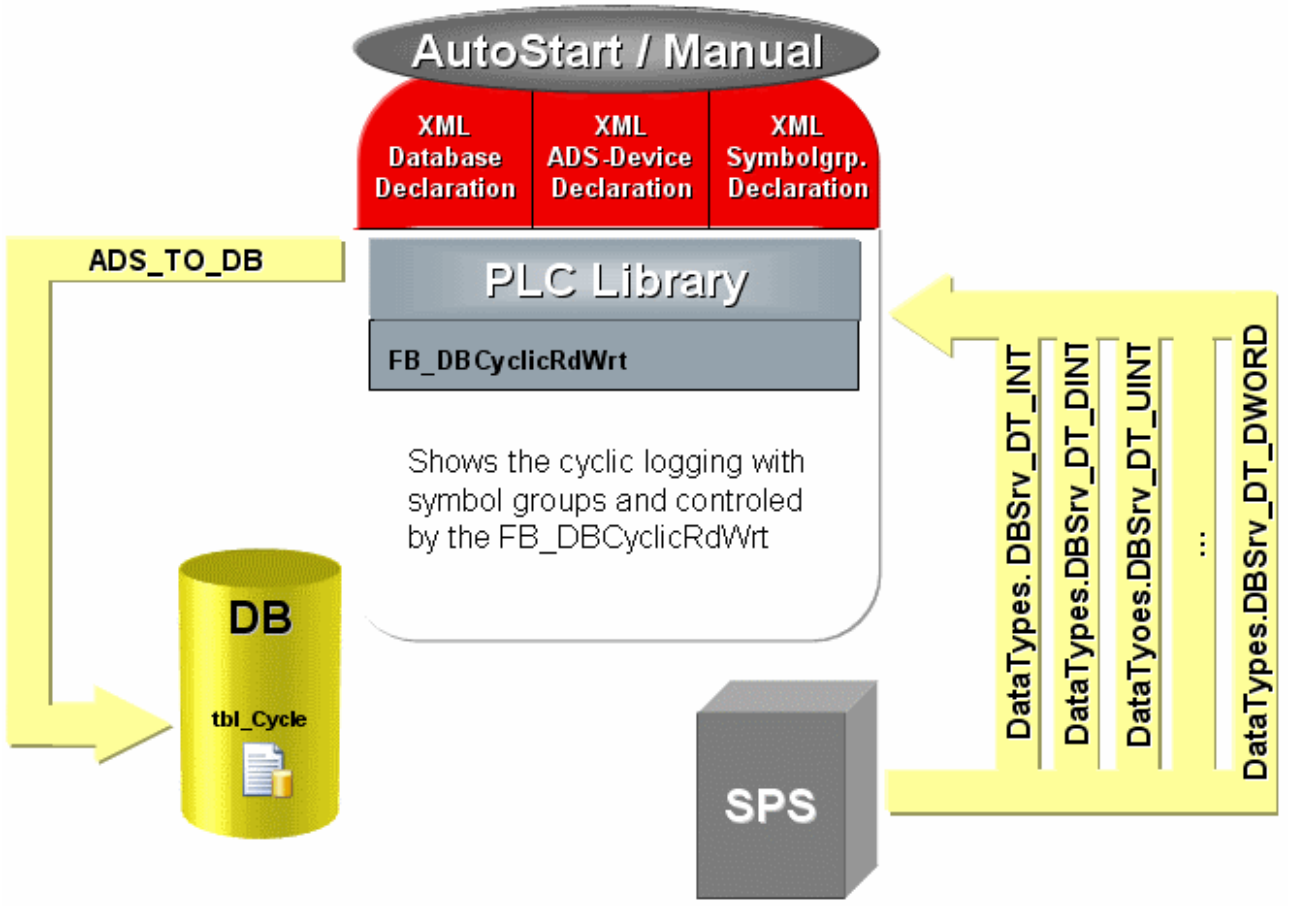
要求

开发环境	目标平台	要连接的 PLC 库
TwinCAT v3.0.0	PC 或 CX（x86）	Tc2_Database

7.2.2 启动/停止，循环记录

本示例说明了从 PLC 开始启动和停止循环记录的过程。

下载：https://infosys.beckhoff.com/content/1033/TF6420_TC3_Database_Server/Resources/3494041099.zip



使用的数据库类型	MS Compact SQL
兼容的数据库类型	ASCII、MS SQL、MS Compact SQL、MS Access、MySQL、PostgreSQL、DB2、Oracle、InterBase/Firebird、XML
使用的功能块	FB_DBCyclicRdWrt
需要使用的库	Tc2_Database、Tc2_System、Tc2_Standard
下载文件列表	TcDBSrv_InfoSysSamples.tszip、CurrentConfigDataBase.xml、TestDB_Cyclic.sdf

在本示例中，通过切换 bStartStop 变量来启动或停止循环采集功能。
循环采集过程随着 bExecute 输入参数的上升沿而开始。
下降沿将再次结束该过程。

变量声明（PRG 数据类型）

```
PROGRAM DataTypes
VAR
    DBSrv_DT_INT     : INT;
    DBSrv_DT_UINT    : UINT;
    DBSrv_DT_DINT    : DINT;
    DBSrv_DT_UDINT   : UDINT;
    DBSrv_DT_REAL    : REAL;
    DBSrv_DT_LREAL   : LREAL;
    DBSrv_DT_BYTE    : BYTE := 16#A1;
    DBSrv_DT_BOOL    : BOOL;
    DBSrv_DT_MYSTRUCT: ST_MyStruct;
    DBSrv_DT_ARRAY   : ARRAY [0..19] OF UDINT;
    DBSrv_DT_WORD    : WORD;
    DBSrv_DT_DWORD   : DWORD;
END_VAR
```

ST_MyStruct 结构体

```
TYPE ST_MyStruct :
STRUCT
    iValue1 : INT;
    iValue2 : UINT;
    iValue3 : BOOL;
```

```
    iValue4 : REAL;
END_STRUCT
END_TYPE
```

变量声明

```
PROGRAM MAIN
VAR
    fbDBCyclicRdWrt1: FB_DBCyclicRdWrt;

    bCyclic         : BOOL :=TRUE;

    bBusy_Cyclic    : BOOL;
    bErr            : BOOL;
    nErrID          : UDINT;
    sSQLState       : ST_DBSQLError;
END_VAR
```

PLC 程序

```
DataTypes;

fbDBCyclicRdWrt(
    sNetID   := ,
    bExecute := bCyclic,
    tTimeout := t#15s,
    bBusy     => bBusy_Cyclic,
    bError    => bErr,
    nErrID    => nErrID,
    sSQLState => sSQLState);
```

为了使用该示例，您只需将 ADS 设备（安装有 TwinCAT 数据库服务器）的 NetID 传输到 sNetID 参数输入端即可。
当您运行程序并将 bCyclic 变量设置为 TRUE 时，在 XML 配置文件的符号组中声明的所有变量都会被记录。

TwinCAT 数据库服务器

在 XML 配置文件中声明的所有 Microsoft SQL Compact 数据库必须存在。它们不是自动生成的。另一方面，如果声明的 ASCII 文件不存在，则会自动生成。

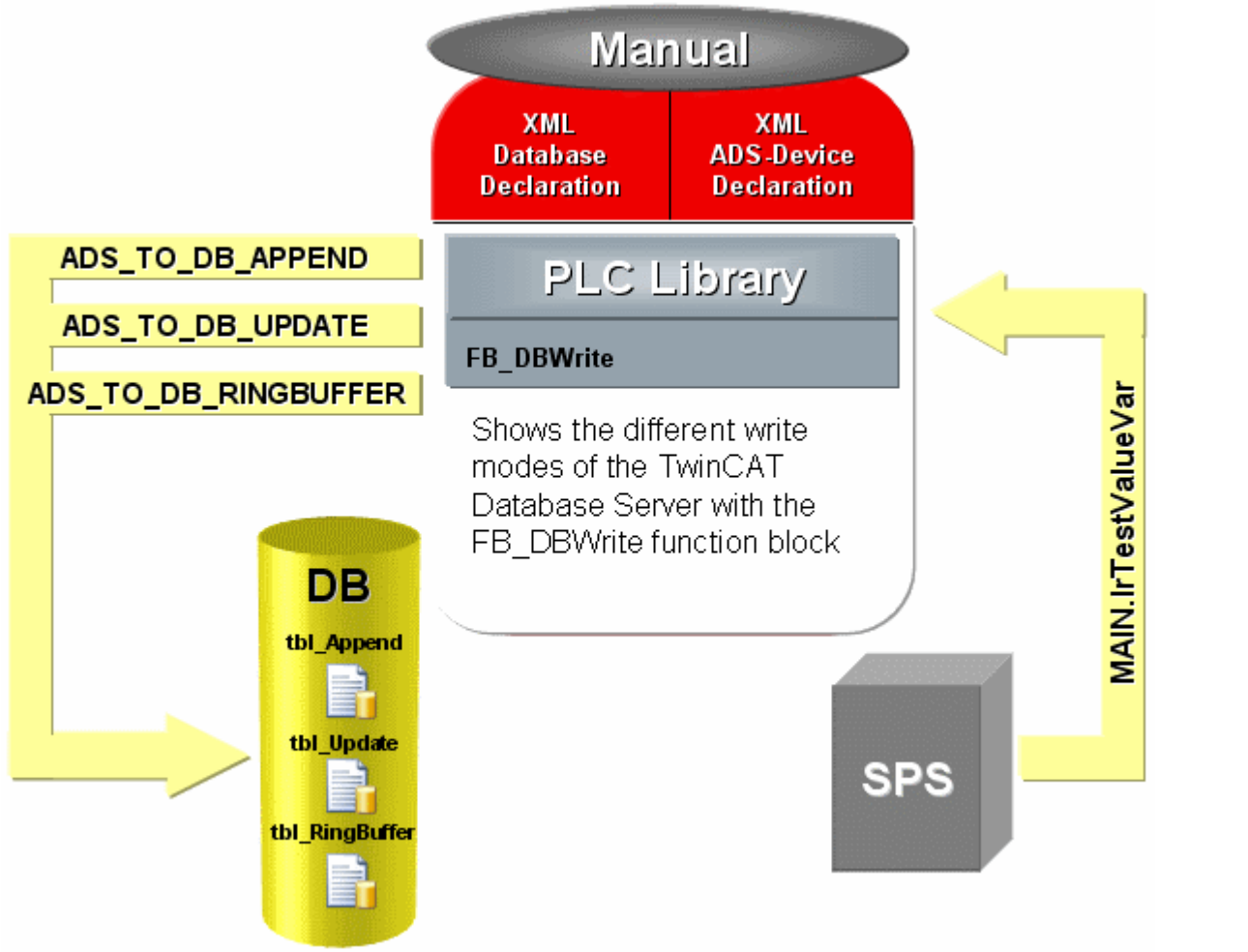
要求

开发环境	目标平台	要连接的 PLC 库
TwinCAT v3.0.0	PC 或 CX（x86）	Tc2_Database

7.2.3 使用 FB_DBWrite 对 PLC 变量进行记录

本示例说明了如何在数据库中对来自 PLC 的 PLC 变量进行记录以及各个写入模式的工作原理。

下载：https://infosys.beckhoff.com/content/1033/TF6420_TC3_Database_Server/Resources/3494041099.zip



使用的数据库类型	MS SQL
兼容的数据库类型	ASCII、MS SQL、MS Compact SQL、MS Access、MySQL、PostgreSQL、DB2、Oracle、InterBase/Firebird、XML
使用的功能块	FB_DBWrite
需要使用的库	Tc2_Database、Tc2_System、Tc2_Standard
下载文件列表	TcDBSrv_InfoSysSamples.tszip、CurrentConfigDataBase.xml、SQLQuery.sql

为了能够使用该示例，您必须在 XML 配置文件（CurrentConfigDataBase.xml）中调整服务器名称和身份验证。在执行 SQLQuery.sql 脚本之前，应确保不存在任何“TestDB”数据库。

示例配置：

变量“eWriteMode”可以用于设置数据采集的写入模式。
然后，通过变量“bSTART”的上升沿可以启动写入操作。

表格赋值：

- **ADS_TO_DB_APPEND** => eWriteAppend -> "tbl_Append"
- **ADS_TO_DB_UPDATE** => eWriteUpdate -> "tbl_Update"
- **ADS_TO_DB_RINGBUFFER** => eWriteRingBuffer -> "tbl_RingBuffer"

使用的表结构

ColumnName	Data type	允许为空	功能
ID	Bigint	否	IDENTITY(1,1)
Timestamp	DateTime	否	
Name	Ntext	否	
值	Float	否	

变量声明

```
PROGRAM MAIN
VAR
(*Test symbol which will be logged into the different database tables*)
    lrTestValueVar       : LREAL := 123.456;

    eState               : E_SampleState := eIdle;
    R_TRIG1              : R_TRIG;

(*With a rising edge at bStart the FB_DBWrite block will be start once*)
    bSTART               : BOOL;

(*With eWriteMode you can select which FB_DBWrite block will be used*)
    eWriteMode           : E_SampleState := eWriteAppend;

    FB_DBWrite_Append    : FB_DBWrite;
    FB_DBWrite_Update    : FB_DBWrite;
    FB_DBWrite_RingBuffer: FB_DBWrite;

(*Status outputs from the three FB_DBWrite blocks*)
    bBusy                : BOOL;
    bErr                 : BOOL;
    bErrid               : UDINT;
    stSqlstate           : ST_DBSQLError;
END_VAR
```

枚举 E_SampleState

```
TYPE E_SampleState :(
    eIdle           := 0,
    eWriteAppend    := 1,
    eWriteUpdate    := 2,
    eWriteRingBuffer:= 3
);
END_TYPE
```

PLC 程序

```
CASE eState OF
    eIdle :
        R_TRIG1(CLK:=bSTART);
        IF R_TRIG1.Q THEN
            lrTestValueVar  := lrTestValueVar + 1;
            eState          := eWriteMode;
            bSTART          := FALSE;
        END_IF
    (*Add a new record to the table tbl_Append*)
    eWriteAppend :
        FB_DBWrite_Append(
            sNetID          := ,
            hDBID           := 1,
            hAdsID          := 1,
            sVarName        := 'MAIN.lrTestValueVar',
            nIGroup         := ,
            nIOffset        := ,
            nVarSize        := ,
            sVarType        := ,
            sDBVarName      := 'lrTestValueVar',
            eDBWriteMode    := eDBWriteMode_Append,
            tRingBufferTime := ,
            nRingBufferCount:= ,
            bExecute        := TRUE,
            tTimeout        := T#15s,
            bBusy            => bBusy,
            bError           => bErr,
            nErrID           => bErrid,
            sSQLState        => stSqlstate);

        IF NOT bBusy THEN
            FB_DBWrite_Append(bExecute := FALSE);
            eState          := eIdle;
        END_IF
    (*Add a new record to the table tbl_Update if it not exist
     else the existing record will be updated*)
    eWriteUpdate :
        FB_DBWrite_Update(
```

```
            sNetID           := ,
            hDBID            := 2,
            hAdsID           := 1,
            sVarName         := 'MAIN.lrTestValueVar',
            nIGroup          := ,
            nIOffset         := ,
            nVarSize         := ,
            sVarType         := ,
            sDBVarName       := 'lrTestValueVar',
            eDBWriteMode     := eDBWriteMode_Update,
            tRingBufferTime  := ,
            nRingBufferCount := ,
            bExecute         := TRUE,
            tTimeout         := T#15s,
            bBusy             => bBusy,
            bError            => bErr,
            nErrID            => bErrid,
            sSQLState         => stSqlstate);

        IF NOT bBusy THEN
            FB_DBWrite_Update(bExecute := FALSE);
            eState           := eIdle;
        END_IF
    (*Add a new record to the table tbl_RingBuffer.
     If the maximum count is reached the records will be deleted in a FIFO process*)
    eWriteRingBuffer :
        FB_DBWrite_RingBuffer(
            sNetID           := ,
            hDBID            := 3,
            hAdsID           := 1,
            sVarName         := 'MAIN.lrTestValueVar',
            nIGroup          := ,
            nIOffset         := ,
            nVarSize         := ,
            sVarType         := ,
            sDBVarName       := 'lrTestValueVar',
            eDBWriteMode     := eDBWriteMode_RingBuffer_Count,
            tRingBufferTime  := ,
            nRingBufferCount := 10,
            bExecute         := TRUE,
            tTimeout         := T#15s,
            bBusy             => bBusy,
            bError            => bErr,
            nErrID            => bErrid,
            sSQLState         => stSqlstate);

        IF NOT bBusy THEN
            FB_DBWrite_RingBuffer(bExecute := FALSE);
            eState           := eIdle;
        END_IF
END_CASE
```

TwinCAT 数据库服务器

在 XML 配置文件中声明的所有 Microsoft SQL Compact 数据库必须存在。它们不是自动生成的。另一方面，如果声明的 ASCII 文件不存在，则会自动生成。

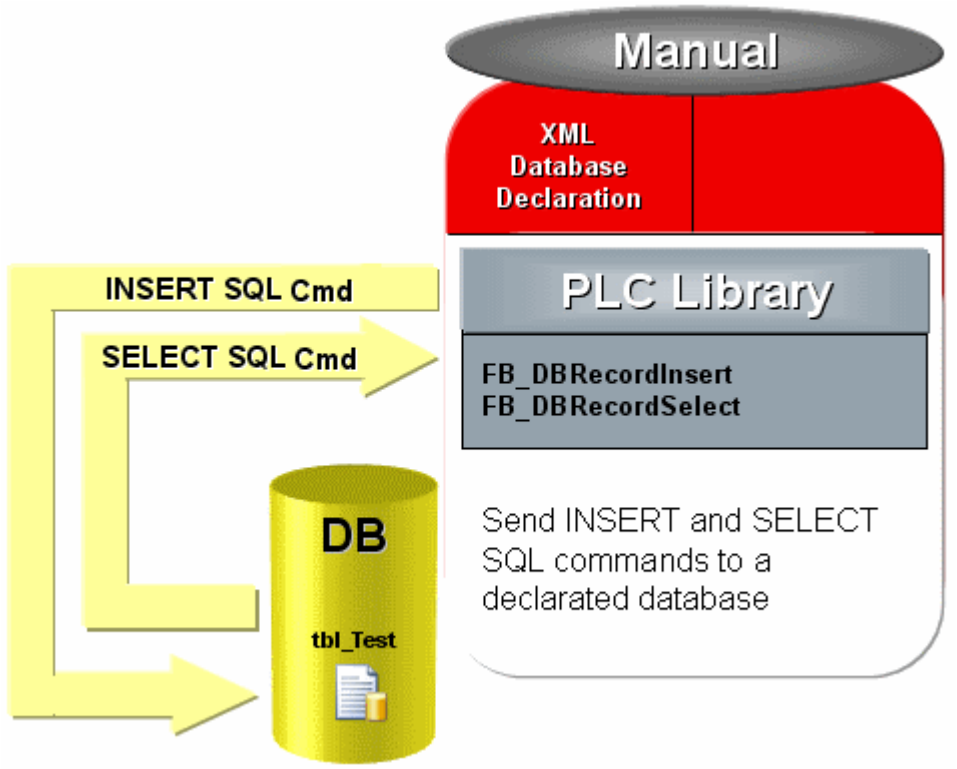
要求

开发环境	目标平台	要连接的 PLC 库
TwinCAT v3.0.0	PC 或 CX（x86）	Tc2_Database

7.2.4 使用 FB_DBRecordInsert 和 FB_DBRecordSelect 功能块的示例

本示例说明了如何使用功能块 FB_DBRecordInsert 在数据库中对来自 PLC 的多个值进行记录。在本示例中，多个 PLC 变量被记录在单个记录中。此外，功能块 FB_DBRecordSelect 可以用于从该数据库中读取记录。

下载：https://infosys.beckhoff.com/content/1033/TF6420_TC3_Database_Server/Resources/3494041099.zip



使用的数据库类型	MS Access
兼容的数据库类型	MS SQL、MS Compact SQL、MS Access、MySQL、PostgreSQL、DB2、Oracle、InterBase/Firebird、XML
使用的功能块	FB_DBRecordInsert、FB_DBRecordSelect
需要使用的库	“**Tc2_Database**”、“Tc2_System”、“Tc2_Standard”、“Tc2_Utilities”
下载文件列表	TcDBSrv_InfoSysSamples.tszip、CurrentConfigDataBase.xml、TestDB_Access.mdb

以下表结构可用于写入：

ColumnName	Data type
Timestamp	DateTime
PLC_TestValue1	float
PLC_TestValue2	float
PLC_TestValue3	float
PLC_TestValue4	String

变量声明

```
(* Declaration *)PROGRAM MAIN
VAR
    eState              : E_SQLStatement;

    NT_GetTime1         : NT_GetTime;
    bTimestart          : BOOL;
    tTime               : TIMESTRUCT;

    FB_FormatStringDateTime: FB_FormatString;
    sDateTimeString     : T_MaxString;

    TestValue1          : REAL := 123.456;
    TestValue2          : REAL := 234.567;
    TestValue3          : REAL := 345.678;
    TestValue4          : STRING(255) := 'No error occurred';

    FB_FormatString1    : FB_FormatString;
    sInsertString       : T_MaxString;
    bError              : BOOL;
```

```
    nErrid            : UDINT;

    FB_DBRecordInsert1: FB_DBRecordInsert;
    bStartstopInsert  : BOOL;
    bBusyInsert       : BOOL;
    bErrInsert        : BOOL;
    nErridInsert      : UDINT;
    stSQLStateInsert  : ST_DBSQLError;

    stRecord          : ST_Record;

    FB_DBRecordSelect1: FB_DBRecordSelect;
    nRecIndex         : UDINT := 0;
    bStartstopSelect  : BOOL;
    bBusySelect       : BOOL;
    bErrorSelect      : BOOL;
    nErrIDSelect      : UDINT;
    stSQLStateSelect  : ST_DBSQLError;
    nRecordCount      : UDINT;
END_VAR
```

枚举 E_SQLStatement

```
TYPEE_SQLStatement:(
    eSQL_INSERT  := 0,
    eSQL_SELECT  := 1
);
END_TYPE
```

结构体 ST_Record

```
TYPEST_Record :
STRUCT
    Timestamp : DT;
    PLC_Value1: REAL;
    PLC_Value2: REAL;
    PLC_Value3: REAL;
    PLC_Value4: STRING;
END_STRUCT
END_TYPE
```

PLC 程序

```
CASEeState OF
    eSQL_INSERT:
        (*Create the timestamp*)
        NT_GetTime1( START:= bTimestart, TIMESTR=> tTime);
        IF NOT NT_GetTime1.BUSY THEN
            bTimestart:= NOT bTimestart;
        END_IF

        FB_FormatStringDateTime(
            sFormat   := '%D.%D.%D %D:%D:%D',
            arg1      := F_WORD(tTime.wYear),
            arg2      := F_WORD(tTime.wMonth),
            arg3      := F_WORD(tTime.wDay),
            arg4      := F_WORD(tTime.wHour),
            arg5      := F_WORD(tTime.wMinute),
            arg6      := F_WORD(tTime.wSecond),
            sOut       => sDateTimeString);

        (*Create the SQL-INSERT command*)
        FB_FormatString1(
            sFormat   := 'INSERT INTO tbl_Test VALUES($'%S$',%F,%F,%F,$'%S$')',
            arg1      := F_STRING(sDateTimeString),
            arg2      := F_REAL(TestValue1),
            arg3      := F_REAL(TestValue2),
            arg4      := F_REAL(TestValue3),
            arg5      := F_STRING(TestValue4),
            sOut       => sInsertString,
            bError     => bError,
            nErrId     => nErrid);

        (*Write the record to the database*)
        FB_DBRecordInsert1(
            sNetID    := ,
            hDBID     := 1,
            sInsertCmd:= sInsertString,
            bExecute  := bStartstopInsert,
            tTimeout  := T#15s,
            bBusy      => bBusyInsert,
            bError     => bErrInsert,
            nErrID     => nErridInsert,
            sSQLState  => stSQLStateInsert);

    eSQL_SELECT:
        (*Read one record from the database*)
        FB_DBRecordSelect1(
            sNetID    := ,
            hDBID     := 1,
            sSelectCmd:= 'SELECT * FROM tbl_Test',
```

```
            nRecordIndex:= nRecIndex,
            cbRecordSize:= SIZEOF(stRecord),
            pDestAddr := ADR(stRecord),
            bExecute  := bStartstopSelect,
            tTimeout  := T#15s,
            bBusy      => bBusySelect,
            bError     => bErrorSelect,
            nErrID     => nErrIDSelect,
            sSQLState  => stSQLStateSelect,
            nRecords   => nRecordCount);
END_CASE
```

要使用此示例，您必须在 XML 配置文件中声明 Access 数据库“Sample7.mdb”。
通过在变量“bStartstopInsert”处生成上升沿，在数据库中可以创建一个带有 4 个 PLC 值和时间戳的记录。

tbl_Test

Timestamp	PLC_Value1	PLC_Value2	PLC_Value3	PLC_Value4
06.09.2012 10:03:34	123,456	234,567	345,678	No error occurred
06.09.2012 10:03:38	123,456	234,567	345,678	No error occurred
06.09.2012 10:04:12	123,456	234,567	345,678	No error occurred
06.09.2012 10:04:23	123,456	234,567	345,678	No error occurred
06.09.2012 10:05:30	123,456	234,567	345,678	No error occurred
06.09.2012 10:05:43	123,456	234,567	345,678	No error occurred
*	0			

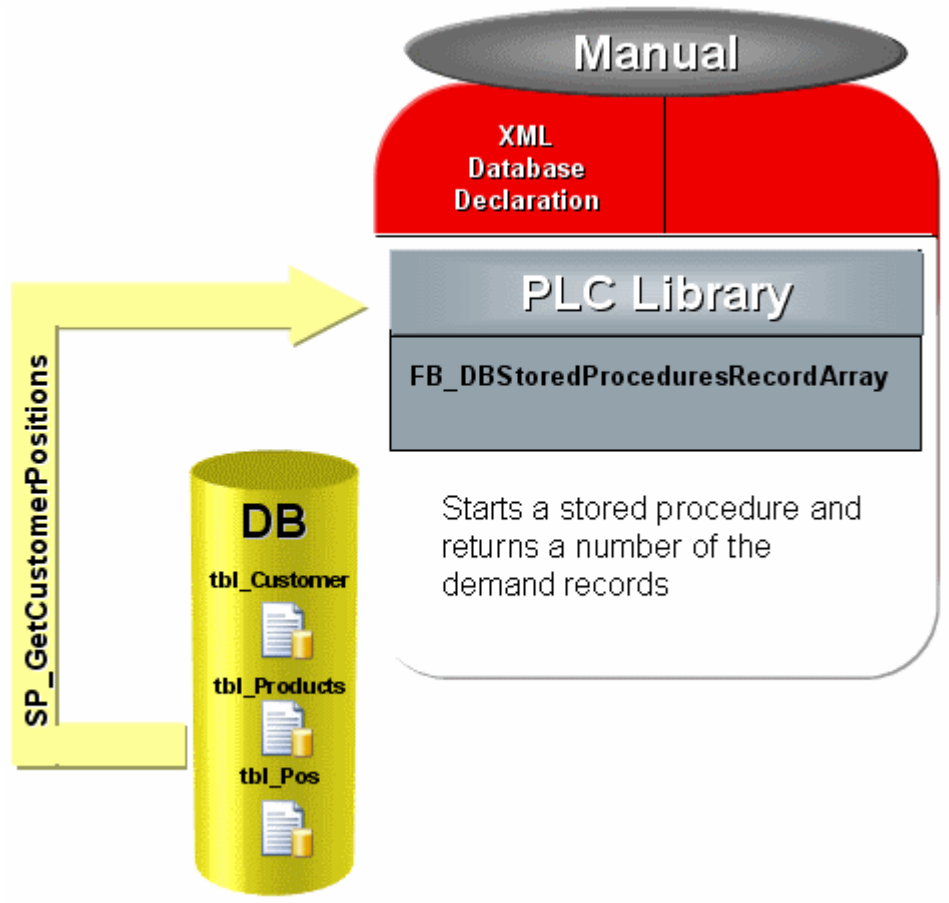
要求

开发环境	目标平台	要连接的 PLC 库
TwinCAT v3.0.0	PC 或 CX（x86）	Tc2_Database

7.2.5 使用 FB_DBStoredProceduresRecordArray 的存储过程

功能块 FB_DBStoredProceduresRecordArray 可以用于将参数声明为 INPUT、OUTPUT 或 INOUT，并将它们传输到存储过程。这样可以在数据库服务器中预先编程复杂的 SQL 命令，然后由 TwinCAT 数据库服务器触发。与功能块 FB_DBStoredProceduresRecordReturn 相比，该功能块可以用于通过一次调用返回多个记录。

下载：https://infosys.beckhoff.com/content/1033/TF6420_TC3_Database_Server/Resources/3494041099.zip



使用的数据库类型	MS SQL（MS SQL 服务器 2008）
兼容的数据库类型	MS SQL、MySQL、Oracle
使用的功能块	FB_DBStoredProceduresRecordArray
需要使用的库	Tc2_Database、Tc2_System、Tc2_Base、Tc2_Utilities
下载文件列表	TcDBSrv_InfoSysSamples.tszip、CurrentConfigDataBase.xml

以下示例说明了一个具有输入参数和返回记录的简单存储过程中的调用。该过程是在 Microsoft SQL 服务器 2008 上创建的。

存储过程 SP_GetAddressByCustomerID 的代码

```
CREATE PROCEDURE [SP_GetAddressByCustomerID]
    @Customer_ID bigint
AS
BEGIN
    SELECT tbl_Customer.ID, tbl_Customer.Name, tbl_Customer.Customer, tbl_Products.SerNum,
     tbl_Products.Product, tbl_Products.Info, tbl_Pos.Timestamp
    FROM
        tbl_Pos JOIN tbl_Customer ON tbl_Pos.CustomerNum = tbl_Customer.ID
        JOIN tbl_Products ON tbl_Pos.ProductNum = tbl_Products.SerNum
    WHERE
        tbl_Pos.CustomerNum = @Customer_ID;
END
```

PLC 中的变量声明

```
PROGRAM MAIN
VAR
    R_TRIG1          : R_TRIG;
    bREAD            : BOOL := FALSE;

    nState           : BYTE;

    arrParaList      : ARRAY [0..0] OF ST_DBParameter;

    nCustomerID      : DINT := 12345;
```

```
    FB_DBStoredProceduresRecordArray1: FB_DBStoredProceduresRecordArray;

    nCustomerID: DINT:= 12345;
    nRecordStartIndex: UDINT;
    stRecordArr      : ARRAY [1..25] OF ST_Record;
    nRecs            : UDINT;

    bBusy            : BOOL;
    bErr             : BOOL;
    nErrid           : UDINT;
    stSqlstate       : ST_DBSQLError;
END_VAR
```

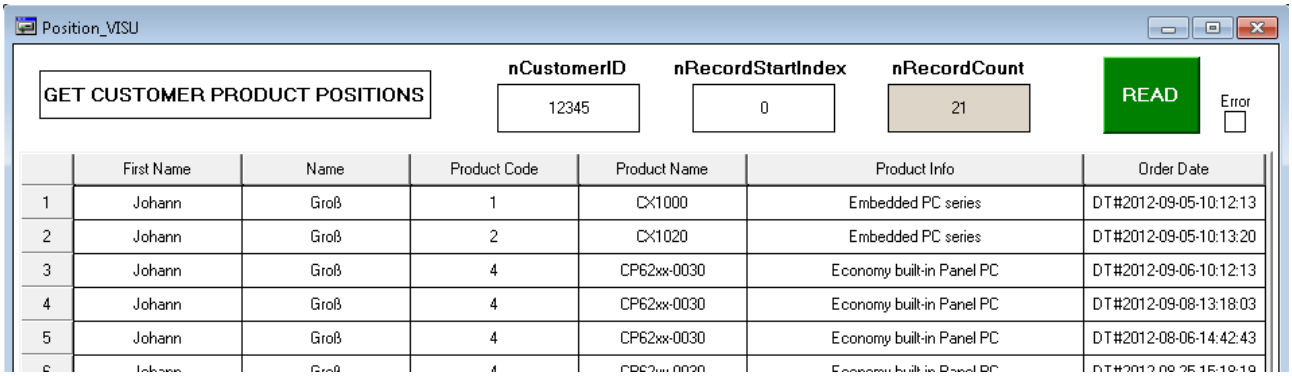
PLC 中的记录结构（ST_Record）

```
TYPE ST_Record :
STRUCT
    nID          : T_ULARGE_INTEGER;
    sCustomer    : STRING(50);
    sName        : STRING(50);
    nProductNum  : DINT;
    sProductName: STRING(50);
    sProductInfo: T_MaxString;
    tTimestamp   : DT;
END_STRUCT
END_TYPE
```

PLC 程序

```
R_TRIG1(CLK:=bREAD);
IF R_TRIG1.Q AND NOT bBusy THEN
    nState := 1;
END_IF

CASE nState OF
 0:
    ;
 1:(*Init of the parameters*)
    arrParaList[0].sParameterName    := '@Customer_ID';
    arrParaList[0].eParameterDataType:= eDBColumn_Integer;
    arrParaList[0].eParameterType    := eDBParameter_Input;
    arrParaList[0].cbParameterValue  := SIZEOF(nCustomerID);
    arrParaList[0].pParameterValue   := ADR(nCustomerID);

    nState := 2;
 2:(*Start the stored procedure "SP_GetCustomerPosition"*)
    FB_DBStoredProceduresRecordArray1(
        sNetID:= ,
        hDBID:= 1,
        sProcedureName   := 'SP_GetCustomerPositions',
        cbParameterList  := SIZEOF(arrParaList),
        pParameterList   := ADR(arrParaList),
        nStartIndex      := nRecordStartIndex,
        nRecordCount     := 25,
        cbRecordArraySize:= SIZEOF(stRecordArr),
        pDestAddr        := ADR(stRecordArr),
        bExecute         := TRUE,
        tTimeout         := T#15s,
        bBusy             => bBusy,
        bError            => bErr,
        nErrID            => nErrid,
        sSQLState         => stSqlstate,
        nRecords          => nRecs);

    IF NOT bBusy THEN
        FB_DBStoredProceduresRecordReturn1(bExecute:= FALSE);
        nState := 0;
    END_IF
END_CASE
```

可视化



	First Name	Name	Product Code	Product Name	Product Info	Order Date
1	Johann	Groß	1	CX1000	Embedded PC series	DT#2012-09-05-10:12:13
2	Johann	Groß	2	CX1020	Embedded PC series	DT#2012-09-05-10:13:20
3	Johann	Groß	4	CP62xx-0030	Economy built-in Panel PC	DT#2012-09-06-10:12:13
4	Johann	Groß	4	CP62xx-0030	Economy built-in Panel PC	DT#2012-09-08-13:18:03
5	Johann	Groß	4	CP62xx-0030	Economy built-in Panel PC	DT#2012-08-06-14:42:43

要求

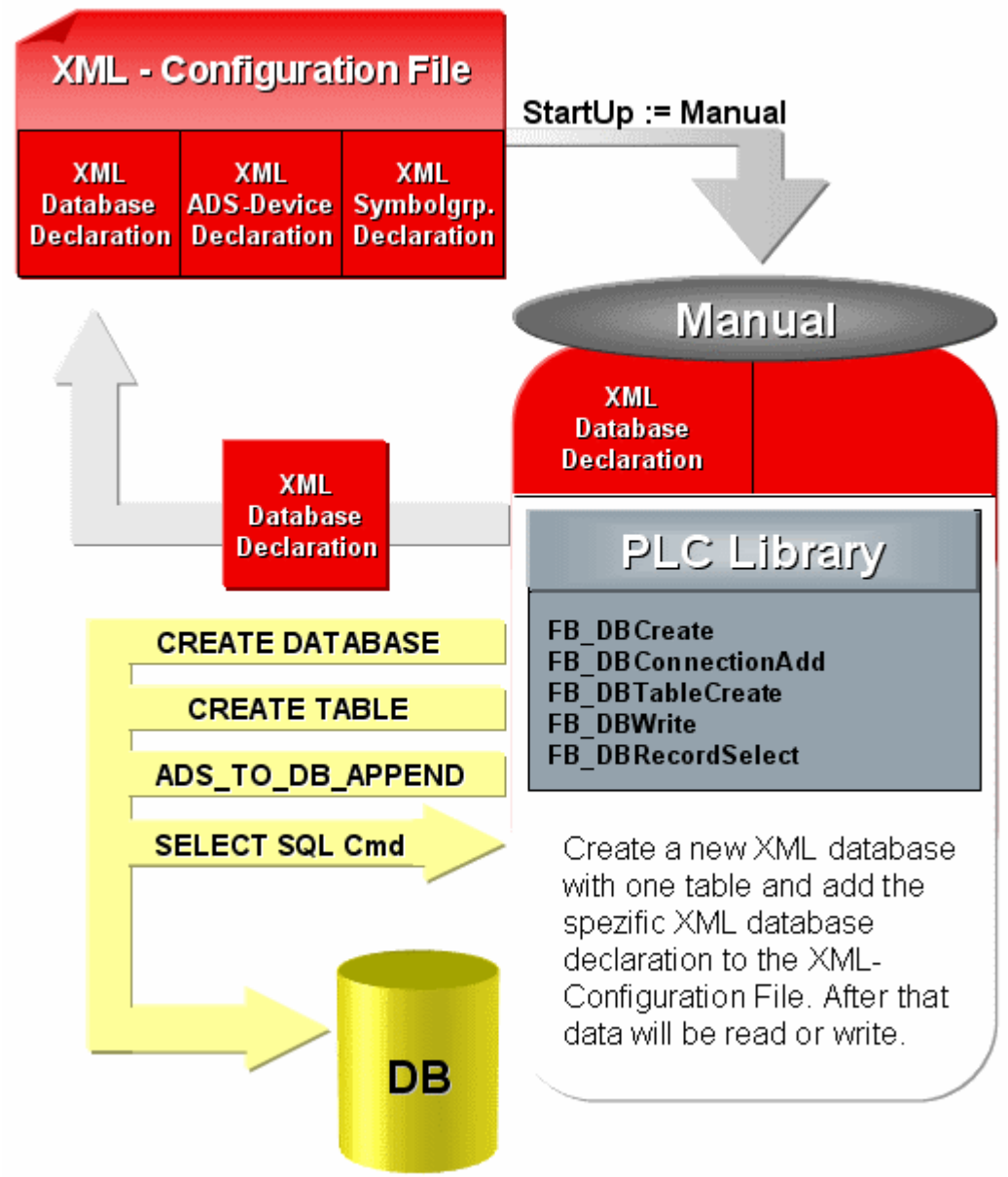
开发环境	目标平台	要连接的 PLC 库
TwinCAT v3.0.0	PC 或 CX（x86）	Tc2_Database

7.2.6 使用 XML 作为数据库

TwinCAT 数据库服务器能够将 XML 文件用作数据库。除“存储过程”功能外，XML 数据库类型还支持所有已知的用于在数据库中读取和写入的功能块。TwinCAT 数据库服务器甚至可以解释那些使用功能块 FB_DBRecordInsert 或 FB_DBRecordSelect 发出的 SQL 命令，并将它们应用到 XML 文件中。

本示例介绍了如何创建 XML 数据库、使用功能块 FB_DBWrite 进行填充，以及使用 SQL SELECT 命令和功能块 FB_DBRecordSelect 进行读取。

下载：https://infosys.beckhoff.com/content/1033/TF6420_TC3_Database_Server/Resources/3494041099.zip



使用的数据库类型	XML
兼容的数据库类型	MS SQL、MS Compact SQL、MS Access、XML
使用的功能块	FB_DBCreate、FB_DBConnectionAdd、FB_DBTableCreate、FB_DBWrite、FB_DBRecordSelect
需要使用的库	“**Tc2_Database**”、“Tc2_System”、“Tc2_Standard”、“Tc2_Utilities”
下载文件列表	TcDBSrv_InfoSysSamples.tszip

MAIN 程序

```
PROGRAM MAIN
VAR
    nState              :BYTE := 0;

    R_TRIG1             : R_TRIG;
    bSTART              : BOOL;

    nCounter            : INT;

    FB_FileDelete1      : FB_FileDelete;
    FB_DBCreate1        : FB_DBCreate;
    FB_DBConnectionAdd1: FB_DBConnectionAdd;
    FB_DBTableCreate1   : FB_DBTableCreate;
    FB_DBWrite1         : FB_DBWrite;
    FB_DBRecordSelect1  : FB_DBRecordSelect;

    bBusy_Delete        : BOOL;
    bBusy_CreateDB      : BOOL;
    bBusy_ConnAdd       : BOOL;
    bBusy_CreateTable   : BOOL;
    bBusy_WriteDB       : BOOL;
    bBusy_SelectRecord  : BOOL;

    bErr                : BOOL;
    nErrid              : UDINT;
    stSQLState          : ST_DBSQLError;
    nRecs               : UDINT;

    nDBid               : UDINT;

    arrTablestrc        : ARRAY [0..3] OF ST_DBColumnCfg :=
     [(sColumnName:='ID',sColumnProperty:='IDENTITY(1,1)',eColumnType:=EDBCOLUMN_BIGINT),
     (sColumnName:='Timestamp',eColumnType:=EDBCOLUMN_DATETIME),
     (sColumnName:='Name',sColumnProperty:='80',eColumnType:=EDBCOLUMN_NTEXT),
     (sColumnName:='Value',eColumnType:=EDBCOLUMN_FLOAT)];

    rTestValue          : LREAL := 1234.56789;
    stRecord            : ST_Record;
END_VAR
```

```
CASE nState OF
    0:
        (*To start this sample you have to set a rising edge to the variable bSTART*)
        R_TRIG1(CLK:=bSTART);
        IF R_TRIG1.Q THEN
            nState  := 1;
            FB_FileDelete1(bExecute:=FALSE);
            FB_DBCreate1(bExecute:=FALSE);
            FB_DBConnectionAdd1(bExecute:=FALSE);
            FB_DBTableCreate1(bExecute:=FALSE);
            FB_DBWrite1(bExecute:=FALSE);
            FB_DBRecordSelect1(bExecute:=FALSE);
            bSTART  := FALSE;
            nCounter:= 0;
        END_IF
    1:
        (*It isn't possible to overwrite an existing database file.
         If the database file exist the FB_FileDelete block will delete the file*)
        FB_FileDelete1(
            sNetId   := ,
            sPathName:= 'C:\TwinCAT\TcDatabaseSrv\Samples\XMLTestDB.xml',
            ePath    := PATH_GENERIC,
            bExecute := TRUE,
            tTimeout := T#5s,
            bBusy    => bBusy_Delete,
            bError   => ,
            nErrId   => );

        IF NOT bBusy_Delete THEN
            nState   := 10;
        END_IF
    10:
        (*It isn't possible to overwrite an existing database file.
         If the database file exist the FB_FileDelete block will delete the file*)
        FB_FileDelete1(
            sNetId   := ,
```

```
            sPathName:= 'C:\TwinCAT\TcDatabaseSrv\Samples\XMLTestDB.xsd',
            ePath    := PATH_GENERIC,
            bExecute := TRUE,
            tTimeout := T#5s,
            bBusy     => bBusy_Delete,
            bError    => ,
            nErrId    => );

        IF NOT bBusy_Delete THEN
            FB_FileDelete1(bExecute:=FALSE);
            nState   := 2;
        END_IF
    2:
        (*The FB_DBCreate block will create the database file
         "C:\TwinCAT\TcDatabaseSrv\Samples\XMLTestDB.xml" and
         C:\TwinCAT\TcDatabaseSrv\Samples\XMLTestDB.xsd "*)
        FB_DBCreate1(
            sNetID   := ,
            sPathName:= 'C:\TwinCAT\TcDatabaseSrv\Samples',
            sDBName  := 'XMLTestDB',
            eDBType  := eDBType_XML,
            bExecute := TRUE,
            tTimeout := T#15s,
            bBusy     => bBusy_CreateDB,
            bError    => bErr,
            nErrID    => nErrid);

        IF NOT bBusy_CreateDB AND NOT bErr THEN
            nState   := 3;
        END_IF
    3:
        (*The FB_DBConnectionAdd adds the connection information to the
         XML configuration file*)
        (*ATTENTION: Each database type has his own connection information*)
        FB_DBConnectionAdd1(
            sNetID      := ,
            eDBType     := eDBType_XML,
            eDBValueType:= eDBValue_Double,
            sDBServer   := 'XMLTestDB',
            sDBProvider := ,
            sDBUrl      := 'C:\TwinCAT\TcDatabaseSrv\Samples\XMLTestDB.xml',
            sDBTable    := 'myTable',
            bExecute    := TRUE,
            tTimeout    := T#15s,
            bBusy        => bBusy_ConnAdd,
            bError       => bErr,
            nErrID       => nErrid,
            hDBID        => nDBid);

        IF NOT bBusy_ConnAdd AND NOT bErr THEN
            nState := 4;
        END_IF
    4:
        (*The FB_DBTableCreate create the table "myTable"*)
        FB_DBTableCreate1(
            sNetID      := ,
            hDBID       := nDBid,
            sTableName  := 'myTable',
            cbTableCfg  := SIZEOF(arrTablestrc),
            pTableCfg   := ADR(arrTablestrc),
            bExecute    := TRUE,
            tTimeout    := T#15s,
            bBusy        => bBusy_CreateTable,
            bError       => bErr,
            nErrID       => nErrid);

         IF NOTbBusy_CreateTable AND NOT bErr THEN
            nState      := 5;
        END_IF
    5:
        (*The FB_DBWrite write five times the value of the plc variable "rTestValue" to
         the database table "myTable"*)
        FB_DBWrite1(
            sNetID          := ,
            hDBID           := nDBid,
            hAdsID          := 1,
            sVarName        := 'MAIN.rTestValue',
            nIGroup         := ,
            nIOffset        := ,
            nVarSize        := ,
            sVarType        := ,
            sDBVarName      := 'rTestValue',
            eDBWriteMode    := eDBWriteMode_Append,
            tRingBufferTime := ,
            nRingBufferCount:= ,
            bExecute        := TRUE,
            tTimeout        := T#15s,
            bBusy            => bBusy_WriteDB,
            bError           => bErr,
            nErrID           => nErrid,
            sSQLState        => stSQLState);

        IF NOT bBusy_WriteDB AND NOT bErr THEN
            FB_DBWrite1(bExecute := FALSE);
```

```
            nCounter        := nCounter + 1;
            IFnCounter = 5 THEN
                nState      := 6;
            END_IF
        END_IF
    6:
        (*The FB_DBRecordSelect select one record of the database table "myTable""*)
        FB_DBRecordSelect1(
            sNetID          := ,
            hDBID           := nDBid,
            sSelectCmd      := 'SELECT * FROM myTable WHERE Name = $'rTestValue$'',
            nRecordIndex    := 0,
            cbRecordSize    := SIZEOF(stRecord),
            pDestAddr       := ADR(stRecord),
            bExecute        := TRUE,
            tTimeout        := T#15s,
            bBusy            => bBusy_SelectRecord,
            bError           => bErr,
            nErrID           => nErrid,
            sSQLState        => stSQLState,
            nRecords         => nRecs);

        IF NOT bBusy_SelectRecord AND NOT bErr THEN
            nState          := 0;
        END_IF
    END_CASE
```

该过程随着切换变量 bSTART 的上升沿开始。

创建以下文件：

XMLTestDB.xml（XML 数据库文件）

```
<?xmlversion="1.0"encoding="UTF-8"?>
<XMLTestDBxmlns:xs="http://www.w3.org/2001/XMLSchema-
instance"xs:noNamespaceSchemaLocation="XMLTestDB.xsd">
  <myTable>
    <rowID="1"Timestamp="2012-05-10T13:48:47"Name="rTestValue"Value="1234.56789" />
    <rowID="2"Timestamp="2012-05-10T13:48:47"Name="rTestValue"Value="1234.56789" />
    <rowID="3"Timestamp="2012-05-10T13:48:47"Name="rTestValue"Value="1234.56789" />
    <rowID="4"Timestamp="2012-05-10T13:48:47"Name="rTestValue"Value="1234.56789" />
    <rowID="5"Timestamp="2012-05-10T13:48:47"Name="rTestValue"Value="1234.56789" />
  </myTable>
</XMLTestDB>
```

XMLTestDB.xsd（XML 模式）

```
<?xmlversion="1.0"?>
<xsd:schemaxmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:simpleTypename="bigint">
    <xsd:restrictionbase="xsd:long" />
  </xsd:simpleType>
  <xsd:simpleTypename="datetime">
    <xsd:restrictionbase="xsd:dateTime" />
  </xsd:simpleType>
  <xsd:simpleTypename="ntext_80">
    <xsd:restrictionbase="xsd:string">
      <xsd:maxLengthvalue="80" />
    </xsd:restriction>
  </xsd:simpleType>
  <xsd:simpleTypename="float">
    <xsd:restrictionbase="xsd:double" />
  </xsd:simpleType>
  <xsd:complexTypename="myTable_Type">
    <xsd:sequence>
      <xsd:elementminOccurs="0"maxOccurs="unbounded"name="row">
        <xsd:complexType>
          <xsd:attributename="ID"type="bigint" />
          <xsd:attributename="Timestamp"type="datetime" />
          <xsd:attributename="Name"type="ntext_80" />
          <xsd:attributename="Value" type="float" />
        </xsd:complexType>
      </xsd:element>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:elementname="XMLTestDB">
    <xsd:complexType>
      <xsd:sequenceminOccurs="1"maxOccurs="1">
        <xsd:elementname="myTable"type="myTable_Type" />
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

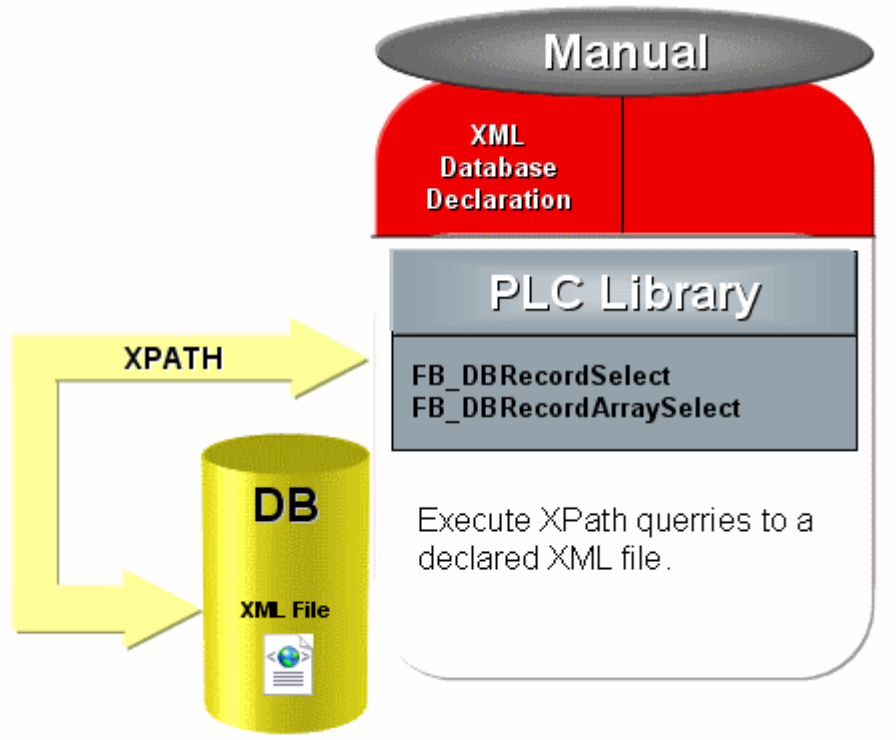
要求

开发环境	目标平台	要连接的 PLC 库
TwinCAT v3.0.0	PC 或 CX（x86）	Tc2_Database

7.2.7 说明不同 SELECT 类型的 XPath 示例

功能块 FB_DBRecordArraySelect/FB_DBRecordSelect 可以用于发出 XPath 命令并从任何 XML 文件中读取 XML 标签。本示例说明了如何通过 TwinCAT 数据库服务器从 XML 文件中读取不同的条目。支持单独的标签、子标签和属性的读取并显示。

下载：https://infosys.beckhoff.com/content/1033/TF6420_TC3_Database_Server/Resources/3494041099.zip



使用的数据库类型	XML
兼容的数据库类型	XML
使用的功能块	FB_DBRecordArraySelect
需要使用的库	Tc2_Database、Tc2_System、Tc2_Standard、Tc2_Utilities
下载文件列表	TcDBSrv_InfoSysSamples.tszip

示例 XML 文件（XMLFactoryXY.xml）

```
<?xmlversion="1.0" encoding="utf-8" ?>
<Factory_XY>
<Name>Sample Factory XY</Name>
<Factory_Info>
<Street>Samplestreet 25</Street>
<City>33415 Verl</City>
<Country>Germany</Country>
<Office_Count>1</Office_Count>
<Employe_Count>6</Employe_Count>
<Manager>Max Mustermann</Manager>
</Factory_Info>
<Employees>
<Employeeid="10001" name="Julia Kingston" department="Development" position="Worker"
hired="2001-08-01" />
<Employeeid="10002" name="Jens Marx" department="Import" position="Worker" hired="2003-08-01" />
<Employeeid="10003" name="Justus Kaiser" department="Export" position="Worker" hired="2003-08-01" />
<Employeeid="10004" name="Marc Klein" department="Production" position="Worker" hired="2005-08-01" /
>
<Employeeid="10005" name="Matt Bloomberg" department="Production" position="Worker"
```

```
hired="2005-08-01" />
<Employeeid="10006" name="Frida Hundt" department="Production" position="Worker"
hired="2010-08-01" />
</Employees>
</Factory_XY>
```

ST_FactoryInfo 结构体

```
TYPEST_FactoryInfo :
STRUCT
 sStreet       : T_MaxString;
 sCity         : T_MaxString;
 sCountry      : T_MaxString;
 sOffice_Count : T_MaxString;
 sEmploye_Count: T_MaxString;
 sManager      : T_MaxString;
END_STRUCT
END_TYPE
```

ST_Employee 结构体

```
TYPEST_Employee :
STRUCT
 sID          : T_MaxString;
 sName        : T_MaxString;
 sDepartment  : T_MaxString;
 sPosition    : T_MaxString;
 sHired       : T_MaxString;
END_STRUCT
END_TYPE
```

MAIN 程序

```
PROGRAM MAIN
VAR
     bSTART    : BOOL;
     R_TRIG1   : R_TRIG;

     nState    : INT;

     sXPath    : T_MaxString;

     fbDBRecordArraySelect : FB_DBRecordArraySelect;

     bBusy_ReadFactoryName : BOOL;
     bError_ReadFactoryName: BOOL;
     nErrID_ReadFactoryName: UDINT;

     bBusy_ReadFactoryInfo : BOOL;
     bError_ReadFactoryInfo: BOOL;
     nErrID_ReadFactoryInfo: UDINT;

     bBusy_ReadEmployee    : BOOL;
     bError_ReadEmployee   : BOOL;
     nErrID_ReadEmployee   : UDINT;

     stSQLState            : ST_DBSQLError;

     sFactoryName          : T_MaxString;
     stFactoryInfo         : ST_FactoryInfo;
     aEmployees            : ARRAY [1..10] OF ST_Employee;
END_VAR
```

```
R_TRIG1(CLK:=bSTART);
IF R_TRIG1.Q THEN
     bSTART:=FALSE;
     fbDBRecordArraySelect(bExecute:=FALSE);
     nState:=1;
END_IF
```

```
CASE nState OF
     0://IDLE
         ;
     1://Read Factory Name
         sXPath:= 'XPATH#Factory_XY/Name';
         fbDBRecordArraySelect(
             sNetID           := ,
             hDBID            := 7,
             pCmdAddr         := ADR(sXPath),
             cbCmdSize        := SIZEOF(sXPath),
             nStartIndex      := 0,
             nRecordCount     := 1,
             pDestAddr        := ADR(sFactoryName),
             cbRecordArraySize:= SIZEOF(sFactoryName),
             bExecute         := TRUE,
             tTimeout         := T#15S,
             bBusy             => bBusy_ReadFactoryName,
             bError            => bError_ReadFactoryName,
             nErrID            => nErrID_ReadFactoryName,
             sSQLState         => stSQLState,
             nRecords          => );
```

```
        IF NOT bBusy_ReadFactoryName THEN
            fbDBRecordArraySelect(bExecute:=FALSE);
            IF NOT bError_ReadFactoryName THEN
                nState        :=2;
            ELSE
                nState        :=255;
            END_IFEND_IF
    2://Read Factory Info
        sXPath                 := 'XPATH#Factory_XY/Factory_Info';
        fbDBRecordArraySelect(
            sNetID             := ,
            hDBID              := 7,
            pCmdAddr           := ADR(sXPath),
            cbCmdSize          := SIZEOF(sXPath),
            nStartIndex        := 0,
            nRecordCount       := 1,
            pDestAddr          := ADR(stFactoryInfo),
            cbRecordArraySize  := SIZEOF(stFactoryInfo),
            bExecute           := TRUE,
            tTimeout           := T#15S,
            bBusy               => bBusy_ReadFactoryInfo,
            bError              => bError_ReadFactoryInfo,
            nErrID              => nErrID_ReadFactoryInfo,
            sSQLState           => stSQLState,
            nRecords            => );

        IF NOT bBusy_ReadFactoryInfo THEN
            fbDBRecordArraySelect(bExecute:=FALSE);
            IF NOT bError_ReadFactoryInfo THEN
                nState         :=3;
            ELSE
                nState         :=255;
            END_IF
        END_IF
    3://Read Employees
        sXPath                  := 'XPATH#Factory_XY/Employees/Employee';
        fbDBRecordArraySelect(
            sNetID             := ,
            hDBID              := 7,
            pCmdAddr           := ADR(sXPath),
            cbCmdSize          := SIZEOF(sXPath),
            nStartIndex        := 0,
            nRecordCount       := 10,
            pDestAddr          := ADR(aEmployees),
            cbRecordArraySize  := SIZEOF(aEmployees),
            bExecute           := TRUE,
            tTimeout           := T#15S,
            bBusy               => bBusy_ReadEmployee,
            bError              => bError_ReadEmployee,
            nErrID              => nErrID_ReadEmployee,
            sSQLState           => stSQLState,
            nRecords            => );

        IF NOT bBusy_ReadEmployee THEN
            fbDBRecordArraySelect(bExecute:=FALSE);
            IF NOT bError_ReadEmployee THEN
                nState         :=0;
            ELSE
                nState         :=255;
            END_IFEND_IF
    255://Error State
        ;
END_CASE
```

变量“bStart”的上升沿可触发发出 XPath 命令，并从 XML 文件中读取各个元素。然后，结果将在变量“sFactoryName”、“stFactoryInfo”和“aEmployees”中显示。

要求

开发环境	目标平台	要连接的 PLC 库
TwinCAT v3.0.0	PC 或 CX（x86）	Tc2_Database

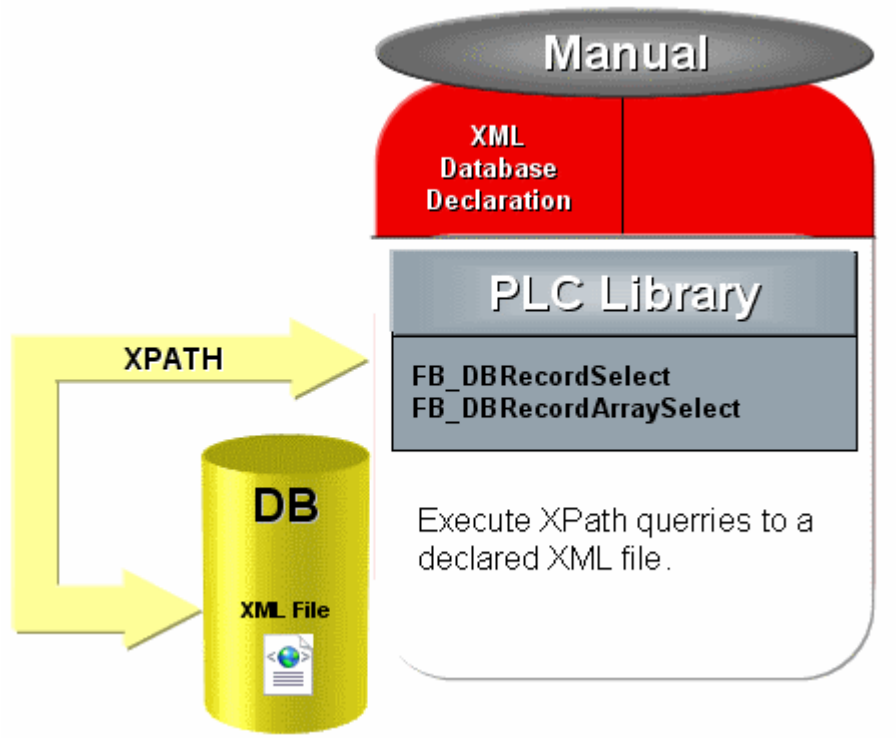
7.2.8 具有 XML 模式的 XPath 示例

功能块 FB_DBRecordSelect 或 FB_DBRecordArraySelect 可以用于发出 XPath 命令并从任何 XML 文件中读取 XML 标签、XML 子标签或 XML 属性。如果存在适合要读取的 XML 文件的 XML 模式，则标签或属性的内容会转换为在模式中定义的相应数据类型。

有关 XML 模式的更多信息，请访问： http://www.edition-w3.de/TR/2001/REC-xmlschema-0-20010502/

在本示例中，FB_DBRecordArraySelect 可用于从具有相应 XML 模式的 XML 文件中读取 2 个不同的子标签。

下载：https://infosys.beckhoff.com/content/1033/TF6420_TC3_Database_Server/Resources/3494041099.zip



使用的数据库类型	XML
兼容的数据库类型	XML
使用的功能块	FB_DBRecordSelect
需要使用的库	Tc2_Database、Tc2_System、Tc2_Standard、Tc2_Utilities
下载文件列表	TcDBSrv_InfoSysSamples.tszip、CurrentConfigDatabase.xml、PLC_Structs.xml、PLC_Structs.xsd

示例 XML 文件（PLC_Structs.xml）

```
<?xml version = "1.0" encoding="utf-8" ?>
<Beckhoff_PLC>
  <PLC_Structs>
    <PLC_Struct Name="ST_TestStruct">
      <Struct Instance="1">
        <nINT64>123456789</nINT64>
        <nUINT16>1234</nUINT16>
        <rREAL64>1234.5678</rREAL64>
        <sSTRING>This is instance one of ST_TestStruct</sSTRING>
        <bBOOL>true</bBOOL>
        <nINT32>-100</nINT32>
      </Struct>
      <Struct Instance="2">
        <nINT64>234567890</nINT64>
        <nUINT16>2345</nUINT16>
        <rREAL64>234.56789</rREAL64>
        <sSTRING>This is instance two of ST_TestStruct</sSTRING>
        <bBOOL>false</bBOOL>
        <nINT32>-50</nINT32>
      </Struct>
      <Struct Instance="3">
        <nINT64>345678901</nINT64>
        <nUINT16>3456</nUINT16>
        <rREAL64>3456.78901</rREAL64>
        <sSTRING>This is instance three of ST_TestStruct</sSTRING>
        <bBOOL>true</bBOOL>
        <nINT32>-150</nINT32>
      </Struct>
    </PLC_Struct>
    <PLC_Struct Name="ST_TestStruct2">
      <Struct2 Instance="1">
        <sSTRING>This is instance one of ST_TestStruct2</sSTRING>
        <bBOOL>false</bBOOL>
        <nINT32>-88</nINT32>
```

```
      </Struct2>
      <Struct2 Instance="2">
        <sSTRING>This is instance two of ST_TestStruct2</sSTRING>
        <bBOOL>true</bBOOL>
        <nINT32>-9</nINT32>
      </Struct2>
    </PLC_Struct>
  </PLC_Structs>
</Beckhoff_PLC>
```

相应的 XML 模式（PLC_Structs.xsd）

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified" xmlns:xs="http://
www.w3.org/2001/XMLSchema">
  <xs:element name="Beckhoff_PLC">
    <xs:complexType >
      <xs:sequence >
        <xs:element name = "PLC_Structs">
          <xs:complexType>
            <xs:sequence>
              <xs:element maxOccurs = "unbounded" name="PLC_Struct">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element minOccurs = "0" maxOccurs="unbounded" name="Struct">
                      <xs:complexType>
                        <xs:sequence>
                          <xs:element name = "nINT64" type="xs:long" />
                          <xs:element name = "nUINT16" type="xs:unsignedShort" />
                          <xs:element name = "rREAL64" type="xs:double" />
                          <xs:element name = "sSTRING" type="xs:string" />
                          <xs:element name = "bBOOL" type="xs:boolean" />
                          <xs:element name = "nINT32" type="xs:int" />
                        </xs:sequence>
                        <xs:attribute name = "Instance" type="xs:unsignedByte" use="required" />
                      </xs:complexType>
                    </xs:element>
                    <xs:element minOccurs = "0" maxOccurs="unbounded" name="Struct2">
                      <xs:complexType>
                        <xs:sequence>
                          <xs:element name = "sSTRING" type="xs:string" />
                          <xs:element name = "bBOOL" type="xs:boolean" />
                          <xs:element name = "nINT32" type="xs:int" />
                        </xs:sequence>
                        <xs:attribute name = "Instance" type="xs:unsignedByte" use="required" />
                      </xs:complexType>
                    </xs:element>
                  </xs:sequence>
                  <xs:attribute name = "Name" type="xs:string" use="required" />
                </xs:complexType>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Structure1 ST_TestStruct

```
TYPE ST_TestStruct :
STRUCT
    nINT64 : T_LARGE_INTEGER;
    nUINT16: UINT;
    rREAL64: LREAL;
    sSTRING: T_MaxString;
    bBOOL  : BOOL;
    nINT32 : DINT;
END_STRUCT
END_TYPE
```

Structure2 ST_TestStruct2

```
TYPE ST_TestStruct2 :
STRUCT
    sSTRING: T_MaxString;
    bBOOL  : BOOL;
    nINT32 : DINT;
END_STRUCT
END_TYPE
```

MAIN 程序

```
PROGRAM MAIN
VAR
 nState                 : BYTE;

 R_TRIG1                : R_TRIG;
 bStartStop             : BOOL;
```

```
 sCmd                    : T_MaxString;

 FB_DBRecordArraySelect1: FB_DBRecordArraySelect;
 arrTestStruct           : ARRAY [0..3] OF ST_TestStruct;
 arrTestStruct2          : ARRAY [0..3] OF ST_TestStruct2;

 bBusy                   : BOOL;
 bError                  : BOOL;
 nErrID                  : UDINT;
 stSQLState              : ST_DBSQLError;

 nRecs1                  : UDINT;
 nRecs2                  : UDINT;
END_VAR
```

```
R_TRIG1(CLK:=bStartStop);
IF R_TRIG1.Q THEN
 FB_DBRecordArraySelect1(bExecute:=FALSE);
 nState                  := 1;
END_IF
```

```
CASE nState OF
 0:(*Idle*)
 ;
 1:
 sCmd:='XPATH<SUBTAG>#/Beckhoff_PLC/PLC_Structs/PLC_Struct[@Name=$'ST_TestStruct$']/Struct';

 FB_DBRecordArraySelect1(
     sNetID              := ,
     hDBID               := 1,
     cbCmdSize           := SIZEOF(sCmd),
     pCmdAddr            := ADR(sCmd),
     nStartIndex         := 0,
     nRecordCount        := 4,
     cbRecordArraySize   := SIZEOF(arrTestStruct),
     pDestAddr           := ADR(arrTestStruct),
     bExecute            := TRUE,
     tTimeout            := T#15s,
     bBusy               => bBusy,
     bError              => bError,
     nErrID              => nErrID,
     sSQLState           => stSQLState,
     nRecords            => nRecs1);

 IF NOT bBusy THEN
    FB_DBRecordArraySelect1(bExecute:=FALSE);
    IF NOT bError THEN
     nState              := 2;
    ELSE
     nState              := 255;
    END_IFEND_IF
 2:
 sCmd:='XPATH<SUBTAG>#Beckhoff_PLC/PLC_Structs/PLC_Struct[@Name=$'ST_TestStruct2$']/Struct2';

 FB_DBRecordArraySelect1(
     sNetID              := ,
     hDBID               := 1,
     cbCmdSize           := SIZEOF(sCmd),
     pCmdAddr            := ADR(sCmd),
     nStartIndex         := 0,
     nRecordCount        := 4,
     cbRecordArraySize   := SIZEOF(arrTestStruct2),
     pDestAddr           := ADR(arrTestStruct2),
     bExecute            := TRUE,
     tTimeout            := T#15s,
     bBusy               => bBusy,
     bError              => bError,
     nErrID              => nErrID,
     sSQLState           => stSQLState,
     nRecords            => nRecs2);

 IF NOT bBusy THEN
    FB_DBRecordArraySelect1(bExecute:=FALSE);
    IF NOT bError THEN
     nState              := 0;
    ELSE
     nState              := 255;
    END_IFEND_IF
 255: (* Error Step*)
 ;
END_CASE
```

读取随着切换变量“bStartStop”的上升沿开始。

MAIN [Online]

TwinCAT_Device.TestPLC.MAIN

Expression	Type	Value
nState	BYTE	0
R_TRIG1	R_TRIG	
bStartStop	BOOL	TRUE
sCmd	STRING(255)	'XPATH<SUBTAG>#Beckhoff_PLC/PLC_Structs/PLC_Struct[@Name='ST_TestStruct2']/Struct2'
FB_DBRecordArraySelect1	FB_DBRecordArraySelect	
arrTestStruct	ARRAY [0..3] OF ST_TestStruct	
arrTestStruct[0]	ST_TestStruct	
nINT64	LINT	123456789
nUINT16	UINT	1234
rREAL64	LREAL	1234.5678
sSTRING	STRING(255)	'This is instance one of ST_TestStruct'
bBOOL	BOOL	TRUE
nINT32	DINT	-100
arrTestStruct[1]	ST_TestStruct	
arrTestStruct[2]	ST_TestStruct	
arrTestStruct[3]	ST_TestStruct	
arrTestStruct2	ARRAY [0..3] OF ST_TestStruct2	
arrTestStruct2[0]	ST_TestStruct2	
sSTRING	STRING(255)	'This is instance one of ST_TestStruct2'
bBOOL	BOOL	FALSE
nINT32	DINT	-88
arrTestStruct2[1]	ST_TestStruct2	
arrTestStruct2[2]	ST_TestStruct2	
arrTestStruct2[3]	ST_TestStruct2	
bBusy	BOOL	FALSE
bError	BOOL	FALSE
nErrID	UDINT	0
stSQLState	ST_DBSQLError	
nRecs1	UDINT	3
nRecs2	UDINT	2

要求

开发环境	目标平台	要连接的 PLC 库
TwinCAT v3.0.0	PC 或 CX（x86）	Tc2_Database

8 附录

8.1 错误代码

8.1.1 Tc3_Database

8.1.1.1 PLC 返回值

Tc3_Database.compiled 库的所有 PLC 块的错误输出均通过 Tc3_Eventlogger.compiled 库中的 I_TcResultEvent 接口进行。这种新的接口结构可以对事件进行更加详细的描述和分类。

```
Interface I_TcMessage [▶ 215]
    nEventId: UDINT;
    EventClass: GUID;
    eSeverity: TcEventSeverity [▶ 216];
    ipSourceInfo: I_TcSourceInfo;
```

nEventID：特定事件代码

EventClass： GUID

EventClassName：使用 RequestEventClassName 方法可以读取相应的事件类名称

eSeverity：事件分类：从“信息”到“严重错误”

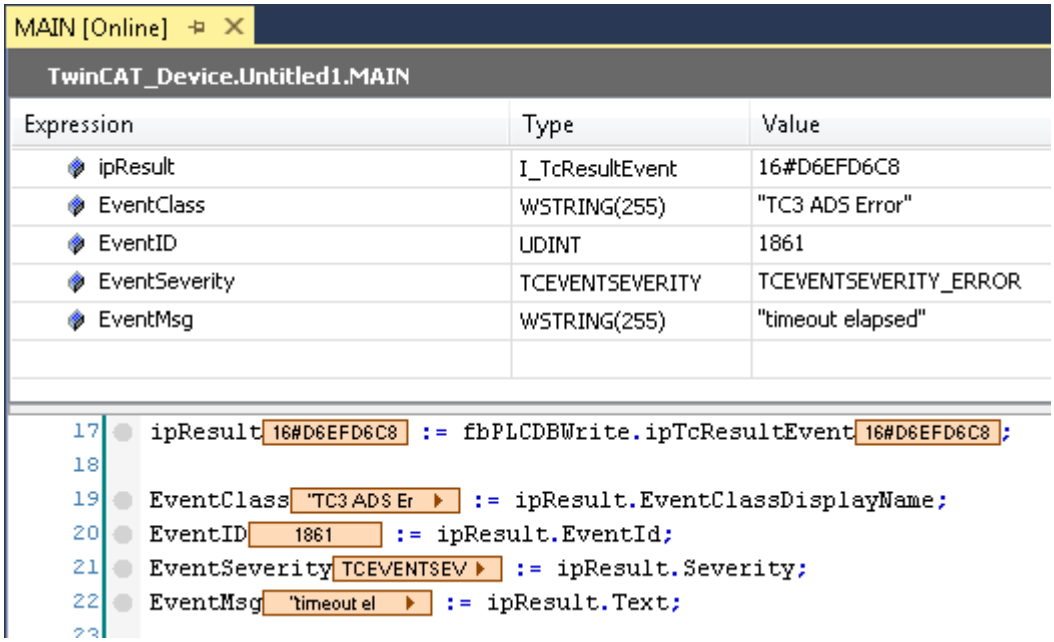
ipSourceInfo：事件位置的路径。

文本：使用 RequestEventText 方法可以读取纯文本形式的事件描述

可能会发生以下事件类：

- **TC3 ADS 错误**
  在与 TwinCAT 数据库服务器通信期间可能会出现的 ADS 错误。
- **TC3 数据库服务器内部错误**
  如果 TwinCAT 数据库服务器配置不正确，则可能会出现内部错误。
- **TC3 数据库服务器数据库错误**
  与相应的数据库通信期间可能会出现的数据库错误。不同的数据库特定错误代码会映射到一个数据库错误列表中。根据需要会将数据库特定错误代码写入 ErrorLog。
- **TC3 数据库服务器 ADS 设备错误**
  在与配置的 ADS 设备进行内部通信期间可能会出现的 ADS 错误。
- **TC3 数据库服务器 NoSQL 错误**
  在与相应的数据库通信时发生的 NoSQL 数据库的数据库错误。

8.1.1.2 ADS 返回代码



全局错误代码

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

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。

路由器错误代码

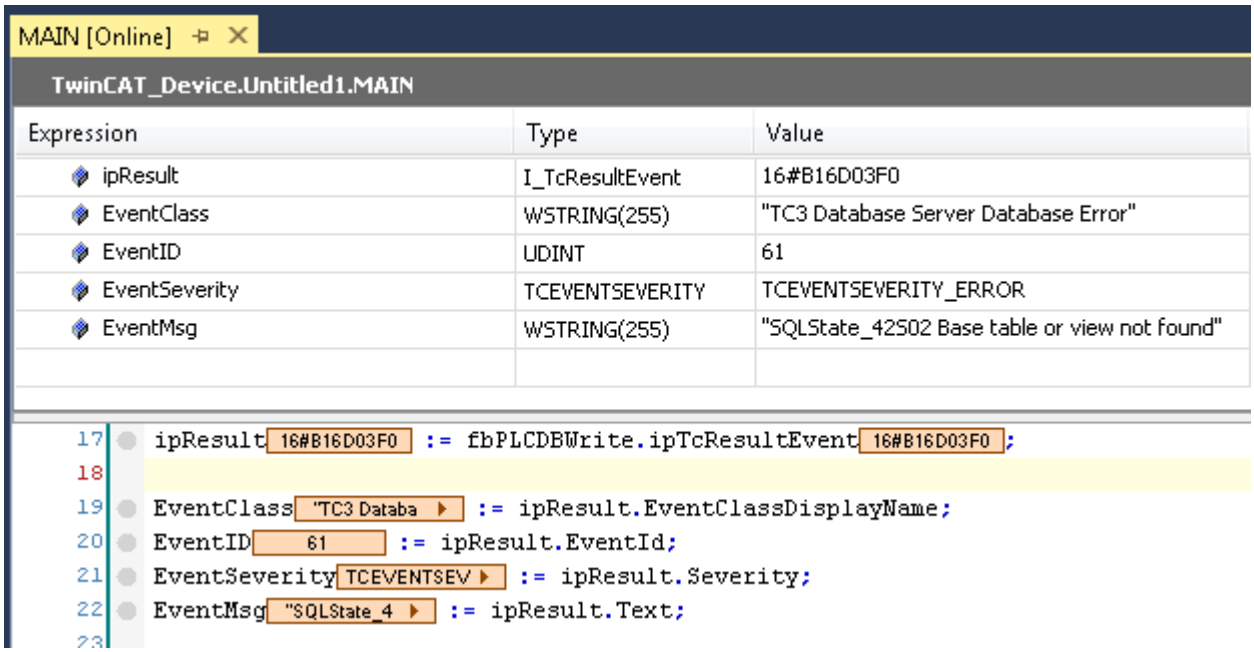
Hex	Dec	HRESULT	名称	描述
0x500	1280	0x98110500	ROUTERERR_NOLOCKEDMEMORY	无法分配锁定的内存。
0x501	1281	0x98110501	ROUTERERR_RESIZEMEMORY	路由器内存大小无法更改。
0x502	1282	0x98110502	ROUTERERR_MAILBOXFULL	邮箱已达到最大消息数。
0x503	1283	0x98110503	ROUTERERR_DEBUGBOXFULL	调试邮箱已达到最大消息数。
0x504	1284	0x98110504	ROUTERERR_UNKNOWNPORTTYPE	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。

ADS 通用错误代码

十六进制	十进制	结构化错误码	名称	描述
0x700	1792	0x98110700	ADSERR_DEVICE_ERROR	设备通用错误。
0x701	1793	0x98110701	ADSERR_DEVICE_SRVNOTSUPP	服务器不支持该服务。
0x702	1794	0x98110702	ADSERR_DEVICE_INVALIDGRP	索引组无效。
0x703	1795	0x98110703	ADSERR_DEVICE_INVALIDOFFSET	索引偏移量无效。
0x704	1796	0x98110704	ADSERR_DEVICE_INVALIDACCESS	不允许读取或写入。 可能有几种原因。例如，创建路由时输入了错误的密码。
0x705	1797	0x98110705	ADSERR_DEVICE_INVALIDSIZE	参数大小不正确。
0x706	1798	0x98110706	ADSERR_DEVICE_INVALIDDATA	无效数据值。
0x707	1799	0x98110707	ADSERR_DEVICE_NOTREADY	设备尚未准备好运行。
0x708	1800	0x98110708	ADSERR_DEVICE_BUSY	设备正忙。
0x709	1801	0x98110709	ADSERR_DEVICE_INVALIDCONTEXT	无效的操作系统运行环境。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 长度过大。
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：许可证设置在未来的时间。
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	系统无法识别来自OEM的公钥。

十六进制	十进制	结构化错误码	名称	描述
0x731	1841	0x98110731	ADSERR_DEVICE_LICENSERESTRICTED	此系统 ID 授权无效。
0x732	1842	0x98110732	ADSERR_DEVICE_LICENSEDEMODENIED	禁止使用演示授权。
0x733	1843	0x98110733	ADSERR_DEVICE_INVALIDFNCID	无效 的函数 ID。
0x734	1844	0x98110734	ADSERR_DEVICE_OUTOFRANGE	超出有效范围。
0x735	1845	0x98110735	ADSERR_DEVICE_INVALIDALIGNMENT	无效对齐。
0x736	1846	0x98110736	ADSERR_DEVICE_LICENSEPLATFORM	无效平台级别。
0x737	1847	0x98110737	ADSERR_DEVICE_FORWARD_PL	上下文 – 转到被动级别。
0x738	1848	0x98110738	ADSERR_DEVICE_FORWARD_DL	上下文 – 转到调度级别。
0x739	1849	0x98110739	ADSERR_DEVICE_FORWARD_RT	上下文 – 转到实时。
0x740	1856	0x98110740	ADSERR_CLIENT_ERROR	客户端错误。
0x741	1857	0x98110741	ADSERR_CLIENT_INVALIDPARM	服务包含无效参数。
0x742	1858	0x98110742	ADSERR_CLIENT_LISTEMPTY	轮询列表为空。
0x743	1859	0x98110743	ADSERR_CLIENT_VARUSED	Var 连接已在使用中。
0x744	1860	0x98110744	ADSERR_CLIENT_DUPLINVOKEID	调用的 ID 已在使用中。
0x745	1861	0x98110745	ADSERR_CLIENT_SYNCTIMEOUT	已发生超时 – 远程终端在指定的 ADS 超时 时间内没有响应。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 同步过程中发生内部错误。
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。

8.1.1.3 数据库返回代码



ErrorCode	ErrorName	ErrorDescription
1	DB_SQLState_01000	一般警告
2	DB_SQLState_01001	游标操作冲突
3	DB_SQLState_01002	断开错误
4	DB_SQLState_01003	在设置功能中消除的 NULL 值
5	DB_SQLState_01004	字符串数据，右侧截断
6	DB_SQLState_01006	权限未撤销
7	DB_SQLState_01007	权限未授予
8	DB_SQLState_01S00	无效的连接字符串属性
9	DB_SQLState_01S01	行中的错误
10	DB_SQLState_01S02	选项值已更改
11	DB_SQLState_01S06	在结果集返回第一行集之前，尝试进行获取
12	DB_SQLState_01S07	小数部分截断
13	DB_SQLState_01S08	保存文件 DSN 出错
14	DB_SQLState_01S09	无效的关键字
15	DB_SQLState_07000	动态 SQL 错误
16	DB_SQLState_07001	错误的参数数量
17	DB_SQLState_07002	COUNT 字段不正确
18	DB_SQLState_07005	预处理语句不是游标规范
19	DB_SQLState_07006	受限数据类型属性违规
20	DB_SQLState_07009	无效的描述符索引
21	DB_SQLState_07S01	使用无效默认参数
22	DB_SQLState_08000	连接异常
23	DB_SQLState_08001	客户端无法建立连接
24	DB_SQLState_08002	连接名称正在使用中
25	DB_SQLState_08003	连接未打开
26	DB_SQLState_08004	服务器已拒绝连接
27	DB_SQLState_08007	事务期间连接失败
28	DB_SQLState_08S01	通信链路故障
29	DB_SQLState_21000	基数违规
30	DB_SQLState_21S01	插入值列表与列列表不匹配
31	DB_SQLState_21S02	导出表的度与列列表不匹配
32	DB_SQLState_22000	数据异常
33	DB_SQLState_22001	字符串数据，右侧截断
34	DB_SQLState_22002	需要但未提供的指示变量
35	DB_SQLState_22003	数值超出范围
36	DB_SQLState_22007	无效的日期时间格式
37	DB_SQLState_22008	日期/时间字段溢出
38	DB_SQLState_22012	除零错误
39	DB_SQLState_22015	区间字段溢出
40	DB_SQLState_22018	无效的强制类型转换规范的字符值
41	DB_SQLState_22019	无效的转义字符
42	DB_SQLState_22025	无效的转义序列

43	DB_SQLState_22026	字符串数据，长度不匹配
44	DB_SQLState_23000	违反完整性约束
45	DB_SQLState_24000	无效的游标状态
46	DB_SQLState_25000	无效的事务状态
47	DB_SQLState_25S01	事务状态
48	DB_SQLState_25S02	事务仍在进行中
49	DB_SQLState_25S03	事务已回滚
50	DB_SQLState_28000	无效的授权规范
51	DB_SQLState_34000	无效的游标名称
52	DB_SQLState_3C000	重复的游标名称
53	DB_SQLState_3D000	无效的目录名称
54	DB_SQLState_3F000	无效的模式名称
55	DB_SQLState_40000	事务回滚
56	DB_SQLState_40001	序列化失败
57	DB_SQLState_40002	违反完整性约束
58	DB_SQLState_40003	语句完成情况未知
59	DB_SQLState_42000	语法错误或访问违规
60	DB_SQLState_42S01	基本表或视图已存在
61	DB_SQLState_42S02	未找到基本表或视图
62	DB_SQLState_42S11	索引已存在
63	DB_SQLState_42S12	未找到索引
64	DB_SQLState_42S21	列已存在
65	DB_SQLState_42S22	未找到列
66	DB_SQLState_44000	WITH CHECK OPTION 违规
67	DB_SQLState_HY000	一般错误
68	DB_SQLState_HY001	内存分配错误
69	DB_SQLState_HY003	无效的应用程序缓冲区类型
70	DB_SQLState_HY004	无效的 SQL 数据类型
71	DB_SQLState_HY007	未准备相关语句
72	DB_SQLState_HY008	操作已取消
73	DB_SQLState_HY009	无效使用空指针
74	DB_SQLState_HY010	功能序列错误
75	DB_SQLState_HY011	现在无法设置属性
76	DB_SQLState_HY012	无效的事务操作代码
77	DB_SQLState_HY013	内存管理错误

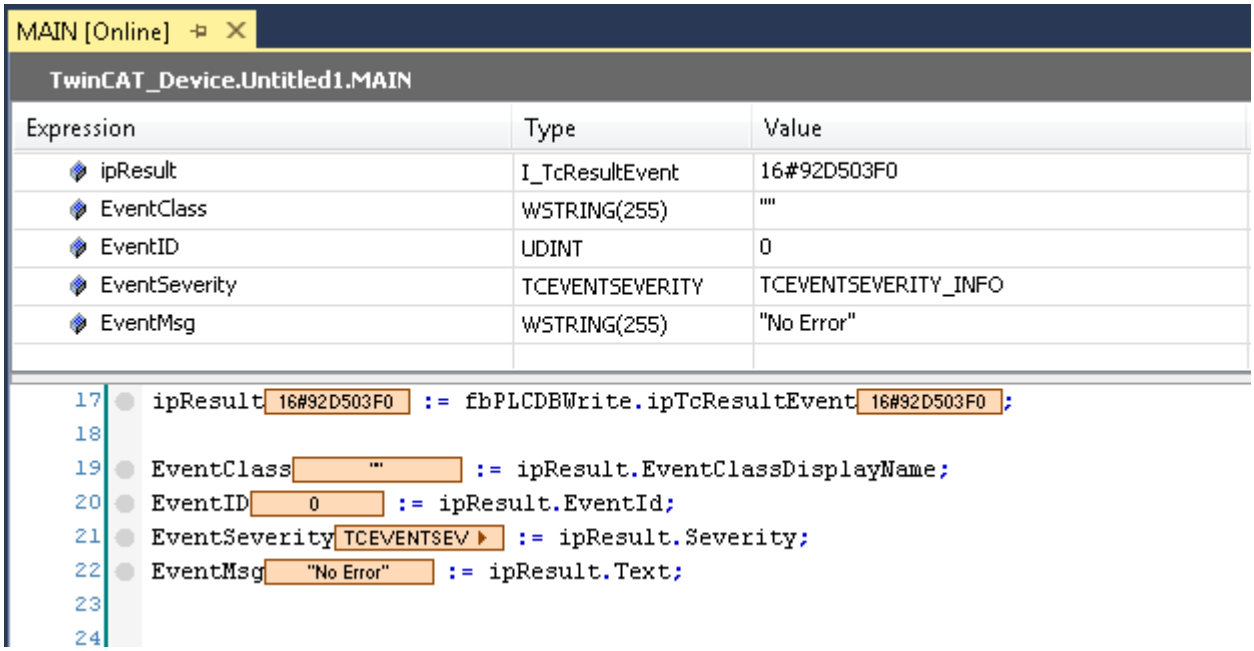
78	DB_SQLState_HY014	超出句柄数量限值
79	DB_SQLState_HY015	没有可用的游标名称
80	DB_SQLState_HY016	无法修改执行行描述符
81	DB_SQLState_HY017	无效使用自动分配的描述符句柄
82	DB_SQLState_HY018	服务器拒绝取消请求
83	DB_SQLState_HY019	分块发送的非字符和非二进制数据
84	DB_SQLState_HY020	尝试连接一个空值
85	DB_SQLState_HY021	不一致的描述符信息
86	DB_SQLState_HY024	无效的属性值
87	DB_SQLState_HY090	无效的字符串或缓冲区长度
88	DB_SQLState_HY091	无效的描述符字段标识符
89	DB_SQLState_HY092	无效的属性/选项标识符
90	DB_SQLState_HY095	功能类型超出范围
91	DB_SQLState_HY096	无效的信息类型
92	DB_SQLState_HY097	列类型超出范围
93	DB_SQLState_HY098	范围类型超出范围
94	DB_SQLState_HY099	可空类型超出范围
95	DB_SQLState_HY100	唯一性选项类型超出范围
96	DB_SQLState_HY101	精度选项类型超出范围
97	DB_SQLState_HY103	无效的检索代码
98	DB_SQLState_HY104	无效的精度或刻度值
99	DB_SQLState_HY105	无效的参数类型
100	DB_SQLState_HY106	获取类型超出范围
101	DB_SQLState_HY107	行值超出范围
102	DB_SQLState_HY109	无效的游标位置
103	DB_SQLState_HY110	无效的驱动程序完成
104	DB_SQLState_HY111	无效的书签值
105	DB_SQLState_HYC00	未执行的可选功能
106	DB_SQLState_HYT00	超时已过
107	DB_SQLState_HYT01	连接超时已过
108	DB_SQLState_IM001	驱动程序不支持该功能
109	DB_SQLState_IM002	未找到数据源名称且未指定默认驱动程序
110	DB_SQLState_IM003	无法加载指定的驱动程序
111	DB_SQLState_IM004	驱动程序在 SQL_HANDLE_ENV 上的 SQLAllocHandle 失败
112	DB_SQLState_IM005	驱动程序在 SQL_HANDLE_DBC 上的 SQLAllocHandle 失败
113	DB_SQLState_IM006	驱动程序的 SQLSetConnectAttr 失败
114	DB_SQLState_IM007	没有禁止指定数据源或驱动程序对话框
115	DB_SQLState_IM008	对话失败
116	DB_SQLState_IM009	无法加载翻译 DLL
117	DB_SQLState_IM010	数据源名称过长
118	DB_SQLState_IM011	驱动程序名称过长
119	DB_SQLState_IM012	DRIVER 关键字语法错误
120	DB_SQLState_IM013	跟踪文件错误
121	DB_SQLState_IM014	无效的文件 DSN 名称
122	DB_SQLState_IM015	文件数据源损坏

200	DB_FunctionNotImplemented	功能未实现
255	DB_SQLState_Unspecified	未指定的错误

8.1.1.4 NoSQL 数据库返回代码

ErrorCode	ErrorName	ErrorDescription
0	NoSql_0	无错误
1	NoSql_Specific_1	MongoDB 内部异常。请查看信息日志，了解更多详情。
2	NoSql_Specific_2	数据库端错误。请查看信息日志，了解更多详情。
8	NoSql_Specific_8	执行命令时发生超时错误
9	NoSql_Specific_9	执行命令时发生格式错误。请查看命令或文档语法。
11	NoSql_Specific_11	RunCommand 执行失败。请查看事件日志，了解更多信息
12	NoSql_Specific_12	该表格没有元数据（包括表格模式）
13	NoSql_Specific_13	命令/筛选器中存在语法错误
14	NoSql_Specific_14	发生连接错误
15	NoSql_Specific_15	身份验证失败
16	NoSql_Specific_16	在写入操作期间发生错误
20	NoSql_Specific_20	TwinCAT 数据库服务器不支持该功能
48	NoSql_Specific_48	Namespace / CollectionName 已存在
51	NoSql_Specific_51	查询的数据为空
141	NoSql_Specific_141	TwinCAT 3 数据库服务器内部错误

8.1.1.5 TwinCAT 数据库服务器代码



ErrorCode	ErrorName	ErrorDescription
0	InternalErrorCode_0	无错误
1	InternalErrorCode_1	不允许使用 NULL 值
2	InternalErrorCode_2	FB_DBRead 选定值为 NULL
3	InternalErrorCode_3	DBID 未知
4	InternalErrorCode_4	ADSDevID 未知
5	InternalErrorCode_5	未找到 DBID: xy 打开的数据库连接
6	InternalErrorCode_6	未找到 ADSDevID: xy 打开的 ADS 设备连接
7	InternalErrorCode_7	AutoLog 组初始化失败
8	InternalErrorCode_8	无法启动 AutoLog。TwinCAT 数据库服务器没有有效的设备状态
9	InternalErrorCode_9	记录选择值返回 => ADS 设备参数大小错误
10	InternalErrorCode_10	无效的参数
11	InternalErrorCode_11	无法在列表中找到 PLCDBWrite 值
12	InternalErrorCode_12	没有有效的符号类型
13	InternalErrorCode_13	参数大小无效
14	InternalErrorCode_14	执行超时
15	InternalErrorCode_15	数据库连接已打开
16	InternalErrorCode_16	数据库连接未初始化
1000	InternalErrorCode_1000	未知的内部错误

8.1.2 Tc2_Database

8.1.2.1 ADS 返回代码

错误代码分组：
全局错误代码：ADS 返回代码 [▶ 374]... （0x9811_0000 ...）
路由器错误代码：ADS 返回代码 [▶ 375]... （0x9811_0500 ...）
一般 ADS 错误：ADS 返回代码 [▶ 375]... （0x9811_0700 ...）
RTime 错误代码：ADS 返回代码 [▶ 377]... （0x9811_1000 ...）

全局错误代码

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

路由器错误代码

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

ADS 通用错误代码

十六进制	十进制	结构化错误码	名称	描述
0x700	1792	0x98110700	ADSERR_DEVICE_ERROR	设备通用错误。
0x701	1793	0x98110701	ADSERR_DEVICE_SRVNOTSUPP	服务器不支持该服务。
0x702	1794	0x98110702	ADSERR_DEVICE_INVALIDGRP	索引组无效。
0x703	1795	0x98110703	ADSERR_DEVICE_INVALIDOFFSET	索引偏移量无效。
0x704	1796	0x98110704	ADSERR_DEVICE_INVALIDACCESS	不允许读取或写入。 可能有几种原因。例如，创建路由时输入了错误的密码。
0x705	1797	0x98110705	ADSERR_DEVICE_INVALIDSIZE	参数大小不正确。
0x706	1798	0x98110706	ADSERR_DEVICE_INVALIDDATA	无效数据值。
0x707	1799	0x98110707	ADSERR_DEVICE_NOTREADY	设备尚未准备好运行。
0x708	1800	0x98110708	ADSERR_DEVICE_BUSY	设备正忙。
0x709	1801	0x98110709	ADSERR_DEVICE_INVALIDCONTEXT	无效的操作系统运行环境。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 长度过大。
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：许可证设置在未来的时间。
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	系统无法识别来自OEM的公钥。

十六进制	十进制	结构化错误码	名称	描述
0x731	1841	0x98110731	ADSERR_DEVICE_LICENSERESTRICTED	此系统 ID 授权无效。
0x732	1842	0x98110732	ADSERR_DEVICE_LICENSEDEMODENIED	禁止使用演示授权。
0x733	1843	0x98110733	ADSERR_DEVICE_INVALIDFNCID	无效 的函数 ID。
0x734	1844	0x98110734	ADSERR_DEVICE_OUTOFRANGE	超出有效范围。
0x735	1845	0x98110735	ADSERR_DEVICE_INVALIDALIGNMENT	无效对齐。
0x736	1846	0x98110736	ADSERR_DEVICE_LICENSEPLATFORM	无效平台级别。
0x737	1847	0x98110737	ADSERR_DEVICE_FORWARD_PL	上下文 – 转到被动级别。
0x738	1848	0x98110738	ADSERR_DEVICE_FORWARD_DL	上下文 – 转到调度级别。
0x739	1849	0x98110739	ADSERR_DEVICE_FORWARD_RT	上下文 – 转到实时。
0x740	1856	0x98110740	ADSERR_CLIENT_ERROR	客户端错误。
0x741	1857	0x98110741	ADSERR_CLIENT_INVALIDPARM	服务包含无效参数。
0x742	1858	0x98110742	ADSERR_CLIENT_LISTEMPTY	轮询列表为空。
0x743	1859	0x98110743	ADSERR_CLIENT_VARUSED	Var 连接已在使用中。
0x744	1860	0x98110744	ADSERR_CLIENT_DUPLINVOKEID	调用的 ID 已在使用中。
0x745	1861	0x98110745	ADSERR_CLIENT_SYNCTIMEOUT	已发生超时 – 远程终端在指定的 ADS 超时 时间内没有响应。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 同步过程中发生内部错误。
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

十六进制	十进制	名称	描述
0x274C	10060	WSAETIMEDOUT	发生连接超时 - 在创建连接时发生错误，因为远程终端在特定时间后未正确响应，或者所建立连接因所连接主机未响应而无法维持。
0x274D	10061	WSAECONNREFUSED	连接遭到拒绝 - 无法建立连接，因为目标计算机明确拒绝了该连接。此错误通常由尝试连接到外部主机上处于非活动状态的服务（即没有运行服务器应用程序的服务）导致。
0x2751	10065	WSAEHOSTUNREACH	不存在至主机的路由 - 套接字操作引用了不可用的主机。
更多 Winsock 错误代码：Win32 错误代码			

8.1.2.2 TwinCAT 数据库服务器错误代码概述

代码（十六进制）	代码（十进制）	描述
0x0001 + ADS 错误代码	65537 - 131071	来自已声明的 ADS 设备的 ADS 错误代码
0x00020001	131073	Microsoft SQL Compact 数据库（错误代码）
0x00040001	262145	Microsoft SQL 数据库（错误代码）
0x00080001	524289	Microsoft Access 数据库（错误代码）
0x00100001	1048577	MySQL 数据库（错误代码）
0x00200001	2097153	Oracle 数据库（错误代码）
0x00400001	4194305	DB2 数据库（错误代码）
0x00800001	8388609	PostgreSQL 数据库（错误代码）
0x01000001	16777217	Interbase/Firebird 数据库（错误代码）
0x02000001	33554433	TwinCAT 数据库服务器错误代码 [▶ 385]
0x04000001	67108865	XML 数据库（错误代码）
0x08000001	134217729	ASCII 数据库（错误代码）

如果功能块的“nErrID”输出端发出上述错误代码之一，则在执行 SQL 语句期间已发生错误。然后，在功能块的“sSQLState”输出端发出 SQL 错误代码。“sSQLState”输出端的数据类型为 ST_DBSQLError [▶ 307]。每种数据库类型都会输出单独的错误代码。

有关 SQLStates 列表，请参见：http://msdn.microsoft.com/en-us/library/ms714687(VS.85).aspx（SQLStates）

Database Type	错误代码参考
Microsoft SQL Compact 数据库	http://technet.microsoft.com/en-us/library/ms171788.aspx / OleDB_Errorcodes.htm [▶ 380]
Microsoft SQL 数据库	OleDB_Errorcodes.htm [▶ 380]
Microsoft Access 数据库	OleDB_Errorcodes.htm [▶ 380]
MySQL 数据库	https://dev.mysql.com/doc/mysql-errors/8.0/en/client-error-reference.html
Oracle 数据库	https://docs.oracle.com/cd/E11882_01/server.112/e17766/toc.htm
DB2 数据库	https://www.ibm.com/docs/en/db2-for-zos/12?topic=diagnostics-sqlstates-odbc-error-reporting
PostgreSQL 数据库	http://www.postgresql.org/docs/current/static/errcodes-appendix.html
Interbase/Firebird 数据库	http://www.firebirdsql.org/file/documentation/reference_manuals/reference_material/Firebird-2.1-ErrorCodes.pdf
XML 数据库	TcDBServer_XML_Errorcodes.htm [▶ 385]
ASCII 数据库	TcDBServer_ASCII_Errorcodes.htm [▶ 384]

8.1.2.3 OleDB 错误代码

值	描述
0x80040E00	访问器无效。
0x80040E01	由于已超出提供程序的最大启用行数，因此无法在行集中插入行。
0x80040E02	访问器受到写入保护。程序失败。
0x80040E03	值违反数据库模式。
0x80040E04	行句柄无效。
0x80040E05	对象打开。
0x80040E06	无效的章节
0x80040E07	由于数据溢出以外的原因，命令中的字面值无法转换为正确类型。
0x80040E08	无效的绑定信息
0x80040E09	权限不足
0x80040E0A	指定的列不包含书签或章节。
0x80040E0B	一些成本限制被拒绝。
0x80040E0C	没有为命令对象指定命令。
0x80040E0D	在指定的成本限制内没有找到查询计划。
0x80040E0E	无效的书签
0x80040E0F	无效的锁定模式
0x80040E10	至少有一个必需参数没有指定值。
0x80040E11	无效的列 ID
0x80040E12	无效的配额
0x80040E13	无效值
0x80040E14	命令至少包含一个错误。
0x80040E15	无法中止当前执行的命令。
0x80040E16	提供程序不支持指定的 SQL 方言。
0x80040E17	具有指定名称的数据源已存在。
0x80040E18	通过实时数据流创建行集，无法重新启动它。
0x80040E19	在当前范围内，没有键符合所描述的特征。
0x80040E1B	提供程序无法确定新添加行的身份。
0x80040E1A	该结构的所有权已转移给提供程序。
0x80040E1C	不支持将非零权重值作为目标信息。因此，该目标被拒绝。当前目标没有改变。
0x80040E1D	不支持所请求的转换。
0x80040E1E	RowsOffset 指向行集结束后的位置，与指定的 cRows 值无关。cRowsObtained 为 0。
0x80040E20	提供程序已在使用者中调用 IRowsetNotify 方法，但尚未收到该方法的返回结果。
0x80040E21	错误
0x80040E22	已指定一个非零控制 IUnknown 对象，并且，当前创建的对象不支持聚合。
0x80040E23	已删除当前行。
0x80040E24	行集不支持后向调用。
0x80040E25	在接收新的 HROW 对象之前，必须释放所有 HROW 对象。
0x80040E26	不支持指定的内存标志。
0x80040E27	比较运算符无效。
0x80040E28	指定的状态标志既不是 DBCOLUMNSTATUS_OK，也不是 DBCOLUMNSTATUS_ISNULL。
0x80040E29	无法反向处理行集。
0x80040E2A	无效的范围句柄。
0x80040E2B	指定的行集与指定监测范围的行不相邻，且没有重叠。
0x80040E2C	已指定从 ALL* 到 MOVE* 或 EXTEND* 的过渡。

值	描述
0x80040E2D	指定的范围不是由指定监测范围句柄标识的范围的有效子范围。
0x80040E2E	提供程序不支持包含多个语句的命令。
0x80040E2F	指定的值违反了列或表格的完整性限制。
0x80040E30	未识别指定的类型名称。
0x80040E31	由于没有更多可用资源，已中止执行。未得到任何结果。
0x80040E32	由于命令层级结构中包含至少一个行集，因此无法克隆该命令对象。
0x80040E33	当前结构无法显示为文本。
0x80040E34	指定的索引已存在。
0x80040E35	指定的索引不存在。
0x80040E36	已使用指定的索引。
0x80040E37	指定的表格不存在。
0x80040E38	行集已使用完全并行性，并且自上次读取操作以来列值已发生变化。
0x80040E39	在复制期间发现错误。
0x80040E3A	精度声明无效。
0x80040E3B	指定的十进制值无效。
0x80040E3C	无效的表格 ID。
0x80040E3D	指定的类型无效。
0x80040E3E	在规范中多次出现列 ID。
0x80040E3F	指定的表格已存在。
0x80040E40	已使用指定的表格。
0x80040E41	不支持指定的范围模式 ID。
0x80040E42	指定的记录编号无效。
0x80040E43	尽管书签格式有效，但是无法找到匹配的行。
0x80040E44	属性的值无效。
0x80040E45	行集没有细分为章节。
0x80040E46	无效的访问器
0x80040E47	无效的内存标志
0x80040E48	该提供程序不支持作为引用进行传输的访问器。
0x80040E49	该提供程序不支持 NULL 访问器。
0x80040E4A	该命令未被预处理。
0x80040E4B	指定的访问器不是参数访问器。
0x80040E4C	指定的访问器受到写入保护。
0x80040E4D	在身份验证期间出错。
0x80040E4E	在通知过程中已中止更改；没有修改任何列。
0x80040E4F	行集由一个章节组成，但该章节未启用。
0x80040E50	无效的源句柄
0x80040E51	提供程序无法获取参数信息，并且未调用 SetParameterInfo。
0x80040E52	数据源对象已经初始化。
0x80040E53	提供程序不支持此方法。
0x80040E54	待修改的行数超过指定的限值。
0x80040E55	指定的列不存在。
0x80040E56	某行数据存在未决更改，但其引用计数器为零。
0x80040E57	命令中的字面量值超出了赋值列的数据类型范围。
0x80040E58	传输的 HRESULT 值无效。
0x80040E59	传输的 LookupID 值无效。
0x80040E5A	传输的 DynamicErrorID 值无效。

值	描述
0x80040E5B	对于尚未更新的新增行，无法检索到可见数据。
0x80040E5C	无效的转换标志
0x80040E5D	未识别指定的参数名称。
0x80040E5E	无法同时打开多个内存对象。
0x80040E5F	无法打开请求的筛选器。
0x80040E60	无法打开请求的序列。
0x80040E65	传输的 columnID 值无效。
0x80040E67	传输的命令没有 DBID 值。
0x80040E68	传输的 DBID 值已存在。
0x80040E69	已达到该提供程序支持的最大会话对象数量。使用者必须释放至少一个当前会话对象，然后才能获取新的会话对象。
0x80040E72	索引 ID 无效。
0x80040E73	指定的初始化字符序列与规范不匹配。
0x80040E74	OLE DB 主枚举器未返回任何与请求的 SOURCES_TYPE 值匹配的提供程序。
0x80040E75	初始化字符序列表示与当前启用的提供程序不匹配的提供程序。
0x80040E76	指定的 DBID 值无效。
0x80040E6A	信任接收者的值无效。
0x80040E6B	信任接收者不适合当前数据源。
0x80040E6C	信任接收者不支持会员资格/列表。
0x80040E6D	对象无效，或提供程序未知。
0x80040E6E	对象没有所有者。
0x80040E6F	传输的访问条目列表无效。
0x80040E70	作为所有者传输的信托接收者无效，或提供程序未知。
0x80040E71	在访问条目列表中传输的权限无效。
0x80040E77	ConstraintType 值无效或提供程序不支持。
0x80040E78	ConstraintType 值不是 DBCONSTRAINTTYPE_FOREIGNKEY，并且 cForeignKeyColumns 不是零。
0x80040E79	可延迟性值无效或提供程序不支持。
0x80040E80	MatchType 值无效或提供程序不支持。
0x80040E8A	UpdateRule 或 DeleteRule 值无效或提供程序不支持。
0x80040E8B	无效的限制 ID。
0x80040E8C	dwFlags 值无效。
0x80040E8D	rguidColumnType 值指向的 GUID 与该列的对象类型不匹配，或者未指定该列。
0x80040E91	不存在源行。
0x80040E92	此 URL 代表的 OLE DB 对象已被至少一个其他进程锁定。
0x80040E93	客户端请求的对象类型只能用于列表。
0x80040E94	调用过程请求对受写入保护的对象进行写入访问。
0x80040E95	提供程序无法与此对象的服务器建立连接。
0x80040E96	提供程序无法与此对象的服务器建立连接。
0x80040E97	绑定到对象时超时
0x80040E98	提供程序无法使用此 URL 创建对象，因为已经存在一个以该 URL 命名的对象。
0x80040E8E	请求的 URL 超出有效范围。
0x80040E90	无法删除该列或限制，因为相关视图或限制引用了它。
0x80040E99	限制已存在。
0x80040E9A	无法使用此 URL 创建对象，因为服务器的物理内存不足。
0x00040EC0	在检索请求的行数期间，已超过该行集支持的活动行的总数。

值	描述
0x00040EC1	至少一个列类型不兼容；在复制期间可能会出现转换错误。
0x00040EC2	调用过程已禁用有关参数类型的信息。
0x00040EC3	已跳过已删除或不相关的行的书签。
0x00040EC5	没有更多的行集可用。
0x00040EC6	已达到行集或章节的起点或终点。
0x00040EC7	提供程序再次执行命令。
0x00040EC8	变量的数据缓冲区已满。
0x00040EC9	没有更多的结果可用。
0x00040ECA	在事务完成之前，服务器无法取消或降级锁定。
0x00040ECB	不支持指定的权重值或已超出支持的限值。值被设置为 0 或限值。
0x00040ECC	因此，数据消费者拒绝进一步通知调用。
0x00040ECD	输入方言已被忽略，文本以另一种方言返回。
0x00040ECE	在此阶段，数据消费者拒绝进一步通知调用。
0x00040ECF	因此，数据消费者拒绝进一步通知调用。
0x00040ED0	操作是异步处理的。
0x00040ED1	要达到行集的起点，提供程序必须再次执行查询。列的顺序已发生变化，或者列已被添加到行集中，或者列已被从行集中移除。
0x00040ED2	方法有多个错误。在错误数组中已返回错误。
0x00040ED3	无效的行句柄
0x00040ED4	指定的 HROW 对象引用了永久删除的行。
0x00040ED5	提供程序无法追踪所有修改。客户端必须通过另一种方法再次检索分配给监测范围的数据。
0x00040ED6	由于没有更多可用资源，已终止执行。截至此时，收到的结果已返回，但无法继续执行。
0x00040ED8	锁定已根据指定的值升级。
0x00040ED9	根据提供程序允许的选项，至少有一个属性已更改。
0x00040EDA	错误
0x00040EDB	指定的参数无效。
0x00040EDC	由于更新了该行，因此必须更新数据源中的多个行。
0x00040ED7	绑定失败，因为提供程序无法满足所有绑定标志或属性。
0x00040EDD	该行不包含特定于行的列。

8.1.2.4 ASCII 错误代码

值	描述
1	功能不可用
2	语法错误
3	无法打开文件。

8.1.2.5 XML 错误代码

值	描述
1	功能不可用
2	无法加载 XML 文件
3	无法加载 XML 模式
4	语法错误
5	无法创建表格
6	INSERT VALUE 列表与列列表不匹配
7	PLC 结构体不够大
8	无法创建 XML 文件
9	未找到 XML 数据库。
10	未找到 XML 表格

8.1.2.6 内部错误代码

错误代码	描述
0x02000001	不允许使用 NULL 值
0x02000002	FB_DBRead 选定值为 NULL
0x02000003	DBID 未知
0x02000004	ADSDevID 未知
0x02000005	未找到 DBID xy 的打开的数据库连接
0x02000006	未找到为 ADSDevID xy 打开的 ADS 设备连接

8.2 FAQ – 常见问题和解答

本部分将对常见问题进行解答，以便于您使用 TwinCAT 数据库服务器。
如果您还有任何其他问题，请联系我们的技术支持部门。

1. TwinCAT 数据库服务器可以实现哪些性能？ [▶ 385]

2. 是否支持所谓的存储过程？ [▶ 386]

3. TwinCAT 数据库服务器支持哪些数据库类型？ [▶ 386]

4. 旧的数据库服务器配置是否仍然可以用于以后的版本？ [▶ 386]

5. 如何将单个变量写入现有的数据库结构或从中读取？ [▶ 386]

6. 是否可以将多个记录同时写入数据库？ [▶ 386]

7. 在网络中可以如何操作 TwinCAT 数据库服务器？ [▶ 386]

8. 使用 “XML” 数据库类型时，TwinCAT 数据库服务器支持哪些功能？ [▶ 386]

9. 数据库服务器配置器目前支持哪些 Visual Studio 版本？ [▶ 386]

TwinCAT 数据库服务器可以实现哪些性能?
这个问题无法用一般性术语来回答。可以实现的性能取决于所使用的硬件、要执行的操作（例如，环形缓冲区记录）和变量的数量。另一个关键因素是所使用的数据库类型。

是否支持所谓的存储过程?
是的，TwinCAT 数据库服务器支持存储过程。为此，在 PLC 库中提供了功能块 FB_SQLStoredProcedure [▶ 194]。SQL 查询编辑器可以用于测试存储过程，并且可以为功能块 FB_SQLStoredProcedure 生成相应的 PLC 代码。并非所有数据库都支持该功能。

TwinCAT 数据库服务器支持哪些数据库类型?
有关受支持的数据库的信息，请参见“数据库 [▶ 120]”部分。

旧的数据库服务器配置是否仍然可以用于以后的版本?
毫无疑问，我们的目标是确保兼容性。在重大版本升级或完全重新设计时也会考虑到这一点（例如，旧版：3.**0**.x，新版：3.**1**.x）。有关更多详情，请参见“兼容性 [▶ 20]”。

如何将单个变量写入现有的数据库结构或从中读取?
功能块 FB_SQLCommand [▶ 189] 可以用于将单个变量写入现有的数据库结构或从中读取。

是否可以将多个记录同时写入数据库?
这取决于所使用的数据库。在使用 Microsoft SQL 数据库时，这可以与功能块 FB_SQLCommand [▶ 189] 结合使用，因为可以将多个 SQL 插入命令（以分号分隔）传输到 PLC 功能块。

在网络中可以如何操作 TwinCAT 数据库服务器?
在网络中可以通过多种方式使用 TwinCAT 数据库服务器。有关支持网络拓扑结构的更多信息，请参见“应用领域和网络拓扑结构 [▶ 18]”部分。

使用“XML”数据库类型时，TwinCAT 数据库服务器支持哪些功能?
“XML”数据库类型支持 TwinCAT 数据库服务器的全部功能，但是“存储过程”除外。XML 文件可以用于通过 SQL 命令与任何其他数据库通信，通过循环写入模式可以将 PLC 值记录在 XML 文件中。此外，还可以执行 XPath 命令并读取相应的 XML 标签。有关更多信息，请参见“XML 数据库 [▶ 131]”部分。

数据库服务器配置器目前支持哪些 Visual Studio 版本?

目前，我们的配置器集成 [▶ 23]支持 Visual Studio® 版本 2013、2015 和 2017。

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

Third-party trademark statements

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

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

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

UNIX is a registered trademark in the United States and other countries, licensed exclusively through X/Open Company Limited.

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