SIEMENS

SIMATIC

使用脚本实现项目自动化

系统手册

打印在线帮助

05/2017

法律资讯

警告提示系统

为了您的人身安全以及避免财产损失，必须注意本手册中的提示。人身安全的提示用一个警告三角表示，仅与财产损失有关的提示不带警告三角。警告提示根据危险等级由高到低如下表示。

⚠ 危险
表示如果不采取相应的小心措施，**将会**导致死亡或者严重的人身伤害。

⚠ 警告
表示如果不采取相应的小心措施，**可能**导致死亡或者严重的人身伤害。

⚠ 小心
表示如果不采取相应的小心措施，可能导致轻微的人身伤害。

注意
表示如果不采取相应的小心措施，可能导致财产损失。

当出现多个危险等级的情况下，每次总是使用最高等级的警告提示。如果在某个警告提示中带有警告可能导致人身伤害的警告三角，则可能在该警告提示中另外还附带有可能导致财产损失的警告。

合格的专业人员

本文件所属的产品/系统只允许由符合各项工作要求的**合格人员**进行操作。其操作必须遵照各自附带的文件说明，特别是其中的安全及警告提示。 由于具备相关培训及经验，合格人员可以察觉本产品/系统的风险，并避免可能的危险。

按规定使用 Siemens 产品

请注意下列说明：

⚠ 警告
Siemens 产品只允许用于目录和相关技术文件中规定的使用情况。如果要使用其他公司的产品和组件，必须得到 Siemens 推荐和允许。正确的运输、储存、组装、装配、安装、调试、操作和维护是产品安全、正常运行的前提。必须保证允许的环境条件。必须注意相关文件中的提示。

商标

所有带有标记符号 ® 的都是西门子股份有限公司的注册商标。本印刷品中的其他符号可能是一些其他商标。若第三方出于自身目的使用这些商标，将侵害其所有者的权利。

责任免除

我们已对印刷品中所述内容与硬件和软件的一致性作过检查。然而不排除存在偏差的可能性，因此我们不保证印刷品中所述内容与硬件和软件完全一致。印刷品中的数据都按规定经过检测，必要的修正值包含在下一版本中。

Siemens AG
Division Digital Factory
Postfach 48 48
90026 NÜRNBERG
德国

℗ 06/2017 本公司保留更改的权利

Copyright © Siemens AG 2017.
保留所有权利

目录

安全说明

1

安全性信息

西门子为其产品及解决方案提供工业安全功能，以支持工厂、解决方案、机器、设备和/或网络的安全运行。

为了防止工厂、系统、机器和网络受到网络攻击，需要实施并持续维护先进且全面的工业安全保护机制。Siemens 的产品和解决方案仅构成此类概念的其中一个要素。

客户负责防止其工厂、系统、机器和网络受到未经授权的访问。只有在必要时并采取适当安全措施（例如，使用防火墙和网络分段）的情况下，才能将系统、机器和组件连接到企业网络或 Internet。

此外，应考虑遵循 Siemens 有关相应安全措施的指南。更多有关工业安全的信息，请访问

http://www.siemens.com/industrialsecurity (http://www.industry.siemens.com/topics/global/en/industrial-security/Pages/Default.aspx)

Siemens 不断对产品和解决方案进行开发和完善以提高安全性。Siemens 强烈建议您及时更新产品并始终使用最新产品版本。如果使用的产品版本不再受支持，或者未能应用最新的更新程序，客户遭受网络攻击的风险会增加。

要及时了解有关产品更新的信息，请订阅 Siemens 工业安全 RSS 源，网址为

http://www.siemens.com/industrialsecurity (http://www.industry.siemens.com/topics/global/en/industrial-security/Pages/Default.aspx)

TIA Portal Openness 自述文件

2

2.1 自述文件

Openness 应用中的安全措施

建议：

- 使用 admin 权限将 openness 应用程序安装到程序文件中。
- 避免从用户区域动态加载程序部件（如，程序集或 DLL 文件）。
- 使用 user 权限，运行 openness 应用程序。

复制 Openness 应用程序

在复制可执行的 Openness 应用程序时，某些条件下可能会发生这种情况：Openness 应用程序将会读取最初创建 Openness 应用程序时使用的目录路径。

解决方法：

如果已将 Openness 应用程序复制到新目录中，请打开然后再关闭属性对话框，以更新 Windows 缓存。

Openness 多用户支持

不建议在多用户项目中使用 Openness，因而 Openness 不支持多用户操作。请注意，有些 Openness 操作会对 TIA portal 的 GUI 所强制的多用户工作流产生实际影响。

如果仍要针对 Openness 进行修改，请先将多用户项目导出到单用户项目。

Openness 性能的改善

要实现最佳的 Openness 性能，可关闭 TIA Portal 的全局搜索功能。请使用 GUI 或 Openness API 调用来关闭全局搜索。完成 Openness 脚本后，全局搜索功能会重新开启。这改善了性能，但如果开启全局搜索功能，所有 Openness 功能也会正常运行。

故障安全和 Openness

使用 Openness 时，存在与故障安全相关的限制。更多信息，请参见文档《SIMATIC Safety - 组态和编程》(SIMATIC Safety - Configuring and Programming)

线程保存程序代码

请注意，用户代码按线程保存，因此事件将在不同的线程中显示。

启用样式时的画面项导出行为

在启用样式时导出画面项将不会导出样式项的属性，而是导出激活样式前的画面项属性。如果选择了一种样式并为画面项选中了 UseDesignColorSchema，则画面项会获取用户界面中样式的属性值，但在选择样式前设置的画面项属性值仍会存储在此画面项的数据库中。Openness 会导出存储在数据库中的实际值。

禁用和启用样式并再次导出画面项后，将为画面项导出与样式项相同的属性值。如果未选中 UseDesignColorSchema，则所选样式项的属性值将保存到相应画面项的数据库中。

这个问题可以通过以下步骤来解决：

1. 将画面项与样式项关联：
   - 数据库包含激活样式前的属性值。
   - 用户界面直接从样式项获取属性。
2. 导出与样式项关联的画面项：
   - XML 文件包含激活样式前数据库中的属性值。
3. 禁用 UseDesignColorSchema：
   - 样式项的属性值将写入数据库中画面项的属性中。
4. 启用 UseDesignColorSchema：
   - 数据库中画面项的属性值不会更改，仍为步骤 3 中的值。
   - 用户界面直接从样式项获取属性。
5. 导出与样式项关联的画面项：
   - XML 文件包含步骤 3 中设置的数据库中的属性值，这些值与样式项中的值相同。

复制 S7-1500 运动控制工艺对象

不能将 TO_CamTrack、TO_OutputCam 或 TO_MeasuringInput 从项目库或全局库复制到项目。

通过 AML 导入 ASi 从站

如果下列 ASi 从站之一通过 aml 文件导入，则不论在何种情况下，设备项的固件版本均会设置为 V13.0：

- ASIsafe FS400 RCV-B: 3SF7 844-*B***-***1
- ASIsafe FS400 RCV-M: 3SF7 844-*M***-***1

- ASIsafe FS400 TRX-M: 3SF7 844-*M***-**T0
- ASIsafe FS400 RCV-C: 3SF7 844-*T***-***1

Openness V14 SP1 有哪些新增功能？ 3

全新的 Openness 对象模型

与 Openness V14 相比，Openness V14 SP1 中的对象模型发生了更改和扩展。例如：

- 库处理能力得到改善
- 通过 AutomationML 可导入和导出 CAx 数据
- 不再需要启动程序和使用文件。
- 通过 Openness API 处理工艺对象 TO
- 通过 Openness API 创建和修改硬件组态

更多详细信息，请参见 V14 SP1 中的主要变更 (页 505)。

说明

Openness V14 及早期版本所使用应用程序的程序代码已更新。

参见

处理异常 (页 331)

主要字符串变更 (页 547)

Openness 对象模型 (页 45)

主要变化 (页 505)

基本知识

4

4.1 TIA Portal Openness 的要求

Openness 应用程序的使用要求

- 在 PC 上安装基于 TIA Portal 的产品，例如“STEP 7 Professional”或“WinCC Professional”。
- PC 上安装了“TIA Portal Openness”附加软件包。
  请参见 TIA Portal Openness 的安装 (页 21)

支持的 Windows 操作系统

下表给出了相互兼容的 Windows 操作系统、TIA portal 和用户应用程序组合：

Windows 操作系统	TIA Portal	用户应用程序
64 位	64 位	32 位和 64 位，“任意 CPU”

Openness 应用程序的编程要求

- 包含 .Net 4.6.2 的 Microsoft Visual Studio 2015 Update 1 或以上版本

用户必备知识

- 作为系统工程师所必备的知识
- 包含 .Net 4.6.2 的 Microsoft Visual Studio 2015 Update 1 或更高版本的高级知识
- C#/VB 和 .Net 的高级知识
- TIA Portal 的用户知识

Openness 远程通道

Openness 远程通道注册为类型 IpcChannel，其中“ensureSecurity”参数设置为“false”。

说明

应该避免使用“false”之外的“ensureSecurity”参数值注册 IpcChannel，并且优先级大于等于“1”。

IpcChannel 使用以下属性定义：

属性	设置
"name" 和 "portName"	在 AppDomain 中注册时设置为 $”{Process.Name}_{Process.Id}” 或 $”{Process.Name}_{Process.Id}_{AppDomain.Id}”，而非应用程序的默认值。
“priority”	设置为默认值“1”。
“typeFilterLevel”	设置为“Full”。
“authorizedGroup”	设置为内置用户帐户（即，所有人）的 NTAccount 值字符串。

参见

向“Siemens TIA Openness”用户组添加用户 (页 22)

4.2 安装

4.2.1 TIA Portal Openness 的安装

简介

安装程序会自动安装“TIA Portal Openness ”附加软件包。安装程序文件为自解压的归档文件，位于产品 DVD 的“Support"目录下。

要求

- PG/PC 的硬件和软件满足系统要求。
- 具有管理员权限。
- 关闭正在运行的程序。
- 禁用自动运行功能。
- 已安装 WinCC 和/或 STEP 7。
- “TIA Portal Openness”附加软件包的版本号与 WinCC 和 STEP 7 的版本号相匹配。

说明

如果已安装早期版本的 TIA Portal Openess，则当前版本将与其并排安装。

步骤

说明

在完成 TIA Portal 安装之前，无法安装“TIA Portal Openness”附加软件包。

要安装附加件包，请按以下步骤操作：

1. 将安装介质置于对应的驱动器中，并切换至“Support”目录。
2. 双击文件“Siemens_TIA_Openness_V14_SP1.exe”。
3. 选择安装语言。
4. 提取安装数据。之后，将开始安装 Openness。
   将打开下面的安装程序对话框：
5. 单击“Next”并选择所需选项。

6. 单击“下一步”(Next) 按钮并选择“TIA Portal Openness”产品。
   如有必要，可更改安装的目标目录，然后单击“Next”。
   请注意，安装路径的长度不能超过 89 个字符。
7. 在下一个安装步骤中，接受所有许可协议并单击“Next”。
   如果在安装 TIA Portal Openness 时需要更改安全和权限设置，则会打开安全设置对话框。
8. 要继续安装，请接受对安全和权限设置的更改，并单击“Next”按钮。
   下一对话框将显示安装设置的概述。
9. 检查所选安装设置并根据需要进行更改。
10. 单击“Install”以启动安装。
    如果安装成功，屏幕上会显示一条有关安装成功的消息。如果安装期间出现错误，则会显示一条错误消息，告知用户这些错误的性质。

结果

PC 上安装了“TIA Portal Openness”附加软件包。此外，还生成了本地用户组“Siemens TIA Openness”。

说明

安装“TIA Portal Openness”附加软件包之后，您仍然无权访问 TIA Portal。您必须是“Siemens TIA Openness”用户组的成员（请参见向“Siemens TIA Openness”用户组添加用户 (页 22)）。

4.2.2 向“Siemens TIA Openness”用户组添加用户

简介

在 PC 上安装 TIA Portal Openness 时，会自动创建“Siemens TIA Openness”用户组。

无论何时通过 Openness 应用程序访问 TIA Portal，TIA Portal 都会验证您是否通过其它用户组直接或间接地成为“Siemens TIA Openness”用户组的成员。如果您是“Siemens TIA Openness”用户组的成员，则 Openness 应用程序将启动并与 TIA Portal 建立连接。

步骤

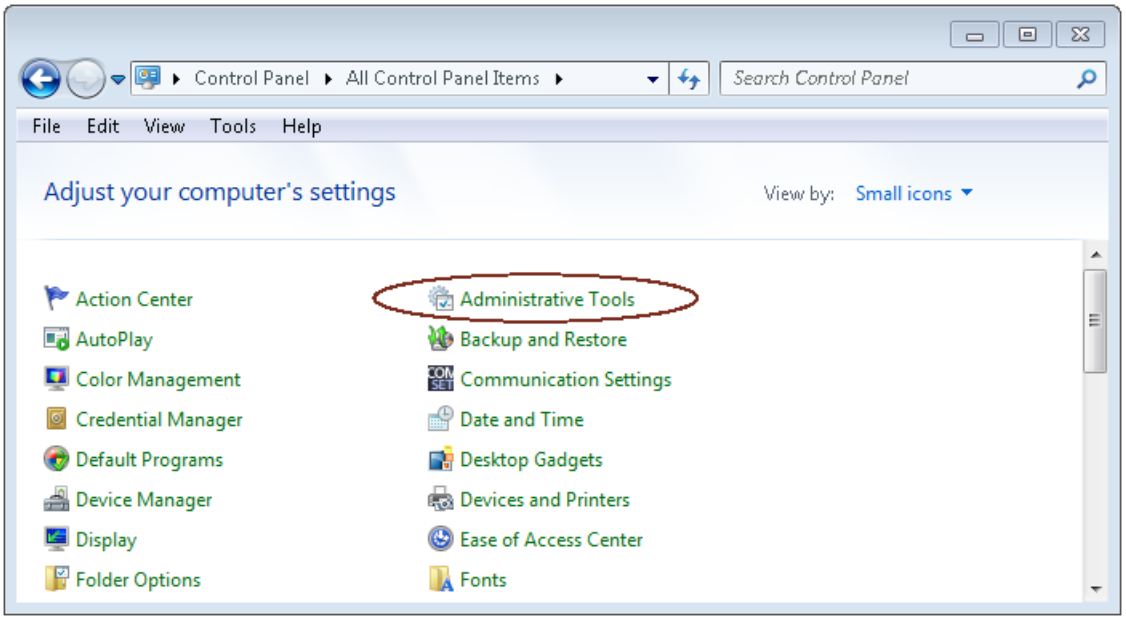
通过操作系统中的应用程序向“Siemens TIA Openness”用户组添加用户。TIA Portal 不支持该操作。

说明

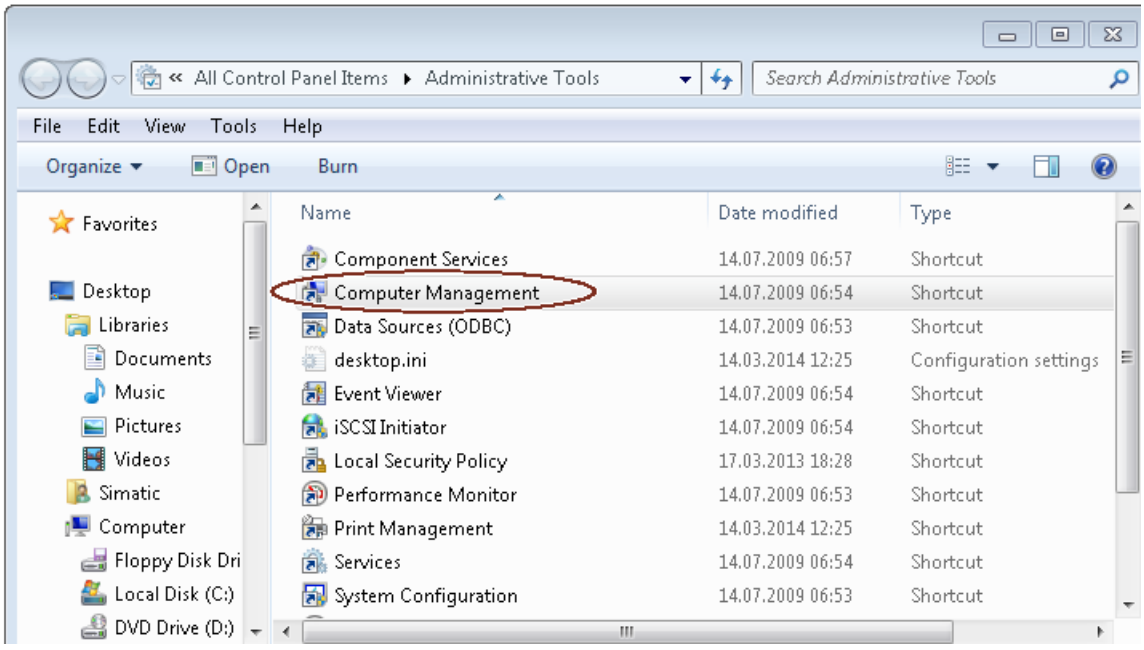
根据您的域或计算机的配置，可能需要使用管理员权限进行登录来扩展用户组。

例如，在 Windows 7 操作系统（语言设置为英语）中，可按以下步骤向用户组添加用户：

1. 选择 "Start" > "Control Panel"。
2. 在控制面板中双击 "Administrative Tools"。

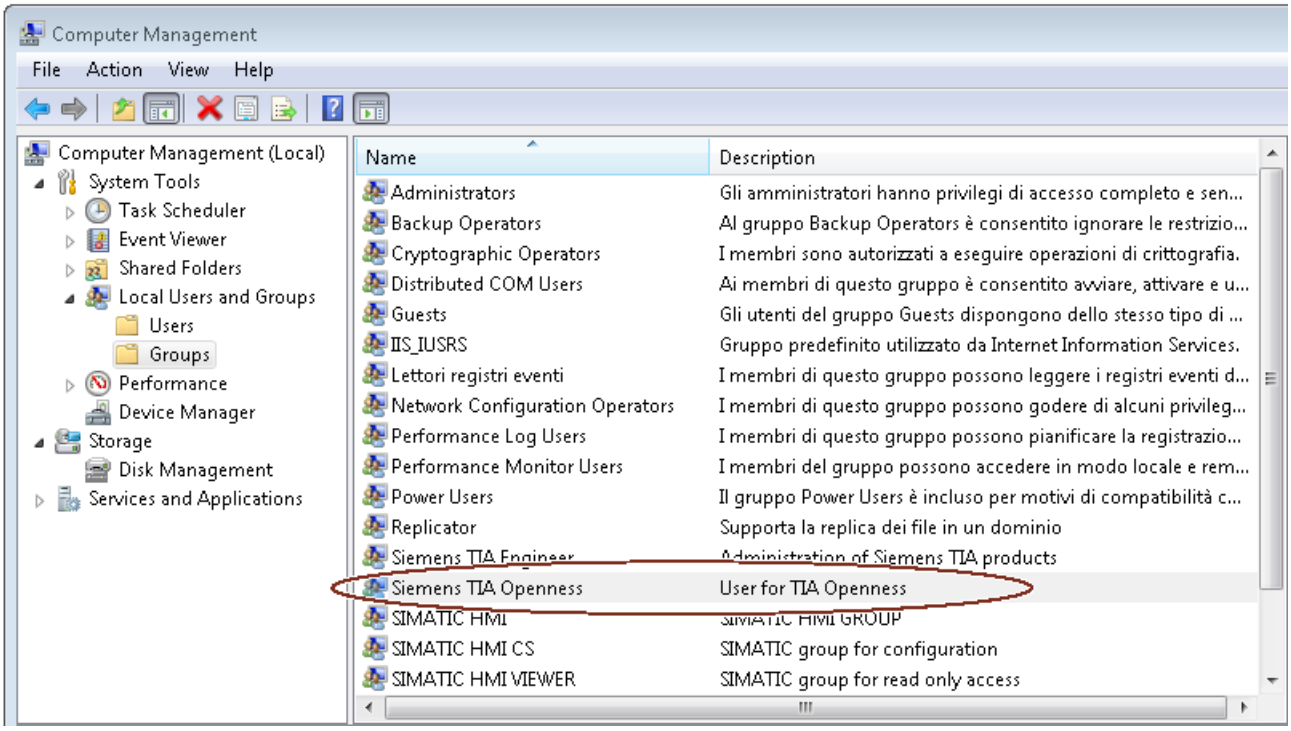


3. 单击 "Computer Management" 打开同名的配置对话框。

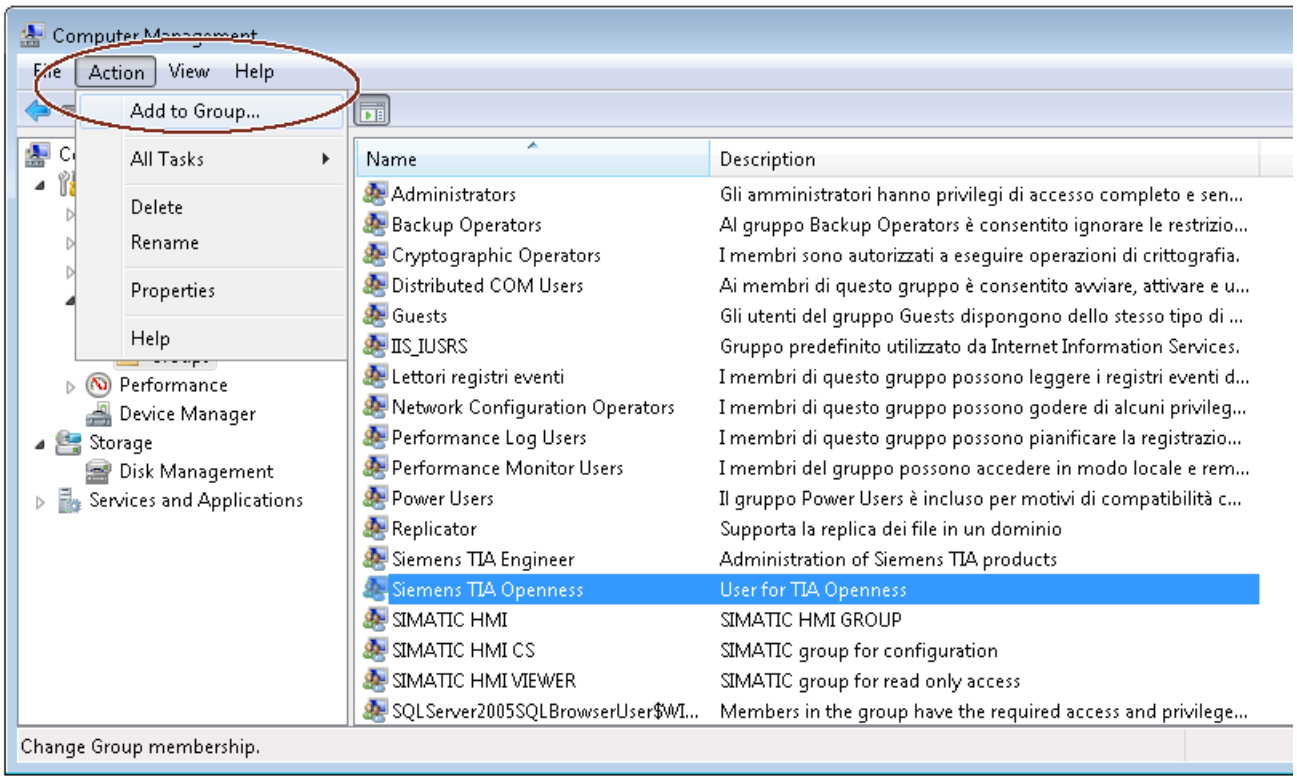


4. 选择 "Local Users and Groups > Groups" 以显示创建的所有用户组。

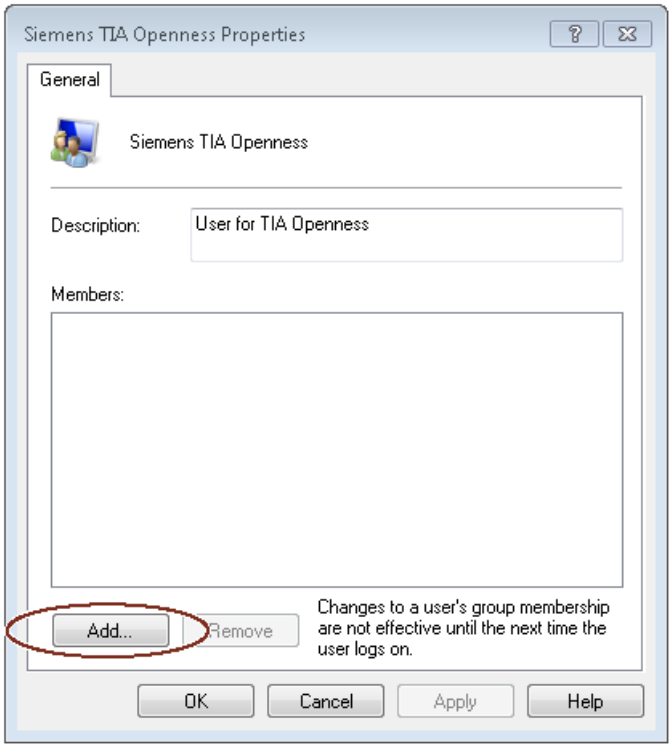
5. 在右侧窗格的用户组列表中选择 "Siemens TIA Openness" 条目。



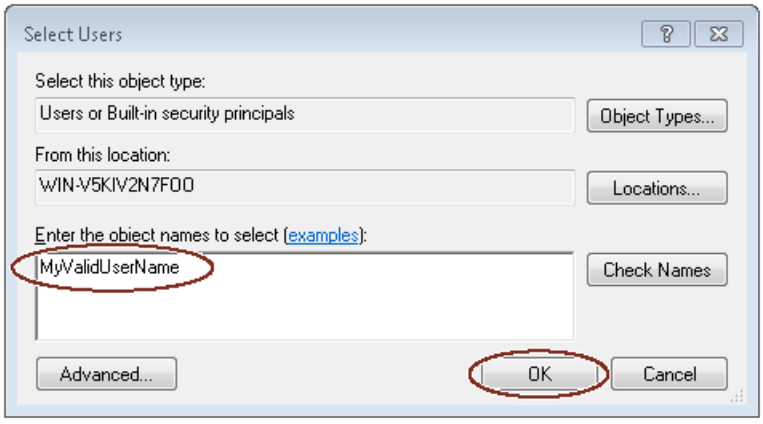
6. 选择“Action > Add to Group...”菜单命令。



将打开用户组属性对话框：



7. 单击“Add"”。
打开的选择对话框将显示可以选择的用户：



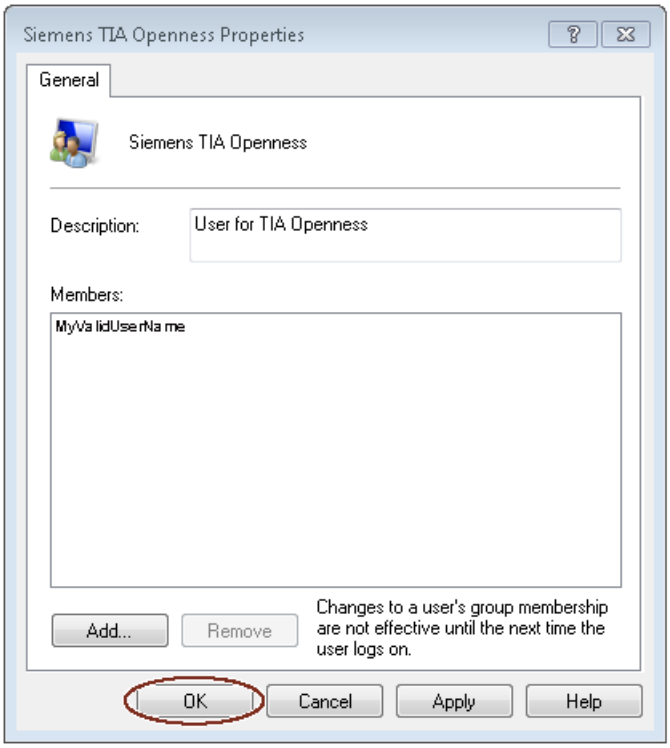
8. 在输入字段中键入有效的用户名。

说明

单击 "Check Names" 以验证已输入用户名是否具有对此域或计算机有效的用户账户。

"From this location" 字段显示已输入用户名的域或计算机名。有关详细信息，请联系您的系统管理员。

9. 单击 "OK" 确认选择。
现在，新用户将显示在用户组的属性对话框中。



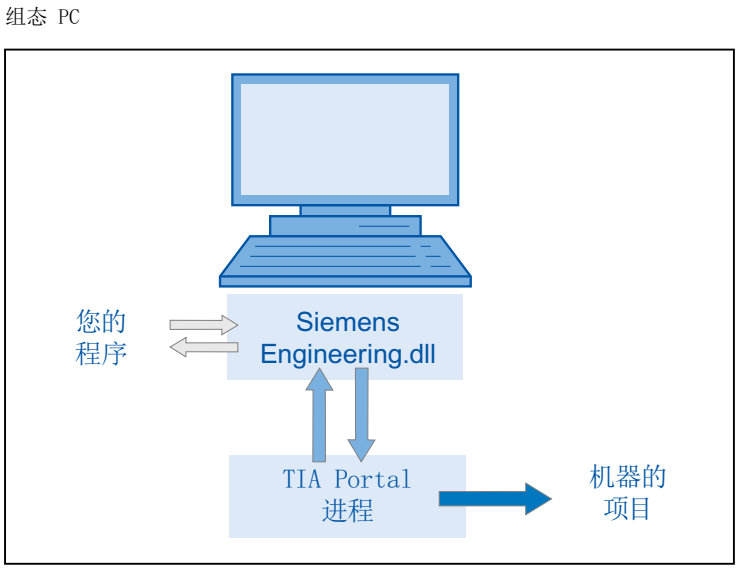
可通过单击 "Add" 按钮注册其它用户。

10.单击 "OK" 结束此操作。

11.重新登录 PC 以使更改生效。

4.2.3 访问 TIA Portal

概述



步骤

1. 设置开发环境以访问和启动 TIA Portal。
2. 实例化程序中门户应用程序的对象以启动 Portal。
3. 找到所需项目并将其打开。
4. 访问项目数据。
5. 关闭项目并退出 TIA Portal。

参见

连接到 TIA Portal (页 69)

终止与 TIA Portal 的连接 (页 81)

4.3 Openness 任务

4.3.1 应用程序

简介

TIA Portal Openness 为您提供了多种访问 TIA Portal 的方式，并且针对定义的任务提供了多个函数供您选择。

可通过 Public API 接口访问 TIA Portal 的以下区域：

- 项目数据
- PLC 数据
- HMI 数据

说明

不得使用公共 API 执行故障安全系统的验收/认证检查或生成相关数据。只能通过使用 STEP 7 安全附加软件包的安全打印输出或通过功能测试进行验收/认证。不能使用公共 API 替代。

访问 TIA Portal

TIA Portal Openness 提供了多种访问 TIA Portal 的方式。无论是否使用用户界面，用户均可在过程中创建外部 TIA Portal 实例。同时，还可以访问当前 TIA Portal 过程。

访问项目和项目数据

访问项目和项目数据时，主要使用 TIA Portal Openness 来执行以下任务：

- 关闭、打开和保存项目
- 枚举和查询对象
- 创建对象
- 删除对象

4.3.2 导出/导入

简介

TIA Portal Openness 支持通过 XML 文件导入和导出项目数据。导入/导出功能支持现有工程数据的外部组态。使用该功能可使工程过程高效无误。

应用

可使用导入/导出功能进行以下操作：

- 数据交换
- 复制项目的一部分
- 对组态数据进行外部处理，例如，使用查找和替换功能对数据进行批量操作
- 基于现有组态对新项目的组态数据进行外部处理
- 导入外部创建的组态数据，例如文本列表和变量
- 为外部应用程序提供项目数据

4.4 对象列表

简介

下表列出了 Runtime Advanced 及之前版本的可用对象，并指明了这些对象是否受 Openness 支持。

在 WinCC 中，Openness 不支持 Runtime Professional 和设备代理文件。

对象

可根据您所使用的 HMI 的设备控制以下项目数据：

表格 4-1 画面

对象	精简系列面板	面板	精智面板	多功能面板	移动面板	RT Advanced
画面	可用	可用	可用	可用	可用	可用
全局画面	可用	可用	可用	可用	可用	可用
模板	可用	可用	可用	可用	可用	可用
永久性区域	不可用	可用	可用	可用	可用	可用
弹出画面	不可用	不可用	可用	不可用	可用	可用
滑入画面	不可用	不可用	可用	不可用	可用	可用

表格 4-2 画面对象

对象	精简系列面板	面板	精智面板	多功能面板	移动面板	RT Advanced
直线	可用	可用	可用	可用	可用	可用
折线	不可用	可用	可用	可用	可用	可用
多边形	不可用	可用	可用	可用	可用	可用
椭圆	可用	可用	可用	可用	可用	可用
部分椭圆	不可用	不可用	不可用	不可用	不可用	不可用
扇形	不可用	不可用	不可用	不可用	不可用	不可用
椭圆弧	不可用	不可用	不可用	不可用	不可用	不可用
摄像机视图	不可用	不可用	不可用	不可用	不可用	不可用
圆弧	不可用	不可用	不可用	不可用	不可用	不可用

对象	精简系列面板	面板	精智面板	多功能面板	移动面板	RT Advanced
圆	可用	可用	可用	可用	可用	可用
PDF 视图	不可用	不可用	不可用	不可用	不可用	不可用
矩形	可用	可用	可用	可用	可用	可用
连接器	不可用	不可用	不可用	不可用	不可用	不可用
文本字段	可用	可用	可用	可用	可用	可用
图形视图	可用	可用	可用	可用	可用	可用
管道	不可用	不可用	不可用	不可用	不可用	不可用
双 T 形管	不可用	不可用	不可用	不可用	不可用	不可用
T 形管	不可用	不可用	不可用	不可用	不可用	不可用
管道弯头	不可用	不可用	不可用	不可用	不可用	不可用
I/O 字段	可用	可用	可用	可用	可用	可用
日期/时间字段	可用	可用	可用	可用	可用	可用
图形 I/O 字段	可用	可用	可用	可用	可用	可用
可编辑的文本字段	不可用	不可用	不可用	不可用	不可用	不可用
列表框	不可用	不可用	不可用	不可用	不可用	不可用
组合框	不可用	不可用	不可用	不可用	不可用	不可用
按钮	可用	可用	可用	可用	可用	可用
圆形按钮	不可用	不可用	不可用	不可用	不可用	不可用
指示灯按钮	不可用	不可用	不可用	不可用	可用	不可用
开关	可用	可用	可用	可用	可用	可用
符号 I/O 字段	可用	可用	可用	可用	可用	可用
钥匙开关	不可用	不可用	不可用	不可用	可用	不可用
棒图	可用	可用	可用	可用	可用	可用
符号库	不可用	可用	可用	可用	可用	可用
滑块	不可用	可用	可用	可用	可用	可用
滚动条	不可用	不可用	不可用	不可用	不可用	不可用
复选框	不可用	不可用	不可用	不可用	不可用	不可用
选项按钮	不可用	不可用	不可用	不可用	不可用	不可用
量表	不可用	可用	可用	可用	可用	可用

对象	精简系列面板	面板	精智面板	多功能面板	移动面板	RT Advanced
时钟	不可用	可用	可用	可用	可用	可用
存储空间视图	不可用	不可用	不可用	不可用	不可用	不可用
功能键	可用	可用	可用	可用	可用	可用
面板实例	不可用	不可用	是	是	是	是
画面窗口	不可用	不可用	不可用	不可用	不可用	不可用
用户视图	可用	可用	可用	可用	可用	可用
HTML 浏览器	不可用	不可用	不可用	不可用	不可用	不可用
打印作业/脚本诊断	不可用	不可用	不可用	不可用	不可用	不可用
配方视图	不可用	不可用	不可用	不可用	不可用	不可用
报警视图	不可用	不可用	不可用	不可用	不可用	不可用
报警指示器	不可用	不可用	不可用	不可用	不可用	不可用
报警窗口	不可用	不可用	不可用	不可用	不可用	不可用
f(x) 趋势视图	不可用	不可用	不可用	不可用	不可用	不可用
f(t) 趋势视图	不可用	不可用	不可用	不可用	不可用	不可用
表格视图	不可用	不可用	不可用	不可用	不可用	不可用
数值表	不可用	不可用	不可用	不可用	不可用	不可用
媒体播放器	不可用	不可用	不可用	不可用	不可用	不可用
通道诊断	不可用	不可用	不可用	不可用	不可用	不可用
WLAN 接收	不可用	不可用	不可用	不可用	不可用	不可用
区域名称	不可用	不可用	不可用	不可用	不可用	不可用
区域信号	不可用	不可用	不可用	不可用	不可用	不可用
有效范围名称	不可用	不可用	不可用	不可用	不可用	不可用
有效范围名称 (RFID)	不可用	不可用	不可用	不可用	不可用	不可用
有效范围信号	不可用	不可用	不可用	不可用	不可用	不可用
充电状况	不可用	不可用	不可用	不可用	不可用	不可用
手轮	不可用	不可用	不可用	不可用	可用	不可用
帮助指示器	不可用	不可用	不可用	不可用	不可用	不可用

对象	精简系列面板	面板	精智面板	多功能面板	移动面板	RT Advanced
Sm@rtClient 视图	不可用	不可用	不可用	不可用	不可用	不可用
状态/强制	不可用	不可用	不可用	不可用	不可用	不可用
系统诊断视图	不可用	不可用	不可用	不可用	不可用	不可用
系统诊断窗口	不可用	不可用	不可用	不可用	不可用	不可用

表格 4-3 动态

对象	精简系列面板	面板	精智面板	多功能面板	移动面板	RT Advanced
显示	可用	可用	可用	可用	可用	可用
可操作性	不可用	可用	可用	可用	可用	可用
可见性	可用	可用	可用	可用	可用	可用
移动	可用	可用	可用	可用	可用	可用

表格 4-4 其它对象

对象	精简系列面板	面板	精智面板	多功能面板	移动面板	RT Advanced
组	可用	可用	可用	可用	可用	可用
软键	可用	可用	可用	可用	可用	可用
周期	可用	可用	可用	可用	可用	可用
VB 脚本	不可用	可用	可用	可用	可用	可用
函数列表	可用	可用	可用	可用	可用	可用
项目图形	可用	可用	可用	可用	可用	可用

表格 4-5 变量

对象	精简系列面板	面板	精智面板	多功能面板	移动面板	RT Advanced
指针变量	可用	可用	可用	可用	可用	可用
数组	可用	可用	可用	可用	可用	可用
用户数据类型	可用	可用	可用	可用	可用	可用
Internal	不可用	可用	可用	可用	可用	可用

对象	精简系列面板	面板	精智面板	多功能面板	移动面板	RT Advanced
基本数据类型的使用点	可用	可用	可用	可用	可用	可用
用户数据类型的使用点	可用	可用	可用	可用	可用	可用
数组的使用点	可用	可用	可用	可用	可用	可用

Openness 还支持通信驱动程序所支持的全部值范围。

连接

Openness 支持同样受相应 HMI 设备支持的非集成连接。有关更多信息，请参见 TIA Portal 在线帮助的“过程可视化 > 控制器通信 > 设备相关”(Process visualization > Controller communication > Device-dependent)。

表格 4-6　列表

对象	精简系列面板	面板	精智面板	多功能面板	移动面板	RT Advanced
文本列表	可用	可用	可用	可用	可用	可用
图形列表	可用	可用	可用	可用	可用	可用

表格 4-7　文本

对象	精简系列面板	面板	精智面板	多功能面板	移动面板	RT Advanced
多语言文本	可用	可用	可用	可用	可用	可用
格式化文本及其实例	不可用	可用	可用	可用	可用	可用

4.5 标准库

在相关代码示例的开头插入以下名称空间语句，以确保代码示例正常运行：

```
using System;
using Siemens.Engineering;
using Siemens.Engineering.CAx;
using Siemens.Engineering.HW;
using Siemens.Engineering.HW.Extension;
using Siemens.Engineering.HW.Features;
using Siemens.Engineering.HW.Utilities;
using Siemens.Engineering.SW;
using Siemens.Engineering.SW.Blocks;
using Siemens.Engineering.SW.TechnologicalObjects;
using Siemens.Engineering.SW.TechnologicalObjects.Motion;
using Siemens.Engineering.SW.ExternalSources;
using Siemens.Engineering.SW.Tags;
using Siemens.Engineering.SW.Types;
using Siemens.Engineering.Hmi;
using Siemens.Engineering.Hmi.Tag;
using Siemens.Engineering.Hmi.Screen;
using Siemens.Engineering.Hmi.Cycle;
using Siemens.Engineering.Hmi.Communication;
using Siemens.Engineering.Hmi.Globalization;
using Siemens.Engineering.Hmi.TextGraphicList;
using Siemens.Engineering.Hmi.RuntimeScripting;
using System.Collections.Generic;
using Siemens.Engineering.Online;
using Siemens.Engineering.Compiler;
using Siemens.Engineering.Library;
using Siemens.Engineering.Library.Types;
using Siemens.Engineering.Library.MasterCopies;
using Siemens.Engineering.Compare;
using System.IO;
```

4.6 关于 TIA Portal Openness 性能的说明

根对象

可以在导入文件中指定多个根对象。

示例：可以在一个 XML 文件中创建多个文本列表，而非只能创建一个文本列表。

Openness 功能

首次调用 Openness 功能所需的时间比后续调用 Openness 功能所需的时间长。

示例：如果相继执行多次组态数据导出操作，则首次导出所需的时间要比后续导出所需的时间长。

简介 5

简介

TIA Portal Openness 介绍了使用 TIA Portal 进行工程组态的开放接口。

TIA Portal Openness 可以通过您创建的程序从外部控制 TIA Portal，从而实现工程组态的自动化。

可借助 TIA Portal Openness 执行以下操作：

- 创建项目数据
- 修改项目和项目数据
- 删除项目数据
- 读取项目数据
- 将项目和项目数据提供给其它应用程序。

说明

对于使用第三方软件通过这些接口传输的信息和数据的兼容性，西门子不会承担责任和提供保证。

我们明确指出：接口的不当使用可能导致数据丢失或停产时间。

说明

本文档中包含的代码片段是使用 C# 语法编写的。

由于所使用的代码片段不足，因此也省去了错误处理的说明。

应用

Openness 接口用于执行以下操作：

- 提供项目数据。
- 访问 TIA Portal 过程。
- 使用项目数据。

使用自动化工程现场中的默认值

- 通过导入外部生成的数据
- 通过远程控制 TIA Portal 生成项目

为外部应用程序提供 TIA Portal 项目数据

- 通过导出项目数据

通过高效的工程组态确保竞争优势

- 无需在 TIA Portal 中组态现有的工程数据。
- 以自动化的工程组态流程替代手动工程组态。
- 与竞争对手相比，较低的工程组态成本巩固了投标地位。

同时使用项目数据

- 测试例程和批量数据处理可以与组态同时执行。

参见

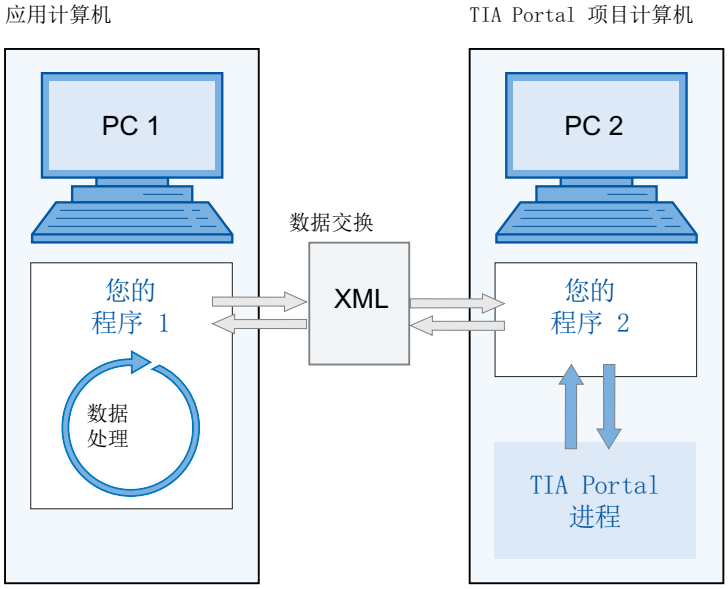
组态 (页 39)

组态

6

使用 TIA Portal Openness 分为以下两种情况：

应用程序和 TIA Portal 位于不同计算机上



- 通过 XML 文件进行数据交换。可通过程序导出或导入 XML 文件。
- 对于从 TIA Portal 项目导出至 PC2 的数据，可在 PC1 上进行修改并重新导入。

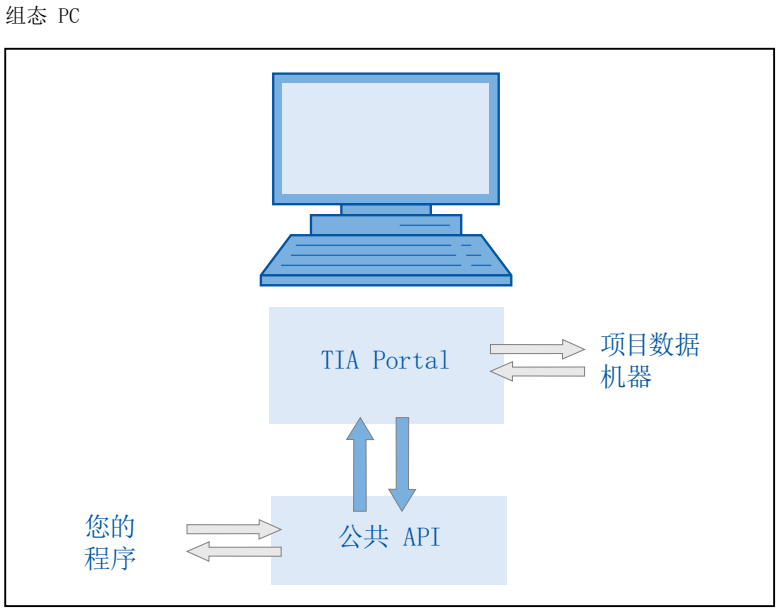
  说明

  必须为 PC2 开发可执行程序“程序 2”，例如“program2.exe”。TIA Portal 与此程序一起在后台运行。

  只能通过公共 API 对 XML 文件进行导入和导出操作。

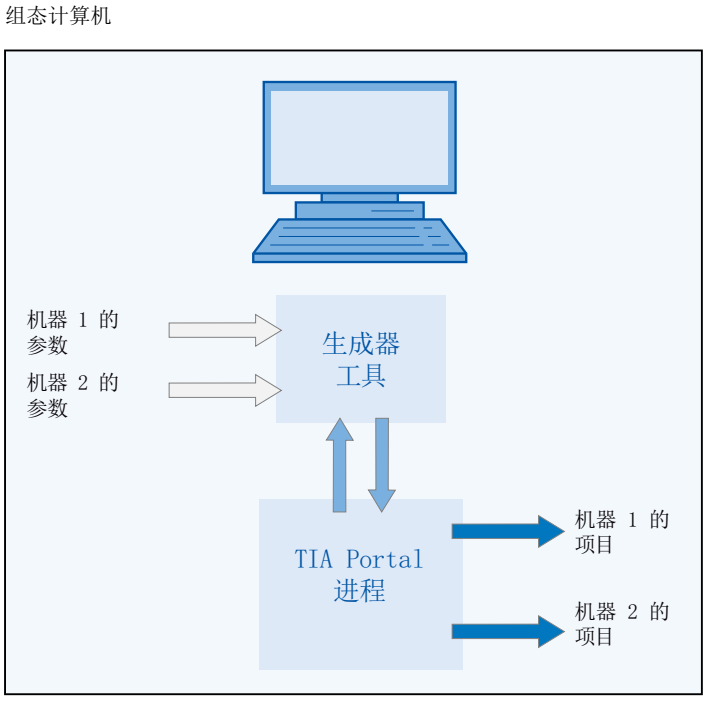
- 可以归档交换的文件以便进行验证。
- 可以在不同的时间和位置对交换的数据进行处理。

应用程序和 TIA Portal 位于同一台计算机上



- 无论是否使用用户界面，程序都可以启动 TIA Portal。程序可打开、保存和/或关闭项目。程序还可连接到正在运行的 TIA Portal。
- 随后可使用 TIA Portal 功能请求、生成和修改项目数据，或启动导入或导出过程。
- 数据在 TIA Portal 处理过程的控制下进行创建，并存储在项目数据中。

模块化机械工程中的典型应用



- 可对类似的机器应用高效的自动化系统。
- TIA Portal 提供包含所有机器型号的组件的项目。
- “生成器”工具控制着针对特定机器型号进行的项目创建。
- “生成器”工具通过读取请求的机器型号的参数来获取默认值。
- “生成器”工具可将相关元素从整个 TIA Portal 项目中过滤出来，并在必要时对其进行修改，以及生成请求的机器项目。

公共 API

7

7.1 简介

概述

TIA Portal Openness 针对定义的任务提供了多个函数供选择，这些函数可从 TIA Portal 外部通过 Public API 进行调用。

说明

如果已安装早期版本的 TIA Portal Openess，则当前版本将与其并排安装。

以下部分概述了典型的编程步骤。以下部分介绍了各代码段的交互方式，以及如何将各个函数集成到完整的程序中。并且还概述了必须针对各个任务进行调整的代码部分。

示例程序

以“在控制台应用程序中创建 API 访问”函数为例对各个编程步骤进行说明。将提供的函数集成到此程序代码中并针对此任务调整相应的代码部分。

函数

以下部分列出了适用于所定义任务的函数，这些函数可在 TIA Portal 外部通过 TIA Portal Openness 进行调用。

参见

应用程序 (页 28)

对象列表 (页 30)

7.2 编程步骤

概述

TIA Portal Openness 需要执行以下编程步骤以通过 Public API 进行访问：

1. 使 TIA Portal 在开发环境中为已知
2. 设置 TIA Portal 的程序访问
3. 激活 TIA Portal 的程序访问
4. 发布并启动 TIA Portal
5. 打开项目
6. 执行命令
7. 保存并关闭项目
8. 终止与 TIA Portal 的连接

说明

允许的字符串

TIA portal 中的字符串仅支持特定字符。通过 Openness 应用程序传送至 TIA Portal 的所有字符串均需遵守这些规则。如果参数中存在无效字符，则会发生异常。

参见

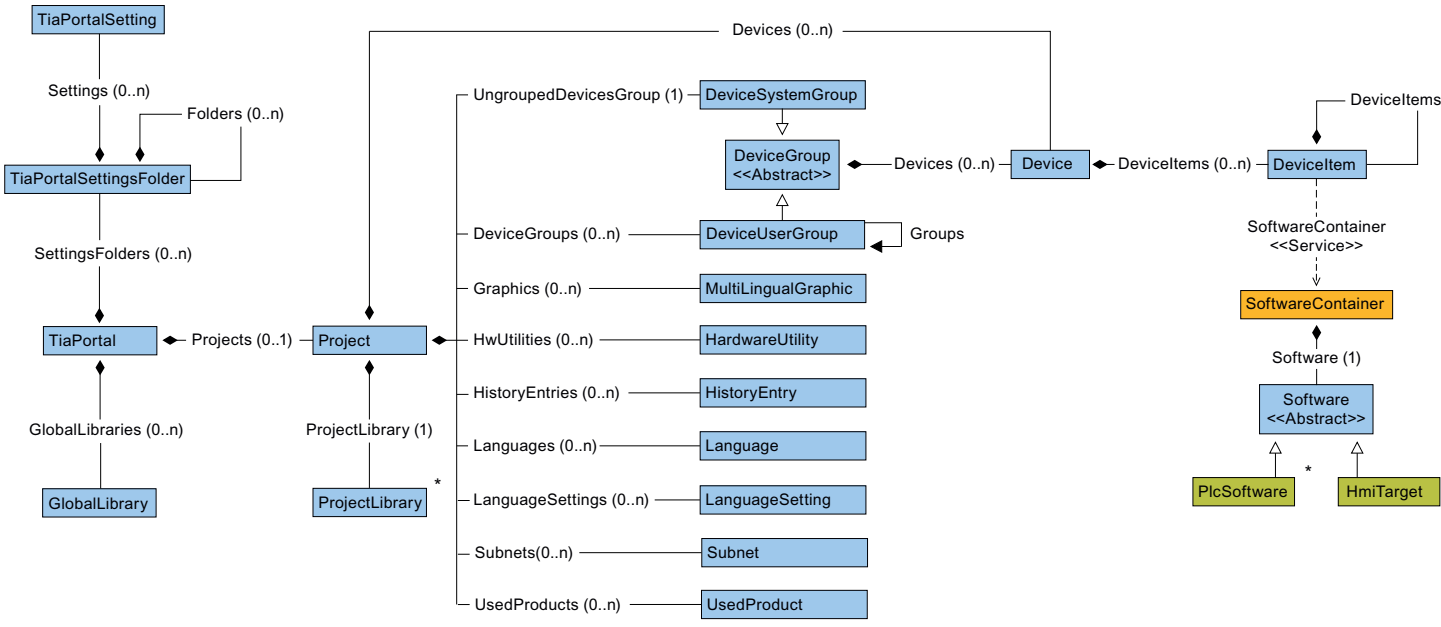
示例程序 (页 61)

使用代码示例 (页 66)

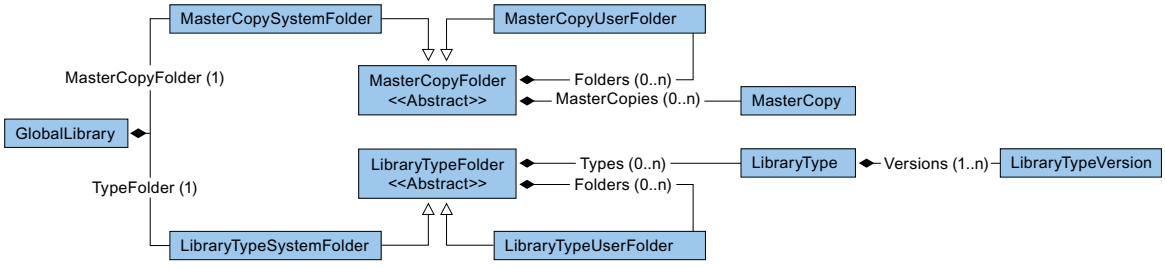
7.3 Openness 对象模型

概述

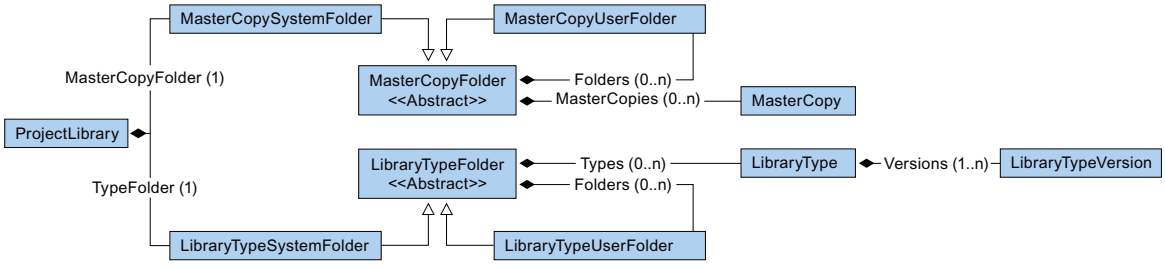
下图介绍了最高级别的 Openness 对象模型：



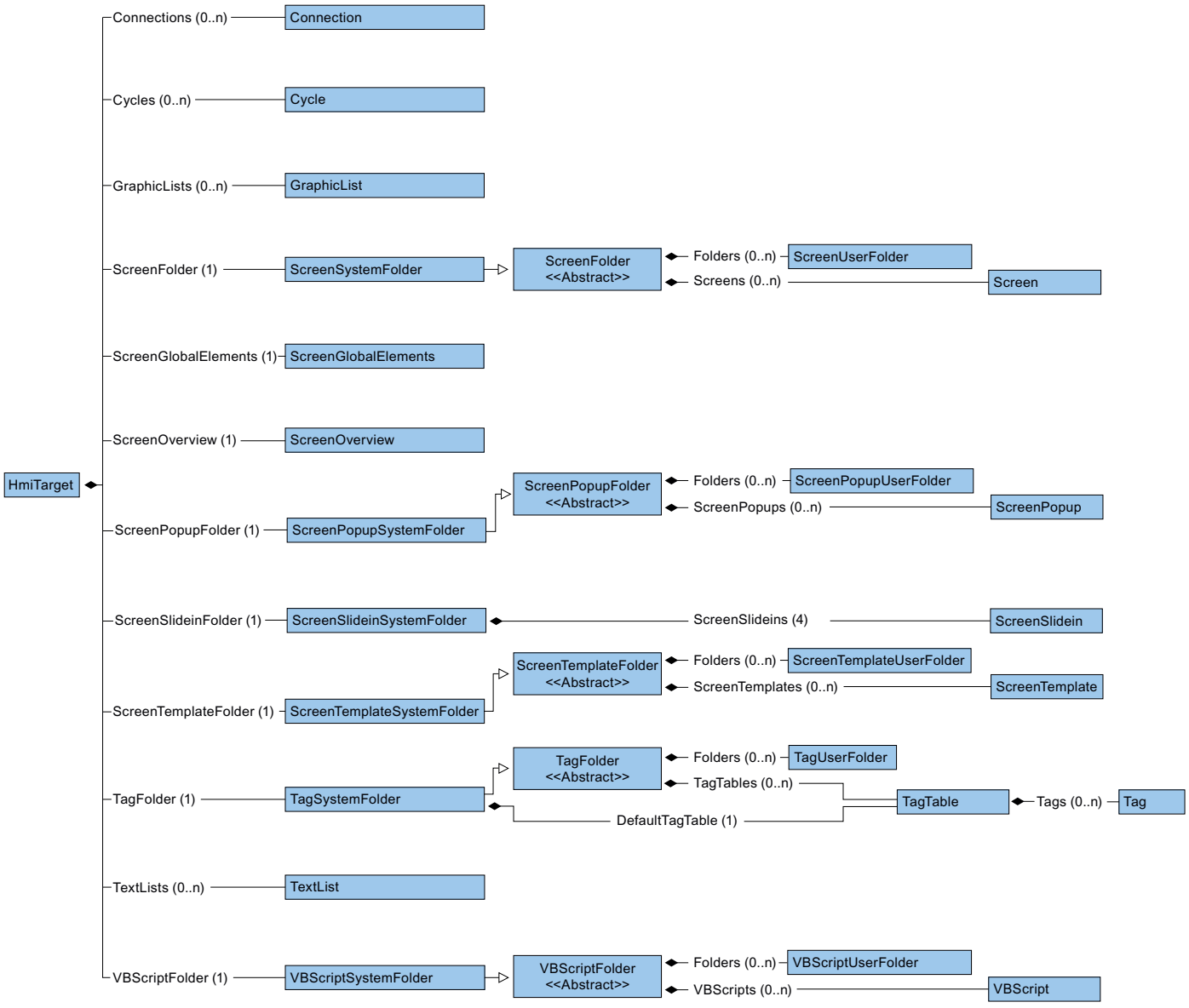
下图显示了位于 GlobalLibrary 中的对象。



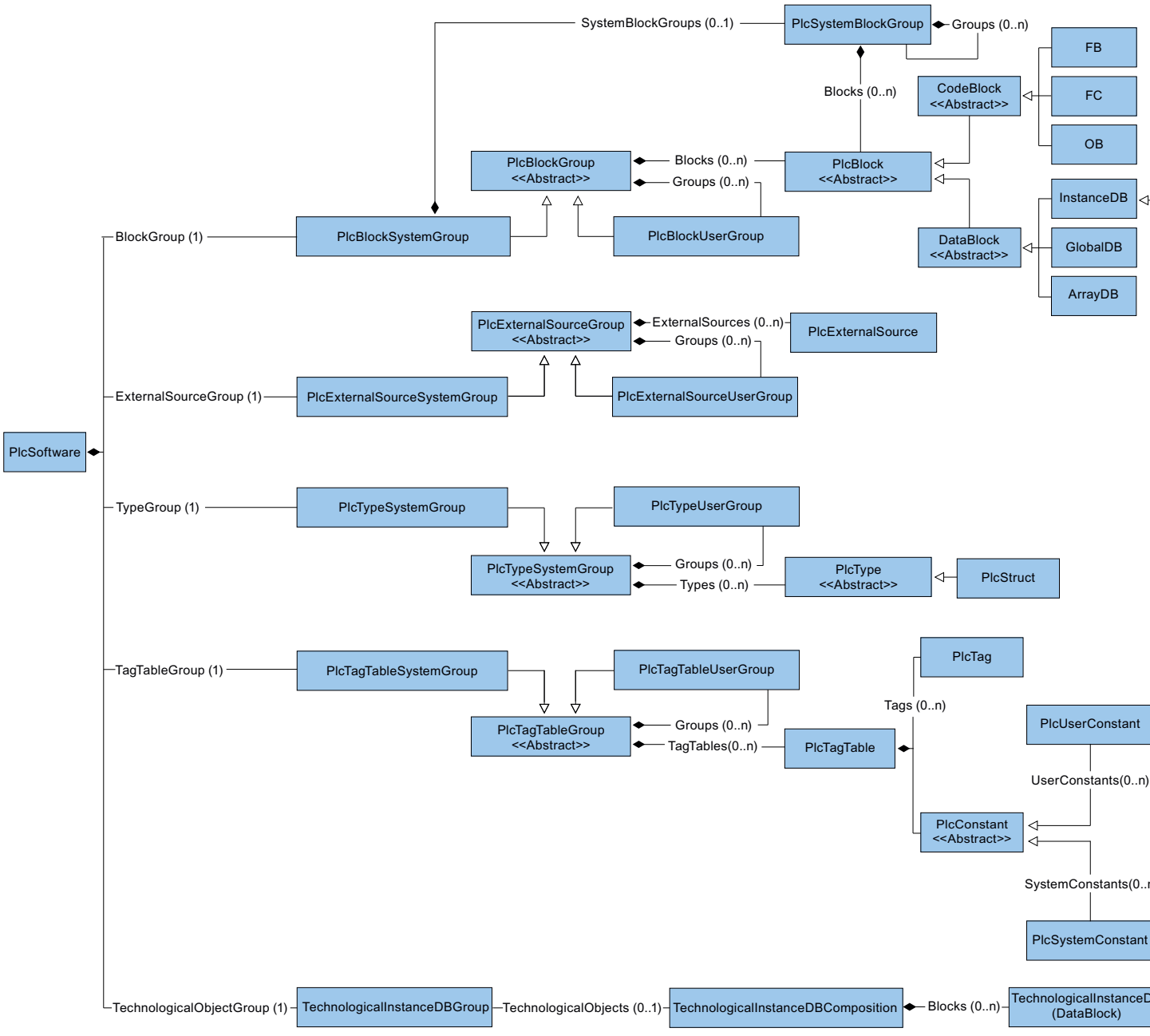
下图所示为 ProjectLibrary 下的对象。



下图所示为 HmiTarget 下的对象。



下图所示为 PlcSoftware 下的对象。


SystemBlockGroups (0..1)
PlcSystemBlockGroup
Groups (0..n)
FB
FC
OB
Blocks (0..n)
CodeBlock
<<Abstract>>
PlcBlockGroup
<<Abstract>>
Blocks (0..n)
Groups (0..n)
PlcBlock
<<Abstract>>
InstanceDB
GlobalDB
ArrayDB
DataBlock
<<Abstract>>
BlockGroup (1)
PlcBlockSystemGroup
PlcBlockUserGroup
PlcExternalSourceGroup
<<Abstract>>
ExternalSources (0..n)
Groups (0..n)
PlcExternalSource
ExternalSourceGroup (1)
PlcExternalSourceSystemGroup
PlcExternalSourceUserGroup
PlcSoftware
TypeGroup (1)
PlcTypeSystemGroup
PlcTypeUserGroup
PlcTypeSystemGroup
<<Abstract>>
Groups (0..n)
Types (0..n)
PlcType
<<Abstract>>
PlcStruct
TagTableGroup (1)
PlcTagTableSystemGroup
PlcTagTableUserGroup
PlcTag
Tags (0..n)
PlcTagTableGroup
<<Abstract>>
Groups (0..n)
TagTables(0..n)
PlcTagTable
PlcUserConstant
UserConstants(0..n)
PlcConstant
<<Abstract>>
SystemConstants(0..n)
PlcSystemConstant
TechnologicalObjectGroup (1)
TechnologicalInstanceDBGroup
TechnologicalObjects (0..1)
TechnologicalInstanceDBComposition
Blocks (0..n)
TechnologicalInstanceDB
(DataBlock)

访问列表中的对象

可使用以下方法访问列表中的对象：

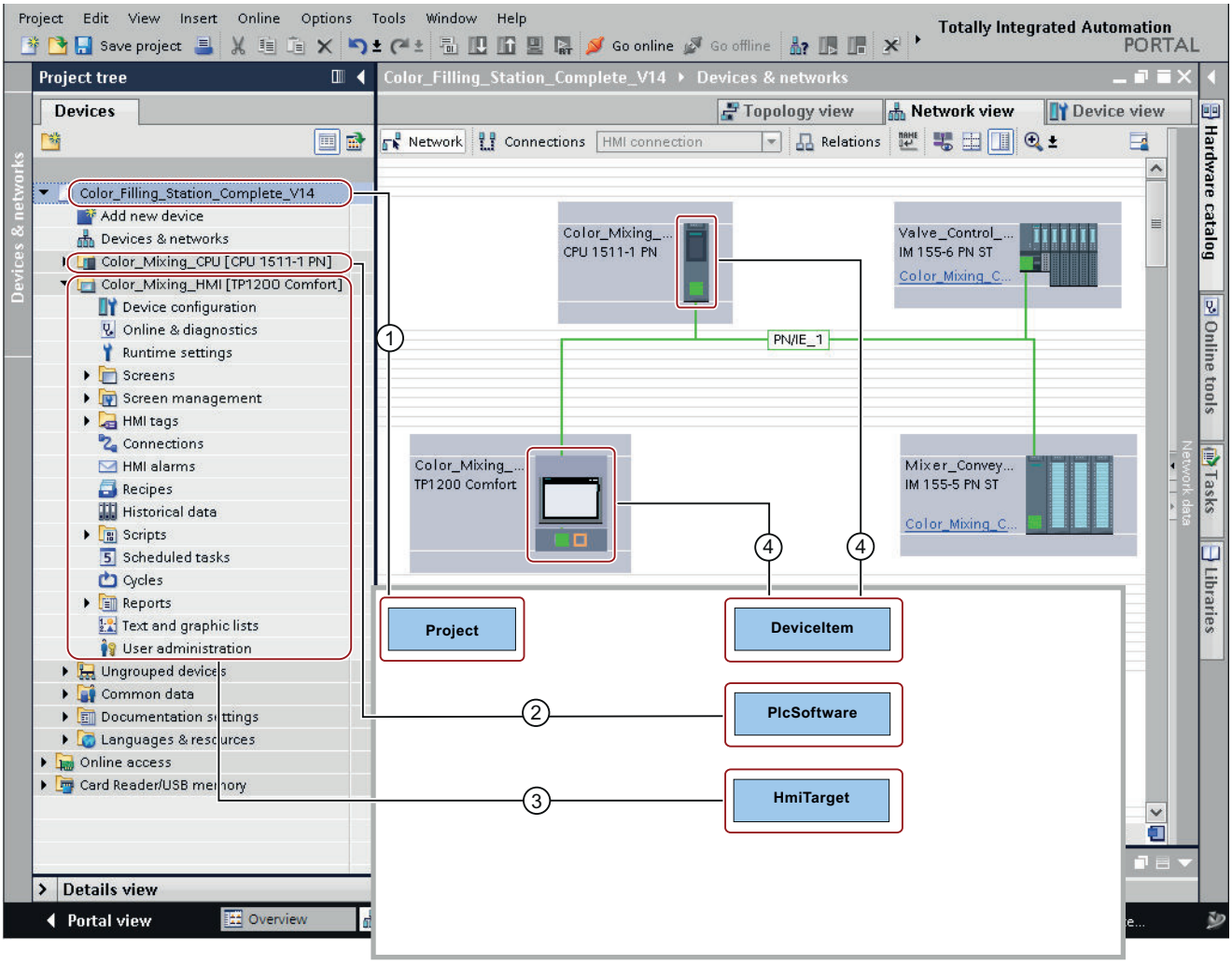
- 通过索引进行寻址。列表中从 0 开始计数。
- 使用 Find 类函数。
  该类函数通过对象的名称进行寻址。可以将该类函数用于构成或列表。Find 类函数未进行递归循环。
  示例：

```
ScreenComposition screens = folder.Screens;
Screen screen = screens.Find("myScreen");
```

- 使用符号名称。

TIA Portal 和 Openness 对象模型之间的关系

下图显示了对象模型和 TIA Portal 中的项目之间的关系：



参见

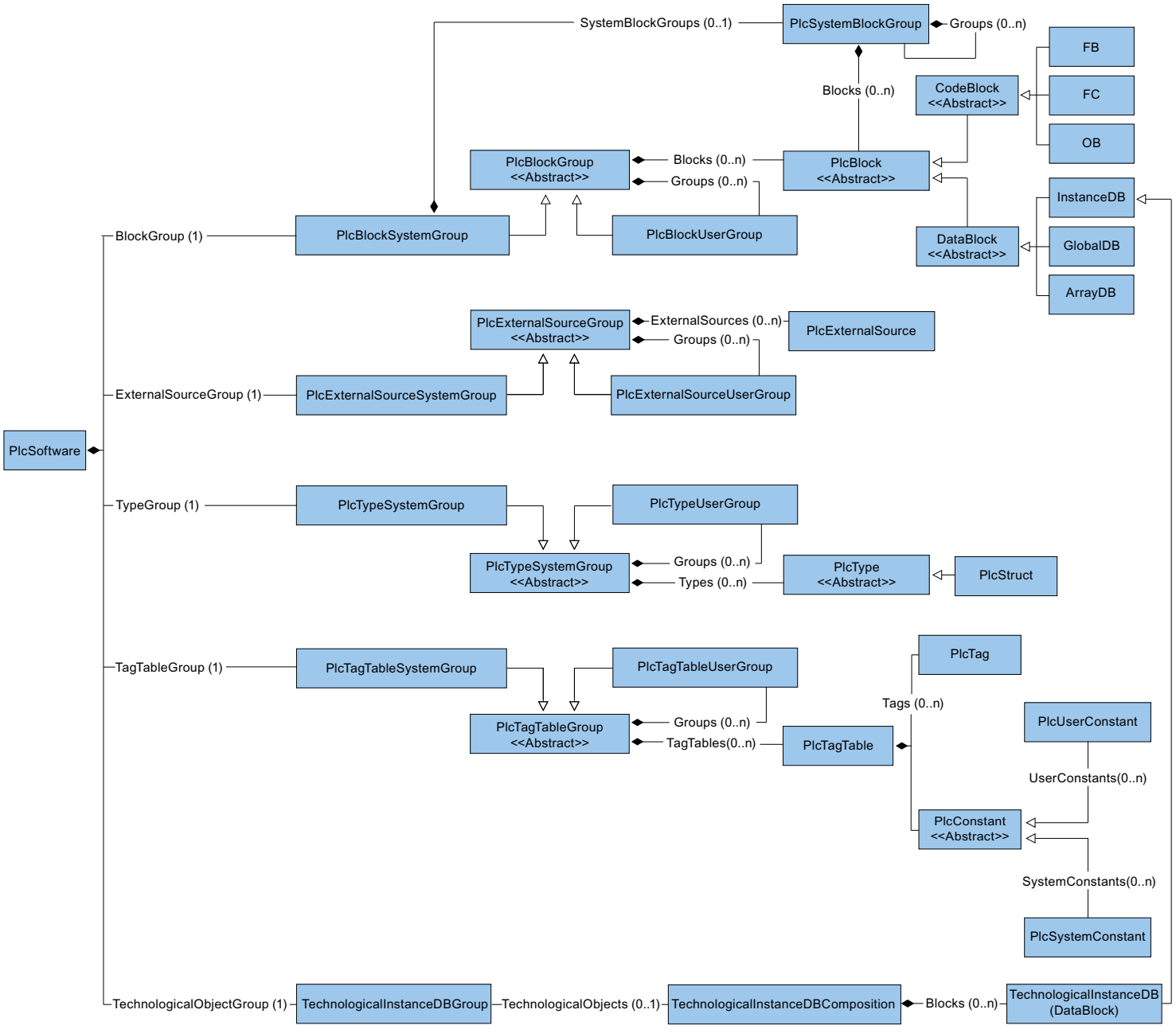
Openness 对象模型的块和类型 (页 50)

对象模型的硬件对象的层级 (页 58)

7.4 Openness 对象模型的块和类型

简介

下图所示为 PLC 的域模型，其中简要介绍了 Openness 中的当前模型。



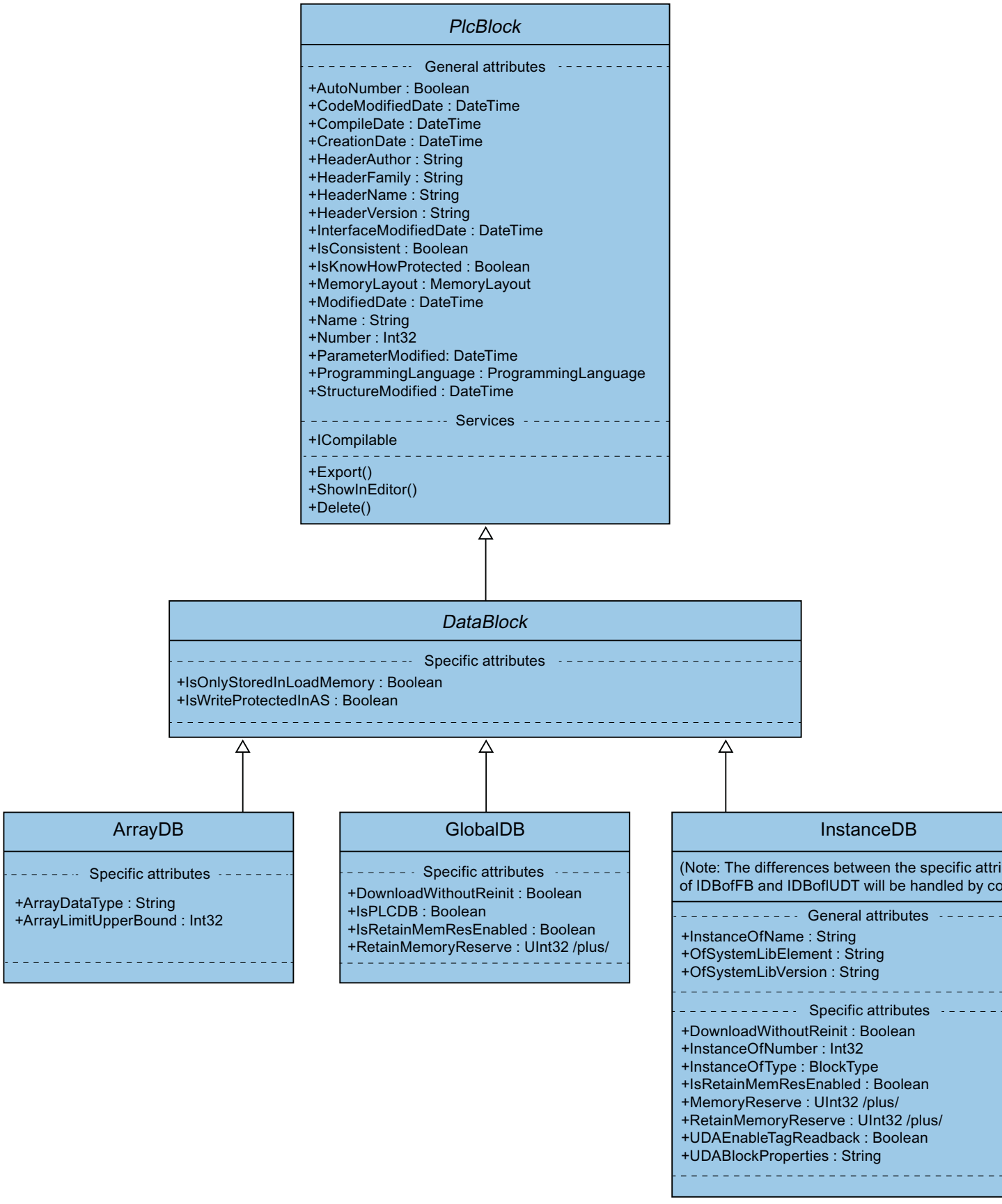
Openness API 中块和类型的表示

块和结构的简化模型部分基于 Openness API 的属性。这些类提供了导出功能，也为块提供了编译功能。

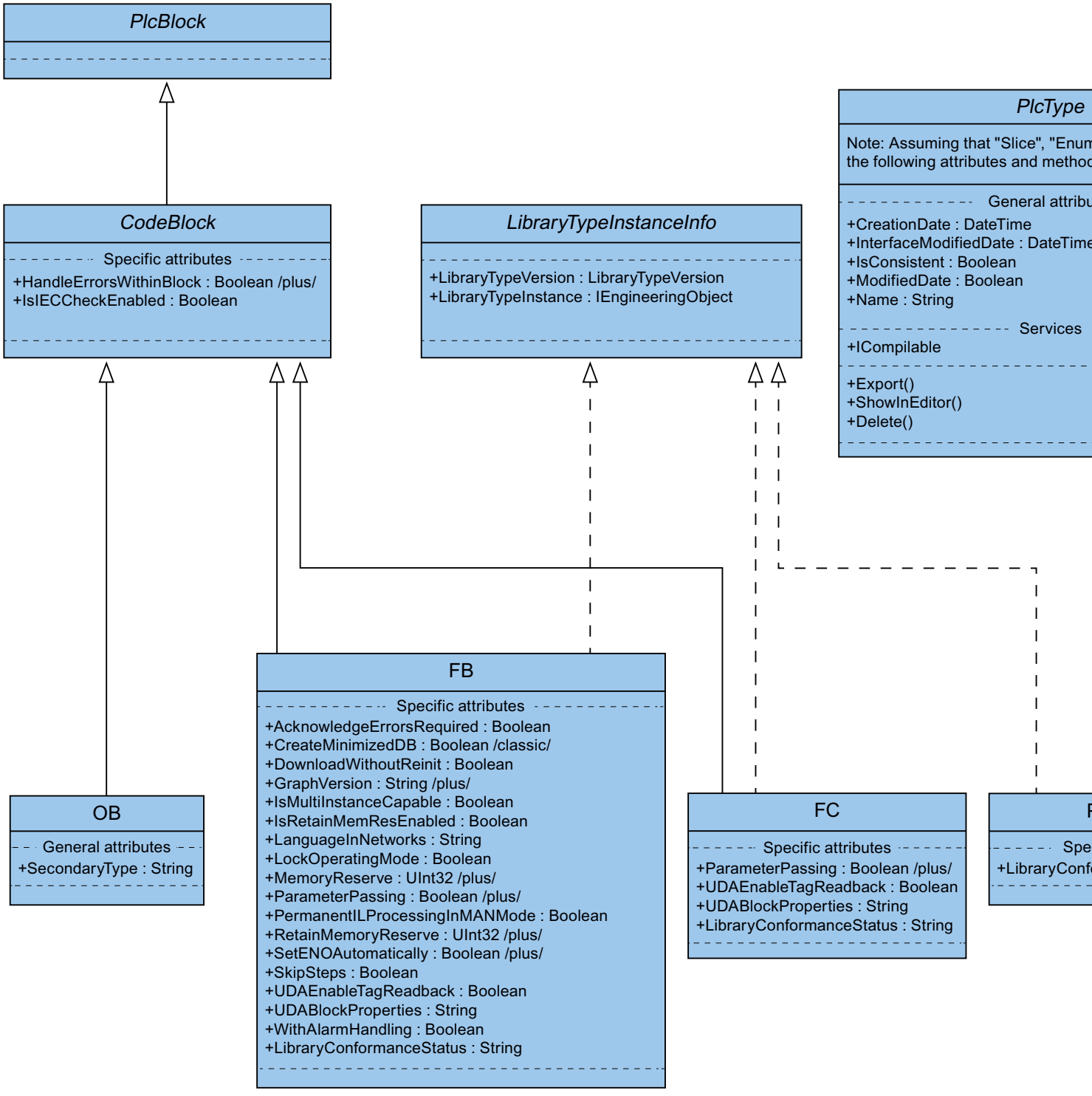
类图

在 Openness 对象模型中，所有类均被定义为抽象概念，未直接实例化。

数据


PlcBlock
General attributes
+AutoNumber : Boolean
+CodeModifiedDate : DateTime
+CompileDate : DateTime
+CreationDate : DateTime
+HeaderAuthor : String
+HeaderFamily : String
+HeaderName : String
+HeaderVersion : String
+InterfaceModifiedDate : DateTime
+IsConsistent : Boolean
+IsKnowHowProtected : Boolean
+MemoryLayout : MemoryLayout
+ModifiedDate : DateTime
+Name : String
+Number : Int32
+ParameterModified: DateTime
+ProgrammingLanguage : ProgrammingLanguage
+StructureModified : DateTime
Services
+ICompilable
+Export()
+ShowInEditor()
+Delete()
DataBlock
Specific attributes
+IsOnlyStoredInLoadMemory : Boolean
+IsWriteProtectedInAS : Boolean
ArrayDB
Specific attributes
+ArrayDataType : String
+ArrayLimitUpperBound : Int32
GlobalDB
Specific attributes
+DownloadWithoutReinit : Boolean
+IsPLCDB : Boolean
+IsRetainMemResEnabled : Boolean
+RetainMemoryReserve : UInt32 /plus/
InstanceDB
(Note: The differences between the specific attri
of IDBofFB and IDBoflUDT will be handled by co
General attributes
+InstanceOfName : String
+OfSystemLibElement : String
+OfSystemLibVersion : String
Specific attributes
+DownloadWithoutReinit : Boolean
+InstanceOfNumber : Int32
+InstanceOfType : BlockType
+IsRetainMemResEnabled : Boolean
+MemoryReserve : UInt32 /plus/
+RetainMemoryReserve : UInt32 /plus/
+UDAEnableTagReadback : Boolean
+UDABlockProperties : String

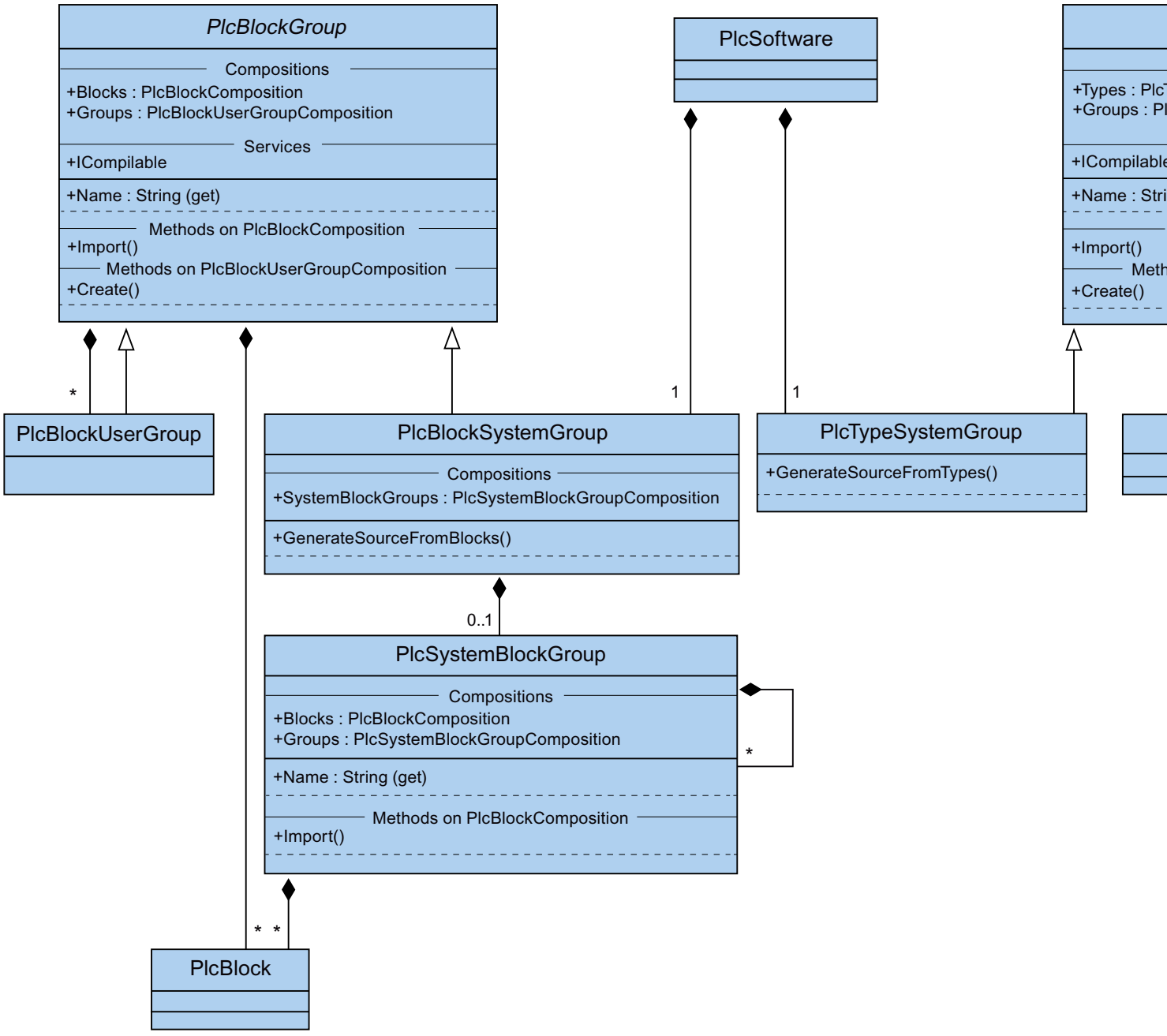
代码和类型


PlcBlock
CodeBlock
Specific attributes
+HandleErrorsWithinBlock : Boolean /plus/
+IsIECCheckEnabled : Boolean
LibraryTypeInstanceInfo
+LibraryTypeVersion : LibraryTypeVersion
+LibraryTypeInstance : IEngineeringObject
PlcType
Note: Assuming that "Slice", "Enum
the following attributes and metho
General attribu
+CreationDate : DateTime
+InterfaceModifiedDate : DateTime
+IsConsistent : Boolean
+ModifiedDate : Boolean
+Name : String
Services
+ICompilable
+Export()
+ShowInEditor()
+Delete()
OB
General attributes
+SecondaryType : String
FB
Specific attributes
+AcknowledgeErrorsRequired : Boolean
+CreateMinimizedDB : Boolean /classic/
+DownloadWithoutReinit : Boolean
+GraphVersion : String /plus/
+IsMultiInstanceCapable : Boolean
+IsRetainMemResEnabled : Boolean
+LanguageInNetworks : String
+LockOperatingMode : Boolean
+MemoryReserve : UInt32 /plus/
+ParameterPassing : Boolean /plus/
+PermanentILProcessingInMANMode : Boolean
+RetainMemoryReserve : UInt32 /plus/
+SetENOAutomatically : Boolean /plus/
+SkipSteps : Boolean
+UDAEnableTagReadback : Boolean
+UDABlockProperties : String
+WithAlarmHandling : Boolean
+LibraryConformanceStatus : String
FC
Specific attributes
+ParameterPassing : Boolean /plus/
+UDAEnableTagReadback : Boolean
+UDABlockProperties : String
+LibraryConformanceStatus : String

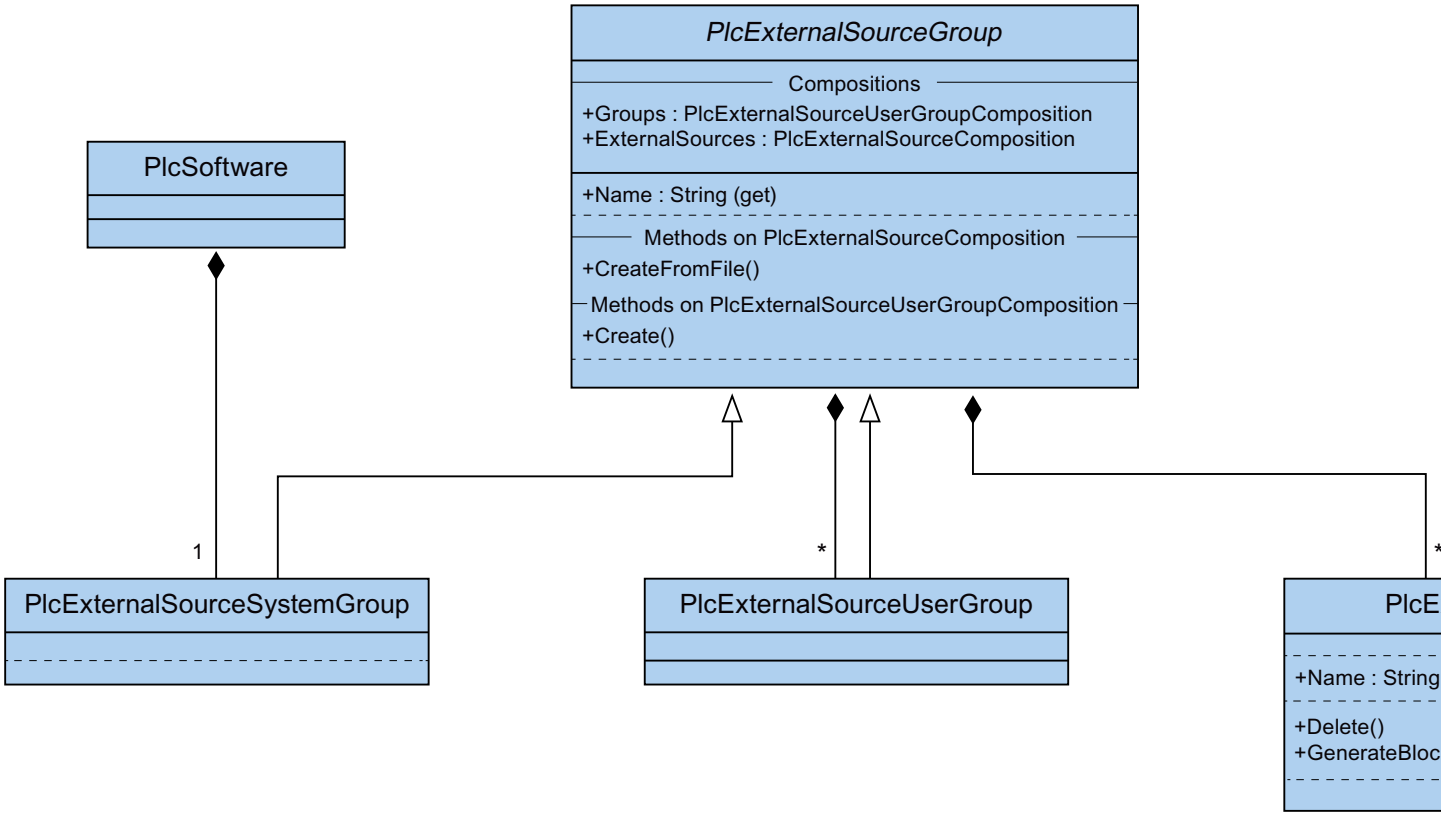
Openness API 中块和类型的组表示

两个高级组“PlcBlocks”（在 TIA Portal 的 GUI 中为“程序块”）和“PlcTypes”（在 TIA Portal 的 GUI 中为“Plc 数据类型”）包含块和类型定义。这些组为各个块提供了导入和编译功能。由于大多数的组功能方法都只能通过集合来实现，因此在“主机”类别下存在集合及其方法的“嵌入”或“紧凑”表示形式。

块和类型


PlcBlockGroup
Compositions
+Blocks : PlcBlockComposition
+Groups : PlcBlockUserGroupComposition
Services
+ICompilable
+Name : String (get)
Methods on PlcBlockComposition
+Import()
Methods on PlcBlockUserGroupComposition
+Create()
PlcSoftware
+Types : PlcT
+Groups : Pl
+ICompilable
+Name : Stri
+Import()
Meth
+Create()
*
1
1
PlcBlockUserGroup
PlcBlockSystemGroup
Compositions
+SystemBlockGroups : PlcSystemBlockGroupComposition
+GenerateSourceFromBlocks()
PlcTypeSystemGroup
+GenerateSourceFromTypes()
0..1
PlcSystemBlockGroup
Compositions
+Blocks : PlcBlockComposition
+Groups : PlcSystemBlockGroupComposition
+Name : String (get)
Methods on PlcBlockComposition
+Import()
*
* *
PlcBlock

外部源



参见

Openness 对象模型 (页 45)

对象模型的硬件对象的层级 (页 58)

7.5 对象模型的硬件对象的层级

TIA Portal 中可见元素与对象模型中建模元素之间的关系

硬件对象	说明
设备 (Device)	用于集中或分布式组态的容器对象。
设备项 (DeviceItem)	每个设备项对象都有一个容器对象。 逻辑关系为“项”。

容器关系相当于设备项对象的模块的关系。

示例：设备包含一个或多个插槽。插槽包含模块。模块包含子模块。

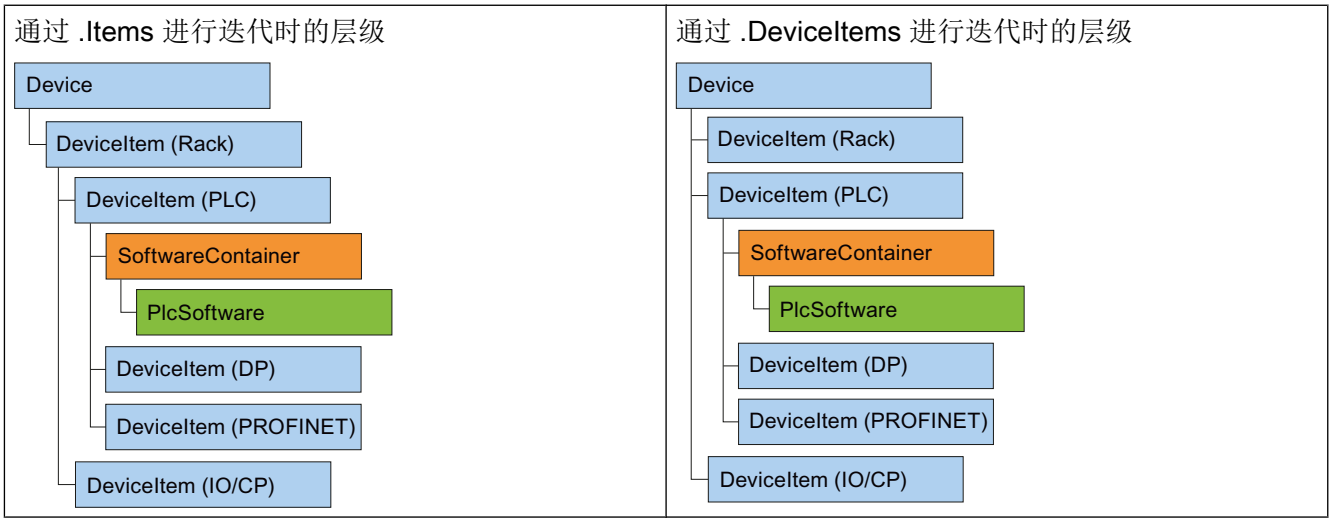
这与 TIA Portal 的网络视图和设备视图中的表示相似。设备项的“PositionNumber”属性在容器的项区域内唯一。

设备项对象间的父子关系是对象模型中的一种纯粹的逻辑关系。如果没有父对象，子对象将无法存在。

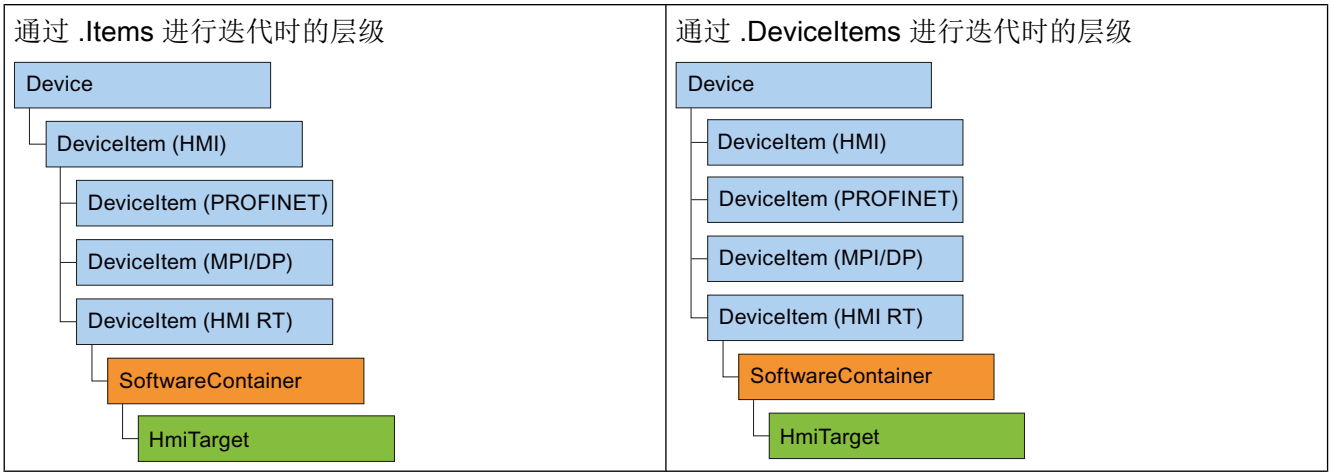
- 如果将子模块做为模块的一部分（子项）进行建模，则无法在没有模块的情况下删除子模块。
- 如果从模块中添加一个子模块，然后再将其移除，则该子项将与模块具有相同的父项。

下图显示了设备之间，以及 PLC 设备与 HMI 设备项之间的层级关系。

PLC 设备的层级关系



HMI 设备的层级关系



参见

Openness 对象模型 (页 45)

Openness 对象模型的块和类型 (页 50)

7.6 已安装 Openness 版本的相关信息

要求

- Openness 和 TIA Portal 已安装

应用

自 Openness V14 起，每个已安装的版本均配有包含版本相关信息的注册表项。因而可以为已安装的各个 Openness 版本自动生成 app.config 文件。

注册表项可位于以下路径下：

HKEY_LOCAL_MACHINE\Software\Siemens\Automation\Openness\14.0\PublicAPI

说明

该路径中的版本号始终表示当前已安装的 TIA Portal 版本号。如果存在多个并排安装的情况，则注册表中包含多组 Openness 的条目。

每个 Openness 版本均有一个注册表项。版本名称与所述程序集中的名称相同，例如，Openness 的注册表项如下：

```
[HKEY_LOCAL_MACHINE\SOFTWARE\Siemens\Automation\Openness\14.0\PublicAPI
\14.0.1.0]"PublicKeyToken"="d29ec89bac048f84"
"Siemens.Engineering"="C:\Program Files\Siemens\Automation\Portal V14\PublicAPI
\V14\Siemens.Engineering.dll"
"Siemens.Engineering.Hmi"="C:\Program Files\Siemens\Automation\Portal V14\PublicAPI
\V14\Siemens.Engineering.Hmi.dll"
"EngineeringVersion"="V14 SP1"
"AssemblyVersion"="14.0.1.0"
```

说明

如果要生成 app.config 文件 (页 69)，可从注册表项中获取 Siemens.Engineering.dll、Siemens.Engineering.Hmi.dll 和公钥标记的路径。

7.7 示例程序

应用示例：在应用程序中创建 API 访问

下面给出了应用示例的完整程序代码。接下来将基于此示例介绍典型的编程步骤。

说明

此应用示例需要使用应用程序组态文件 (页 69)。

```
using System;
using Siemens.Engineering;
using Siemens.Engineering.HW;
using Siemens.Engineering.HW.Features;
using Siemens.Engineering.SW;
using Siemens.Engineering.SW.Blocks;
using Siemens.Engineering.SW.ExternalSources;
using Siemens.Engineering.SW.Tags;
using Siemens.Engineering.SW.Types;
using Siemens.Engineering.Hmi;
using HmiTarget = Siemens.Engineering.Hmi.HmiTarget;
using Siemens.Engineering.Hmi.Tag;
using Siemens.Engineering.Hmi.Screen;
using Siemens.Engineering.Hmi.Cycle;
using Siemens.Engineering.Hmi.Communication;
using Siemens.Engineering.Hmi.Globalization;
using Siemens.Engineering.Hmi.TextGraphicList;
using Siemens.Engineering.Hmi.RuntimeScripting;
using System.Collections.Generic;
using Siemens.Engineering.Compiler;
using Siemens.Engineering.Library;
using System.IO;


namespace HelloTIA
{
    internal class Program
    {
        private static void Main(string[] args)
        {
            RunTiaPortal();
        }

        private static void RunTiaPortal()
        {
            Console.WriteLine("Starting TIA Portal");
            using (TiaPortal tiaPortal = new TiaPortal(TiaPortalMode.WithUserInterface))
            {
                Console.WriteLine("TIA Portal has started");
                ProjectComposition projects = tiaPortal.Projects;

                Console.WriteLine("Opening Project...");

                FileInfo projectPath = new FileInfo("C:\Demo\AnyCompanyProject.ap14"); //
edit the path according to your project
                Project project = null;
                try
                {
                    project = projects.Open(projectPath);
                }
```

```
                catch (Exception)
                {
                    Console.WriteLine(String.Format("Could not open project {0}",
projectPath.FullName));
                    Console.WriteLine("Demo complete hit enter to exit");
                    Console.ReadLine();
                    return;
                }

                Console.WriteLine(String.Format("Project {0} is open",
project.Path.FullName));

                IterateThroughDevices(project);

                project.Close();

                Console.WriteLine("Demo complete hit enter to exit");
                Console.ReadLine();
            }
        }

        private static void IterateThroughDevices(Project project)
        {
            if (project == null)
            {
                Console.WriteLine("Project cannot be null");
                return;
            }

            Console.WriteLine(String.Format("Iterate through {0} device(s)",
project.Devices.Count));

            foreach (Device device in project.Devices)
            {
                Console.WriteLine(String.Format("Device: \"{0}\".", device.Name));
            }

            Console.WriteLine();
        }
    }
}
```

程序步骤

1.使 TIA Portal 在开发环境中为已知

在开发环境中，创建对 "C:\Program Files\Siemens\Automation\PortalV..\PublicAPI\V.." 目录中所有“dll 文件”的引用。

下面以“Siemens.Engineering.dll”文件为例介绍此过程。

"Siemens.Engineering.dll" 文件位于目录 "C:\Program Files\Siemens\Automation\PortalV..\PublicAPI\V.." 中。在开发环境中，创建对“"Siemens.Engineering.dll"”文件的引用。

说明

确保为引用属性中的“CopyLocal”参数分配值“False”。

2.发布 TIA Portal 的名称空间

添加以下代码：

```
using Siemens.Engineering;
```

3.发布并启动 TIA Portal

要发布并启动 TIA Portal，请插入以下代码：

```
using (TiaPortal tiaPortal = new TiaPortal())
{
    // Add your code here
}
```

4.打开项目

例如，可使用以下代码打开项目：

```
ProjectComposition projects = tiaPortal.Projects;
Console.WriteLine("Opening Project...");
FileInfo projectPath = new FileInfo("C:\Demo\AnyCompanyProject.ap14");
Project project = null;
try
{
    project = projects.Open(projectPath);
}
catch (Exception)
{
    Console.WriteLine(String.Format("Could not open project {0}", projectPath.FullName));
    Console.WriteLine("Demo complete hit enter to exit");
    Console.ReadLine();
    return;
}
Console.WriteLine(String.Format("Project {0} is open", project.Path.FullName));
```

5.枚举项目的设备

插入如下代码以枚举项目的所有设备：

```
static private void IterateThroughDevices(Project project)
    {
        if (project == null)
        {
            Console.WriteLine("Project cannot be null");
            return;
        }

        Console.WriteLine();
        Console.WriteLine(String.Format("Iterate through {0} device(s)",
project.Devices.Count));
        foreach (Device device in project.Devices)
        {
            Console.WriteLine(String.Format("Device: \"{0}\".", device.Name));
        }
        Console.WriteLine();
    }
```

6.保存并关闭项目

插入如下代码以保存并关闭项目：

```
project.Save();
project.Close();
```

7.8 使用代码示例

代码片段的结构

本文档中的每个代码片段均实现为不带返回值的函数（对象引用作为传递参数）。为了增加程序的可读性，此处省略了对象的处理过程。可使用 Find 类函数按照 TIA Portal 对象的名称进行寻址。

```
//Deletes a single screen from a user folder or a system folder
private static void DeleteScreenFromFolder(HmiTarget hmiTarget)
{
    //The screen "MyScreen" will be deleted if it is existing in the folder
"myScreenFolder".
    //If "myScreen" is stored in a subfolder of "myScreenFolder" it will not be deleted.
    string screenName = "MyScreen";
    ScreenUserFolder folder = hmiTarget.ScreenFolder.Folders.Find("myScreenFolder");
    ScreenComposition screens = folder.Screens;
    Screen screen = screens.Find(screenName);
    if (screen != null)
    {
        screen.Delete();
    }
}
```

要执行此代码片段，需要以下各项：

- WinCC 项目以及所含组中至少存在一个画面的 HMI 设备。
- 实例化 HMI 设备的函数。

说明

指定目录路径时，请使用绝对目录路径，例如“C:/path/file.txt”。

仅通过 XML 文件进行导入和导出操作时支持相对目录路径，例如“file.txt”或“C:/path01/.../path02/file.txt”。

代码片段的执行示例

使用以下示例执行“Hello TIA”示例程序的“DeleteScreenFromFolder”代码片段部分：

```
//In the sample program "Hello TIA" replace the function call
//"IterateThroughDevices(project)" by the following functions calls:
    HmiTarget hmiTarget = GetTheFirstHmiTarget(project);
    DeleteScreenFromFolder(hmiTarget);

//Put the following function definitions before or after the
//function definition of "private static void IterateThroughDevices(Project project)":
private static HmiTarget GetTheFirstHmiTarget(Project project)
{
    if (project == null)
    {
        Console.WriteLine("Project cannot be null");
        throw new ArgumentNullException("project");
    }
    foreach (Device device in project.Devices)
    //This example looks for devices located directly in the project.
    //Devices which are stored in a subfolder of the project will not be affected by this
example.
    {
        foreach (DeviceItem deviceItem in device.DeviceItems)
        {
            DeviceItem deviceItemToGetService = deviceItem as DeviceItem;
            SoftwareContainer container =
deviceItemToGetService.GetService<SoftwareContainer>();
            if (container != null)
            {
                HmiTarget hmi = container.Software as HmiTarget;
                if (hmi != null)
                {
                    return hmi;
                }
            }
        }
    }
    return null;
}

//Deletes a single screen from a user folder or a system folder
private static void DeleteScreenFromFolder(HmiTarget hmiTarget)
{
    string screenName = "MyScreen";
    ScreenUserFolder folder = hmiTarget.ScreenFolder.Folders.Find("myScreenFolder");
    ScreenComposition screens = folder.Screens;
    Screen screen = screens.Find(screenName);
    if (screen != null)
    {
        screen.Delete();
    }
}
```

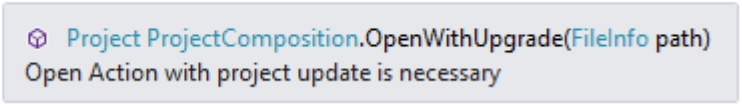
7.9 常规函数

7.9.1 Openness 智能感知支持

应用

Openness 的智能感知支持可通过工具提示信息帮助用户获取可用的属性或方法。其中可能包含所需参数的数目、名称和类型等相关信息。在下面的示例中，第一行以粗体显示的参数表示在您输入函数时所需的下一个参数。

```
project = tiaPortal.Projects.OpenWithUpgrade(projectInfo);
```



可以通过单击“编辑智能感知/参数信息”(Edit IntelliSense/Parameter Info)、输入 CTRL +SHIFT+SPACE 或单击编辑器工具栏上的“参数信息”(Parameter Info) 按钮手动调用“参数信息”(Parameter Info)。

7.9.2 连接到 TIA Portal

简介

使用 TIA Portal Openness 启动 TIA Portal 或连接到已在运行的 TIA Portal。使用 Openness 应用程序启动 TIA Portal 时，可指定在有或没有图形化用户界面的情况下启动 TIA Portal。在没有用户界面的情况下运行 TIA Portal 时，TIA Portal 仅作为一个进程由操作系统启动。如有必要，可使用 Openness 应用程序创建多个 TIA Portal 实例。

说明

如果使用 Openness 操作 TIA Portal 界面，则无法使用 HMI 编辑器。可以手动或使用 Public API 打开“设备和网络”编辑器或编程编辑器。

使用 Openness 应用程序启动 TIA Portal 时可选择以下方式：

- 使用应用程序组态文件（在大多数情况下都建议使用该文件）。
- 使用“AssemblyResolve”类函数（使用副本部署时建议采用）。
- 将 Siemens.Engineering.dll 复制到 Openness 应用目录中。

说明

建议通过 app.config 文件加载 Siemens.Engineering.dll。如果使用该类函数，则会涉及强名称，对 engineering.dll 的恶意修改会导致加载错误。如果使用 AssemblyResolve 类函数，则不会检测到上述情况。

使用应用程序组态文件启动 TIA Portal

请参考应用程序组态文件中所需的全部程序库。将应用程序组态文件与 Openness 应用程序一起分发。

将应用程序组态文件“app.config”与 Openness 应用程序存储在同一个目录下，同样，将其合并到应用程序中。检查各个代码中的文件路径是否与 TIA Portal 安装路径相匹配。

可以在应用程序组态文件中使用以下代码片段：

```
<?xml version="1.0"?>
<configuration>
    <runtime>
        <assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
            <dependentAssembly>
                <assemblyIdentity name="Siemens.Engineering" culture="neutral"
publicKeyToken="d29ec89bac048f84"/>
                <!-- Edit the following path according to your installation -->
              <codeBase version="14.0.1.0" href="FILE://C:\Program Files\Siemens\Automation
\Portal V14\PublicAPI\V14 SP1\Siemens.Engineering.dll"/>
            </dependentAssembly>
            <dependentAssembly>
                <assemblyIdentity name="Siemens.Engineering.Hmi" culture="neutral"
publicKeyToken="d29ec89bac048f84"/>
                <!-- Edit the following path according to your installation -->
              <codeBase version="14.0.1.0" href="FILE://C:\Program Files\Siemens\Automation
\Portal V14\PublicAPI\V14 SP1\Siemens.Engineering.Hmi.dll"/>
            </dependentAssembly>
        </assemblyBinding>
    </runtime>
</configuration>
```

使用以下程序代码以通过应用程序组态文件打开新的 TIA Portal 实例：

```
//Connect an openness application via API using
using System;
using System.IO;
using Siemens.Engineering;

namespace UserProgram
{
    internal class MyProgram
    {
        public static void Main(string[] args)
        {
            // To start TIA Portal with user interface:
            // using (TiaPortal tiaPortal = new TiaPortal(TiaPortalMode.WithUserInterface))
            //
            // To start TIA Portal without user interface:
            // using (TiaPortal tiaPortal = new
TiaPortal(TiaPortalMode.WithoutUserInterface))
            using (TiaPortal tiaPortal = new TiaPortal(TiaPortalMode.WithUserInterface))
            {
                //begin of code for further implementation
                //...
                //end of code
            }
        }
    }
}
```

使用“AssemblyResolve”类函数启动 TIA Portal

设计 Openness 应用程序的程序代码，以实现在出现事件 "AssemblyResolve" 时尽早注册。将对 TIA Portal 的访问封装到附加对象或类函数中。

使用一个封装的解析程序类函数对封装的工程组态系统会进行解析时，必须小心谨慎。如果在程序集解析程序运行前使用了工程程序集的任何类型，程序会崩溃。这是因为，实时编译器（JIT 编译器）只有在需要执行类函数时才会对其进行编译。例如，如果在 Main 中使用了工程程序集类型，JIT 编译器会在程序运行时试图编译 Main，但不会编译成功，因为它不清楚如何找到工程程序集。Main 中程序集解析程序的注册不会对此有所改变。该类函数需要先进行编译，然后再运行，并且需要在注册程序集解析程序之前运行。要解决这一问题，可将使用封装工程组态系统中类型的业务逻辑放置到一个单独的类函数中，且该类函数仅使用 JIT 编译器支持的类型。在本示例中，类函数将返回空值且不带参数，并包含所有使用的业务逻辑。JIT 编译器编译 Main 时，会成功完成编译，因为它了解 Main 中的所有类型。运行时，如果我们调用 RunTiaPortal，程序集解析程序将完成注册，因此在 JIT 编译器试图查找业务逻辑类型时，它清楚如何找到工程程序集。

使用以下程序代码打开新的 TIA Portal 实例。

```
using System;
using System.IO;
using System.Reflection;
using Siemens.Engineering;

namespace UserProgram
{
    static class MyProgram
    {
        public static void Main(string[] args)
        {
            AppDomain.CurrentDomain.AssemblyResolve += MyResolver;
            RunTiaPortal();
        }
        private static void RunTiaPortal()
        {
            // To start TIA Portal with user interface:
            // using (TiaPortal tiaPortal = new TiaPortal(TiaPortalMode.WithUserInterface))
            //
            // To start TIA Portal without user interface:
            // using (TiaPortal tiaPortal = new
TiaPortal(TiaPortalMode.WithoutUserInterface))
            using (TiaPortal tiaPortal = new TiaPortal(TiaPortalMode.WithUserInterface))
            {
                //begin of code for further implementation
                //...
                //end of code
            }
        }
        private static Assembly MyResolver(object sender, ResolveEventArgs args)
        {
            int index = args.Name.IndexOf(',');
            if (index == -1)
            {
                return null;
            }
            string name = args.Name.Substring(0, index) + ".dll";
            // Edit the following path according to your installation
            string path = Path.Combine(@"C:\Program Files\Siemens\Automation\Portal
V14\PublicAPI\V14 SP1\", name);
            string fullPath = Path.GetFullPath(path);
            if (File.Exists(fullPath))
            {
                return Assembly.LoadFrom(fullPath);
            }
            return null;
        }
    }
}
```

访问 TIA Portal 的运行实例

为使用 Openness 应用程序连接到 TIA Portal 的运行实例，需要首先枚举 TIA Portal 的实例。可以连接到 Windows 会话中的多个实例。运行实例可以为具有或不具有已启动用户界面的 TIA Portal：

```
foreach (TiaPortalProcess tiaPortalProcess in TiaPortal.GetProcesses())
    {
        //...
    }
```

如果已知 TIA Portal 的实例的进程 ID，则可使用此进程 ID 访问该对象。TIA Portal 需要一定时间完成启动，才能将 Openness 应用程序连接到 TIA Portal。

连接到 TIA Portal 的运行实例时，将出现 Openness 防火墙的连接提示。连接提示提供下列选项：

- 允许连接一次
- 不允许连接
- 始终允许来自此应用程序的连接
  更多信息，请参见“Openness 防火墙 (页 75)”。

说明

如果注册表提示被拒绝三次，则系统会触发 `EngineeringSecurityException` 类型的例外。

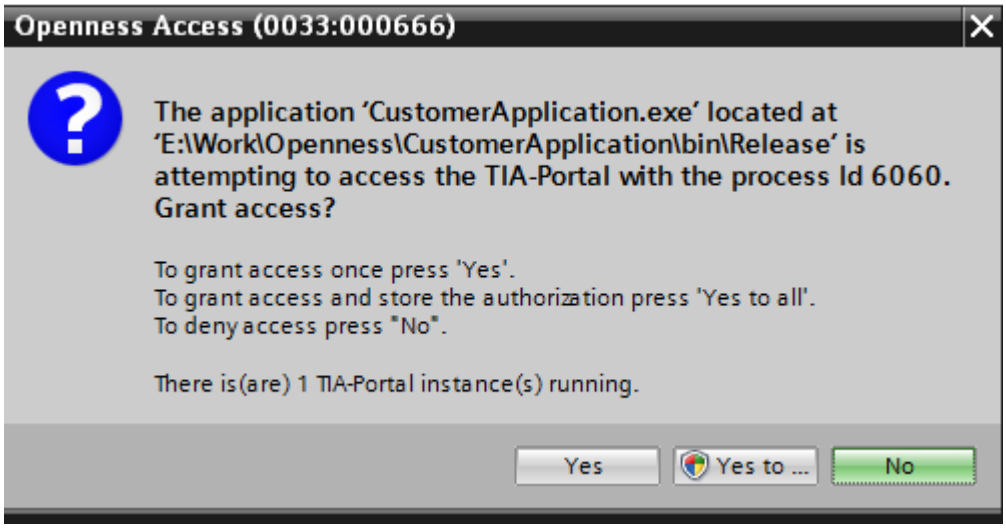
连接到进程后，可使用以下属性获取有关 TIA Portal 实例的信息：

属性	信息
`InstalledSoftware as IList<TiaPortalProduct>`	返回已安装产品的相关信息。
`Mode as TiaPortalMode`	返回 TIA Portal 的启动模式 (WithoutUserInterface/WithUserInterface)。
`AttachedSessions as IList<TiaPortalSession>`	返回连接到 TIA Portal 的应用程序的列表。
`ProjectPath as FileInfo`	返回在 TIA Portal 中打开的项目的文件名（包括文件夹），例如，`"D:\WinCCProjects\ColorMixing\ColorMixing.ap14"` 如果未打开任何项目，则返回空字符串。
`ID as int`	返回 TIA Portal 实例的过程 ID。
`Path as FileInfo`	返回 TIA Portal 可执行项目的路径。

7.9.3 Openness 防火墙

Openness 防火墙提示

尝试通过 Openness 连接到运行的 TIA Portal 时，TIA Portal 会提示您接受或拒绝连接，如下面的截图中所示。



允许连接一次 TIA Portal

如果您想要使 Openness 应用程序仅连接一次 TIA Portal，请在出现提示后单击“是”(Yes)。Openness 应用程序下次尝试连接 TIA Portal 时，会再次显示相应的提示。

通过连接 TIA Portal 添加白名单条目

要为 Openness 应用程序创建白名单条目，请按照以下步骤进行操作：

1. 出现提示后，单击“全部接受”(Yes to all)，以显示“用户帐户控制”(User Account Control) 对话框。
2. 在“用户帐户控制”(User Account Control) 对话框中单击“是”(Yes)，将应用程序添加到 Windows 注册表白名单中，并会与 TIA Portal 相连。

不使用 TIA Portal 添加白名单条目

如果要在不使用 TIA portal 的情况下将条目添加到白名单中，可创建一个 Reg 文件，如下所示：

```
Windows Registry Editor Version 5.00
[HKEY_LOCAL_MACHINE\SOFTWARE\Siemens\Automation\Openness\14.0\Whitelist
\CustomerApplication.exe]
[HKEY_LOCAL_MACHINE\SOFTWARE\Siemens\Automation\Openness\14.0\Whitelist
\CustomerApplication.exe\Entry]
"Path"="E:\\Work\\Openness\\CustomerApplication\\bin\\Release\\CustomerApplication.exe"
"DateModified"="2014/06/10 15:09:44.406"
"FileHash"="0rXRKUCNzMWHOMFrT52OwXzqJef10ran4UykTeBraaY="
```

以下示例显示了如何计算文件哈希以及上次修改的日期：

```
string applicationPath = @"E:\\Work\\Openness\\CustomerApplication\\bin\\Release\
\CustomerApplication.exe";
string lastWriteTimeUtcFormatted = String.Empty;
DateTime lastWriteTimeUtc;
HashAlgorithm hashAlgorithm = SHA256.Create();
FileStream stream = File.OpenRead(applicationPath);
byte[] hash = hashAlgorithm.ComputeHash(stream);
// this is how the hash should appear in the .reg file
string convertedHash = Convert.ToBase64String(hash);
lastWriteTimeUtc = fileInfo.LastWriteTimeUtc;
// this is how the last write time should be formatted
lastWriteTimeUtcFormatted = lastWriteTimeUtc.ToString(@"yyyy/MM/dd HH:mm:ss.fff");
```

7.9.4 事件处理程序

Openness 应用程序中的事件处理程序

TIA Portal 实例提供了以下事件，可在 Openness 应用程序中通过事件处理程序对这些事件做出响应。可以访问通知的属性并相应地定义这些响应。

事件	响应
`Disposed`	可通过此事件，使用 Openness 应用程序对 TIA Portal 的关闭做出响应。
`Notification`	可通过此事件，使用 Openness 应用程序对 TIA Portal 的通知做出响应。对通知进行确认即可，如“确定”(OK)。
`Confirmation`	可通过此事件，使用 Openness 应用程序对 TIA Portal 的确认做出响应。确认时通常需要您做出决定，如“是否要保存项目？”(Do you want to save the project?)。

程序代码

在 Openness 应用程序中，修改以下程序代码以注册事件处理程序：

```
//Register event handler for Disposed-Event
    ....
        tiaPortal.Disposed +=TiaPortal_Disposed;
....

private static void TiaPortal_Disposed(object sender, EventArgs e)
{
    ....
}

//Register event handler for Notification-Event
    ....
        tiaPortal.Notification += TiaPortal_Notification;
....

private static void TiaPortal_Notification(object sender, NotificationEventArgs e)

{
    ....
}

//Register event handler for Confirmation-Event
    ....
        tiaPortal.Confirmation += TiaPortal_Confirmation;
....

private static void TiaPortal_Confirmation(object sender, ConfirmationEventArgs e)
{
    ....
}
```

TIA Portal 通知的属性

TIA Portal 通知具有以下属性：

属性	说明
Caption	返回确认的姓名。
DetailText	返回确认的详细文本。
Icon	返回确认的图标。
IsHandled	返回确认或指定其是否仍处于未决状态。

属性	说明
MessageID	返回服务中的唯一 ID。
ServiceID	返回触发了确认的服务 ID。
Text	返回确认的文本。

确认属性

确认具有下列属性：

属性	说明
Caption	返回确认的姓名。
Choices	返回用于接受确认的选项。
DetailText	返回确认的详细文本。
Icon	返回确认的图标。
IsHandled	返回确认或指定其是否仍处于未决状态。
MessageID	返回服务中的唯一 ID
Result	返回确认的结果或指定结果。
ServiceID	返回触发了确认的服务 ID。
Text	返回确认的文本。

参见

由程序控制系统事件对话框的确认 (页 79)

7.9.5 由程序控制系统事件对话框的确认

要求

- Openness 应用程序连接至 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)
- 事件处理程序已注册。
  请参见连接到 TIA Portal (页 69)

应用

运行带用户界面的 TIA Portal 时，对某些程序操作会显示系统事件对话框。可基于这些系统事件决定继续执行的方式。

通过 Openness 应用程序访问 TIA Portal 时，这些系统事件必须通过相应的“.NET”事件加以确认。

Choices 列表中包含所允许的确认方式：

- Abort
- Cancel
- Ignore
- No
- NoToAll
- None
- OK
- Retry
- Yes
- YesToAll

ConfirmationEventArgs.Result 的值必须为上述条目之一。否则，会触发异常。

程序代码

修改以下程序代码以便对确认事件做出响应：

```
private void TiaPortalOnConfirmation(object sender, ConfirmationEventArgs e)
{
    int serviceId = e.ServiceId;
    int messageId = e.MessageId;
    if (serviceId == MyExpectedServiceId)
    {
        if (messageId == MyExpectedMessageId && ((e.Choices & ConfirmationChoices.Yes) ==
ConfirmationChoices.Yes))
        {
            e.Result = ConfirmationResult.Yes;
            e.IsHandled = true;
        }
    }
}
```

修改以下程序代码以通知项目工程师所执行的 Openness 应用程序操作：

```
//Handles notifications
using (TiaPortal tiaPortal = new TiaPortal())
{
    tiaPortal.Notification += Notification;
    try
        {
            //perform actions that will result in a notification event
        }
    finally
        {
            tiaPortal.Notification -= Notification;
        }
}
```

```
private void TiaPortalOnNotification(object sender, NotificationEventArgs e)
{
    if (e.MessageId == MyExpectedMessageId && e.ServiceId == MyExpectedServiceId)
    {
        e.IsHandled = true;
        }
    }
```

7.9.6 终止与 TIA Portal 的连接

简介

如果启动了不带用户界面的 TIA Portal 实例且如果您的应用程序是唯一与 TIA Portal 相连的 Openness 客户端，可使用 Openness 应用程序关闭该 TIA Portal 实例。否则，需要断开 Openness 应用程序与 TIA Portal 实例的连接。

请使用 `IDisposable.Dispose()` 方法断开或关闭激活的 TIA Portal 实例。

可通过如下方式使用 `IDisposable.Dispose()` 方法：

- 通过正在使用的语句。
- 采用 try-finally 块环绕对象描述并调用该 finally 块中的 `IDispose.Dispose()` 方法。

关闭激活的 TIA Portal 实例后，将不能再访问 TIA Portal。

说明

如果组态工程师在 Openness 应用程序访问 TIA Portal 实例过程中关闭该实例，则在下次进行 API 访问时会在 Openness 应用程序中触发“NonRecoverableException”类的例外。可订阅处理事件以便在关闭 TIA Portal 时收到相关调用。

程序代码

修改以下程序代码以断开或关闭至 TIA Portal 的连接：

```
// Add code to dispose the application if the application is still instantiated
if (tiaPortal != null)
{
    tiaPortal.Dispose();
}
```

参见

事件处理程序 (页 77)

7.9.7 TIA Portal 的诊断接口

应用

可以采用静态方法从 TIA Portal 的运行实例中检索某些诊断信息。诊断接口位于 TiaPortalProcess 对象上，可针对当前正在运行的 TIA Portal 实例进行检索。

诊断接口未受阻，因而无论 TIA Portal 是否处于忙碌状态，均可检索 TiaPortalProcess 对象以及访问其成员。诊断接口包含以下成员：

类 TiaPortalProcess

成员	类型	函数
AcquisitionTime	DateTime	采集 TiaPortalProcess 对象的时间。由于 TiaPortalProcess 对象表示给定时间点处 TIA Portal 状态的完全静态快照，因而其包含的信息可能会过期。
Attach	TiaPortal	连接到给定的 TiaPortalProcess，它将返回一个 TiaPortal 实例。
AttachedSessions	IList<TiaPortalSession>	当前连接到同一 TIA Portal 的所有其它会话的集合。该集合可为空。每个会话均由一个 TiaPortalSession 对象进行表示。
Attaching	EventHandler<AttachingEventArgs>	该事件可使应用程序允许尝试连接 TIA Portal。如果有其它应用程序尝试连接 TIA Portal，该事件的订户会收到通知，并有 10 秒钟的时间批准连接。如果有任一订户忽略了该事件或未能及时做出响应，则会被视为拒绝其它应用程序进行连接。崩溃的应用程序未能对该事件做出响应，且无法拒绝应用程序进行连接。
Dispose	void	关闭相关联的 TIA Portal 实例。
Id	int	TIA Portal 的过程 ID。
InstalledSoftware	IList<TiaPortalProduct>	当前作为部分 TIA Portal 进行安装的所有产品的集合。每个产品均由一个 TiaPortalProduct 对象进行表示，如下所述。

成员	类型	函数
Mode	TiaPortalMode	TIA Portal 的启动模式。当前值为 WithUserInterface 和 WithoutUserInterface。
Path	FileInfo	TIA Portal 可执行项目的路径。
ProjectPath	FileInfo	当前在 TIA Portal 中处于打开状态的项目的路径。如果项目均未打开，该属性将为空。

类 TiaPortalSession

成员	类型	函数
AccessLevel	TiaPortalAccessLevel	会话的访问等级。该等级以标志枚举的形式表示，其中允许存在多个访问等级。下面详细介绍了 TiaPortalAccessLevel。
AttachTime	DateTime	建立与 TIA Portal 的连接的时间。
Id	int	当前会话的 ID。
IsActive	bool	如果 TIA Portal 当前正在处理来自运行中的会话的调用，则返回“true”。
Dispose	void	断开与 TIA Portal 的过程连接。该方法不会像 System.Diagnostics.Process.Kill 那样终止过程。连接已被终止的应用程序仍会获取处置事件，但不会再指示终止连接的原因。
ProcessId	int	连接过程的过程 ID。
ProcessPath	FileInfo	连接过程的可执行项目路径。

成员	类型	函数
TrustAuthority	TiaPortalTrustAuthority	指示当前会话是否已由签名的过程启动，并指示其是否为 Openness 证书。TrustAuthority 为标志枚举，下文对其进行了介绍。
UtilizationTime	TimeSpan	过程使用 TIA Portal 所消耗的时间。与 AttachTime 属性相结合，用于确定使用百分比或类似的数据。
Version	string	与会话相连的 Siemens.Engineering.dll 的版本。

枚举 TiaPortalAccessLevel

枚举值	函数
None	此为无效值。将该值包含在内的原因是，TiaPortalAccessLevel 为标志枚举，需要使用相应的“零值”来表示未设置任何标志，但在实际使用中绝不会显示该值，因为在无访问权限的情况下无法启动会话。
Published	会话有权访问发布的功能。
Modify	会话具备修改权限。

枚举 TiaPortalTrustAuthority

枚举值	函数
None	未使用证书对所连接过程的主模块进行签名。
Signed	已使用证书（非 Openness 证书）对主模块进行签名。
Certified	已使用 Openness 证书对主模块进行签名。
CertifiedWithExpiration	已使用 Openness 证书对主模块进行签名，该证书将在到期后变为无效状态。

类 TiaPortalProduct

成员	类型	函数
Name	string	产品名称（例如，STEP 7 Professional）。
Options	IList<TiaPortalProduct>	属于相连 TIA Portal 的所有可选数据包集合，以 TiaPortalProduct 对象的形式进行表示。如果某个选件包自身具备选件包，则会保留这种嵌套关系。
Version	string	产品版本字符串。

以下代码片段即为有关如何使用诊断接口查询信息然后在应用程序中使用该信息的示例。

```
public void TiaPortalDiagnostics()
{
    IList<TiaPortalProcess> tiaPortalProcesses = TiaPortal.GetProcesses();
    foreach (TiaPortalProcess tiaPortalProcess in tiaPortalProcesses)
    {
        Console.WriteLine("Process ID: {0}", tiaPortalProcess.Id);
        Console.WriteLine("Path: {0}", tiaPortalProcess.Path);
        Console.WriteLine("Project: {0}", tiaPortalProcess.ProjectPath);
        Console.WriteLine("Timestamp: {0}", tiaPortalProcess.AcquisitionTime);
        Console.WriteLine("UI Mode: {0}", tiaPortalProcess.Mode);
        //See method body below.
        Console.WriteLine("Installed Software:");
        EnumerateInstalledProducts(tiaPortalProcess.InstalledSoftware);
        Console.WriteLine("Attached Openness Applications:");
        foreach (TiaPortalSession session in tiaPortalProcess.AttachedSessions)
        {
            Console.WriteLine("Process: {0}", session.ProcessPath);
            Console.WriteLine("Process ID: {0}", session.ProcessId);
            DateTime attachTime = session.AttachTime;
            TimeSpan timeSpentAttached = DateTime.Now - attachTime;
            TimeSpan utilizationTime = session.UtilizationTime;
            long percentageTimeUsed = (utilizationTime.Ticks / timeSpentAttached.Ticks) *
100;
            Console.WriteLine("AttachTime: {0}", attachTime);
            Console.WriteLine("Utilization Time: {0}", utilizationTime);
            Console.WriteLine("Time spent attached: {0}", timeSpentAttached);
            Console.WriteLine("Percentage of attached time spent using TIA Portal: {0}",
percentageTimeUsed);
            Console.WriteLine("AccessLevel: {0}", session.AccessLevel);
            Console.WriteLine("TrustAuthority: {0}", session.TrustAuthority);
            if ((session.TrustAuthority & TiaPortalTrustAuthority.Certified) !=
TiaPortalTrustAuthority.Certified)
            {
                Console.WriteLine("TrustAuthority doesn't match required level, attempting
to terminate connection to TIA Portal."); session.Dispose();
            }
        }
    }
}
public void EnumerateInstalledProducts(IEnumerable<TiaPortalProduct> products)
{
    foreach (TiaPortalProduct product in products)
    {
        Console.WriteLine("Name: {0}", product.Name);
        Console.WriteLine("Version: {0}", product.Version);
        //recursively enumerate all option packages
        Console.WriteLine("Option Packages \n:");
        EnumerateInstalledProducts(product.Options);
    }
```

```
}
```

安全相关的信息

由于无需连接 TIA Portal 即可使用诊断接口，因此可以写入 Windows 服务，该服务使用连接事件检查试图连接 TIA Portal 的应用程序（例如，仅限以允许连接的公司名开头的应用程序）。还可以始终授权访问，但应向日志中写入连接过程的相关信息。以下程序代码即为用于检查传入连接的事件处理程序示例：

```
public void OnAttaching(object sender, AttachingEventArgs e)
{
    string name = Path.GetFileNameWithoutExtension(e.ProcessPath);
    TiaPortalAccessLevel requestedAccessLevel = e.AccessLevel &
TiaPortalAccessLevel.Published;
    TiaPortalTrustAuthority certificateStatus = e.TrustAuthority
&TiaPortalTrustAuthority.Certified;
    if (requestedAccessLevel == TiaPortalAccessLevel.Published &&
       certificateStatus == TiaPortalTrustAuthority.Certified &&
       name.StartsWith("SampleCustomerName"))
    {
        e.GrantAccess();
    }
}
```

7.9.8 Exclusive access

要求

- Openness 应用程序连接到 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 项目已经打开。
  请参见打开项目 (页 96)

应用

“TIA Portal”类会提供方法“ExclusiveAccess(String text)”来建立附加 TIA Portal 过程的独占访问。即使不强制使用独占访问，也强烈建议使用。

在“using”语句中使用“ExclusiveAccess”可确保正确对其进行处理，即使发生异常或应用程序关闭，亦能如此。

说明

如果在开放式独占访问范围内试图创建另一个独占访问，则会产生可恢复的异常。

修改以下示例即可对实例进行“ExclusiveAccess”：

```
...
[assembly: AssemblyTitle("MyApplication")]
// This will be used for the exclusive access dialog when present....
TiaPortal tiaPortal = ...;
using (ExclusiveAccess exclusiveAccess = tiaPortal.ExclusiveAccess("My Activity"))
{
    ...
}
```

针对给定 TIA Portal 过程获得“ExclusiveAccess”实例后，会显示一个对话框。该对话框会显示实例化期间提供的消息。此外，还会显示客户端应用程序的以下信息：

- 清单数据的程序集标题（如果可用）；否则会显示过程名称
- 过程 ID
- SID

说明

对于某个已知的 Openness 客户端应用程序，可能存在多个活动的会话，这是因为可能存在多个与同一个 TIA Portal 过程相关联的 TiaPortal 实例。

客户端应用程序还会通过使用新值设置“文本”(Text) 属性的方式更新独占访问对话框的所显示内容。修改以下程序代码以调用该行为：

```
exclusiveAccess = ...;
...
exclusiveAccess.Text = "My Activity Phase 1";
...
exclusiveAccess.Text = "My Activity Phase 2";
...
exclusiveAccess.Text = String.Empty; // or null;
...
```

可以请求通过选择“取消”(Cancel) 按钮来取消独占访问。修改以下程序代码以调用该行为：

```
exclusiveAccess = ...;
...
if (exclusiveAccess.IsCancellationRequested)
{
    // stop your activity
    ...
}
else
{
    // continue your activity
    ...
}
...
```

7.9.9 事务处理

要求

- Openness 应用程序连接到 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 项目已经打开。
  请参见打开项目 (页 96)

操作

对于相关 TIA Portal 过程中打开的保留设置（项目和库等），可通过 Openness 客户端应用程序进行修改。可以通过单次操作或通过系列操作实现此修改。根据具体活动合理地将这些操作组成单一撤消单元，以用于更多的逻辑工作流。此外，将操作组成单一撤消单元还展现出了性能优势。为此，“ExclusiveAccess”类提供了方法“Transaction(ITransactionSupport persistence, string undoDescription)”。调用该方法时会对“事务”类型的新可处置对象进行实例化。用户必须提供事务内容的相关说明（文本属性不能为 null 或空）。如果尚未处理该实例，在相关的 TIA Portal 过程中所有客户端应用程序操作均会组成单一撤消单元。

修改以下程序代码获取“Transaction”实例：

```
ExclusiveAccess exclusiveAccess = ...;
Project project = ...;
using (Transaction transaction = exclusiveAccess.Transaction(project, "My Operation"))
{
    ...
}
```

说明

使用“using”语句实例化“Transaction”，以确保正确对其进行处理，即使发生异常进而回滚事务，亦能如此。

一致提交或回滚

使用客户端应用程序中的“Transaction”有助于确保以可预测的方式提交或回滚系列修改。客户端应用程序必须决定是否提交对默认设置的修改。为此，应用程序必须要求在通过调用“Transaction.CommitOnDispose()”方法处理事务时，提交活动事务范围内的修改。如果从未在代码流中调用该方法，则在处理活动事务范围内的修改时，会自动回滚此类修改。

如果在发出请求后出现异常，活动事务范围内的所有修改仍会在处理期间进行回滚。

修改以下程序代码以在包含两个“Create”修改的连接 TIA Portal 中创建一个撤消单元：

```
ExclusiveAccess exclusiveAccess = ...;
Project project = ...;
using (Transaction transaction = exclusiveAccess.Transaction(project, "My Operation")
{
    project.DeviceGroups.Create("My Group 1");
    project.DeviceGroups.Create("My Group 2");
    transaction.CommitOnDispose();
}
```

限制

事务内不允许以下操作。调用这些操作会导致可恢复异常：

- Compile
- Go Online
- Go Offline

- ProjectText Import
- ProjectText Export
- Open Global Library
- Close Global Library
- Project Create
- Project Open
- Project OpenWithUpgrade
- Project Save
- Project Close

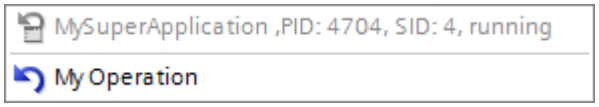
撤消行为

Openness 客户端应用程序执行的操作可能导致在连接的 TIA Portal 过程中出现撤消单元。将在位置条目下组合其中的各个撤消条目。该位置条目会组合客户端应用程序中的以下信息：

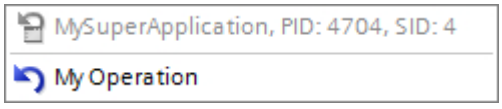
- 清单数据的程序集标题（如果可用）；不可用的话则为过程名称
- 过程 ID
- SID
- 表明客户端过程仍在运行的指示（可选）

这些条目将属于以下两种类型之一：

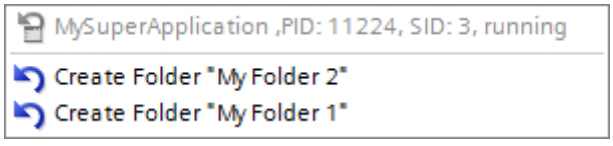
1. 对于因使用“事务”而被集合到一个撤消事务中的操作，实例化“事务”时客户端应用程序提供了相应的说明。
   - 正在运行的客户端应用程序的撤消条目：

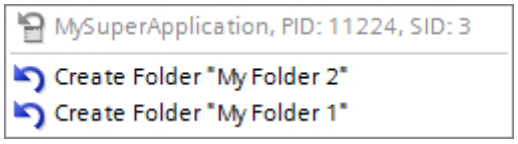
   - 已停止运行的客户端应用程序的撤消条目：

2. 对于单独执行的操作，其各个撤消条目介绍了相应的命令元数据中所定义的操作。
   - 正在运行的客户端应用程序的撤消条目：

   - 已停止运行的客户端应用程序的撤消条目：

7.9.10 创建 DirectoryInfo/FileInfo 对象

应用

`DirectoryInfo` 和 `FileInfo` 类别的实例必须包含绝对路径。否则，使用 `DirectoryInfo` 或 `FileInfo` 对象的方法将导致异常。

程序代码

修改以下程序代码以创建 DirectoryInfo 或 FileInfo 对象。

```
..
    //Create a DirectoryInfo object
    string directoryPath = @"D:\Test\Project 1";
    DirectoryInfo directoryInfo = new DirectoryInfo(directoryPath);


    //Create a FileInfo object
    string fileName = @"D:\Test\Project 1\Project 1.ap14");
    FileInfo fileInfo = new FileInfo(fileName);
...
```

7.10 项目和项目数据的功能

7.10.1 打开项目

要求

- Openness 应用程序连接到 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 要打开的项目不可在任何其它 TIA Portal 实例中打开。

说明

撤消项目升级

如果在将项目连接到 Openness 之后撤消将项目升级到 V14SP1 的过程，则会发生冲突。

应用

使用 `Projects.Open` 方法打开项目。在 `Projects.Open` 方法中，输入到所需项目的路径。

`Projects.Open` 方法只能访问由当前版本 TIA Portal 创建的项目或已升级到当前版本的项目。如果使用 `Projects.Open` 方法访问先前版本的项目，将返回异常。使用 `OpenWithUpgrade` 方法打开通过早期版本的 TIA Portal 创建的项目。

说明

不能访问只读项目

TIA Portal Openness 只能访问具有读写权限的项目。

程序代码

修改以下程序代码以打开项目：

```
Project project =tiaPortal.Projects.Open(new FileInfo(@"D:\Some\Path\Here\Project.apXX"));
if (project != null)
{
    try
    {
        ...
    }
    finally
    {
        project.Close();
    }
}
```

打开使用早期版本创建的项目

使用 OpenWithUpgrade 方法打开通过早期版本的 TIA Portal 创建的项目。此方法将创建一个新的已升级项目并将其打开。

如果访问使用比先前版本更早的版本创建的项目，则会返回异常。

说明

如果访问使用当前版本创建的项目，则会打开此项目。

程序代码

修改以下程序代码以通过 OpenWithUpgrade 方法打开项目：

```
Project project = tiaPortal.Projects.OpenWithUpgrade(new FileInfo(@"D:\Some
\Path\Here\Project.apXX"));
if (project != null)
{
    try
    {
        ...
    }
    finally
    {
        project.Close();
    }
}
```

7.10.2 保存项目

要求

- Openness 应用程序连接到 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)

应用

使用 `project.Save()` 方法可保存项目。

程序代码

修改以下程序代码以保存并打开项目：

```
public static void SaveProject(TiaPortal tiaPortal)
{
    Project project = null;
    //Use the code in the try block to open and save a project
    try
    {
        project = tiaPortal.Projects.Open(new FileInfo(@"Some\Path\MyProject.ap14"));
        //begin of code for further implementation
        //...
        //end of code
        project.Save();
    }
        //Use the code in the final block to close a project
    finally
    {
      if (project != null)
      project.Close();
    }
}
```

7.10.3 编译项目

要求

- Openness 应用程序已连接 TIA Portal。
  参见“连接到 TIA Portal (页 69)”
- 项目已经打开。
  参见“打开项目 (页 96)”
- 所有设备都处于“离线”状态。

应用

API 接口支持设备和程序块的编译。编译结果作为对象返回。将根据对象类型提供 HW、SW 或 HW/SW 编译。支持以下对象类型：

- `Device - HW & SW`
  - `Device`，配有故障安全 CPU - `SW`
- `DeviceItem - HW`
- `CodeBlock - SW`
- `DataBlock - SW`
- `HmiTarget - SW`
- `PlcSoftware - SW`
- `PlcType - SW`
- `PlcBlockSystemGroup - SW`
- `PlcBlockUserGroup - SW`
- `PlcTypeSystemGroup - SW`
- `PlcTypeUserGroup - SW`

说明

时间戳格式

所有时间戳均为 UTC 时间。如果要查看当地时间，可以使用 DateTime.ToLocalTime()。

签名

使用 `ICompilable` 类函数进行编译。

```
ICompilable compileService =
iEngineeringServiceProvider.GetService<ICompilable>();
CompilerResult result = compileService.Compile();
```

说明

开始编译前，所有设备必须为“离线”状态。

程序代码

修改以下程序代码以编译 HmiTarget 类型对象的软件变更：

```
public static void CompileHmiTarget(HmiTarget hmiTarget)
{
    ICompilable compileService = hmiTarget.GetService<ICompilable>();
    CompilerResult result = compileService.Compile();
}
```

修改以下程序代码以编译 PlcSoftware 类型对象的软件变更：

```
public static void CompilePlcSoftware(PlcSoftware plcSoftware)
{
    ICompilable compileService = plcSoftware.GetService<ICompilable>();
    CompilerResult result = compileService.Compile();
}
```

修改以下程序代码以编译 CodeBlock 类型对象的软件变更：

```
public static void CompileCodeBlock(PlcSoftware plcSoftware)
{
    CodeBlock block = plcSoftware.BlockGroup.Blocks.Find("MyCodeBlock") as CodeBlock;
    if (block != null)
    {
        ICompilable compileService = block.GetService<ICompilable>();
        CompilerResult result = compileService.Compile();
    }
}
```

修改以下程序代码以评估编译结果：

```
private void WriteCompilerResults(CompilerResult result)
{
    Console.WriteLine("State:" + result.State);
    Console.WriteLine("Warning Count:" + result.WarningCount);
    Console.WriteLine("Error Count:" + result.ErrorCount);
    RecursivelyWriteMessages(result.Messages);
}
private void RecursivelyWriteMessages(CompilerResultMessageComposition messages, string
indent = "")
{
    indent += "\t";
    foreach (CompilerResultMessage message in messages)
    {
        Console.WriteLine(indent + "Path: " + message.Path);
        Console.WriteLine(indent + "DateTime: " + message.DateTime);
        Console.WriteLine(indent + "State: " + message.State);
        Console.WriteLine(indent + "Description: " + message.Description);
        Console.WriteLine(indent + "Warning Count: " + message.WarningCount);
        Console.WriteLine(indent + "Error Count: " + message.ErrorCount);
        RecursivelyWriteMessages(message.Messages, indent);
    }
}
```

参见

导入组态数据 (页 345)

7.10.4 创建一个项目

要求

- Openness 应用程序已连接 TIA Portal。
  参见“AUTOHOTSPOT”

应用

可采用以下几种方式通过公共 API 进行项目创建。

- 在 ProjectComposition 中，调用 “创建”(Create) 类函数。
- 在 IEngineeringComposition 中，调用 “创建”(Create) 类函数。

ProjectComposition.Create

修改以下程序代码：

```
TiaPortal tiaPortal = ...;
ProjectComposition projectComposition = tiaPortal.Projects;
DirectoryInfo targetDirectory = new DirectoryInfo(@"D:\TiaProjects");

// Create a project with name MyProject
Project project = projectComposition.Create(targetDirectory, "MyProject");
```

在本示例中，

- 将创建一个“D:\TiaProjects\MyProject”文件夹。
- 将创建一个“D:\TiaProjects\MyProject\MyProject.aPXX”项目文件。

说明

有关参数 targetDirectory

参数 targetDirectory 也可用于指示一个 UNC (Universal Naming Convention) 路径，因此可在网络共享驱动中创建项目。

IEngineeringComposition.Create

修改以下程序代码：

```
TiaPortal tiaPortal = ...;
ProjectComposition projectComposition = tiaPortal.Projects;

//allows the user to give optional create parameters like author, comment in addition to
mandatory create parameters (targetdirectory, projectname)

IEnumerable<KeyValuePair<string, object>> createParameters = new [] {
          new KeyValuePair<string, object>("TargetDirectory", new DirectoryInfo(@"D:
\TiaProjects")), // Mandatory
          new KeyValuePair<string, object>("Name", "MyProject"), // Mandatory
          new KeyValuePair<string, object>("Author", "Bob"), // Optional
          new KeyValuePair<string, object>("Comment", "This project was created with
Openness") // Optional };

// Create a project with both mandatory and optional parameters
((IEngineeringComposition)projectComposition).Create(typeof (Project), createParameters);
```

在本示例中，

- 将创建一个“D:\TiaProjects\MyProject”文件夹。
- 将创建一个“D:\TiaProjects\MyProject\MyProject.aPXX”项目文件。其中，项目属性 Author 为“Bob”，Comment 为“This project was created with openness”。

使用可选项目属性创建项目的参数

参数	数据类型	类型	说明
Author	String	必需	项目的作者。
Comment	String	可选	项目的注释信息。
Name	String	可选	项目的名称
TargetDirectory	DirectoryInfo	必需	包含所创建项目文件夹的目录。

7.10.5 关闭项目

要求

- Openness 应用程序连接到 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已通过 Openness 应用程序打开一个项目。
  请参见打开项目 (页 96)

程序代码

修改以下程序代码以关闭项目：

```
public static void CloseProject(Project project)
{
    project.Close();
}
```

7.10.6 TIA Portal 的常规访问设置

要求

- Openness 应用程序连接到 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)

应用

可通过 Openness 访问 TIA Portal 的常规设置：

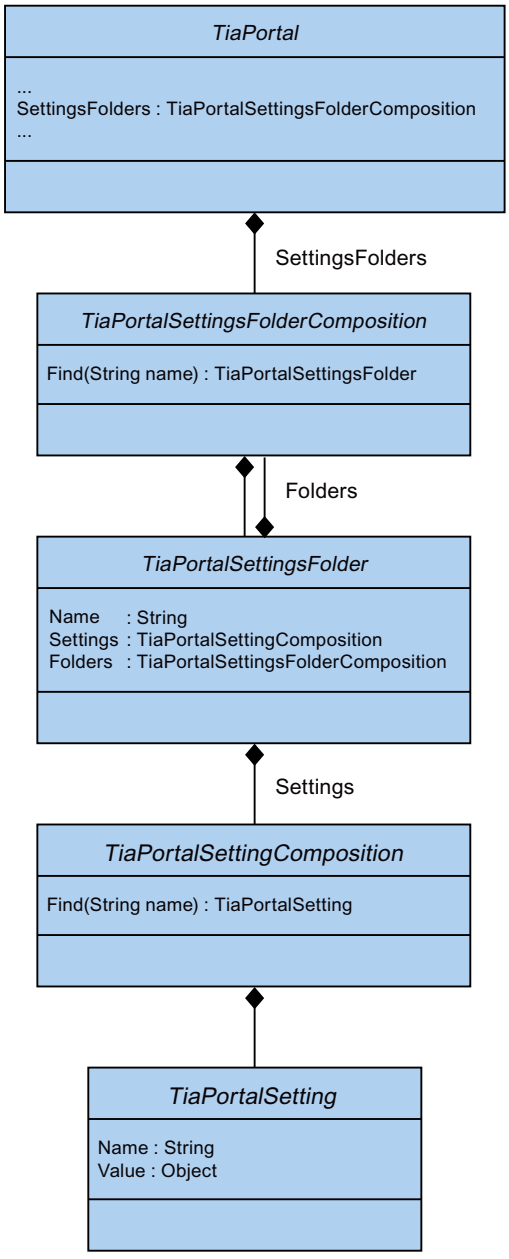
- 当前用户界面语言
- “在项目中搜索”(Search in project) 选项创建在项目内搜索所需的搜索索引。

下表显示了 TIA portal 设置的“常规”(General) 部分中可访问设置的详细信息。`TiaPortalSettingsFolder` 实例的名称将为“常规”。

设置名称	数据类型	可写入	说明
`"SearchInProject"`	System.Boolean	r/w	启用或禁用创建在项目内搜索所需的搜索索引。
`"UserInterfaceLanguage"`	System.CultureInfo	r/w	指示 TIA Portal 的活动用户界面语言或活动用户界面语言的规范。

通过 `TiaPortalSettingsFolder` 类别提供对这些设置的访问。可通过 `TiaPortal` 类别上的 `Settings` 属性访问 `TiaPortalSettingsFolder` 类别。

下图显示了 Openness 中的特殊设置：


TiaPortal
...
SettingsFolders : TiaPortalSettingsFolderComposition
...
SettingsFolders
TiaPortalSettingsFolderComposition
Find(String name) : TiaPortalSettingsFolder
Folders
TiaPortalSettingsFolder
Name : String
Settings : TiaPortalSettingComposition
Folders : TiaPortalSettingsFolderComposition
Settings
TiaPortalSettingComposition
Find(String name) : TiaPortalSetting
TiaPortalSetting
Name : String
Value : Object

程序代码：在项目中搜索

修改以下程序代码以激活/禁用 “在项目中搜索”(Search in project) 选项：

```
private static void SetSearchInPoject(Project project)
{
    TiaPortalSettingsFolder generalSettingsFolder =
tiaPortal.SettingsFolders.Find("General");
    TiaPortalSetting searchSetting =
generalSettingsFolder.Settings.Find("SearchInProject");

        if (((bool)searchSetting.Value))
        {
            searchSetting.Value = false;
        }
   }
```

程序代码：用户界面语言

修改以下程序代码以访问当前用户界面语言。

```
private static void SetUILanguage(Project project)
{

    TiaPortalSettingsFolder generalSettingsFolder =
tiaPortal.SettingsFolders.Find("General");

    TiaPortalSetting UILanguageSetting =
generalSettingsFolder.Settings.Find("UserInterfaceLanguage");

        if (((CultureInfo)UILanguageSetting.Value) != CultureInfo.GetCultureInfo("de-DE"))
        {
            UILanguageSetting .Value = CultureInfo.GetCultureInfo("de-DE");
        }

}
```

参见

对象模型的硬件对象的层级 (页 58)

7.10.7 访问语言

要求

- Openness 应用程序连接到 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)

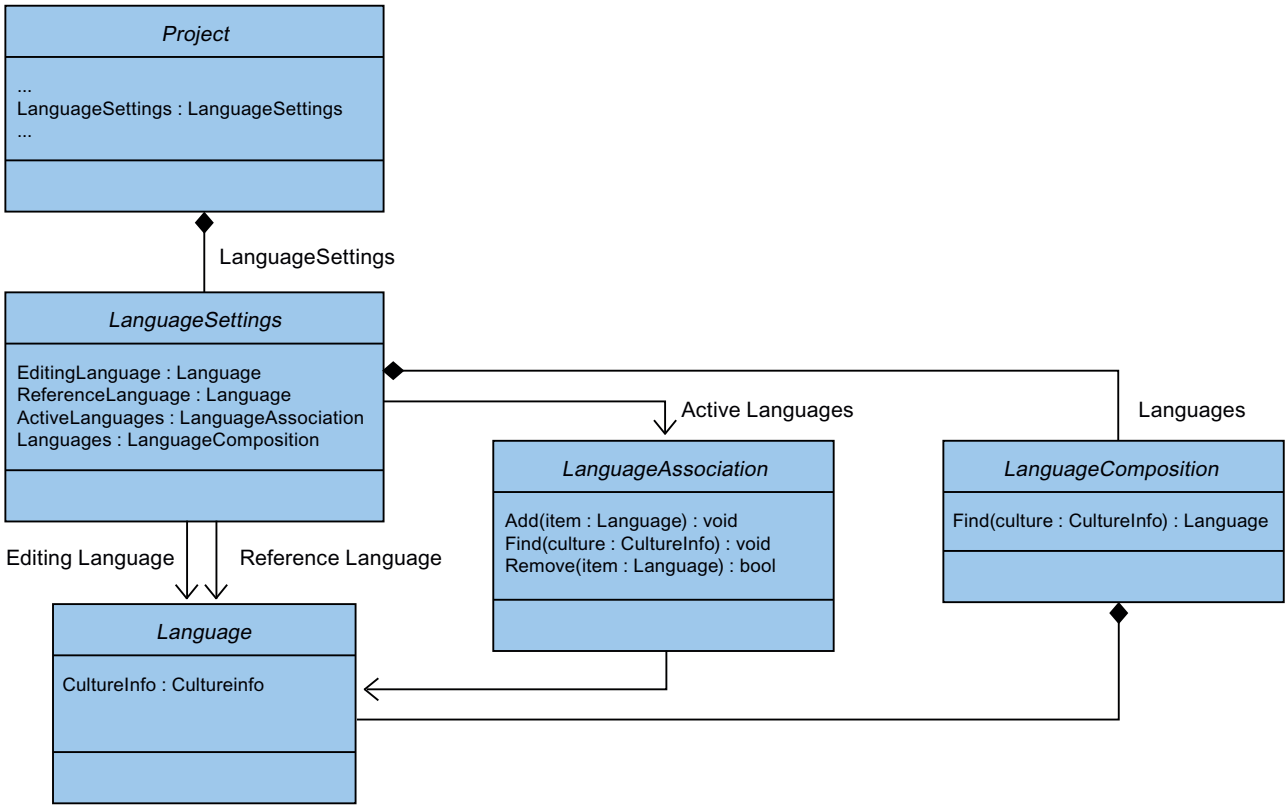
应用

在 TIA Portal 中，可在“项目语言”(Project languages) 编辑器中设置和管理项目语言。

Openness 支持对项目语言进行以下访问：

- 通过支持的语言进行迭代。
- 由 `System.Globalization.CultureInfo` 通过支持的语言集合进行搜索。
- 访问各个语言。每个语言对象将包含一个 `System.Globalization.CultureInfo.` 类型的只读属性 `Culture`。
- 访问活动语言的集合。
- 由 `System.Globalization.CultureInfo` 通过活动的语言集合进行搜索。
- 向活动的语言集合添加语言。
- 从活动的语言集合删除语言。
- 设置编辑语言。
- 设置参考语言。

功能由 `LanguageSettings` 对象提供。下图显示了 Openness 提供的模型：



程序代码：设置语言

修改以下程序代码以设置语言：如果通过 Openness 设置非活动语言，则会将该语言将添加到活动语言集合。

```
Project project = ...;

    LanguageSettings languageSettings = project.LanguageSettings;

    LanguageComposition supportedLanguages = languageSettings.Languages;
    LanguageAssociation activeLanguages = languageSettings.ActiveLanguages;

    Language supportedGermanLanguage =
supportedLanguages.Find(CultureInfo.GetCultureInfo("de-DE"));
    activeLanguages.Add(supportedGermanLanguage);

    languageSettings.EditingLanguage = supportedGermanLanguage;
    languageSettings.ReferenceLanguage = supportedGermanLanguage;
```

程序代码：禁用活动语言

修改以下程序代码以禁用活动语言。如果禁用用作参考或编辑语言的语言，所选语言将与用户界面中的行为一致。

```
Project project = ...;

    LanguageSettings languageSettings = project.LanguageSettings;
    LanguageAssociation activeLanguages = languageSettings.ActiveLanguages;
    Language activeGermanLanguage = activeLanguages.Find(CultureInfo.GetCultureInfo("de-
DE"));
    activeLanguages.Remove(activeGermanLanguage);
```

参见

对象模型的硬件对象的层级 (页 58)

7.10.8 读取项目相关的属性

要求

- Openness 应用程序已连接 TIA Portal。
  参见“连接到 TIA Portal (页 69)”
- 项目已打开。
  参见“打开项目 (页 96)”

应用

使用该功能，可从公共 API 中获取项目相关的属性。提供的信息包含项目属性、项目历史以及项目所使用的产品。

项目属性

项目属性提供了以下信息：

属性名称	数据类型	可写	说明
Author	System.String	r/o	项目的作者
Comment	Siemens.Engineering.MultilingualText	r/o	项目的注释信息

属性名称	数据类型	可写	说明
Copyright	System.String	r/o	项目的版权声明
CreationTime	System.DateTime	r/o	项目的创建时间
Family	System.String	r/o	项目的所属系列
IsModified	System.Boolean	r/o	如果项目已修改，则返回“true”。
LanguageSettings	Siemens.Engineering.LanguageSettings	r/o	项目的处理语言
LastModified	System.DateTime	r/o	项目上一次的修改时间
LastModifiedBy	System.String	r/o	上一次的修改人员
Name	System.String	r/o	参数的名称
Path	System.IO.FileInfo	r/o	项目的绝对路径
Size	System.Int64	r/o	项目的大小 (KB)
Version	System.String	r/o	项目的版本

修改以下程序代码以访问项目相关的属性：

```
Project project = ...;
string author = project.Author;
string name = project.Name;
string path = project.Path;
DateTime creationTime = project.CreationTime;
DateTime modificationTime = project.LastModified;
string lastModifiedBy = project.LastModifiedBy;
string version = project.Version;
MultilingualText comment = project.Comment;
string copyright = project.Copyright;
string family = project.Family;
Int64 size = project.Size;
LanguageSettings languageSettings = project.LanguageSettings;
```

修改以下程序代码以枚举项目语言：

```
Project project = ...;
LanguageComposition languages = project.LanguageSettings.Languages;
foreach (Language language in languages)
{
    CultureInfo lang = language.Culture;
}
```

修改以下程序代码以获取注释文本：

```
Project project = ...;
Language english =
project.LanguageSettings.ActiveLanguages.Find(CultureInfo.GetCultureInfo("en-US"));

MultilingualText projectComment = project.Comment;
MultilingualTextItem textItem = project.Comment.Items.Find(english);
string text = textItem.Text;
```

项目历史

项目历史即为 HistoryEntry 对象的组合，其中包含以下信息：

属性名称	数据类型	可写	说明
文本	System.String	r/o	事件说明
DateTime	System.DateTime	r/o	发生事件的时间

修改以下程序代码以通过 HistoryEntries 进行枚举和访问相应的属性：

```
Project project = ...;
HistoryEntryComposition historyEntryComposition = project.HistoryEntries;
foreach (HistoryEntry historyEntry in historyEntryComposition)
{
    string entryText = historyEntry.Text;
    DateTime entryTime = historyEntry.DateTime;
}
```

说明

HistoryEntry 的文本属性包含以 UI 语言进行表示的字符串。如果在没有 UI 的情况下将 Openness 应用程序连接到 TIA Portal，则字符串始终以英语进行表示。

使用的产品

对象 UsedProduct 包括以下信息：

属性名称	数据类型	可写	说明
名称	System.String	r/o	所用的产品名称
版本	System.String	r/o	产品版本

修改以下程序代码以通过 UsedProduct 进行枚举和访问相应的属性。

```
Project project = ...;
UsedProductComposition usedProductComposition = project.UsedProducts;
foreach (UsedProduct usedProduct in usedProductComposition)
{
    string productName = usedProduct.Name;
    string productVersion = usedProduct.Version;
}
```

7.10.9 确定对象结构和属性

要求

- Openness 应用程序连接到 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已通过 Openness 应用程序打开一个项目。
  请参见打开项目 (页 96)

应用

可使用 IEngineeringObject 接口通过对象层级确定浏览结构。结果将以列表形式返回：

- 子对象
- 子组合
- 全部属性

签名

使用 GetAttributeInfos 方式确定属性。

```
IList<EngineeringAttributeInfo>
IEngineeringObject.GetAttributeInfos();
```

程序代码：确定对象或组合

使用以下程序代码可显示所有组合名称：

```
public static void DisplayCompositionInfos(IEngineeringObject obj)
{
    IList<EngineeringCompositionInfo> compositionInfos = obj.GetCompositionInfos();
    foreach (EngineeringCompositionInfo compositionInfo in compositionInfos)
    {
        Console.WriteLine(compositionInfo.Name);
    }
}
```

若知道返回值则修改以下程序代码：

```
public static DeviceItemComposition GetDeviceItemComposition(Device device)
{
    IEngineeringCompositionOrObject composition = ((IEngineeringObject)
device).GetComposition("DeviceItems");
    DeviceItemComposition deviceItemComposition = (DeviceItemComposition)composition;
    return deviceItemComposition;
}
```

程序代码：确定属性

修改以下程序代码以返回具有列表中特定访问权限的对象属性：

```
public static void DisplayAttributenInfos(IEngineeringObject obj)
{
    IList<EngineeringAttributeInfo> attributeInfos = obj.GetAttributeInfos();
    foreach (EngineeringAttributeInfo attributeInfo in attributeInfos)
    {
        Console.WriteLine("Attribute: {0} - AccessMode {1} ",
        attributeInfo.Name, attributeInfo.AccessMode);
        switch (attributeInfo.AccessMode)
        {
            case EngineeringAttributeAccessMode.Read: Console.WriteLine("Attribute: {0} -
Read Access", attributeInfo.Name);
            break;
           case EngineeringAttributeAccessMode.Write: Console.WriteLine("Attribute: {0} -
Write Access", attributeInfo.Name);
            break;
          case EngineeringAttributeAccessMode.Read | EngineeringAttributeAccessMode.Write:
Console.WriteLine("Attribute: {0} - Read and Write Access", attributeInfo.Name);
            break;
        }
    }
}
public static string GetNameAttribute(IEngineeringObject obj)
{
    Object nameAttribute = obj.GetAttribute("Name");
    return (string)nameAttribute;
}
```

7.10.10 访问软件目标

要求

- Openness 应用程序连接到 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 项目已经打开。
  请参见打开项目 (页 96)

程序代码

修改以下程序代码以使软件目标可用：

```
SoftwareContainer softwareContainer =
((IEngineeringServiceProvider)deviceItem).GetService<SoftwareContainer>();
if (softwareContainer != null)
{
    Software software = softwareContainer.Software;
}
```

修改以下程序代码以访问软件属性：

```
SoftwareContainer softwareContainer =
((IEngineeringServiceProvider)deviceItem).GetService<SoftwareContainer>();
if (softwareContainer != null)
{
    PlcSoftware software = softwareContainer.Software as PlcSoftware;
    string name = software.Name;
}
```

7.10.11 访问和枚举多语言文本

要求

- Openness 应用程序连接到 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)

应用

TIA Portal 中的多语言文本示例包括：Project.Comment、PlcTag.Comment 等。在 Opennes 中，多语言文本由 `MultilingualText` 对象表示。`MultilingualText` 对象由 `MultilingualTextItemComposition` 组成。

`MultilingualTextItemComposition` 支持以下 `Find` 方法：

- `Find(<language: Siemens.Engineering.Language>):MultilingualTextItem`

每个 MultilingualTextItem 具有以下属性：

属性名称	数据类型	可写入	说明
Language	Siemens.Engineering.Language	r/o	此项目的语言。
Text	System.String	r/w	为此语言提供的文本。

程序代码：设置多语言文本

```
...
    Language englishLanguage = project.LanguageSettings.Languages.Find(new CultureInfo("en-
US"));
    MultilingualText comment = project.Comment;
    MultilingualTextItemComposition mltItemComposition = comment.Items;
    MultilingualTextItem englishComment = mltItemComposition.Find(englishLanguage);
    englishComment.Text = "English comment";
...
```

7.10.12 库操作函数

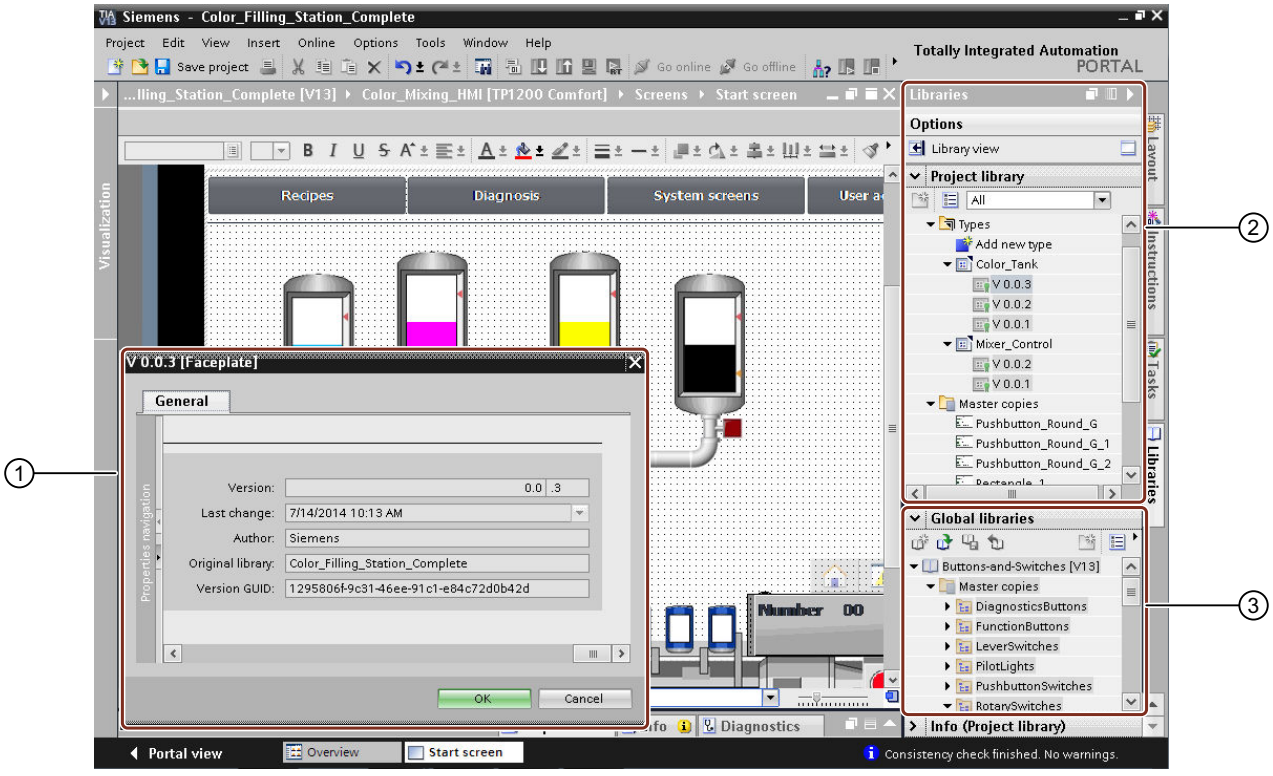
7.10.12.1 对象和实例操作函数

访问类型和实例

可以使用公共 API 接口访问项目库或全局库中的类型、类型版本和主副本。可以确定类型版本和实例间的连接。还可以更新项目中的实例以及在全局库和项目库之间同步更改。公共 API 接口还支持类型版本与实例的比较。

对象和实例操作函数

可以使用公共 API 接口访问类型、类型版本、主副本和实例的以下操作函数：



① 显示类型属性、类型版本、主副本和实例

① 项目库中提供以下功能：

- 更新类型实例
- 实例化项目中的类型版本
- 在库组内进行导航
- 删除组、类型、类型版本和主副本

① 全局库中提供以下功能：

- 更新类型实例
- 实例化项目中的类型版本
- 在库组内进行导航

7.10.12.2 访问全局库

要求

- Openness 应用程序连接到 TIA Portal。
  请参见连接到 TIA Portal (页 69)

应用

全局库有三种类型。

- 系统全局库：这些全局库随 TIA Portal 一起安装并采用 .as14 文件扩展名。所有系统全局库均为只读。
- 企业全局库：这些全局库已由管理员选择，可在 TIA Portal 启动时进行预加载。所有企业全局库均为只读。
- 用户全局库：这些全局库已由 TIA Portal 用户创建。用户全局库能够以只读模式或读写模式打开。
  如果用户全局库已采用某种模式打开，则不能使用另一种模式打开同一用户全局库。
  之前版本的用户全局库只能在只读模式下打开。

使用 Openness 打开的全局库还可添加至 TIA Portal UI 的全局库集合，并显示在 TIA Portal UI 中（若 UI 存在）。

程序代码：可用的全局库

修改以下程序代码以获取所有可用全局库的相关信息：

```
TiaPortal tia = ...;
var availableLibraries = tia.GlobalLibraries.GetGlobalLibraryInfos();
foreach (GlobalLibraryInfo info in availableLibraries)
{
    //work with the global library info
    Console.WriteLine("Library Name: ", info.Name);
    Console.WriteLine("Library Path: ", info.Path);
    Console.WriteLine("Library Type: ", info.LibraryType);
    Console.WriteLine("Library IsOpen: ", info.IsOpen);
}
```

GlobalLibrary 属性

值	数据类型	说明
Author	String	全局库的作者。
Comment	MultilingualText	全局库的注释。
IsReadOnly	Boolean	全局库为只读时为真。
IsModified	Boolean	全局库的内容已更改时为真。
Name	String	全局库的名称。
Path	FileInfo	全局库的路径。

GlobalLibraryInfo 属性

值	返回类型	说明
IsReadOnly	Boolean	全局库为只读时为真。
IsOpen	Boolean	全局库已打开时为真。
LibraryType	GlobalLibraryType	全局库的类型： • System：系统全局库 • Corporate：企业全局库 • User：用户全局库
Name	String	全局库的名称。
Path	FileInfo	全局库的路径。

参见

在库中访问文件夹 (页 123)

7.10.12.3 打开库

要求

- Openness 应用程序连接到 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已通过 Openness 应用程序打开一个项目。此要求仅适用于访问项目库。
  请参见打开项目 (页 96)

应用

全局库可采用 System.IO.FileInfo 通过本地存储介质或网络存储的库文件路径打开。只有用户全局库才可通过路径打开。从系统全局库或企业全局库获取的路径无法用于打开操作。

自 V14 SP1 起的全局库可使用 GlobalLibraryInfo 打开。OpenMode 可在 GlobalLibraryInfo 中指定。

早期版本的 TIA Portal 的用户全局库可用当前版本的 TIA Portal 升级和打开。V13 或更早版本的全局库无法用升级版本打开。这些库必须首先升级至 V13 SP1。

使用 Openness 打开的库还可添加至 TIA Portal 的全局库集合，并显示在 TIA Portal 的用户界面中。

程序代码：使用 System.IO.FileInfo 打开库

修改以下程序代码：

```
TiaPortal tia = ...
FileInfo fileInfo = ....

UserGlobalLibrary userLib = tia.GlobalLibraries.Open(fileInfo, OpenMode.ReadWrite);
```

程序代码：使用 GlobalLibraryInfo 打开库

修改以下程序代码：

```
TiaPortal tia = ...
IList<GlobalLibraryInfo> libraryInfos = tia.GlobalLibraries.GetGlobalLibraryInfos();
GlobalLibraryInfo libInfo = ...; //check for the info you need from the list, e.g.
GlobalLibrary libraryOpenedWithInfo;
if (libInfo.Name == "myLibrary")
libraryOpenedWithInfo = tia.GlobalLibraries.Open(libInfo);
```

程序代码：升级库

修改以下程序代码：

```
TiaPortal tia = ...
FileInfo fileInfo = .... //library from previous TIA Portal version

UserGlobalLibrary userLib = tia.GlobalLibraries.OpenWithUpgrade(fileInfo);
```

OpenMode

值	说明
ReadOnly	库的读访问。
ReadWrite	库的读写访问。

7.10.12.4 枚举打开的库

要求

- Openness 应用程序连接至 TIA Portal。
  请参见连接到 TIA Portal (页 69)

应用

TIA Portal 中所有打开（不论是通过 API 打开还是通过用户界面打开）的全局库均可枚举。

如果早期版本的 TIA Portal 的全局库以写访问的方式打开，则无法对其进行枚举。

程序代码

修改以下程序代码以枚举打开的全局库：

```
TiaPortal tia = ...
foreach (GlobalLibrary globLib in tia.GlobalLibraries)
{
    ////work with the global library
}
```

参见

打开项目 (页 96)

7.10.12.5 保存并关闭库

要求

- Openness 应用程序连接到 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 库已打开。
  请参见打开库 (页 119)

应用

用户全局库可关闭或保存。对全局库的所有更改不会自动保存。若关闭全局库，则所有未保存的更改均将丢弃且不会给予任何提示。

系统全局库和企业全局库无法关闭或保存。

程序代码

修改以下程序代码：

```
UserGlobalLibrary userLib = ...

// close and discard changes
userLib.Close();

// save changes and close library
userLib.Save();
userLib.Close();
```

7.10.12.6 创建全局库

要求

- Openness 应用程序已连接 TIA Portal。
  参见“连接到 TIA Portal (页 69)”

应用

在 GlobalLibraryComposition 中调用 Create 类函数，基于公共 API 创建各种全局库。此时，系统将返回一个 UserGlobalLibrary

GlobalLibraryComposition.Create

修改以下程序代码：

```
TiaPortal tia= ...;
DirectoryInfo targetDirectory = new DirectoryInfo(@"D:\GlobalLibraries");
UserGlobalLibrary globalLibrary =
tia.GlobalLibraries.Create<UserGlobalLibrary>(targetDirectory, "Library1")
```

在本示例中，

- 将创建文件夹“D:\GlobalLibraries\Library1”
- 将创建全局库文件“D:\GlobalLibraries\Library1\Library1.alXX”

用于创建全局库的参数

参数	数据类型	类型	说明
Author	String	必需	全局库的作者。
Comment	String	可选	全局库的注释。
Name	String	可选	全局库的名称。
TargetDirectory	DirectoryInfo	必需	将包含全局库文件夹的目录。

参见

打开项目 (页 96)

7.10.12.7 在库中访问文件夹

要求

- Openness 应用程序连接到 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已通过 Openness 应用程序打开一个项目。
  请参见打开项目 (页 96)
- 您具有所需库的访问权限。
  请参见访问全局库 (页 118)。

应用

您可以通过 Public API 接口在库中访问有关类型和主副本的系统文件夹。您可以访问系统文件夹中的类型、类型版本、主副本以及用户定义文件夹。

您可通过 `Find` 方法随时访问用户定义的文件夹，例如 `libTypeUserFolder.Folders.Find("SomeUserFolder");`。

程序代码：访问系统文件夹

修改以下程序代码以访问库中类型的系统文件夹：

```
public static void AccessTypeSystemFolder(ILibrary library)
{
    LibraryTypeSystemFolder libTypeSystemFolder = library.TypeFolder;
}
```

修改以下程序代码以访问库中主副本的系统文件夹：

```
public static void AccessMasterCopySystemFolder(ILibrary library)
{
    MasterCopySystemFolder libMasterCopySystemFolder = library.MasterCopyFolder;
}
```

程序代码：通过 Find() 方法访问用户自定义文件夹：

修改以下程序代码：

```
...
LibraryTypeUserFolderComposition userFolderComposition = ...
LibraryTypeUserFolder userFolder = userFolderComposition.Find("Name of user folder");
...
```

程序代码：枚举用户自定义文件夹

修改以下程序代码以枚举类型系统文件夹中的用户自定义子文件夹：

```
public static void EnumerateUserFoldersInTypeSystemFolder(ILibrary library)
{
    // Enumerating user folders in type system folder:
    LibraryTypeSystemFolder libTypeSystemFolder = library.TypeFolder;
    foreach (LibraryTypeUserFolder libTypeUserFolder in libTypeSystemFolder.Folders)
    {
        //...
    }
}
```

修改以下程序代码以枚举主副本系统文件夹中的用户自定义子文件夹：

```
public static void EnumerateUserFoldersInMasterCopySystemFolder(ILibrary library)
{
    // Enumerating user folders in master copy system folder:
    MasterCopySystemFolder libMasterCopySystemFolder = library.MasterCopyFolder;
    foreach (MasterCopyUserFolder libMasterCopyUserFolder in
libMasterCopySystemFolder.Folders)
    {
        //..
    }
}
```

修改以下程序代码以枚举类型用户自定义文件夹中的用户自定义子文件夹：

```
public static void EnumerateAllUserFolders(LibraryTypeUserFolder libUserFolder)
{
    foreach (LibraryTypeUserFolder libSubUserFolder in libUserFolder.Folders)
    {
       EnumerateAllUserFolders(libSubUserFolder);
    }
}
```

修改以下程序代码以枚举主副本用户自定义文件夹中的用户自定义子文件夹：

```
public static void EnumerateAllUserFolders(MasterCopyUserFolder libUserFolder)
{
    foreach (MasterCopyUserFolder libSubUserFolder in libUserFolder.Folders)
    {
       EnumerateAllUserFolders(libSubUserFolder);
    }
}
```

程序代码：创建用户自定义文件夹

修改以下程序代码以创建类型用户自定义文件夹：

```
var typeFolderComposition = ProjectLibrary.TypeFolder.Folders;
var newTypeUserFolder = typeFolderComposition.Create("NewTypeUserFolder");
```

修改以下程序代码以创建主副本用户自定义文件夹：

```
var masterCopyFolderComposition = projectProjectLibrary.MasterCopyFolder.Folders;
MasterCopyUserFolder newMasterCopyUserFolder =
masterCopyFolderComposition.Create("NewMasterCopyUserFolder);
```

程序代码：重命名用户自定义文件夹

修改以下程序代码以创建类型用户自定义文件夹：

```
var typeUserFolder =
project.ProjectLibrary.TypeFolder.Folders.Find("SampleTypeUserFolderName");
typeUserFolder.Name = "NewTypeUserFolderName";
```

```
var typeUserFolder = ProjectLibrary.TypeFolder.Folders.Find("SampleTypeUserFolderName");
typeUserFolder.SetAttributes(new[] {new KeyValuePair<string,object>("Name",
"NewTypeUserFolderName")});
```

修改以下程序代码以创建主副本用户自定义文件夹：

```
var masterCopyUserFolder =
project.ProjectLibrary.MasterCopyFolder.Folders.Find("SampleMasterCopyUserFolderName");
masterCopyUserFolder.Name = "NewMasterCopyUserFolderName";
```

```
var masterCopyUserFolder =
ProjectLibrary.MasterCopyFolder.Folders.Find("SampleMasterCopyUserFolderName");
masterCopyUserFolder.SetAttributes(new[] {new KeyValuePair<string,object>("Name",
"NewMasterCopyUserFolderName")});
```

参见

访问主副本 (页 137)

7.10.12.8 访问类型

要求

- Openness 应用程序连接到 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已通过 Openness 应用程序打开一个项目。
  请参见打开项目 (页 96)
- 您具有所需库的访问权限。
  请参见访问全局库 (页 118)。
- 您具有类型组的访问权限。
  请参见在库中访问文件夹 (页 123)。

应用

您可以通过公共 API 接口访问包含在库中的类型。

- 您可以枚举类型。
- 可以重命名类型。
- 您可以访问各个类型的下列属性：

属性	数据类型	说明
Author	String	返回作者的姓名。
Comment	MultilingualText	返回注释。
Guid	Guid	返回类型的 GUID。[1]
Name	String	返回类型的名称。[2]

[1] 您可使用此属性在库中查找一个具体类型。该搜索是递归的。

[2] 您可使用此属性在文件夹中查找一个具体类型。子文件夹不包括在搜索范围内。类型名称不唯一。不同组中可能会有多个具有相同名称的类型。但 Guid 类型是唯一的。

库类型对象的子类别

利用公共 API，您可以通过子类别访问库类型对象。存在以下子类别：

- Siemens.Engineering.Hmi.Faceplate.FaceplateLibraryType
- Siemens.Engineering.Hmi.RuntimeScripting.VBScriptLibraryType
- Siemens.Engineering.Hmi.RuntimeScripting.CScriptLibraryType
- Siemens.Engineering.Hmi.Screen.ScreenLibraryType
- Siemens.Engineering.Hmi.Screen.StyleLibraryType
- Siemens.Engineering.Hmi.Screen.StyleSheetLibraryType
- Siemens.Engineering.Hmi.Tag.HmiUdtLibraryType
- Siemens.Engineering.SW.Blocks.CodeBlockLibraryType
- Siemens.Engineering.SW.Types.PlcTypeLibraryType

以下代码为库类型子类别使用方法示例

```
ProjectLibrary library = project.ProjectLibrary;
VBScriptLibraryType vbScriptType = ...;

VBScriptLibraryType libraryTypeAsVbScript = libraryType as VBScriptLibraryType;
```

程序代码

修改以下程序代码以枚举库的系统文件夹中的所有类型：

```
public static void EnumerateTypesInTypesSystemFolder(LibraryTypeSystemFolder
libraryTypeSystemFolder)
{
    foreach (LibraryType libraryType in libraryTypeSystemFolder.Types)
    {
        //...
    }
}
```

修改以下程序代码以枚举库的用户自定义文件夹中的所有类型：

```
public static void EnumerateTypesInTypesUserFolder (LibraryTypeUserFolder
libraryTypeUserGroup)
{
    foreach (LibraryType libraryType in libraryTypeUserGroup.Types)
    {
        //...
    }
}
```

修改以下程序代码以访问一个类型的属性：

```
public static void InspectPropertiesOfType (LibraryType libTypeObject)
{
    string typeAuthor = libTypeObject.Author;
    MultilingualText typeComment = libTypeObject.Comment;
    string typeName = libTypeObject.Name;
    Guid typeGUID = libTypeObject.Guid;
}
```

修改以下程序代码以通过名称或 GUID 查找单个类型：

```
public static void FindTypeObjectInLibrary(ILibrary library)
{
    // Find type object by its GUID in a given library:
    System.Guid targetGuid = ...;
    LibraryType libTypeByGUID = library.FindType(targetGuid);
    // Find type object by its name in a given group:
    LibraryTypeFolder libTypeSystemFolder = library.TypeFolder;
    LibraryType libTypeByName = libTypeSystemFolder.Types.Find("myTypeObject");
}
```

修改以下程序代码以重命名类型：

```
// Setting the name property
var type = project.ProjectLibrary.TypeFolder.Types.Find("SampleTypeName");
type.Name = "NewTypeName";

//Setting the name property dynamically
var type = project.ProjectLibrary.TypeFolder.Types.Find("SampleTypeName");
type.SetAttributes(new[] {new KeyValuePair<string,object>("Name", "NewTypeName")});
```

7.10.12.9 访问类型版本

要求

- Openness 应用程序已连接 TIA Portal。
  参见“连接到 TIA Portal (页 69)”
- 已通过 Openness 应用程序打开一个项目。
  参见“打开项目 (页 96)”
- 需具有指定库的访问权限。
  参见“访问全局库 (页 118)”。
- 需具有该类型组的访问权限。
  参见“在库中访问文件夹 (页 123)”。

应用

通过公共 API 接口访问类型版本。

- 可以枚举一个类型的多个类型版本。
- 可以确定类型版本所属的类型。
- 可以枚举一个类型版本的多个实例。
- 可以为一个类型版本创建新实例。
- 可以从某个实例导航至其所连接的版本对象。
- 可以访问各个类型版本的以下属性：

属性	数据类型	说明
Author	String	返回作者的姓名。
Comment	MultilingualText	返回注释。
Guid	Guid	返回类型版本的 GUID。[1]
ModifiedDate	DateTime	返回类型版本设为“Committed”状态的日期和时间。
State	LibraryTypeVersionState	返回版本的状态： • InWork：对应于状态“进行中”(In progress) 或“正在测试”(In testing)，具体取决于关联的类型。 • Committed：对应于状态“已发布” (Released)。
TypeObject	LibraryType	返回此类型版本所属的类型。
VersionNumber	Version	返回三位版本标识符格式的版本号，例如“1.0.0”。[2]

[1] 使用此属性在库中查找一个具体类型版本。

[2] 使用此属性在“LibraryTypeVersion”组合中查找某个具体类型版本。

枚举某一类型的所有类型版本

修改以下程序代码：

```
//Enumerate the type versions of a type
public static void EnumerateVersionsInType(LibraryType libraryType)
{
    foreach (LibraryTypeVersion libraryTypeVersion in libraryType.Versions)
    {
        //...
    }
}
```

访问类型版本的属性

修改以下程序代码：

```
//Acessing the properties of a type version
public static void InspectPropertiesOfVersion(LibraryTypeVersion libTypeVersion)
{
    string versionAuthor = libTypeVersion.Author;
    MultilingualText versionComment = libTypeVersion.Comment;
    Guid versionGUID = libTypeVersion.Guid; DateTime versionModifiedDate =
libTypeVersion.ModifiedDate;
    LibraryTypeVersionState versionStateLibrary = libTypeVersion.State;
    LibraryType versionParentObject = libTypeVersion.TypeObject;
    Version versionNumber = libTypeVersion.VersionNumber;
}
```

创建类型版本的实例

可以为一个类型版本创建新实例。支持以下对象：

- 块 (FB/FC)
- PLC 用户数据类型
- 画面
- VB 脚本

此时，将在全局库和项目库中创建一个类型版本的实例。当在全局库中创建类型版本实例时，类型版本会首先与项目库同步。

如果无法在目标中创建实例，则会出现可恢复异常。可能的原因有：

- 库类型版本正在使用中
- 库类型版本的实例已存在于目标设备中

修改以下程序代码：

```
VBScriptLibraryTypeVersion scriptVersion = ...;
VBScriptComposition vbscripts = ...;

//Using the CreateFrom method to create an instance of the version in the VBScripts
composition
VBScript newScript = vbscripts.CreateFrom(scriptVersion);
```

修改以下程序代码：

```
ScreenLibraryTypeVersion screenVersion = ...;
ScreenComposition screens = ...;

//Using the CreateFrom method to create an instance of the version in the screens
composition
Screen newScreen = screens.CreateFrom(screenVersion);
```

修改以下程序代码：

```
CodeBlockLibraryTypeVersion blockVersion = ...;
PlcBlockComposition blocks = ...;

//Using the CreateFrom method to create an instance of the version in the blocks composition
PlcBlock newBlock = blocks.CreateFrom(blockVersion);
```

修改以下程序代码：

```
PlcTypeLibraryTypeVersion plcTypVersione=...;
PlcTypeComposition types=...;

//Using the CreateFrom method to create an instance of the version in the types composition
PlcType newType = types.CreateFrom(plcTypeVersion);
```

确定类型版本的用途

类型版本分为以下用途：

- 类型版本使用库中的其它类型版本。
  示例：在程序块中使用一个用户数据类型。程序块必须具有该用户数据类型的访问权限。这意味着程序块取决于用户数据类型。
  当通过 GetDependencies() 方式访问代码块库版本的相关性 (Dependents) 属性时，将返回 LibraryTypeVersion 列表。
- 此类型正在被库中的另一类型版本使用。
  示例：在程序块中使用一个用户数据类型。程序块必须具有该用户数据类型的访问权限。该用户数据类型具有相关联的程序块。程序块取决于用户数据类型。
  当通过 GetDependents() 方式访问 PLC 类型库类型版本的相关性 (Dependents) 属性时，将返回 LibraryTypeVersion 列表。

这两种属性都会返回一个列表，其中包含 LibraryTypeVersion 类型的对象。如果未进行任何使用，则将返回一个空列表。

说明

如果对“InWork”状态的类型版本使用这些属性，则将出现异常。

修改以下程序代码：

```
//Determine the uses of a type version in a library
public static void GetDependenciesAndDependentsOfAVersion(LibraryTypeVersion
libTypeVersion)
{
    IList<LibraryTypeVersion> versionDependents = libTypeVersion.Dependents();
    IList<LibraryTypeVersion> versionDependencies = libTypeVersion.Dependencies();
}
```

程序代码

修改以下程序代码以确定类型版本所属的类型：

```
public static void GetParentTypeOfVersion(LibraryTypeVersion libTypeVersion)
{
    LibraryType parentType = libTypeVersion.TypeObject;
}
```

修改以下程序代码以确定包含类型版本实例的主副本：

```
public static void GetMasterCopiesContainingInstances(LibraryTypeVersion libTypeVersion)
{
    MasterCopyAssociation masterCopies = libTypeVersion.MasterCopiesContainingInstances;
}
```

修改以下程序代码以通过版本号查找单个类型版本：

```
public static void FindVersionInLibrary(ILibrary library, Guid versionGUID)
{
    LibraryTypeVersion libTypeVersionByVersionNumber = library.FindVersion(versionGUID);
}
```

7.10.12.10 访问实例

要求

- Openness 应用程序连接到 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已通过 Openness 应用程序打开一个项目。
  请参见 打开项目 (页 96)
- 您具有所需库的访问权限。
  请参见访问全局库 (页 118)。
- 您具有类型组的访问权限。
  请参见在库中访问文件夹 (页 123)。

应用

您可通过 Public API 接口访问类型版本的实例。

使用 `FindInstances(IInstanceSearchScope searchScope)` 方法找到某一类型版本的所有实例。

您可使用 `searchScope` 参数指定要搜索的项目区域。以下类实现 `IInstanceSearchScope` 接口，并可用于进行实例搜索：

- `PlcSoftware`
- `HmiTarget`

此方法会返回一个列表，其中包含 LibraryTypeInstanceInfo 类型的对象。如果未使用任何实例，则将返回一个空列表。

说明

HMI 用户数据类型和面板的实例通常链接到关联的类型版本。
所有其它对象的实例（例如程序块或画面）可链接到某个类型版本。

枚举一个类型版本的多个实例

修改以下程序代码：

```
//Enumerate  the instances of a type version in the project
LibraryTypeVersion version = ...;
PlcSoftware plcSoftware = ...;

IInstanceSearchScope searchScope = plcSoftware as IInstanceSearchScope;

if(searchScope==null)
{
    //No search possible
}

IList<LibraryTypeInstanceInfo> instanceInfos = version.FindInstances(searchScope);
IEnumerable<IEngineeringObject> instances = instanceInfos.Select(instanceInfo =>
instanceInfo.LibraryTypeInstance);
```

从某个实例导航至其所连接的版本对象

使用 LibraryTypeInstanceInfo 服务的 LibraryTypeVersion 属性从某个实例导航至其所连接的版本对象。

以下对象可提供 LibraryTypeInstanceInfo 服务：

- 块 FB
- 块 FC
- PLC 用户数据类型
- 画面
- VB 脚本

如果某个实例对象未连接至版本对象，则将不会提供“LibraryTypeInstanceInfo”服务。

```
FC fc = ...;
//Using LibraryTypeInstanceInfo service

LibraryTypeInstanceInfo instanceInfo = fc.GetService<LibraryTypeInstanceInfo>();
if(instanceInfo != null)
{
    LibraryTypeVersion connectedVersion = instanceInfo.LibraryTypeVersion;
    FC parentFc = instanceInfo.LibraryTypeInstance as FC; //parentFc == fc
}
```

程序代码

修改以下程序代码：

7.10.12.11 访问主副本

要求

- Openness 应用程序连接至 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)
- 您具有所需库的访问权限。
  请参见访问全局库 (页 118)
- 您具有主副本组的访问权限。
  请参见在库中访问文件夹 (页 123)

应用

公共 API 接口支持对全局库和项目库中的主副本进行访问：

- 创建主副本
- 枚举系统文件夹和用户定义文件夹中的主副本
- 重命名主副本
- 查询主副本中的信息
- 查询主副本中对象的信息

属性	数据类型	说明
Author	String	返回作者的姓名。
ContentDescriptions	MasterCopyContentDescriptionComposition	返回 MasterCopy 内容的说明。
CreationDate	DateTime	返回创建日期。
Name	String	返回主副本的名称。

程序代码

修改以下程序代码以枚举某一库系统文件夹中的所有主副本：

```
public static void EnumerateMasterCopiesInSystemFolder
(MasterCopySystemFolder masterCopySystemFolder)
{
    foreach (MasterCopy masterCopy in masterCopySystemFolder.MasterCopies)
    {
        //...
    }
}
```

修改以下程序代码以通过查找方法访问单个主副本：

```
...
    MasterCopySystemFolder systemFolder = projectLibrary.MasterCopyFolder;
    MasterCopyComposition mastercopies = systemFolder.MasterCopies;
    MasterCopy masterCopy = mastercopies.Find("Copy of ...");
...
```

修改以下程序代码以枚举主副本组和子组：

```
private static void EnumerateFolder(MasterCopyFolder folder)
{
    EnumerateMasterCopies(folder.MasterCopies);
    foreach (MasterCopyUserFolder subFolder in folder.Folders)
    {
        EnumerateFolder(subFolder); // recursion
    }
}
private static void EnumerateMasterCopies(MasterCopyComposition masterCopies)
{
    foreach (MasterCopy masterCopy in masterCopies)
    {
        ...
    }
}
```

修改以下程序代码以通过查找方法访问 MasterCopyUserFolder：

```
...
    MasterCopyUserFolderComposition userFolderComposition = ...
    MasterCopyUserFolder userFolder = userFolderComposition.Find("Name of user folder");
...
```

修改以下程序代码以重命名主副本：

```
//Setting the name property
var masterCopy = projectLibrary.MasterCopyFolder.MasterCopies.Find("SampleMasterCopyName");
masterCopy.Name = "NewMasterCopyName";

//Setting the name property dynamically
var masterCopy = projectLibrary.MasterCopyFolder.MasterCopies.Find("SampleMasterCopyName");
masterCopy.SetAttributes(new[] {new KeyValuePair<string,object>("Name",
"NewMasterCopyName")});
```

查询主副本中的信息

修改以下程序代码以获取某一主副本的信息：

```
public static void GetMasterCopyInformation(MasterCopy masterCopy)
{
    string author = masterCopy.Author;
    DateTime creationDate = masterCopy.CreationDate;
    string name = masterCopy.Name;
}
```

查询主副本中对象的信息

MasterCopy 对象包含名为 ContentDescriptions 的导航器，它是 MasterCopyContentDescriptions 的一个组合。

一个主副本可能包含多个对象。对于直接包含于主副本中的各个对象，MasterCopy 包含相应的 ContentDescriptions。如果主副本包含的某一文件夹中也包含某些项，则 MasterCopy 对象仅包含该文件夹的一个 ContentDescription。

```
MasterCopy multiObjectMasterCopy = ...;

//Using ContentDescriptions
MasterCopyContentDescriptionComposition masterCopyContentDescriptions =
multiObjectMasterCopy.ContentDescriptions;
MasterCopyContentDescription contentDescription= masterCopyContentDescriptions.First();

string name = contentDescription.ContentName;
Type type = contentDescription.ContentType;
```

7.10.12.12 创建库中项目的主副本

要求

- Openness 应用程序连接到 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 项目已经打开。
  请参见打开项目 (页 96)

应用

如果库为读写库，则可在目标位置创建 IMasterCopySource 的主副本。

```
MasterCopy
MasterCopyComposition.Create(Siemens.Engineering.Library.MasterCopies.IMasterCopySource
sourceObject);
```

在以下情况下，会出现工程异常：

- 目标位置为只读型
- 系统拒绝创建源的主副本

以下项被定义为 IMasterCopySources：

- Device - HW
- DeviceItem - HW
- DeviceUserGroup - HW
- CodeBlock - SW
- DataBlock - SW
- PlcBlockUserGroup - SW
- PlcTag - SW
- PlcTagTable - SW
- PlcTagTableUserGroup - SW
- PlcType - SW
- PlcTypeUserGroup - SW
- VBScript - HMI
- VBScriptUserFolder - HMI
- Screen - HMI
- ScreenTemplate - HMI
- ScreenTemplateUserFolder - HMI
- ScreenUserFolder - HMI
- Tag - HMI

- TagTable - HMI
- TagUserFolder - HMI

程序代码

使用以下程序代码：

```
// create a master copy from a code block in the project library
public static void Create(Project project, PlcSoftware plcSoftware)
{
    MasterCopySystemFolder masterCopyFolder = project.ProjectLibrary.MasterCopyFolder;
   CodeBlock block = plcSoftware.BlockGroup.Groups[0].Blocks.Find("Block_1") as CodeBlock;
    MasterCopy masterCopy = masterCopyFolder.MasterCopies.Create(block);
}
```

7.10.12.13 从主副本创建对象

要求

- Openness 应用程序连接至 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)
- PLC 未处于在线状态。

应用

Public API 接口支持在项目中使用主副本。可使用 CreateFrom 方法从项目库或全局库中的主副本在对象的组成中创建对象。

返回类型与相应组成的返回类型相对应。

CreateFrom 方法仅支持包含单个对象的主副本。如果调用操作的组成与源文件主副本不兼容（例如，源文件主副本包含 plc 变量表，而组成是 plc 块组成），将发生可恢复的异常。

支持以下组成：

- Siemens.Engineering.HW.DeviceComposition
- Siemens.Engineering.HW.DeviceItemComposition
- Siemens.Engineering.SW.Blocks.PlcBlockComposition

- Siemens.Engineering.SW.Tags.PlcTagTableComposition
- Siemens.Engineering.SW.Tags.PlcTagComposition
- Siemens.Engineering.SW.Types.PlcTypeComposition
- Siemens.Engineering.SW.TechnologicalObjects.TechnologicalInstanceDBComposition
- Siemens.Engineering.SW.Tags.PlcUserConstantComposition
- Siemens.Engineering.Hmi.Tag.TagTableComposition
- Siemens.Engineering.Hmi.Tag.TagComposition
- Siemens.Engineering.Hmi.Screen.ScreenComposition
- Siemens.Engineering.Hmi.Screen.ScreenTemplateComposition
- Siemens.Engineering.Hmi.RuntimeScripting.VBScriptComposition
- Siemens.Engineering.HW.SubnetComposition
- Siemens.Engineering.HW.DeviceUserGroupComposition
- Siemens.Engineering.SW.Blocks.PlcBlockUserGroupComposition
- Siemens.Engineering.SW.ExternalSources.PlcExternalSourceUserGroupComposition
- Siemens.Engineering.SW.Tags.PlcTagTableUserGroupComposition
- Siemens.Engineering.SW.Types.PlcTypeUserGroupComposition

程序代码：从主副本创建 PLC 块

修改以下程序代码，从库的主副本创建 PLC 块：

```
var plcSoftware = ...;
MasterCopy copyOfPlcBlock = ...;
PlcBlock plcSoftware.BlockGroup.Blocks.CreateFrom(copyOfPlcBlock);
```

程序代码：从主副本创建设备

修改以下程序代码，从库的主副本创建设备：

```
Project project = ...;
MasterCopy copyOfDevice = ...;
Device newDevice = project.Devices.CreateFrom(copyOfDevice);
```

程序代码：从主副本创建设备项

修改以下程序代码，从库的主副本创建设备项：

```
Device device = ...;
MasterCopy copyOfDeviceItem = ...;
DeviceItem newDeviceItem = device.DeviceItems.CreateFrom(copyOfDeviceItem);
```

程序代码：从主副本创建子网

修改以下程序代码，从库的主副本创建子网：

```
Project project = ...;
MasterCopy copyOfSubnet = ...;
Subnet newSubnet = project.Subnets.CreateFrom(copyOfSubnet);
```

程序代码：从主副本创建设备文件夹

修改以下程序代码，从库的主副本创建设备文件夹：

```
Project project = ...;
MasterCopy copyOfDeviceGroup = ...;
DeviceGroup newDeviceGroup= project.DeviceGroups.CreateFrom(copyOfDeviceGroup);
```

参见

访问主副本 (页 137)

7.10.12.14 复制主副本

要求

- Openness 应用程序连接至 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)

应用

Public API 接口支持在库内复制主副本，也可使用 CreateFrom 操作在多个库之间复制主副本。该操作会创建一个基于新对象的源主副本，并将它置于调用该操作的组合中。该操作试图创建具有与源主副本相同名称的新主副本。如果该名称不可用，系统将为新主副本提供一个新名称。随后即可返回新主副本。

如果调用“CreateFrom”操作的组合位于只读全局库中，会出现可恢复的异常情况。

程序代码

修改以下程序代码：

```
ProjectLibrary projectLibrary = ...;

MasterCopy copiedMasterCopy =
projectLibrary.MasterCopyFolder.MasterCopies.CreateFrom(sampleMasterCopy)
```

参见

访问主副本 (页 137)

7.10.12.15 确定过期类型实例

要求

- Openness 应用程序连接至 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)

- 您具有所需库的访问权限。
  请参见访问全局库 (页 118)
- 您具有类型文件夹的访问权限。
  请参见在库中访问文件夹 (页 123)。

应用

Public API 接口可用于确定已打开项目中的实例下属的类型版本。Public API 将为每个实例返回以下两种状态之一：

- 该实例引用了过期的类型版本。
- 该实例引用的是最新的类型版本。

确定版本时，以下规则适用：

- 版本的确定取决于相关库和要通过 Public API 接口打开的项目。
- 确定版本时，实例不会进行更新。

签名

使用 UpdateCheck 方法确定实例对应的类型版本：
`UpdateCheck(Project project, UpdateCheckMode updateCheckMode)`

参数	功能
`Project`	指定在其中确定实例对应类型版本的具体项目。
`UpdateCheckMode`	指定确定的版本： • `ReportOutOfDateOnly`：仅返回“已过期”(out of date) 类型的状态。 • `ReportOutOfDateAndUpToDate`： 返回“已过期”(out of date) 和“最新”(up to date) 类型的状态。

结果

确定版本时会从上至下扫描项目的所有设备。每个设备都将进行检查，以确定其组态数据是否包含指定库中类型版本的实例。UpdateCheck 方法将以层级结构的顺序返回版本检查的结果。

下表显示了一个版本检查的结果，其中使用了参数 `UpdateCheck.ReportOutOfDateAndUpToDate`:

Update check for: HMI_1				
	Update check for library element Screen_1 0.0.3			
		Out-of-date		
			\HMI_1\Screens	Screen_4 0.0.1
			\HMI_1\Screens	Screen_2 0.0.2
		Up-to-date		
			\HMI_1\Screens	Screen_1 0.0.3
			\HMI_1\Screens	Screen_10 0.0.3
Update check for: HMI_2				
	Update check of library element Screen_4 0.0.3			
		Out-of-date		
			\Screens folder1	Screen_02 0.0.1
			\Screens folder1	Screen_07 0.0.2
		Up-to-date		
			\Screens folder1	Screen_05 0.0.3
			\Screens folder1	Screen_08 0.0.3

程序代码

项目和全局库在处理更新检查时没有区别。

修改以下程序代码以确定项目中实例全局或项目库的类型版本：

```
public static void UpdateCheckOfGlobalLibrary(Project project, ILibrary library)
{
    // check for out of date instances and report only out of date instances in the returned
feedback
    UpdateCheckResult result = library.UpdateCheck(project,
UpdateCheckMode.ReportOutOfDateOnly);

    //Alternatively, check for out of date instances and report both out of date and up to
date instances in the returned feedback
    UpdateCheckResult alternateResult = library.UpdateCheck(project,
UpdateCheckMode.ReportOutOfDateAndUpToDate);

    //Show result
    RecursivelyWriteMessages(result.Messages);

    // Alternatively, show result and access single message parts
    RecursivelyWriteMessageParts(result.Messages);
}
```

修改以下程序代码以输出版本检查结果并逐条处理所有消息：

```
private static void RecursivelyWriteMessages (UpdateCheckResultMessageComposition
messages, string indent = "")
{
    indent += "\t";
    foreach (UpdateCheckResultMessage message in messages)
    {
        Console.WriteLine(indent + message.Description);
        RecursivelyWriteMessages(message.Messages, indent);
    }
}
```

修改以下程序代码以逐条访问版本检查结果中的消息部分：

```
private static void RecursivelyWriteMessageParts (UpdateCheckResultMessageComposition
messages, string indent= "")
{
    indent += "\t";
    foreach (UpdateCheckResultMessage message in messages)
    {
        Console.WriteLine(indent + "Full description: " + message.Description);
        foreach (KeyValuePair<string, string> messagePart in message.MessageParts)
        {
             // first level
             // part 1: device name
             // second level:
             // part 1: Name of the type in the global library
             // part 2: version of the type in the global library
             // third level:
             // part 1: title (either "Out-of-date" or "Up-to-date");
             // fourth level:
             // part 1: Path hierarchy to instance
             // part 2: Instance name in project
             // part 3: Version of the instance in the project
             Console.WriteLine(indent + "*Key: {0} Value:{1}", messagePart.Key,
messagePart.Value);
        }
     RecursivelyWriteMessageParts(message.Messages,indent);
    }
}
```

7.10.12.16 更新项目

要求

- Openness 应用程序连接到 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已通过 Openness 应用程序打开一个项目。
  请参见打开项目 (页 96)
- 您具有所需库的访问权限。
  请参见访问全局库 (页 118)。
- 您具有类型文件夹的访问权限。
  请参见在库中访问文件夹 (页 123)。

应用

Public API 接口可用于在项目的类型文件夹中更新所选类型的实例。

进行更新时，项目中所用的实例将根据最新发布的类型版本进行更新。如果要更新全局库中的实例，则应预先执行同步操作。

签名

使用 UpdateProject 方法可更新实例。

针对实现 LibraryTypes 接口的类使用以下调用：

```
void UpdateProject(IUpdateProjectScope updateProjectScope)
```

针对实现 ILibrary 接口的类使用以下调用：

```
void UpdateProject(IEnumerable<ILibraryTypeOrFolderSelection>
selectedTypesOrFolders, IEnumerable <IUpdateProjectScope>
updateProjectScope)
```

每个调用都输入在项目目录的日志文件中。

参数	功能
IEnumerable<ILibraryTypeOrFolderSelection> selectedTypesOrFolders	指定要同步的文件夹或类型或者待更新项目中的相应实例。
IUpdateProjectScope updateProjectScope IEnumerable <IUpdateProjectScope> updateProjectScope	在要更新其中实例的用途的项目中，指定项目对象。支持以下对象： • PlcSoftware • HmiTarget

程序代码

修改以下程序代码以更新类型文件夹内所选类型的实例：

```
private static void UpdateInstances(ILibrary myLibrary, LibraryTypeFolder
singleFolderContainingTypes, LibraryType singleType, PlcSoftware plcSoftware, HmiTarget
hmiTarget)
{
    //Update Instances of multiple types (subset of types and folders)
    IUpdateProjectScope[] updateProjectScopes =
    {
        plcSoftware as IUpdateProjectScope, hmiTarget as IUpdateProjectScope
    };
    myLibrary.UpdateProject(new ILibraryTypeOrFolderSelection[] {singleType,
singleFolderContainingTypes}, updateProjectScopes);
    //Update Instances of multiple types (all types in library)
    myLibrary.UpdateProject(new[] {myLibrary.TypeFolder}, updateProjectScopes);
}
```

7.10.12.17 更新库

要求

- Openness 应用程序连接到 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已通过 Openness 应用程序打开一个项目。
  请参见打开项目 (页 96)
- 您具有所需库的访问权限。
  请参见访问全局库 (页 118)。
- 您具有类型文件夹的访问权限。
  请参见在库中访问文件夹 (页 123)。

应用

Public API 接口支持在项目库中进行以下更新：

- 同步两个库中的所选类型。

不可在执行同步时修改文件夹结构。待更新的类型通过 GUID 进行识别，然后进行更新：

- 如果库中的类型包含待更新的库中所缺少的类型版本，则复制该类型版本。
- 如果库中的类型包含具有不同 GUID 的类型版本，将中止更新过程，并会触发 Exception。

签名

使用 UpdateLibrary 方法可同步类型版本。

针对实现 LibraryTypes 接口的类使用以下调用：

```
void UpdateLibrary(ILibrary targetLibrary)
```

针对实现 ILibrary 接口的类使用以下调用：

```
void UpdateLibrary(IEnumerable<LibraryTypeOrFolderSelection>
selectedTypesOrFolders, ILibrary targetLibrary)
```

参数	功能
IEnumerable<ILibraryTypeOrFolderSelection> selectedTypesOrFolders	指定要同步的文件夹或类型或者项目中待更新的相应实例。
ILibrary targetLibrary	指定将与库进行内容同步的库。 如果源库和目标库完全相同，则会触发异常。

程序代码

修改以下程序代码以同步项目库与全局库的类型：

```
sourceType.UpdateLibrary(projectLibrary);
```

修改以下程序代码以同步全局库与项目库之间类型文件夹内的所选类型：

```
globalLibrary.UpdateLibrary(new[]{globalLibrary.TypeFolder}, projectLibrary);
```

7.10.12.18 删除库内容

要求

- Openness 应用程序连接到 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已通过 Openness 应用程序打开一个项目。
  请参见 打开项目 (页 96)
- 您具有所需库的访问权限。
  请参见访问全局库 (页 118)。
- 您具有类型文件夹的访问权限。
  请参见在库中访问文件夹 (页 123)。

应用

您可通过 Public API 接口删除以下项目库内容：

- 类型
- 类型版本
- 用户定义的类型文件夹
- 主副本
- 用户定义的主副本文件夹

说明

删除类型和用户定义的类型文件夹

如要删除一个类型或用户定义的文件夹类型，必须符合“版本删除规则”。空类型文件夹可随时删除。

说明

版本删除规则

您只能删除“已提交”(Committed) 状态的版本。删除版本时，以下规则同样适用：

- 如果“InWork”状态的新版本刚在“Committed”状态的版本基础上创建，则只有在新版本已被放弃或获得“Committed”状态时，您才能删除“Committed”状态的版本。
- 如果一个类型只有一个版本，则该类型也会被删除。
- 如果版本 A 取决于另一类型的版本 B，则首先删除版本 A，然后再删除版本 B。
- 如果版本 A 有多个实例，则只有在删除全部实例后，才能删除版本 A。如果某个实例还包含在主副本中，则主副本也将被删除。

程序代码

修改以下程序代码以删除类型或用户自定义类型文件夹：

```
public static void DeleteMultipleTypesOrTypeUserFolders(ILibrary library)
{
    LibraryType t1 = library.TypeFolder.Types.Find("type1");
    LibraryType t2 = library.TypeFolder.Types.Find("type2");
    LibraryTypeUserFolder f1 = library.TypeFolder.Folders.Find("folder1");
    t1.Delete();
    t2.Delete();
    f1.Delete();
}
```

修改以下程序代码以删除单个类型或用户自定义类型文件夹：

```
public static void DeleteSingleTypeOrTypeUserFolder(ILibrary library)
{
    //Delete a single type
    LibraryType t1 = library.TypeFolder.Types.Find("type1");
    t1.Delete();

    //Delete a single folder
    LibraryTypeFolder parentFolder = library.TypeFolder;
    LibraryTypeUserFolder f1 = parentFolder.Folders.Find("folder1");
    f1.Delete();
}
```

修改以下程序代码以删除版本：

```
public static void DeleteVersion(ILibrary library)
{
    LibraryType singleType = library.TypeFolder.Types.Find("type1");
    LibraryTypeVersion version1 = singleType.Versions.Find(new System.Version(1, 0, 0));
    version1.Delete();
}
```

修改以下程序代码以删除主副本或用户自定义主副本文件夹：

```
public static void DeleteMasterCopies(ILibrary library)
{
    // Delete master copy
    MasterCopy masterCopy = library.MasterCopyFolder.MasterCopies.Find("myMasterCopy");
    masterCopy.Delete();

    // Delete master copy user folder
    MasterCopyUserFolder masterUserFolder =
library.MasterCopyFolder.Folders.Find("myFolder");
    masterUserFolder.Delete();
}
```

参见

访问主副本 (页 137)

7.10.13 删除项目图形

要求

- Openness 应用程序连接到 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)

程序代码

修改以下程序代码以删除项目图形：

```
//Deletes a single project graphic entry
public static void DeletesSingleProjectGraphicEntry(Project project)
{
    MultiLingualGraphicComposition graphicsAggregation = project.Graphics;
    MultiLingualGraphic graphic = graphicsAggregation.Find("Graphic XYZ");
    graphic.Delete();
}
```

7.10.14 属性、导航器、操作和服务的自述支持

应用

在 Openness 中，公共 API 的各个 IEngineeringServiceProvider 均介绍了其潜在调用功能。

IEngineeringObject 的自述支持

方法名称	返回值
GetCompositionInfos	返回 EngineeringCompositionInfo 对象的集合，其中介绍了这类对象的各种组合。下面对 EngineeringCompositionInfo 进行了介绍。
GetAttributeInfos	返回 EngineeringAttributeInfo 对象的集合，其中介绍了这类对象的各种属性。下面对 EngineeringAttributeInfo 进行了介绍。
GetInvocationInfos	返回 EngineeringInvocationInfo 对象的集合，其中介绍了这类对象的各种操作。下面对 EngineeringInvocationInfo 进行了介绍。

IEngineeringServiceProvider 的自述支持

方法名称	返回值
GetServiceInfos	返回 EngineeringServiceInfo 对象的集合，其中介绍了这类对象的各种服务。下面对 EngineeringServiceInfo 进行了介绍。

类 EngineeringCompositionInfo

属性名称	返回值
Name	组合名称

类 EngineeringAttributeInfo

属性名称	返回值
AccessMode	属性所支持的访问等级。该属性为可组合属性，下文对其进行了详细介绍。
Name	属性名称。

类 EngineeringInvocationInfo

属性名称	返回值
Name	操作名称。
ParameterInfos	EngineeringInvocationParameterInfo 对象的集合，其中介绍了操作可能需要的所有参数。下面对 EngineeringInvocationParameterInfo 进行了介绍。

类 EngineeringServiceInfo

属性名称	返回值
Type	用作 System.Type 对象的服务类型。

Enum AccessMode

枚举值	返回值
None	此为无效选项。
Read	可读取属性。
Write	可写入属性。

类 EngineeringInvocationParameterInfo

属性名称	返回值
Name	参数的名称。
Type	用作 System.Type 对象的参数类型

程序代码

AccessMode 为标志枚举，其值可组合成以下程序代码的形式：

```
EngineeringAttributeAccessMode value = EngineeringAttributeAccessMode.Read|
EngineeringAttributeAccessMode.Write;
```

修改以下程序代码以查找 IEngineeringObject 的全部属性和更改这些属性的访问模式。

```
...
IEngineeringObject engineeringObject = ...;
IList<EngineeringAttributeInfo> attributeInfos = engineeringObject.GetAttributeInfos();
foreach(EngineeringAttributeInfo attributeInfo in attributeInfos)
{
    switch (attributeInfo.AccessMode)
        {
        case EngineeringAttributeAccessMode.Read:
            ...
            break;
        case EngineeringAttributeAccessMode.Write:
            ...
            break;
        case EngineeringAttributeAccessMode.Read|EngineeringAttributeAccessMode.Write:
            ...
            break;
        }
}
...
```

7.11 访问设备、网络和连接的功能

7.11.1 打开“设备和网络”编辑器

要求

- Openness 应用程序连接到 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)

应用

可采用以下两种方法之一，通过 API 接口打开“设备和网络”编辑器：

- `ShowHwEditor(View.Topology 或 View.Network 或 View.Device)`：从项目打开“设备和网络”编辑器。
- `ShowInEditor(View.Topology 或 View.Network 或 View.Device)`：在“设备和网络”编辑器中显示指定的设备。

使用 `View` 参数定义打开编辑器时显示的视图：

- `View.Topology`
- `View.Network`
- `View.Device`

程序代码

修改以下程序代码以打开“设备和网络”(Devices & networks) 编辑器：

```
// Open topology view from project
private static void OpenEditorDevicesAndNetworksFromProject(Project project)
{
    project.ShowHwEditor(Siemens.Engineering.HW.View.Topology);
}
```

修改以下程序代码以打开设备的“设备和网络”(Devices & networks) 编辑器：

```
// Open topology view for given device
private static void OpenEditorDevicesAndNetworksFromDevice(Device device)
{
    device.ShowInEditor(Siemens.Engineering.HW.View.Topology);
}
```

参见

导入组态数据 (页 345)

7.11.2 查询 PLC 和 HMI 目标

要求

- Openness 应用程序连接到 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)

应用

您可以决定软件基础可在公共 API 中用作 PLC 目标 (PlcSoftware) 还是 HMI 目标。

程序代码：PLC 目标

修改以下程序代码以确定某一设备项是否可用作 PLC 目标：

```
// Returns PlcSoftware
private PlcSoftware GetPlcSoftware(Device device)
{
    DeviceItemComposition deviceItemComposition = device.DeviceItems;
    foreach (DeviceItem deviceItem in deviceItemComposition)
    {
        SoftwareContainer softwareContainer = deviceItem.GetService<SoftwareContainer>();
        if (softwareContainer != null)
        {
            Software softwareBase = softwareContainer.Software;
            PlcSoftware plcSoftware = softwareBase as PlcSoftware;
            return plcSoftware;
        }
    }
    return null;
 }
```

程序代码：HMI 目标

修改以下程序代码以确定某一设备项是否可用作 HMI 目标：

```
//Checks whether a device is of type hmitarget
private HmiTarget GetHmiTarget(Device device)
{
    DeviceItemComposition deviceItemComposition = device.DeviceItems;
    foreach (DeviceItem deviceItem in deviceItemComposition)
    {
        SoftwareContainer softwareContainer = deviceItem.GetService<SoftwareContainer>();
        if (softwareContainer != null)
        {
            Software softwareBase = softwareContainer.Software;
            HmiTarget hmiTarget = softwareBase as HmiTarget;
            return hmiTarget;
        }
    }
    return null;
}
```

参见

枚举设备 (页 215)

7.11.3 端口到端口连接的可组态属性

要求

- Openness 应用程序连接到 TIA Portal。
  请参见打开项目 (页 96)
- 项目已经打开。
  请参见打开项目 (页 96)

应用

端口互连的属性位于端口设备项。通过 Openness 进行的属性读写访问与在 UI 中的访问相同。

端口接口设置

端口接口设置具有以下属性：

属性名称	数据类型	可写入	访问	说明
MediumAttachmentType	MediumAttachmentType	r/o	动态属性	
CableName	CableName	r/w	动态属性	
AlternativePartnerPorts	Boolean	r/w	动态属性	仅在支持工具转换器功能时可用，例如在 CPU1516 上。
SignalDelaySelection	SignalDelaySelection	r/w	动态属性	
CableLength	CableLength	r/w	动态属性	
SignalDelayTime	Double	r/w	动态属性	

为属性 MediumAttachmentType 提供以下 ENUM 值：

值	说明
MediumAttachmentType.None	连接类型无法确定。
MediumAttachmentType.Copper	连接类型为铜。
MediumAttachmentType.FibreOptic	连接类型为光纤。

为属性 Cablename 提供以下 ENUM 值：

值	说明
CableName.None	未指定电缆名称
CableName.FO_Standard_Cable_9	FO 标准电缆 GP (9 µm)
CableName.Flexible_FO_Cable_9	柔性 FO 电缆 (9 µm)
CableName.FO_Standard_Cable_GP_50	FO 标准电缆 GP (50 µm)
CableName.FO_Trailing_Cable_GP	FO 拖曳式电缆/GP
CableName.FO_Ground_Cable	FO 接地电缆
CableName.FO_Standard_Cable_62_5	FO 标准电缆 (62.5 µm)
CableName.Flexible_FO_Cable_62_5	柔性 FO 电缆 (62.5 µm)
CableName.POF_Standard_Cable_GP	POF 标准电缆 GP
CableName.POF_Trailing_Cable	POF 拖曳式电缆
CableName.PCF_Standard_Cable_GP	PCF 标准电缆 GP
CableName.PCF_Trailing_Cable_GP	PCF 拖曳式电缆/GP
CableName.GI_POF_Standard_Cable	GI-POF 标准电缆
CableName.GI_POF_Trailing_Cable	GI-POF 拖曳式电缆
CableName.GI_PCF_Standard_Cable	GI-PCF 标准电缆
CableName.GI_PCF_Trailing_Cable	GI-PCF 拖曳式电缆

为属性 SignalDelaySelection 提供以下 ENUM 值：

值	说明
SignalDelaySelection.None	
SignalDelaySelection.CableLength	CableLength 用于定义信号延迟。
SignalDelaySelection.SignalDelayTime	SignalDelayTime 用于定义信号延迟。

为属性 CableLength 提供以下 ENUM 值：

值	说明
CableLength.None	未指定电缆长度。
CableLength.Length20m	电缆长度为 20m。
CableLength.Length50m	电缆长度为 50m。
CableLength.Length100m	电缆长度为 100m。
CableLength.Length1000m	电缆长度为 1000m。
CableLength.Length3000m	电缆长度为 3000m。

端口选项

端口选项具有以下属性：

属性名称	数据类型	可写入	访问
PortActivation	bool	r/w	动态属性
TransmissionRateAndDuplex	TransmissionRateAndDuplex	r/w	动态属性
PortMonitoring	bool	r/w	动态属性
TransmissionRateAutoNegotiation	bool	r/w	动态属性
EndOfDetectionOfAccessibleDevices	bool	r/w	动态属性
EndOfTopologyDiscovery	bool	r/w	动态属性
EndOfSyncDomain	bool	r/w	动态属性

为属性 TransmissionRateAndDuplex 提供以下 ENUM 值：

值	说明
TransmissionRateAndDuplex.None	
TransmissionRateAndDuplex.Automatic	自动
TransmissionRateAndDuplex.AUI10Mbps	10 Mbps AUI
TransmissionRateAndDuplex.TP10MbpsHalfDuplex	TP 10 Mbps 半双工
TransmissionRateAndDuplex.TP10MbpsFullDuplex	TP 10 Mbps 全双工
TransmissionRateAndDuplex.AsyncFiber10MbpsHalfDuplex	异步光纤 10 Mbps 半双工模式
TransmissionRateAndDuplex.AsyncFiber10MbpsFullDuplex	异步光纤 10 Mbps 全双工模式
TransmissionRateAndDuplex.TP100MbpsHalfDuplex	TP 100 Mbps 半双工
TransmissionRateAndDuplex.TP100MbpsFullDuplex	TP 100 Mbps 全双工
TransmissionRateAndDuplex.FO100MbpsFullDuplex	FO 100 Mbps 全双工
TransmissionRateAndDuplex.X1000MbpsFullDuplex	X1000 Mbps 全双工
TransmissionRateAndDuplex.FO1000MbpsFullDuplexLD	FO 1000 Mbps 全双工 LD
TransmissionRateAndDuplex.FO1000MbpsFullDuplex	FO 1000 Mbps 全双工
TransmissionRateAndDuplex.TP1000MbpsFullDuplex	TP 1000 Mbps 全双工
TransmissionRateAndDuplex.FO10000MbpsFullDuplex	FO 10000 Mbps 全双工

值	说明
TransmissionRateAndDuplex.FO100MbpsFullDuplexLD	FO 100 Mbps 全双工 LD
TransmissionRateAndDuplex.POFPCF100MbpsFullDuplexLD	POF/PCF 100 Mbps 全双工

参见

连接到 TIA Portal (页 69)

7.11.4 地址对象的访问属性

要求

- Openness 应用程序连接至 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)
- 对于写入访问，PLC 已处于离线状态。

应用

可以使用 Public API 接口来获取或设置地址对象属性。

还可为 OB 指定当前过程映像。

可访问以下属性：

属性名称	数据类型	可写入	访问	说明
IsochronousMode	BOOL	r/w	动态属性	激活/禁用等时模式
ProcessImage	Int32	r/w	动态属性	设置/获取过程映像分区号。
InterruptObNumber	Int64	r/w	动态属性	设置/获取中断组织块编号（仅传统控制器）
StartAddress	Int32	r/w	模型化属性	设置/获取新的 StartAddress 值。

限制

- 属性 StartAddress
  - 设置 StartAddress 可能隐式改变相同模块上相对 IO 类型的 StartAddress。更改输入地址会更改输出地址。
  - 并非所有设备均支持写访问。
  - Openness 中不支持压缩地址
  - 通过 Openness 更改地址不会重新连接已分配的标签。
- 属性 InterruptObNumber
  - 只有使用 S7-300 或 S7-400 控制器时才能在设置中访问。S7-400 控制器支持写访问。

程序代码：获取或设置地址对象的属性

修改以下程序代码以访问地址对象的等时模式：

```
Address address= ...;

  // read attribute
bool attributeValue = (bool)address.GetAttribute("IsochronousMode");

 // write attribute
address.SetAttribute("IsochronousMode", true);
```

修改以下程序代码以访问地址对象的 ProcessImage 属性：

```
Address address= ...;

  // read attribute
int attributeValue = (int)address.GetAttribute("ProcessImage");

 // write attribute
address.SetAttribute("ProcessImage", 7);
```

修改以下程序代码以访问地址对象的 InterruptObNumber 属性：

```
Address address= ...;

  // read attribute
long attributeValue = (long)address.GetAttribute("InterrruptObNumber");

 // write attribute
address.SetAttribute("InterrruptObNumber", 42L);

//default value = 40
```

修改以下程序代码以访问地址对象的 StartAddress 属性：

```
Address address= ...;

  // read attribute
int attributeValue = (int)address.GetAttribute("StartAddress");

 // write attribute
address.StartAddress = IntValueStartAddress;
```

程序代码：为 OB 指定当前过程映像。

修改以下程序代码以便为 OB 指定当前过程映像。

```
OB obX =…
Address address= ...;

// assign PIP 5 to obX

address.SetAttribute("ProcessImage", 5);

try
{
        address.AssignProcessImageToOrganizationBlock(obX);
}    catch(RecoverableException e) {
       Console.WriteLine(e.Message);
}

// remove this PIP-OB assignment

try
{
        address.AssignProcessImageToOrganizationBlock(null);
}    catch(RecoverableException e) {
       Console.WriteLine(e.Message);
}
```

7.11.5 访问模块的通道

要求

- Openness 应用程序连接至 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)

应用

对于模拟量输入模块等信号模块，通常单个模块内具有多个通道。通常，通道可多次提供相似功能，例如，一个四通道的模拟量输入模块可同时测量四个电压值。

要访问某一模块的所有通道，请使用设备项的通道属性。

程序代码：通道属性

修改以下程序代码以访问通道的属性：

```
DeviceItem aiModule = ...
ChannelComposition channels = aiModule.Channels;
foreach (Channel channel in channels)
{
    ... // Work with the channel
}
```

程序代码：标识属性

修改以下程序代码以获取每个通道的标识属性：

```
Channel channel = ...
int channelNumber = channel.Number;
ChannelType type = channel.Type;
ChannelIoType ioType = channel.IoType;
```

程序代码：访问单通道

修改以下程序代码以使用标识属性直接访问通道：

```
DeviceItem aiModule = ...
Channel channel = aiModule.Channels.Find(ChannelType.Analog, ChannelIoType.Input, 0);
... // Work with the channel
```

通道类型

值	说明
ChannelType.None	无效通道类型。
ChannelType.Analog	模拟量通道类型。
ChannelType.Digital	数字量通道类型。
ChannelType.Technology	工艺通道类型。

通道 IO 类型

值	说明
ChannelIOType.None	无效的通道 IO 类型。
ChannelIOType.Input	输入通道。
ChannelIOType.Output	输出通道。
ChannelIOType.Complex	复杂 IO 类型，例如工艺通道。

7.11.6 网络功能

7.11.6.1 创建子网

要求

- Openness 应用程序连接至 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)

应用

可以用两种不同方式创建子网：

- 创建连接至接口的子网：子网创建处的接口的类型决定了子网的类型
- 创建未连接至接口的子网。

程序代码：创建连接至接口的子网

修改以下程序代码以创建子网：

```
Node node = ...;
Subnet subnet = node.CreateAndConnectToSubnet("NameOfSubnet");
```

使用下列标识符：

- System:Subnet.Ethernet
- System:Subnet.Profibus

- System:Subnet.Mpi
- System:Subnet.Asi

程序代码：创建未连接至接口的子网

修改以下程序代码以创建子网：

```
Project project = ...;
SubnetComposition subnets = project.Subnets;

Subnet newSubnet = subnets.Create("System:Subnet.Ethernet", "NewSubnet");
```

可使用下列标识符：

- System:Subnet.Ethernet
- System:Subnet.Profibus
- System:Subnet.Mpi
- System:Subnet.Asi

7.11.6.2 访问子网

要求

- Openness 应用程序连接至 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)

应用

为实现某些网络相关的功能，例如，将接口分配给某一子网，必须访问项目中的子网。通常，子网会直接在项目级别聚合。

程序代码：访问某一项目的所有子网

修改以下程序代码以访问某一项目的所有子网（内部子网除外）：

```
Project project = ...
foreach (Subnet net in project.Subnets)
{
    ... // Work with the subnet
}
```

程序代码：访问特定子网

修改以下程序代码以通过名称访问特定子网：

```
Project project = ...
Subnet net = project.Subnets.Find("PROFIBUS_1");
{
    ... // Work with the subnet
}
```

子网的属性

子网具有以下属性：

```
Subnet net = ...;
string name = net.Name;
NetType type = net.NetType;
```

网络类型

值	说明
NetType.Unknown	网络类型未知。
NetType.Ethernet	网络类型为以太网。
NetType.Profibus	网络类型为 Profibus。
NetType.Mpi	网络类型为 MPI。
NetType.ProfibusIntegrated	网络类型为集成 Profibus。
NetType.Asi	网络类型为 ASi。
NetType.PcInternal	网络类型为 PC 内网。

值	说明
NetType.Ptp	网络类型为 PtP。
NetType.Wan	网络类型为广域网

7.11.6.3 访问内部子网

要求

- Openness 应用程序连接至 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)

应用

如果一个设备项能够构建一个子网，则该设备项可提供附加功能“子网所有者”。若要访问此附加功能，必须使设备项的特定服务。

程序代码：获取子网所有者角色

修改以下程序代码以获取子网所有者角色：

```
SubnetOwner subnetOwner =
((IEngineeringServiceProvider)deviceItem).GetService<SubnetOwner>();
if (subnetOwner != null)
{
    // work with the role
}
```

程序代码：子网所有者的属性

修改以下程序代码以访问子网所有者的子网：

```
foreach(Subnet subnet in subnetOwner.Subnets)
{
    Subnet interalSubnet = subnet;
}
```

7.11.6.4 获取子网的类型标识符

要求

- Openness 应用程序连接至 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)

应用

属性 `TypeIdentifier` 用于识别子网。`TypeIdentifier` 字符串包含多个部分：
`<TypeIdentifierType>:<SystemIdentifier>`

`TypeIdentifierType` 可能具有的值为：

- `System`

SystemIdentifier

子网类型	系统标识符
PROFIBUS	Subnet.Profibus
MPI	Subnet.Mpi
工业以太网	Subnet.Ethernet
ASI	Subnet.Asi
Ptp	Subnet.Ptp
集成 PROFIBUS	Subnet.ProfibusIntegrated
PC 内网	*null*

程序代码

修改以下程序代码以获取用户可管理并可单独创建的 GSD 对象的类型标识符：

```
Subnet subnet = ...;
string typeIdentifier = subnet.TypeIdentifier;
```

7.11.6.5 访问子网属性

要求

- Openness 应用程序连接至 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)

应用

子网提供了某些可供读取和/或写入的强制属性。这些属性仅当其在 UI 中可用时才可用。通常仅当属性可以由用户在 UI 中修改时才允许写入。具体取决于子网类型。例如，只有当 IsochronousMode 为“真”且 DpCycleMinTimeAutoCalculation 为“假”时用户才能设置 DpCycleTime。

ASI 类型子网的属性

属性	数据类型	可写	访问	说明
Name	string	读取/写入		子网的名称。
NetType	NetType	读取		子网的类型。
SubnetId	string	读取	动态	子网的唯一标识。S7 子网 ID 由两个数字组成，中间以连字符连接。一个数字表示项目，另一数字表示子网。例如，4493-1。

以太网类型子网的属性

属性	数据类型	可写	访问	说明
Name	string	读取/写入		子网的名称。
NetType	NetType	读取		子网的类型。

属性	数据类型	可写	访问	说明
SubnetId	string	读取/写入	动态	子网的唯一标识。S7 子网 ID 由两个数字组成，中间以连字符连接。一个数字表示项目，另一数字表示子网。例如，4493-1。
DefaultSubnet	bool	读取/写入	动态	如果子网是默认子网，则为“真”。一个项目中最多只有一个默认子网。

MPI 类型子网的属性

属性	数据类型	可写	访问	说明
Name	string	读取/写入		子网的名称。
NetType	NetType	读取		子网的类型。
SubnetId	string	读取/写入	动态	子网的唯一标识。S7 子网 ID 由两个数字组成，中间以连字符连接。一个数字表示项目，另一数字表示子网。例如，4493-1。
HighestAddress	int	读取/写入	动态	子网中的最高 MPI 地址。
TransmissionSpeed	BaudRate	读取/写入	动态	如果子网是默认子网，则为“真”。一个项目中最多只有一个默认子网。

PC 内部类型子网的属性

属性	数据类型	可写	访问	说明
Name	string	读取		子网的名称。
NetType	NetType	读取		子网的类型。
SubnetId	string	读取	动态	子网的唯一标识。S7 子网 ID 由两个数字组成，中间以连字符连接。一个数字表示项目，另一数字表示子网。例如，4493-1。

PROFIBUS 类型子网的属性

属性	数据类型	可写	访问	说明
Name	string	读取/写入		子网的名称。
NetType	NetType	读取		子网的类型。
SubnetId	string	读取/写入	动态	子网的唯一标识。S7 子网 ID 由两个数字组成，中间以连字符连接。一个数字表示项目，另一数字表示子网。例如，4493-1。
HighestAddress	int	读取/写入	动态	子网中的最高 PROFIBUS 地址。
TransmissionSpeed	BaudRate	读取/写入	动态	如果子网是默认子网，则为“真”。一个项目中最多只有一个默认子网。
BusProfile	BusProfile	读取/写入	动态	PROFIBUS 配置文件。
PbCableConfiguration	bool	读取/写入	动态	如果为“真”，则启用其它 PROFIBUS 网络设置
PbRepeaterCount	int	读取/写入	动态	铜缆的中继器数量
PbCopperCableLength	double	读取/写入	动态	铜缆的长度
PbOpticalComponentCount	int	读取/写入	动态	光纤电缆的 OLM 和 OBT 的数量。
PbOpticalCableLength	double	读取/写入	动态	PROFIBUS 网络的光纤电缆长度，单位为 km。
PbOpticalRing	bool	读取/写入	动态	如果根据光纤环网调整总线参数，则为“真”
PbOlmP12	bool	读取/写入	动态	如果启用了 OLM/P12 用于总线参数计算，则为“真”
PbOlmG12	bool	读取/写入	动态	如果启用了 OLM/G12 用于总线参数计算，则为“真”
PbOlmG12Eec	bool	读取/写入	动态	如果启用了 OLM/G12-EEC 用于总线参数计算，则为“真”
PbOlmG121300	bool	读取/写入	动态	如果启用了 OLM/G12-1300 用于总线参数计算，则为“真”

属性	数据类型	可写	访问	说明
PbAdditionalNetworkDevices	bool	读取/写入	动态	如果在计算总线时间时将项目中不存在的其它总线设备考虑其中，则为“真”。
PbAdditionalDpMaster	int	读取/写入	动态	未组态的 DP 主站的数量。
PbTotalDpMaster	int	读取	动态	DP 主站总数
PbAdditionalPassiveDevice	int	读取/写入	动态	未组态的 DP 从站或被动设备的数量。
PbTotalPassiveDevice	int	读取	动态	DP 从站或被动设备的总数。
PbAdditionalActiveDevice	int	读取/写入	动态	带 FDL/FMS/S/ 通信负载的未组态的主动设备的数量。
PbTotalActiveDevice	int	读取	动态	带 FDL/FMS/S/ 通信负载的主动设备总数。
PbAdditionalCommunicationLoad	CommunicationLoad	读取/写入	动态	通信负载的粗略量化
PbDirectDateExchange	bool	读取/写入	动态	优化直接数据交换。
PbMinimizeTslotForSlaveFailure	bool	读取/写入	动态	最小化从站故障的时间分配。
PbOptimizeCableConfiguration	bool	读取/写入	动态	优化电缆组态。
PbCyclicDistribution	bool	读取/写入	动态	如果启用总线参数的总线参数的周期性分配，则为“真”。
PbTslotInit	int	读取/写入	动态	Tslot 的默认值。
PbTslot	int	读取	动态	等待接收的时间（插槽时间）
PbMinTsdr	int	读取/写入	动态	协议处理的最短时间
PbMaxTsdr	int	读取/写入	动态	协议处理的最长时间
PbTid1	int	读取	动态	空闲时间 1
PbTid2	int	读取	动态	空闲时间 2
PbTrdy	int	读取	动态	准备时间
PbTset	int	读取/写入	动态	设置时间

属性	数据类型	可写	访问	说明
PbTqui	int	读取/写入	动态	调节器的稳态时间
PbTtr	int64	读取/写入	动态	t_Bit 中的 Ttr 值
PbTtrTypical	int64	读取	动态	总线上的平均响应时间
PbWatchdog	int64	读取/写入	动态	看门狗
PbGapFactor	int	读取/写入	动态	更新因数的间隔
PbRetryLimit	int	读取/写入	动态	最大重试次数
IsochronousMode	bool	读取/写入	动态	如果启用了恒定总线循环时间，则为“真”。
PbAdditionalPassivDeviceForIsochronousMode	int	读取/写入	动态	未在此网络视图中进行组态的附加 OP/PG/TD 等的数量。
PbTotalPassivDeviceForIsochronousMode	int	读取	动态	组态和未组态设备（如 OP/PG/TD 等）的总数。
DpCycleMinTimeAutoCalculation	bool	读取/写入	动态	如果启用了自动计算和设置最短 DP 循环时间，则为“真”。
DpCycleTime	double	读取/写入	动态	DP 循环时间。
IsochronousTiToAutoCalculation	bool	读取/写入	动态	如果启用了自动计算和设置 IsochronTi 和 IsochronTo 值，则为“真”。
IsochronousTi	double	读取/写入	动态	时间 Ti（在过程值中读取）
IsochronousTo	double	读取/写入	动态	时间 To（输出过程值）

PROFIBUS Integrated 类型子网的属性

属性	数据类型	可写	访问	说明
Name	string	读取/写入		子网的名称。
NetType	NetType	读取		子网的类型。

属性	数据类型	可写	访问	说明
SubnetId	string	读取/写入	动态	子网的唯一标识。S7 子网 ID 由两个数字组成，中间以连字符连接。一个数字表示项目，另一数字表示子网。例如，4493-1。
IsochronousMode	bool	读取	动态	已启用恒定总线循环时间。
DpCycleMinTimeAutoCalculation	bool	读取/写入	动态	如果启用了自动计算和设置最短 DP 循环时间，则为“真”。
DpCycleTime	double	读取/写入	动态	DP 循环时间。
IsochronousTiToAutoCalculation	bool	读取/写入	动态	如果启用了自动计算和设置 IsochronTi 和 IsochronTo 值，则为“真”。
IsochronousTi	double	读取/写入	动态	时间 Ti（在过程值中读取）
IsochronousTo	double	读取/写入	动态	时间 To（输出过程值）

程序代码

修改以下程序代码以获取或设置子网的属性：

```
Subnet subnet = ...;

string nameValue = subnet.Name;
NetType nodeType = (NetType)subnet.NetType;
string subnetId = ((IEngineeringObject)subnet).GetAttribute("SubnetId");

subnet.Name = "NewName";
subnet.SetAttribute("Name", "NewName");

bool isDefaultSubnet = ((IEngineeringObject)subnet).GetAttribute("DefaultSubnet");
```

波特率

值	说明
BaudRate.None	波特率未知。
BaudRate.Baud9600	9.6 kBaud
BaudRate.Baud19200	19.2 kBaud

值	说明
BaudRate.Baud45450	45.45 kBaud
BaudRate.Baud93700	93.75 kBaud
BaudRate.Baud187500	187.5 kBaud
BaudRate.Baud500000	500 kBaud
BaudRate.Baud1500000	1.5 MBaud
BaudRate.Baud3000000	3 MBaud
BaudRate.Baud6000000	6 MBaud
BaudRate.Baud12000000	12 MBaud

总线配置文件

值	说明
BusProfile.None	总线配置文件未知。
BusProfile.DP	网络类型为 DP。
BusProfile.Standard	网络类型为标准类型。
BusProfile.Universal	网络类型为通用类型。
BusProfile.UserDefined	网络类型为用户自定义类型。

通信负载

值	说明
CommunicationLoad.None	无有效通信负载。
CommunicationLoad.Low	通常用于 DP，除 DP 之外没有大的数据通信。
CommunicationLoad.Medium	通常用于当 DP 具有严格时间要求时具有 DP 和其它通信服务的混合式操作（如 S7 通信），以及用于一般量的非周期性通信。
CommunicationLoad.High	用于当 DP 具有宽松的时间要求时具有 DP 和其它通信服务的混合式操作（如 S7 通信），以及用于量比较大的非周期性通信。

7.11.6.6 删除全局子网

要求

- Openness 应用程序连接至 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)

程序代码

修改以下程序代码以删除项目中的全局子网：

```
Project project = ...;
SubnetComposition subnets = projects.Subnets;

// delete subnet
Subnet subnetToDelete = ...;
subnetToDelete.Delete();
```

7.11.6.7 枚举子网的所有参与者

要求

- Openness 应用程序连接至 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)

应用

枚举子网上的所有参与者。

程序代码

修改以下程序代码以枚举子网的 DP 主站系统：

```
Subnet subnet = ...;
foreach (Node node in subnet.Nodes)
    {
        // work with the node
    }
```

7.11.6.8 枚举子网的 io 系统

要求

- Openness 应用程序连接至 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)

应用

枚举 IoSystem 可提供所有存在于子网上的 io 系统。主站系统和 io 系统均由类别 IoSystem 表示。

程序代码

修改以下程序代码以枚举子网的 DP 主站系统：

```
Subnet subnet = ...;
foreach (IoSystem ioSystem in subnet.IoSystems)
    {
        // work with the io system
    }
```

7.11.6.9 访问节点

要求

- Openness 应用程序连接至 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)

应用

角色接口聚合节点访问与接口地址和子网分配相关的属性。

节点名称可在 TIA Portal 的接口属性中查看。聚合在接口的节点，每个节点均具有一个唯一的标示符 NodeId，只能通过 Openness 查看其值。

程序代码

修改以下程序代码以访问某一接口的所有节点：

```
NetworkInterface itf = ...
foreach (Node node in itf.Nodes)
{
    ... // Work with the node
}
```

多数接口仅提供一个节点，因此通常使用第一个节点：

```
NetworkInterface itf = ...
Node node in itf.Nodes.First()
{
    ... // Work with the node
}
```

节点提供其名称、类型以及 NodeId 作为属性：

```
Node node = ...
string name = node.Name;
NetType type = node.NodeType;
string id = node.NodeId;
```

网络类型

值	说明
NetType.Unknown	网络类型未知。
NetType.Ethernet	网络类型为以太网。
NetType.Profibus	网络类型为 Profibus。
NetType.Mpi	网络类型为 MPI。
NetType.ProfibusIntegrated	网络类型为集成 Profibus。
NetType.Asi	网络类型为 ASi。
NetType.PcInternal	网络类型为 PC 内网。
NetType.Ptp	网络类型为 PtP。

7.11.6.10 访问节点的属性

要求

- Openness 应用程序连接至 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)

应用

设备项提供了某些可供读取和/或写入的强制属性。这些属性仅当其在 UI 中可用时才可用。通常仅当属性可以由用户在 UI 中修改时才允许写入。具体取决于设备项类型。例如，只有当 RouterUsed 为“真”时用户才能设置 RouterAddress。如果用户修改了 IO 控制器上的 SubnetMask，则所有 IO 设备上的子网掩码也将更改为同一值。

ASI 类型节点的属性

属性	数据类型	可写	访问	说明
Name	string	读取		节点名称。
NodeId	string	读取		节点 ID。

属性	数据类型	可写	访问	说明
NodeType	NetType	读取		节点从子网中获取其类型。
Address	string	读取/写入	动态	AS-i 从站的附加属性。

以太网类型节点的属性

属性	数据类型	可写	访问	说明
Name	string	读取		节点名称。
NodeId	string	读取		节点 ID。
NodeType	NetType	读取/写入或读取		节点从子网中获取其类型。
UseIsoProtocol	bool	读取/写入	动态	
MacAddress	string	读取/写入	动态	例如：01-80-C2-00-00-00
UseIpProtocol	bool	读取/写入	动态	即使在相应的 TIA UI 控件中看不到该值，也可以读取该值。
IpProtocolSelection	enum	读取/写入	动态	
Address	string	读取/写入	动态	只支持 IPv4，不支持 IPv6
SubnetMask	string	读取/写入	动态	
UseRouter	bool	读取/写入	动态	
RouterAddress	string	读取/写入	动态	
DhcpClientId	string	读取/写入	动态	
PnDeviceNameSetDirectly	bool	读取/写入	动态	在设备中直接设置 PROFINET 设备名称。不适用于每个设备。

属性	数据类型	可写	访问	说明
PnDeviceNameAutoGeneration	bool	读取/写入	动态	自动创建 PROFINET 设备名称。
PnDeviceName	string	读取/写入	动态	子网中唯一的设备名称。
PnDeviceNameConverted	string	读取	动态	对设备名称进行转换以供系统内部使用。

MPI 类型节点的属性

属性	数据类型	可写	访问	说明
Name	string	读取		节点名称。
NodeId	string	读取		节点 ID。
NodeType	NetType	读取		节点从子网中获取其类型。
Address	string	读取/写入	动态	

PC 内部类型节点的属性

属性	数据类型	可写	访问	说明
Name	string	读取		节点名称。
NodeId	string	读取		节点 ID。
NodeType	NetType	读取		节点从子网中获取其类型。

PROFIBUS 类型节点的属性

属性	数据类型	可写	访问	说明
Name	string	读取		节点名称。
NodeId	string	读取		节点 ID。
NodeType	NetType	读取		节点从子网中获取其类型。
Address	string	读取/写入	动态	

PROFIBUS Integrated 类型节点的属性

属性	数据类型	可写	访问	说明
Name	string	读取		节点名称。
NodeId	string	读取		节点 ID。
NodeType	NetType	读取		节点从子网中获取其类型。
Address	string	读取	动态	

程序代码：节点的属性

修改以下程序代码以获取或设置节点的属性：

```
Node node = ...;
string nameValue = node.Name;
NetType nodeType = node.NodeType;
node.NodeType = NetType.Mpi;
```

程序代码：动态属性

修改以下程序代码以获取或设置动态节点属性：

```
Node node = ...;
var attributeNames = new[]
{
    "Address", "SubnetMask", "RouterAddress", "UseRouter", "DhcpClientId",
"IpProtocolSelection"
};
foreach (var attributeName in attributeNames)
{
    object attributeValue = ((IEngineeringObject)node).GetAttribute(attributeName);
}
```

协议选择

值	说明
IpProtocolSelection.None	错误值
IpProtocolSelection.Project	在项目中进行组态的 IP 套件。

值	说明
IpProtocolSelection.Dhcp	通过 DHCP 协议管理的 IP 套件。需要 DHCP 客户端 ID。
IpProtocolSelection.UserProgram	通过 FB（功能块）设置的 IP 套件。
IpProtocolSelection.OtherPath	IP 套件通过其它方法设置，例如 PST 工具。
IpProtocolSelection.ViaIoController	运行时，IP 套件通过 IO 控制器进行设置。

网络类型

值	说明
NetType.None	网络类型未知。
NetType.Profibus	网络类型为 PROFIBUS。
NetType.Mpi	网络类型为 MPI。
NetTypeEthernet	网络类型为以太网。
NetTypeAsi	网络类型为 ASI。
NetType.PcInternal	网络类型为 PC 内部。
NetType.ProfibusIntegrated	网络类型为 PROFIBUS Integrated。

7.11.6.11 连接节点到子网

要求

- Openness 应用程序连接至 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)

程序代码

修改以下程序代码以将节点（设备、接口）分配给网络：

```
Node node = ...;
Subnet subnet = ...;
node.ConnectToSubnet(subnet);
```

7.11.6.12 从子网断开节点

要求

- Openness 应用程序连接至 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)

程序代码

修改以下程序代码以断开节点（设备、接口）与网络的连接：

```
Node node = ...;
node.DisconnectFromSubnet();
```

7.11.6.13 创建一个 io 系统

要求

- Openness 应用程序连接至 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)

应用

通过在 IoController 类型的对象上调用操作 IoController.CreateIoSystem("name") 创建 io 系统。如果 name 为空或 String.Empty，则使用默认名称。通过访问 NetworkInterface 上的属性 IoControllers 对象获取 io 控制器。 IoControllers 导航器将返回一个 IoController 对象。

创建 io 系统的先决条件：

- io 控制器的接口连接到子网。
- io 控制器无 io 系统。

程序代码

修改以下程序代码以创建 io 系统：

```
using System.Linq;
...

NetworkInterface interface = ...;
IoSystem ioSystem = null;

// Interface is configured as io controller
if((interface.InterfaceOperatingMode & InterfaceOperatingModes.IoController) != 0)
{
    IoControllerComposition ioControllers = interface.IoControllers;
    IoController ioController = ioControllers.First();
    if(ioController != null)
    {
        ioSystem = ioController.CreateIoSystem("io system");
    }
}
```

7.11.6.14 访问 io 系统的属性

要求

- Openness 应用程序连接至 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)

应用

主站系统和 io 系统均由 IoSystem 类别表示。

程序代码：io 系统的属性

修改以下程序代码以获取 IoSystem 的属性：

```
NetworkInterface itf = ...
foreach (IIoController ioController in itf.IoControllers)
{
    IoSystem ioSystem = ioController.IoSystem;
    int ioSystemNumber = ioSystem.Number;
    string ioSystemName = ioSystem.Name;
}
```

程序代码：io 系统的子网

修改以下程序代码以导航至分配有该 io 系统的子网：

```
Subnet subnet = ioSystem.Subnet;
```

7.11.6.15 将 io 连接器连接到 io 系统

要求

- Openness 应用程序连接至 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)

应用

使用 IoConnector 的 ConnectToIoSystem(IoSystem ioSystem) 操作将 profinet 或 dp IoConnector 连接至现有 io 系统。

使用操作 GetIoController 导航至远程 IoController。有关如何导航至本地 IoConnector 和 io 系统的更多信息，请参见 获取接口的主站系统或 io 系统 (页 194)。

先决条件：

- IoConnector 未连接到 io 系统。
- IoConnector 接口与所需 IoController 的接口连接到相同的子网。

程序代码

修改以下程序代码：

```
IoSystem ioSystem = ...;
IoConnector ioConnector = ...;
ioConnector.ConnectToIoSystem(ioSystem);
IoController ioController = ioConnector.GetIoController();
```

7.11.6.16 获取接口的主站系统或 io 系统

要求

- Openness 应用程序连接至 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)

应用

服务 NetworkInterface 可提供导航 IoControllers，每个 IoController 又可提供导航 IoSystem。主站系统和 io 系统均由类别 IoSystem 表示。io 设备和从站被统称为 io 设备。

- 若网络接口具有 io 系统，则 IoControllers 导航将返回 IoController 对象。此时仅会返回一个 io 控制器。
- 如果网络接口可作为 io 设备连接至 io 系统，则 IoConnectors 导航将返回 IoConnector 对象。此时仅会返回一个 io 连接器。

程序代码：获取 IoController 的 io 系统

修改以下程序代码以获取 IoController 的 io 系统：

```
NetworkInterface itf = ...
foreach (IoController ioController in itf.IoControllers)
{
    IoSystem ioSystem = ioController.IoSystem;
    // work with the io system
}
```

程序代码：获取 IoConnector 的 io 系统

修改以下程序代码以获取 IoConnector 的 io 系统：

```
NetInterface itf = ...
foreach (IoConnector ioConnector in itf.IoConnectors)
{
    IoSystem ioSystem = ioConnector.ConnectedIoSystem;
    // work with the io system
}
```

7.11.6.17 获取 IO 控制器

要求

- Openness 应用程序连接至 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)

程序代码

当前仅可配置一个 IoController。IoController 不提供任何建模属性或操作。

程序代码

修改以下程序代码以获取 IO 控制器：

```
NetworkInterface itf = ...
foreach (IoController ioController in itf.IoControllers)
{
    // work with the io controller
}
```

7.11.6.18 获取 IO 连接器

要求

- Openness 应用程序连接至 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)

应用

IoConnector 不提供任何建模属性或操作。

程序代码

修改以下程序代码以获取 IO 连接器：

```
NetworkInterface itf = ...
foreach (IoConnector ioConnector in itf.IoConnectors)
{
    // work with the IoConnector
}
```

7.11.6.19 从 io 系统或 dp 主站系统断开 io 连接器

要求

- Openness 应用程序连接至 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)

应用

使用 IoConnector 的 DisconnectFromIoSystem() 操作从现有 io 系统或现有 dp 主站系统中断开 IoConnector。

有关如何导航至本地 IoConnector 和 io 系统的更多信息，请参见 获取接口的主站系统或 io 系统 (页 194)。

程序代码

修改以下程序代码：

```
IoSystem ioSystem = ...;
IoConnector ioConnector = ...;

ioConnector.DisconnectFromIoSystem(ioSystem);
```

7.11.6.20 访问 dp 主站系统的属性

要求

- Openness 应用程序连接至 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)

应用

DP 主站系统提供了某些可供读取和/或写入的属性。这些属性仅当其在 UI 中可用时才可用。通常仅当属性可以由用户在 UI 中修改时才允许写入。具体取决于分配给此 DP 主站系统的 DP 主站和 DP 从站。

dp 主站系统的属性

属性	数据类型	可写	访问	说明
Name	string	读取/写入		
Number	int	读取/写入		属性 Number 接受无法通过 UI 设置的值。这种情况下，编译将失败。

程序代码：获得属性

修改以下程序代码以获取属性：

```
IoSystem dpMastersystem = ...;

string name = dpMastersystem.Name;
int number = dpMastersystem.Number;
```

程序代码：设置属性

修改以下程序代码以设置属性：

```
IoSystem dpMastersystem = ...;

dpMastersystem.Name ="myDpMastersystem"
dpMastersystem.Number=42;
```

7.11.6.21 访问 profinet io 系统的属性

要求

- Openness 应用程序连接至 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)

应用

IO 系统提供了某些可供读取和/或写入的属性。这些属性仅当其在 UI 中可用时才可用。通常仅当属性可以由用户在 UI 中修改时才允许写入。具体取决于分配给此 IO 系统的 IO 控制器和 IO 设备。

PROFINET IO 系统的属性

属性	数据类型	可写	访问	说明
MultipleUseIoSystem	bool	读取/写入	动态	
Name	string	读取/写入		
Number	int	读取/写入		属性 Number 接受无法通过 UI 设置的值。这种情况下，编译将失败。
UseIoSystemNameAsDeviceNameExtension	bool	读取/写入	动态	如果 MultipleUseIoSystem 设置为“真”，UseIoSystemNameAsDeviceNameExtension 将设置为“假”，并且无法进行写访问。
MaxNumberIWlanLinksPerSegment	int	读取/写入	动态	

程序代码：获得属性

修改以下程序代码以获取属性：

```
IoSystem ioSystem = ...;
string name = ioSystem.Name;
```

程序代码：设置属性

修改以下程序代码以设置属性：

```
IoSystem ioSystem = ...;
ioSystem.Name = "IOSystem_1";
```

程序代码：获取动态访问的属性

修改以下程序代码以获取动态属性的值：

```
IoSystem ioSystem = ...;
var attributeNames = new[]
{
    "MultipleUseIoSystem", "UseIoSystemNameAsDeviceNameExtension",
"MaxNumberIWlanLinksPerSegment"
};
foreach (var attributeName in attributeNames)
{
    object attributeValue = ((IEngineeringObject)ioSystem).GetAttribute(attributeName);
}
```

程序代码：设置动态访问的属性

修改以下程序代码以设置动态属性的值：

```
IoSystem ioSystem = ...;
((IEngineeringObject)ioSystem).SetAttribute("MultipleUseIoSystem", true);
```

7.11.6.22 删除 dp 主站系统

要求

- Openness 应用程序连接至 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)

程序代码：删除 PROFINET io 系统

修改以下程序代码以删除 PROFINET io 系统：

```
IoController ioController = ...;
IoSystem ioSystem = ioController.IoSystem;

ioSystem.Delete();
```

7.11.6.23 删除 profinet io 系统

要求

- Openness 应用程序连接至 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)

程序代码

修改以下程序代码以删除 profinet io 系统：

```
IoController ioController = ...;
IoSystem ioSystem = ioController.IoSystem;
ioSystem.Delete();
```

7.11.6.24 创建 dp 主站系统

要求

- Openness 应用程序连接至 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)

应用

通过在 IoController 类型的对象上调用 CreateIoSystem(string nameOfIoSystem) 操作创建 DP 主站系统。通过访问 NetworkInterface 上的 IoControllers 属性对象获取 io 控制器。

创建 DP 主站系统的先决条件：

- io 控制器的接口连接到子网。
- io 控制器无 io 系统。

程序代码

修改以下程序代码以创建 dp 主站系统：

```
using System.Linq;
...
NetworkInterface interface = ...;
IoSystem dpMasterSystem = null;

// Interface is configured as master or as master and slave
if((interface.InterfaceOperatingMode & InterfaceOperatingModes.IoController) != 0)
{
    IoControllerComposition ioControllers = interface.IoControllers;
    IoController ioController = ioControllers.First();
    if(ioController != null)
    {
        dpMasterSystem = ioController.CreateIoSystem("dp master system");
    }
}
```

7.11.6.25 访问端口设备项的端口互连信息

要求

- Openness 应用程序连接至 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)

应用

NetworkPort 提供链接 ConnectedPorts，该链接用于对端口进行枚举，从而访问某个端口的所有互连伙伴端口。

只能对在 TIA UI 中同样可以互连的端口进行互连，例如，不能对同一以太网接口的两个端口进行互连。以下情况会触发可恢复的异常：

- 已经存在到同一伙伴端口的互连
- 尝试互连两个不能互连的端口时
- 尝试创建到不支持替代伙伴的端口的第二个互连时

程序代码：获取端口互连

修改以下程序代码以获取端口设备项的端口互连信息：

```
NetworkPort port = ...;
foreach (NetworkPort partnerPort in port.ConnectedPorts)
{
    // work with the partner port
}
```

程序代码：创建端口互连

修改以下程序代码：

```
NetworkPort port1 = ...;
NetworkPort port2 = ...;
port1.ConnectToPort(port2);

// port supports alternative partners
NetworkPort port1 = ...;
NetworkPort port2 = ...;
NetworkPort port3 = ...;
port1.ConnectToPort(port2);
port1.ConnectToPort(port3);
```

程序代码：删除端口互连

修改以下程序代码：

```
NetworkPort port1 = ...;
NetworkPort port2 = ...;
port1.DisconnectFromPort(port2);
```

7.11.6.26 访问端口属性

要求

- Openness 应用程序连接至 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)

应用

如果设备项为端口，则除简单的设备项功能外，其还可提供附加功能。

- 可以访问该端口的已链接伙伴端口
- 可以访问该端口的接口

若要访问此附加功能，即 NetworkPort 功能，必须使用设备项的特定服务。

程序代码：访问端口

修改以下程序代码以访问通道的属性：

```
NetworkPort port = ((IEngineeringServiceProvider)deviceItem).GetService<NetworkPort>();
if (port != null)
{
    ... // Work with the port
}
```

端口的属性

端口具有以下属性：

```
NetworkPort port = ...;
var connectedPorts = port.ConnectedPorts;
var myInterface = port.Interface;
```

7.11.6.27 枚举子网的 dp 主站系统

要求

- Openness 应用程序连接至 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)

应用

枚举 IoSystem 可提供所有存在于子网上的 dp 主站系统。主站系统和 io 系统均由类别 IoSystem 表示。

程序代码

修改以下程序代码以枚举子网的 DP 主站系统：

```
Subnet subnet = ...;
foreach (IoSystem ioSystem in subnet.IoSystems)
    {
        // work with the io system
    }
```

7.11.6.28 枚举已分配的 IO 连接器

要求

- Openness 应用程序连接至 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)

应用

主站系统和 io 系统均由类别 IoSystem 表示。

它用于：

- 枚举 dp 主站系统的 IO 连接器
- 枚举 profinet io 系统的 IO 连接器

程序代码

修改以下程序代码以枚举 DP 主站系统已分配的 IO 连接器：

```
IoSystem ioSystem = ...;
foreach (IoConnector ioConnector in ioSystem.ConnectedIoDevices)
    {
        // work with the io connector
    }
```

7.11.6.29 将 dp io 连接器连接到 dp 主站系统

要求

- Openness 应用程序连接至 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)

应用

使用 IoConnector 的 ConnectToIoSystem(IoSystem ioSystem) 操作将 IoConnector 连接至现有 DP 主站系统。

使用操作 GetIoController 导航至远程 IoController。有关如何导航至本地 IoConnector 和 io 系统的更多信息，请参见 获取接口的主站系统或 io 系统 (页 194)。

先决条件：

- IoConnector 未连接到 io 系统。
- IoConnector 接口与所需 IoController 的接口连接到相同的子网。

程序代码

修改以下程序代码：

```
IoSystem ioSystem = ...;
IoConnector ioConnector = ...;
ioConnector.ConnectToIoSystem(ioSystem);
IoController ioController = ioConnector.GetIoController();
```

7.11.7 设备和设备项的功能

7.11.7.1 设备项的强制属性

要求

- Openness 应用程序连接至 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)

应用

每个设备或设备项均具备可进行读取和/或写入的特定强制属性。这些属性始终与 TIA Portal 用户界面中的属性相同。

Openness 支持以下属性：

属性名称	数据类型	可写	访问	注释
Author	string	读/写	动态	
Classification	DeviceItemClassification	读		
Comment	string	读/写	动态	有时仅限读访问
FirmwareVersion	string	读	动态	
InterfaceOperatingMode	InterfaceOperatingModes	读/写	动态	适用于提供 `NetworkInterface` 功能的设备项
InterfaceType	NetType	读/写	动态	适用于提供 `NetworkInterface` 功能的设备项

属性名称	数据类型	可写	访问	注释
IsBuiltIn	bool	读		对于可由用户创建的对象而言为 FALSE
IsGsd	bool	读		TRUE，前提是通过 GSD/GSDML 安装设备说明
IsPlugged	bool	读		对于已插入的设备而言为 TRUE
Label	string	读	动态	适用于提供 `NetworkPort` 或 `NetworkInterface` 功能的设备项。 如果接口或端口不具备标签，`Label` 将为 `String.Empty`。
LocationIdentifier	string	读/写	动态	
Name	string	读/写		有时仅限读访问
OrderNumber	string	读/写	动态	有时仅限读访问
PlantDesignation	string	读/写	动态	
PositionNumber	int	读		
TypeIdentifier	string	读		
TypeName	string	读	动态	和语言无关的类型名称。可用于用户无法管理的设备项（例如，自动创建的项或固定子模块）。

设备项分类

值	说明
DeviceItemClassifications.None	无分类。
DeviceItemClassifications.CPU	设备项为 CPU
DeviceItemClassifications.HM	设备项为头模块。

程序代码：设备项的强制属性

修改以下程序代码以获取设备项的强制属性：

```
DeviceItem deviceItem = ...;
string nameValue = deviceItem.Name;
string typeIdentfierValue = deviceItem.TypeIdentifier;
int positionNumberValue = deviceItem.PositionNumber;
bool isBuiltInValue = deviceItem.IsBuiltIn;
bool isPluggedValue = deviceItem.IsPlugged;
```

程序代码：动态访问的强制属性

修改以下程序代码以通过动态访问获取属性项：

```
Device device = ...;
var attributeNames = new[] {
    "TypeName", "Author", "Comment", "OrderNumber", "FirmwareVersion", "PlantDesignation",
"LocationIdentifier"
    };
foreach (var attributeName in attributeNames) {
    object attributeValue = ((IEngineeringObject)deviceItem).GetAttribute(attributeName);
    }
```

7.11.7.2 设备的强制属性

要求

- Openness 应用程序连接至 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)

应用

每个设备或设备项均具备可进行读取和/或写入的特定强制属性。这些属性始终与 TIA Portal 用户界面中的属性相同。

Openness 支持以下属性：

属性名称	数据类型	可写入	访问	注释
Author	string	读/写	动态	
Comment	string	读/写	动态	有时仅限读访问
IsGsd	bool	读		TRUE，前提是通过 GSD/GSDML 安装设备说明
Name	string	读/写		有时仅限读访问
TypeIdentifier	string	读		
TypeName	string	读	动态	

程序代码：设备的强制属性

修改以下程序代码以获取设备的强制属性：

```
Device device = ...;
string nameValue = device.Name;
bool isGsdValue = device.IsGsd;
```

程序代码：动态访问的强制属性

修改以下程序代码以通过动态访问获取属性项：

```
Device device = ...;
var attributeNames = new[] {
    "TypeName", "Author", "Comment"
    };
foreach (var attributeName in attributeNames) {
    object attributeValue = ((IEngineeringObject)device).GetAttribute(attributeName);
    }
```

7.11.7.3 获取设备和设备项的类型标识符

要求

- Openness 应用程序已连接 TIA Portal。
  参见“连接到 TIA Portal (页 69)”
- 项目已打开。
  参见“打开项目 (页 96)”

应用

属性 TypeIdentifier 用于识别可通过 Public API 创建的硬件对象。TypeIdentifier 字符串包含多个部分：<TypeIdentifierType>:<Identifier>

TypeIdentifierType 可能具有的值为：

- OrderNumber
- GSD
- System

OrderNumber

OrderNumber 是硬件目录中所有模块的通用 TypeIdentifier。

类型标识符的格式	示例	详细信息
<OrderNumber>	OrderNumber:3RK1 200-0CE00-0AA2	
<OrderNumber>/ <FirmwareVersion>	OrderNumber:6ES7 510-1DJ01-0AB0/V2.0	固件版本可用于系统中不存在或只有一个版本的情况。注意
<OrderNumber>// <AdditionalTypeIdentifier>	OrderNumber:6AV2 124-2DC01-0AX0//Landscape	附加类型标识符可用于 OrderNumber 和 FirmwareVersion 不会导致系统中存在唯一匹配的情况。

说明

硬件目录中存在几个订货号包含“通配符”字符的模块，这类字符用于表示实际硬件的特定集群，例如不同长度的 S7-300 机架。对于这种情况，可使用特定 OrderNumber 和“通配符”OrderNumber 创建硬件对象的实例。不过，不能将通配符通用于任意位置。

GSD

这是用于通过 GSD 或 GSDML 添加至 TIA Portal 的模块的标识符。

类型标识符的格式	示例	详细信息
<GsdName>/<GsdType>	GSD:SIEM8139.GSD/DAP	GsdName 为大写字母形式的 GSD 或 GSDML 名称。 GsdType 则可以为： • D：设备 • R：机架 • DAP：前端模块 • M：模块 • SM：子模块 GsdId 是该类型的标识符。
<GsdName>/<GsdType>/<GsdId>	GSD:SIEM8139.GSD/M/4	

System

此为无法使用 OrderNumber 或 GSD 确定的对象的标识符。

类型标识符的格式	示例	详细信息
<SystemTypeIdentifier>	System:Device.S7300	SystemTypeIdentifier 是对象的主要标识符。 SystemTypeIdentifier 不唯一时，可能需要 AdditionalTypeIdentifier。某些对象类型的前缀为： • Connection. • Subnet. • Device. • Rack.
<SystemTypeIdentifier>/<AdditionalTypeIdentifier>	GSD:SIEM8139.GSD/M/4	

程序代码

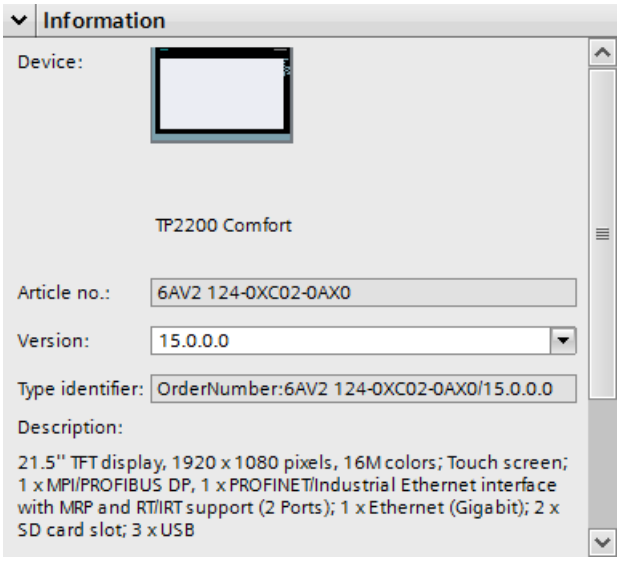
修改以下程序代码以获取用户可管理并可单独创建的 GSD 对象的类型标识符：

```
HardwareObject hardwareObject = ...;
string typeIdentifier = hardwareObject.TypeIdentifier;
```

在 TIA Portal 中显示类型标识符

如果要确定一个类型标识符，则可在 TIA Portal 中执行以下查询操作：

1. 在“选项 > 设置 > 硬件配置 > 显示类型标识符”(Options > Settings > Hardware configuration > Display of the type identifier) 中，启用设置“显示设备和模块的类型标识符”(Enable display of the type identifier for devices and modules)。
2. 打开“设备与网络”(Devices & Networks) 编辑器。
3. 在产品目录中选择一个设备。
   “类型标识符”将显示在“信息”(Information) 窗口中。



7.11.7.4 创建设备

要求

- Openness 应用程序连接至 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)

应用

可以通过两种方法创建设备，即在项目中或在设备组中：

- 通过设备项类型标识符（如 TIA 硬件目录 `Device CreateWithItem(DeviceItemTypeId, DeviceItemName, DeviceName)` 中）创建设备对象：
- 仅创建设备 `Device Create(DeviceTypeId, DeviceName)`

名称	类型	说明
DeviceItemTypeId	string	设备项的类型标识符
DeviceTypeId	string	设备的类型标识符
DeviceItemName	string	已创建设备项的名称
DeviceName	string	已创建设备的名称

参见：类型标识符 (页 211)

程序代码：通过类型标识符创建设备

修改以下代码以通过类型标识符创建设备对象：

```
DeviceComposition devices = ...;
Device device = devices.CreateWithItem("OrderNumber:6ES7 510-1DJ01-0AB0/V2.0", "PLC_1",
"NewDevice");
Device gsdDevice = devices.CreateWithItem("GSD:SIEM8139.GSD/M/4 ", "GSD Module",
"NewGsdDevice");
```

程序代码：仅创建设备

修改以下代码以仅创建设备对象：

```
DeviceComposition devices = ...;
Device deviceOnly = devices.Create("System:Device.S7300", "S7300Device");
Device gsdDeviceOnly = devices.Create("GSD:SIEM8139.GSD/D", "GSD Device");
```

7.11.7.5 枚举设备

要求

- Openness 应用程序已连接 TIA Portal。
  参见“连接到 TIA Portal (页 69)”
- 项目已打开。
  参见“打开项目 (页 96)”

应用：枚举设备

使用公共 API 定位设备的方式与 TIA Portal 中项目导航的方式非常类似。PNV.

- 使用项目组合“Devices”集合作为直接子项目的设备
- 使用文件夹组合“Devices”集成设备文件夹中的设备。

说明

请遵守对象模型的硬件对象的层级 (页 58)。

使用以下一个选项枚举项目的设备：

- 枚举根级别的所有设备
- 枚举组或子组中的所有设备
- 枚举不包含任何设备组的项目的所有设备
- 枚举未分组设备系统组的所有设备

可以枚举的设备的示例：

- Central station
- PB-Slave / PN-IO device
- HMI Device

程序代码：枚举根级别的设备

修改以下程度代码以枚举根级别的设备：

```
private static void EnumerateDevicesInProject(Project project)
{
    DeviceComposition deviceComposition = project.Devices;
    foreach (Device device in deviceComposition)
    {
    // add code here
    }
}
```

修改以下程序代码以访问单个设备。

```
private static void AccessSingleDeviceByName(Project project)
{
    DeviceComposition deviceComposition = project.Devices;
    // The parameter specifies the name of the device
    Device device = deviceComposition.Find("MyDevice");
}
```

程序代码：枚举组或子组中的设备

要访问组中的设备，首先需要导航至该组，然后再导航至相应的设备。

修改以下程序代码：

```
//Enamerate devices in groups or sub-groups
private static void EnumerateDevicesInGroups(Project project)
{
    foreach (DeviceUserGroup deviceUserGroup in project.DeviceGroups)
    {
        EnumerateDeviceUserGroup(deviceUserGroup);
    }
}
private static void EnumerateDeviceUserGroup(DeviceUserGroup deviceUserGroup)
{
    EnumerateDeviceObjects(deviceUserGroup.Devices);
    foreach (deviceUserGroup subDeviceUserGroup in deviceUserGroup.Groups)
    {
        // recursion
        EnumerateDeviceUserGroup(subDeviceUserGroup);
    }
}
private static void EnumerateDeviceObjects(DeviceComposition deviceComposition)
{
    foreach (Device device in deviceComposition)
    {
    // add code here
    }
}
```

程序代码：查找特定设备

要按名称查找特定设备，则需修改以下程序代码：

```
//Find a specific device by name
Project project = ...
Device plc1 = project.Devices.First(d => d.Name == "Mydevice");
... // Work with the device
```

修改以下程序代码以通过“查找”方法查找特定设备：

```
//Find a specific device via "Find" method
Project project = ...
Device plc1 = project.Devices.Find("MyDevice");
... // Work with the device
```

程序代码：枚举不包含任何设备组的项目的设备

修改以下程序代码：

```
//Enumerate all devices which are located directly under a project that contains no device
groups
Project project = ...
foreach (Device device in project.Devices)
{
    ... // Work with the devices
}
```

程序代码：枚举位于某一文件夹中的所有设备

修改以下程序代码：

```
//Enumerate all devices located in a folder
Project project = ...
DeviceUserGroup sortingGroup = project.DeviceGroups.Find ("Sorting");
Device plc1 = sortingGroup.Devices.First(d => d.Name == "MyStationName");
... // Work with the device
```

程序代码：枚举未分组设备系统组的设备

为构建项目，已将分散设备置于 UngroupedDevices 组中。要访问该组中的设备，首先需要导航至该组，然后再导航至相应的设备。

修改以下程序代码：

```
//Enumerate devices of the ungrouped device system group
Project project = ...
DeviceSystemGroup group = project.UngroupedDevicesGroup;
Device plc1 = group.Devices.First(d => d.Name == "MyStationName");
... // Work with the device
```

7.11.7.6 访问设备

要求

- Openness 应用程序连接到 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)

应用

每个基于 GSD 或 GSDML 的 IO 设备都具有属性。其中某些属性可用于识别特定类型的设备。

名称	数据类型	可写入	访问	说明
Author	string	读/写	动态	
Comment	string	读/写	动态	
GsdName	string	读	动态	GSD 文件的名称。
GsdType	string	读	动态	硬件对象的类型。对于设备，该值始终为“D”。
GsdId	string	读	动态	硬件对象的特定标识符。对于设备，始终为空
IsGsd	bool	读		对于 GSD 设备或 GSDML 设备为 TRUE
Name	string	读/写		
TypeIdentifier	string	读		

程序代码：获取标识属性

修改以下程序代码以获取属性：

```
Device device = ...;
var attributeNames = new[] {
    "GsdName", "GsdType", "GsdId"
    ;
foreach (var attributeName in attributeNames) {
    object attributeValue = device.GetAttribute(attributeName);
    }
```

程序代码：属性

修改以下程序代码以获取属性：

```
Device device = ...;
string nameValue = device.Name;
bool isGsdValue = device.IsGsd;
```

程序代码：动态访问的属性

修改以下程序代码以获取属性：

```
Device device = ...;
var attributeNames = new[] {
    "GsdName", "GsdType", "GsdId"
    ;
foreach (var attributeName in attributeNames) {
    object attributeValue = device.GetAttribute(attributeName);
    }
```

GSD 设备特性

如果设备是为 GSD 设备，则该设备可提供附加功能。要获取 GsdDevice 功能，请使用 GetService 方法。

```
GsdDevice gsdDevice = ((IEngineeringServiceProvider)deviceItem).GetService<GsdDevice>();
if (gsdDevice != null) {
    ... // work with the GSD device
    };
```

程序代码：GSD 设备的属性

修改以下程序代码以获取属性：

```
Device device = ...;
GsdDevice gsdDevice = ...;
string gsdId = gsdDevice.GsdId;
string gsdName = gsdDevice.GsdName;
string gsdType = gsdDevice.GsdType;
bool isProfibus = gsdDevice.IsProfibus;
bool isProfinet = gsdDevice.IsProfinet;
```

7.11.7.7 删除设备

要求

- Openness 应用程序连接至 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)

程序代码

修改以下程序代码以删除设备：

```
Project project = ...;
Device deviceToDelete = project.UngroupedDevices.Devices.Find("......");

// delete device
deviceToDelete.Delete();
```

7.11.7.8 创建和插入设备项

要求

- Openness 应用程序连接至 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)

应用

使用 `HardwareObject` 的 `PlugNew(string typeIdentifier, string name, int positionNumber)` 操作

- 创建新的设备项并将其插入现有硬件对象中
- 创建新的子设备项（例如，子模块）并将其插入某一设备项中

如果操作成功，将返回已创建的设备项对象，否则将触发可恢复的异常。

通过使用操作 `CanPlugNew(string typeIdentifier, string name, int positionNumber)`，您可以确定是否可以执行创建和插入操作。如果不能执行，则操作会返回 `false`。

下表列出了所需的方法参数：

名称	类型	说明
`typeIdentifier`	字符串	已创建设备项的类型标识符
`name`	字符串	已创建设备项的名称
`positionNumber`	int	已创建设备项的位置编号

程序代码

修改以下程序代码以将设备项插入到现有硬件对象中：

```
HardwareObject hwObject = ...;
string typeIdentifier = ...;
string name = ...;
int positionNumber = ...;
if(hwObject.CanPlugNew(typeIdentifier, name, positionNumber))
{
    DeviceItem newPluggedDeviceItem = hwObject.PlugNew(typeIdentifier, name,
positionNumber);
}
```

7.11.7.9 将设备项移到另一插槽中

要求

- Openness 应用程序连接至 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)

应用

HardwareObject 的 PlugMove(DeviceItem deviceItem, int positionNumber) 操作可用于移动现有设备项并将其插入现有硬件对象中。如果无法在 UI 中插入模块，则只有在极少数情况下 PlugMove 方法才能起作用。在这些情况下将出现编译错误。

CanPlugMove(DeviceItem deviceItem, int positionNumber) 操作可用于确定是否可以进行移动。若不可能，CanPlugMove 将返回“假”。但是，如果方法返回“真”，该操作仍然可能会因意外原因而失败。

操作失败的可能原因如下：

- 位置编号已由另一个设备项使用
- 当前设备项无法插入相应位置（尽管该位置空闲）
- 容器不提供位置编号
- 设备项的名称已由同一容器中的现有设备项所使用
- 设备项无法插入到容器
- 用户无法插入设备项
- 用户无法移除设备项
- 设备在线

程序代码

修改以下程序代码：

```
HardwareObject hwObject = ...;
int positionNumber = ...;
if(hwObject.CanPlugMove(deviceItemToMove, positionNumber)
{
    DeviceItem movedDeviceItem = hwObject.PlugMove(deviceItemToMove, positionNumber);
}
```

7.11.7.10 复制设备项

要求

- Openness 应用程序连接至 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)

应用

使用 HardwareObject 的 PlugCopy(DeviceItem deviceItem, int positionNumber) 操作将项目中的设备复制并插入现有的硬件。如果无法在 UI 中插入模块，则只有在极少数情况下 PlugCopy 方法才能起作用。此时，复制后会发生编译错误。当 PlugCopy 成功时，将返回设备项对象的副本，否则，会触发可恢复的异常。

操作失败的可能原因：

- 位置编号已由另一个设备项使用
- 当前设备项无法插入相应位置（尽管该位置空闲）
- 容器不提供位置编号
- 设备项的名称已由同一容器中的现有设备项所使用
- 设备项无法插入到容器
- 设备项无法插入到 UI
- ...

使用 CanPlugCopy(DeviceItem deviceItem, int positionNumber) 操作确定是否可以进行复制。当无法执行复制操作时，CanPlugCopy 将返回“假”。但是，如果方法返回“真”，该操作仍然可能会因意外原因而失败。

程序代码

修改以下程序代码：

```
HardwareObject hwObject = ...;
DeviceItem deviceItemToCopy = ...;
int positionNumber = ...;
if(hwObject.CanPlugCopy(deviceItemToCopy, positionNumber))
{
   DeviceItem copiedDeviceItem = hwObject.PlugCopy(deviceItemToCopy, positionNumber);
}
```

7.11.7.11 删除设备项

要求

- Openness 应用程序连接至 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)

程序代码

修改以下程序代码以删除设备项：

```
Project project = ...;
var device = project.UngroupedDevicesGroup.Devices.Find("......");
var deviceItem = deviceItem.DeviceItems.First();

// delete device item
deviceItem.Delete();
```

7.11.7.12 枚举设备项

要求

- Openness 应用程序连接至 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)

应用

为获取设备项，请使用 HardwareObject。硬件对象的项目为插入硬件对象时 TIA Portal 用户所看到的内容：

- 位于设备中的机架
- 位于机架中的模块
- 位于模块中的子模块
- 位于子模块中的子模块

说明

有关此主题的更多详细信息，可参考对象模型的硬件对象的层级 (页 58)部分。

程序代码：枚举设备的设备项

修改以下程序代码以枚举硬件对象的设备项：

```
private static void EnumerateDeviceItems(HardwareObject hardwareObject)
{
    foreach (DeviceItem deviceItem in hardwareObject.Items)
    {
        // add code here
    }
}
```

程序代码：通过组合层级进行枚举

若要通过组合层级来枚举某一设备的设备项，则修改以下程序代码：

```
//Enumerates devices using an composition
private static void EnumerateDeviceItems(Device device)
{
    DeviceItemComposition deviceItemComposition = device.DeviceItems;
    foreach (DeviceItem deviceItem in deviceItemComposition)
    {
        // add code here
    }
}
```

程序代码：使用关联枚举设备项

修改以下程序代码以使用关联枚举设备项：

```
//Enumerates devices using an association
private static void EnumerateDeviceItemsWithAssociation(Device device)
{
    DeviceItemAssociation deviceItemAssociation = device.Items;
    foreach (DeviceItem deviceItem in deviceItemAssociation)
    {
        // add code here
    }
}
```

7.11.7.13 访问设备项

要求

- Openness 应用程序连接到 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)

应用：访问设备项

要访问“DeviceItem”类型的对象，请使用以下属性：

- 名称 (string)：设备项的名称
- 容器 (HardwareObject)：用于插入设备项的容器

名称	数据类型	可写入	访问	说明
Author	string	读/写	动态	
Comment	string	读/写	动态	
FirmwareVersion	string	读	动态	仅用于头模块
GsdName	string	读	动态	GSD 文件的名称。
GsdType	string	读	动态	硬件对象的类型。对于设备，该值始终为“D”。
GsdId	string	读	动态	硬件对象的特定标识符。对于设备，始终为空

名称	数据类型	可写入	访问	说明
IsBuiltIn	bool	读		
IsGsd	bool	读		对于 GSD 设备或 GSDML 设备为 TRUE
IsPlugged	bool	读		
IsProfibus	bool	读		
IsProfinet	bool	读		
Name	string	读/写		
OrderNumber	string	读	动态	仅用于头模块
PositionNumber	bool	读		
TypeIdentifier	string	读		

程序代码：访问设备项

修改以下程序代码以访问设备项：

```
public static DeviceItem AccessDeviceItemFromDevice(Device device)
{
    DeviceItem deviceItem = device.DeviceItems[0];
    return deviceItem;
}
```

程序代码：访问设备项的设备项

修改以下程序代码以访问设备项的设备项：

```
public static DeviceItem AccessDeviceItemFromDeviceItem(DeviceItem deviceItem)
{
    DeviceItem subDeviceItem = deviceItem.DeviceItems[0];
    return subDeviceItem;
}
```

程序代码：导航至设备项的容器

修改以下程序代码以通过 DeviceItem 的 "Container" 属性导航回设备项的容器：

```
DeviceItem deviceItem = ...;
HardwareObject container = deviceItem.Container;
```

程序代码：获取标识属性

修改以下程序代码以获取属性：

```
Device device = ...;
var attributeNames = new[] {
    "GsdName", "GsdType", "GsdId" };
foreach (var attributeName in attributeNames) {
    object attributeValue = ((IEngineeringObject)deviceItem).GetAttribute(attributeName);
    }
```

程序代码：获取属性

修改以下程序代码以获取属性：

```
DeviceItem deviceItem = ...;
GsdDeviceItem gsdDeviceItem =
((IEngineeringServiceProvider)deviceItem).GetService<GsdDeviceItem>();

string gsdName = gsdDeviceItem.GsdName;
string gsdType = gsdDeviceItem.GsdType;
string gsdId = gsdDeviceItem.GsdId;
bool isProfinet = gsdDeviceItem.IsProfinet;
bool isProfibus = gsdDeviceItem.IsProfibus;;
```

程序代码：获取动态访问的属性

修改以下程序代码以获取属性：

```
DeviceItem deviceItem = ...;
GsdDeviceItem gsdDeviceItem =
((IEngineeringServiceProvider)deviceItem).GetService<GsdDeviceItem>();

var attributeNames = new[] {
    "TypeName", "Author", "Comment", ...
    ;
foreach (var attributeName in attributeNames) {
    object attributeValue =
((IEngineeringObject)gsdDeviceItem).GetAttribute(attributeName);
    }
```

程序代码：设置属性

修改以下程序代码以设置属性：

```
DeviceItem deviceItem = ...;
((IEngineeringObject)deviceItem).SetAttribute("Comment", "This is a comment.");
```

程序代码：获取头模块的 prm 数据

修改以下程序代码以获取 prm 数据：

```
DeviceItem deviceItem = ...;
GsdDeviceItem gsdDeviceItem =
((IEngineeringServiceProvider)deviceItem).GetService<GsdDeviceItem>();

int dsNumber = 0;       // For Profibus GSDs, dataset number zero must be used!
int byteOffset = 0;
int lengthInBytes = 5;

// read complete data set:
byte[] prmDataComplete = gsdDeviceItem.GetPrmData(dsNumber, byteOffset, lengthInBytes);

// read partial data set (only second byte):
byteOffset = 1;
lengthInBytes = 1;
byte[] prmDataPartial = gsdDeviceItem.GetPrmData(dsNumber, byteOffset, lengthInBytes);
```

程序代码：设置头模块的 prm 数据

修改以下程序代码以获取 prm 数据：

```
DeviceItem deviceItem = ...;
GsdDeviceItem gsdDeviceItem =
((IEngineeringServiceProvider)deviceItem).GetService<GsdDeviceItem>();

// The parameters byteOffset and the length of the byte array prmData define the range
within the
// dataset which is written to.
// For Profibus GSDs, dataset number zero must be used!

// Change the highlighted bytes 2-4 from 0x0 to 0x1
// to write only the first two bytes: byte[] prmData = {0x05, 0x21};

int dsNumber = 0;
int byteOffset = 0;
byte[] prmData = {0x05, 0x21, 0x01, 0x01, 0x01};

gsdDeviceItem.SetPrmData(dsNumber, byteOffset, prmData);
```

参见

对象模型的硬件对象的层级 (页 58)

7.11.7.14 I/O 设备接口的访问属性

要求

- Openness 应用程序连接至 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)
- 对于写入访问，PLC 已处于离线状态。

应用

可以使用 Public API 接口来获取或设置 I/O 设备接口上 IRT 及等时模式的属性。

访问 I/O 控制器的接口

可访问 I/O 控制器的接口的以下属性。控制器必须是同步主站：

属性名称	数据类型	可写入	访问	说明
PnSendClock	Int64	r/w	动态属性	发送时钟（以纳秒为单位）

访问 I/O 系统的接口

可访问 I/O 系统的接口的以下属性。Ti/To 值可供 I/O 系统的所有模块和子模块使用。

属性名称	数据类型	可写入	访问
IsochronousTiToAutoCalculation	BOOL	r/w	动态属性
IsochronousTi	DOUBLE	r/w	动态属性
IsochronousTo	DOUBLE	r/w	动态属性

访问 I/O 设备的接口

可访问 I/O 设备的接口的以下属性。Ti/To 值可供 I/O 系统的所有模块和子模块使用。

属性名称	数据类型	可写入	访问
IsochronousMode	BOOL	r/w	动态属性
IsochronousTiToCalculationMode	IsochronousTiToCalculationMode	r/w	动态属性
IsochronousTi	DOUBLE	r/w	动态属性
ochronousTo	DOUBLE	r/w	动态属性

为属性 IsochronousTiToCalculationMode 提供以下 ENUM 值：

值	说明
IsochronousTiToCalculationMode.None	
IsochronousTiToCalculationMode.FromOB	使用 OB（在 IoSystem 中组态的）的 Ti/To 值。
IsochronousTiToCalculationMode.FromSubnet	PROFINET 接口不使用此值。

值	说明
IsochronousTiToCalculationMode.AutomaticMinimum	为 IO 设备自动计算 Ti/To 值。
IsochronousTiToCalculationMode.Manual	用户可以手动输入此 IO 设备的 Ti/To 值。

程序代码：获取或设置 I/O 设备接口的属性

修改以下程序代码以访问发送时钟值：

```
DeviceItem pnInterface = ...;

// read attribute
long attributeValue = (long)pnInterface.GetAttribute("PnSendClock");
// write attribute
long sendClock = 2000000;
pnInterface.SetAttribute("PnSendClock", sendClock);
```

修改以下程序代码以访问 OB 的 Ti/To 值：

```
IoSystem ioSystem = ...;
bool titoAutoCalculation = (bool)ioSystem.GetAttribute("IsochronousTiToAutoCalculation");
ioSystem.SetAttribute("IsochronousTiToAutoCalculation", true);
```

修改以下程序代码以访问 I/O 设备接口的等时设置：

```
DeviceItem pnInterface = ...;
bool isochronousMode = (bool)pnInterface.GetAttribute("IsochronousMode");
pnInterface.SetAttribute("IsochronousMode", true);
```

7.11.7.15 访问地址控制器

要求

- Openness 应用程序连接至 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)

应用

如果设备项为地址控制器，则可提供附加功能：要访问地址控制器的注册地址，请使用角色 AddressController。

程序代码：获取地址控制器

修改以下程序代码以获取地址控制器角色：

```
AddressController addressController =
((IEngineeringServiceProvider)deviceItem).GetService<AddressController>();
    if (addressController != null)
        {
            ... // work with the address controller
        }
```

地址控制器的属性

地址控制器的属性为：

- RegisteredAddresses

修改以下程序代码以获取地址控制器的属性：

```
AddressController addressController = ...;
foreach (Address registeredAddress in addressController.RegisteredAddresses)
        {
            ...
        }
```

7.11.7.16 访问地址

要求

- Openness 应用程序连接至 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)

应用

通过设备项的组合链接 Addresses 获取地址对象。属性 Addresses 将返回可枚举的 AddressComposition 类型集合。

程序代码：获取设备项的地址

要获取设备项的地址，请修改以下程序代码：

```
AddressComposition addresses = deviceItem.Addresses;
    foreach(Address address in addresses)
        {
            // work with the address
        }
```

程序代码：获取 IO 控制器地址

要获取 io 控制器的地址，请修改以下程序代码：

```
AddressComposition addresses = ioController.Addresses;
    foreach(Address address in addresses)
        {
            // work with the address
        }
```

属性

地址支持以下属性：

属性名称	数据类型	可写入	访问	注释
AddressControllers	AddressControllerAssociation	读		
Context	enum: AddressContext	读	动态	仅适用于特殊设备项的诊断地址
IoType	enum: AddressIoType	读		
StartAdress	Int32	读/写		
Length	Int32	读		

值	说明
AddressIoType.Diagnosis	地址 io 的类型为诊断。
AddressIoType.Input	地址 io 的类型为输入。
AddressIoType.Output	地址 io 的类型为输出。
AddressIoType.Substitute	地址 io 的类型为替代。
AddressIoType.None	地址 io 的类型为指定的 mot。

值	说明
AddressContext.None	地址上下文不适用
AddressContext.Device	设备地址内容
AddressContext.Head	一个前段地址内容

程序代码：读取特性

修改以下程序代码以获取属性：

```
AddressControllerAssociation addressControllers = address.AddressControllers;
Int32 startAddress = address.StartAddress;
AddressIoType addressType = address.IoType;
Int32 adressLength = address.Length;
```

程序代码：写入特性

修改以下程序代码以写入特性：

```
Address addressControllers = ...;

address.StartAddress = intValueStartAddress;
```

程序代码：动态访问的属性

修改以下程序代码以获取属性：

```
Address address= ...;

object attributeValue = ((IEngineeringObject)address).GetAttribute("Context");
```

7.11.7.17 访问硬件标识符

要求

- Openness 应用程序连接至 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)

应用

从下列对象中获取硬件标识符对象：

- Device
- DeviceItem
- IoSystem

硬件标识可由类别 `HwIdentifier` 表示，并通过以下属性访问：`HwIdentifiers`.

程序代码：获取硬件标识符

若要使 HwIdentifier 可用，可修改以下程序代码：

```
var hwObject = ...
    foreach(HwIdentifier hardwareIdentifier in hwObject.HwIdentifiers)
       {
          // Work with the HwIdentifer
       }
```

硬件标识符的属性

```
HwIdentifierControllerAssociation controllers = hwIdentifier.HwIdentifierControllers;
Int64 Identifier = hwIdentifier.Identifier;
```

7.11.7.18 访问硬件标识符控制器

要求

- Openness 应用程序连接至 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)

应用

如果设备项为硬件标识符控制器，则可以访问已注册的硬件标识符。要访问这些 HwIdentifierController，清使用设备项的特定服务。

程序代码：获取硬件标识符控制器

要获取 HwIdentifierController，请修改以下程序代码：

```
HwIdentifierController hwIdentifierController =
((IEngineeringServiceProvider)deviceItem).GetService<HwIdentifierController>();
    if (hwIdentifierController != null)
        {
            ... // work with the hardware identifier controller
        }
```

程序代码：硬件标识符控制器的属性

地址控制器的属性为：

- `RegisteredHwIdentifiers`:在其中注册硬件标识符的硬件标识符控制器。

修改以下程序代码以获取地址控制器的属性：

```
HwIdentifierController hwIdentifierController = ...;
HwIdentifierAssociation controllers = hwIdentifierController.RegisteredHwIdentifiers;
```

7.11.7.19 访问设备项的通道

要求

- Openness 应用程序连接至 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)

应用

通道可由 Channel 类别表示。通道可通过 DeviceItem 类别的属性 Channels 从设备项获取。属性 Channels 将返回可枚举的 ChannelComposition 实施。如果设备项无通道，属性 Channels 将返回一个空集合。

强制属性

通道支持以下强制属性：

属性名称	数据类型	可写入	访问	注释
IoType	ChannelIoType	读		
Type	ChannelType	读		
Number	Int32	读		
ChannelAddress	Int32	读	动态	通道的地址，单位为位
ChannelWidth	UInt32	读	动态	通道的宽度，单位为位

程序代码：获取设备项的通道

修改以下程序代码以获取设备项的通道：

```
ChannelComposition channels = deviceItem.Channels
foreach(Channel channel in channels)
    {
        // work with the channel
    }
```

程序代码：通道的强制属性

修改以下程序代码以获取设备项的通道：

```
Channel channel = ...;
int channelNumber = channel.Number;
ChannelType type = channel.Type;
ChannelIoType ioType = channel.IoType;
```

程序代码：通过动态访问获取属性值

修改以下程序代码以获取动态属性的值：

```
Channel channel = ...;
Int32 channelAddress = (Int32)((IEngineeringObject)channel).GetAttribute("ChannelAddress");
UInt32 channelWidth = (UInt32)((IEngineeringObject)channel).GetAttribute("ChannelWidth");
```

程序代码：设置动态属性的值

修改以下程序代码以设置可写入动态属性的值：

```
Channel channel = ...;
((IEngineeringObject)channel).SetAttribute("AnAttribute", 1234);
```

7.12 HMI 设备的数据访问函数

7.12.1 画面

7.12.1.1 创建用户定义的画面文件夹

要求

- Openness 应用程序连接至 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)

程序代码

修改以下程序代码以创建用户自定义画面文件夹：

```
//Creates a screen folder
private static void CreateScreenFolder(HmiTarget hmitarget)
{
    ScreenUserFolder myCreatedFolder =
hmitarget.ScreenFolder.Folders.Create("myScreenFolder");
}
```

7.12.1.2 从文件夹中删除画面

要求

- Openness 应用程序连接至 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)

应用

说明

无法删除永久性区域。永久性区域是指始终存在的系统画面。

程序代码

修改以下程序代码以从指定文件夹中删除画面：

```
public static void DeleteScreenFromFolder(HmiTarget hmiTarget)
{
    ScreenUserFolder screenUserFolder =
hmiTarget.ScreenFolder.Folders.Find("myScreenFolder");
    ScreenComposition screens = screenUserFolder.Screens;
    Screen screen = screens.Find("myScreenName");
    if (screen != null)
    {
        screen.Delete();
    }
}
```

7.12.1.3 从文件夹中删除画面模板

要求

- Openness 应用程序连接至 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)
- 此项目包含 HMI 设备。

程序代码

修改以下程序代码以从指定文件夹中删除画面模板：

```
private static void DeleteScreenTemplateFromFolder(HmiTarget hmiTarget)
{
    string templateName = "MyScreenTemplate";
    ScreenTemplateUserFolder folder =
hmiTarget.ScreenTemplateFolder.Folders.Find("myScreenTemplateFolder");
    ScreenTemplateComposition templates = folder.ScreenTemplates;
    ScreenTemplate template = templates.Find(templateName);
    if (template != null)
    {
        template.Delete();
    }
}
```

7.12.1.4 从文件夹中删除所有画面

要求

- Openness 应用程序连接至 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)

应用

说明

无法删除永久性区域。永久性区域是指始终存在的系统画面。

程序代码

修改以下程序代码以从指定文件夹中删除所有画面：

```
private static void DeleteAllScreensFromFolder(HmiTarget hmitarget)
//Deletes all screens from a user folder or a system folder
{
    ScreenUserFolder folder = hmitarget.ScreenFolder.Folders.Find("myScreenFolder");
    //or ScreenSystemFolder folder = hmitarget.ScreenFolder;
    ScreenComposition screens = folder.Screens;
    List<Screen> list = new List<Screen>();
    foreach(Screen screen in screens)
    {
        list.Add(screen);
    }
    foreach (Screen screen in list)
    {
        screen.Delete();
    }
}
```

7.12.2 周期

7.12.2.1 删除周期

要求

- Openness 应用程序连接至 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)
- 此项目包含 HMI 设备。

应用

无法删除标准周期。

可确定周期实际上是否已根据相应周期的对象模型中的组合 (composition count) 进行删除。无法再访问这些周期。

程序代码

修改以下程序代码以从 HMI 设备中删除周期：

```
public static void DeleteCycle(HmiTarget hmiTarget)
{
    CycleComposition cycles = hmiTarget.Cycles;
    Cycle cycle = cycles.Find("myCycle");
    cycle.Delete();
}
```

7.12.3 文本列表

7.12.3.1 删除文本列表

要求

- Openness 应用程序连接至 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)
- 此项目包含 HMI 设备。

程序代码

修改以下程序代码以从 HMI 设备中删除所选文本列表和所有相关列表条目：

```
public static void DeleteTextList(HmiTarget hmiTarget)
{
    TextListComposition textLists = hmiTarget.TextLists;
    TextList textList = textLists.Find("myTextList");
    textList.Delete();
}
```

7.12.4 图形列表

7.12.4.1 删除图形列表

要求

- Openness 应用程序连接至 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)
- 此项目包含 HMI 设备。

程序代码

修改以下程序代码以从 HMI 设备中删除所选图形列表和所有相关列表条目：

```
private static void DeleteGraphicList(HmiTarget hmiTarget)
{
    GraphicListComposition graphicLists = hmiTarget.GraphicLists;
    GraphicList graphicList = graphicLists.Find("myGraphicList");
    graphicList.Delete();
}
```

7.12.5 连接

7.12.5.1 删除连接

要求

- Openness 应用程序连接至 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)
- 此项目包含 HMI 设备。

程序代码

修改以下程序代码以从 HMI 设备中删除所选通信连接：

```
private static void DeleteConnection(HmiTarget hmiTarget)
{
    ConnectionComposition connections = hmiTarget.Connections;
    Connection connection = connections.Find("HMI_connection_1");
    connection.Delete();
}
```

7.12.6 变量表

7.12.6.1 创建用户定义的 HMI 变量文件夹

要求

- Openness 应用程序连接至 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)

程序代码

修改以下程序代码以创建用户自定义的 HMI 变量文件夹：

```
private static void CreateUserfolderForHMITags(HmiTarget hmitarget)
// Creates an HMI tag user folder
{
    TagSystemFolder folder = hmitarget.TagFolder;
    TagUserFolder myCreatedFolder = folder.Folders.Create("MySubFolder");
}
```

7.12.6.2 枚举 HMI 变量表的变量

要求

- Openness 应用程序连接至 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)

程序代码

修改以下程序代码以枚举 HMI 变量表中的所有变量：

```
private static void EnumerateTagsInTagtable(HmiTarget hmitarget)
// //Enumerates all tags of a tag table
{
    TagTable table = hmitarget.TagFolder.TagTables.Find("MyTagtable");
    // Alternatively, you can access the default tag table:
    // TagTable defaulttable = hmitarget.TagFolder.DefaultTagTable;

    TagComposition tagComposition = table.Tags;
    foreach (Tag tag in tagComposition)
    {
        // Add your code here
    }
}
```

7.12.6.3 从 HMI 变量表中删除单个变量

要求

- Openness 应用程序连接至 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)

程序代码

修改以下程序代码以从 HMI 变量表中删除特定变量：

```
private static void DeleteATag(HmiTarget hmiTarget)
{
    string tagName = "MyTag";
    TagTable defaultTagTable = hmiTarget.TagFolder.DefaultTagTable;
    TagComposition tags = defaultTagTable.Tags;
    Tag tag = tags.Find(tagName);
    tag.Delete();
}
```

7.12.6.4 从文件夹中删除变量表

要求

- Openness 应用程序连接至 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)
- 此项目包含 HMI 设备。

应用

无法删除默认变量表

程序代码

修改以下程序代码：

```
// Delete a tag table from a specific folder
private static void DeleteTagTable(HmiTarget hmiTarget)
{
    string tableName = "myTagTable";
    TagSystemFolder tagSystemFolder = hmiTarget.TagFolder;
    TagTableComposition tagTables = tagSystemFolder.TagTables;
    TagTable tagTable = tagTables.Find(tableName);
    tagTable.Delete();
}
```

7.12.7 VB 脚本

7.12.7.1 创建用户定义的脚本文件夹

要求

- Openness 应用程序连接至 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)

程序代码

修改以下程序代码以在系统文件夹或其它用户自定义的文件夹下创建用户自定义的脚本子文件夹：

```
private static void CreateFolderInScriptfolder(HmiTarget hmitarget)
//Creates a script user subfolderVBScriptSystemFolder
{
    VBScriptSystemFolder vbScriptFolder = hmitarget.VBScriptFolder;
    VBScriptUserFolderComposition vbScriptFolders = vbScriptFolder.Folders;
    VBScriptUserFolder vbScriptSubFolder = vbScriptFolder.Folders.Create("mySubfolder");
}
```

7.12.7.2 从文件夹中删除 VB 脚本

要求

- Openness 应用程序连接至 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)
- 此项目包含 HMI 设备。

程序代码

修改以下程序代码以从指定文件夹中删除 VB 脚本：

```
//Deletes a vbscript from a script folderVBScriptSystemFolder
private static void DeleteVBScriptFromScriptFolder(HmiTarget hmitarget)
{
    VBScriptUserFolder vbscriptfolder =
hmitarget.VBScriptFolder.Folders.Find("MyScriptFolder");
    var vbScripts = vbscriptfolder.VBScripts;
    if (null != vbScripts)
    {
        var vbScript = vbScripts.Find("MyScript");
        vbScript.Delete();
    }
}
```

7.12.8 删除 HMI 设备的用户自定义文件夹

要求

- Openness 应用程序连接到 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 项目已经打开。
  请参见打开项目 (页 96)

程序代码

修改以下程序代码以删除 HMI 设备的用户自定义文件夹：

```
HmiTarget hmiTarget = ...;
ScreenUserFolder screenUserGroup = hmiTarget.ScreenFolder.Folders.Find("MyUserFolder");
screenUserGroup.Delete();
```

7.13 PLC 设备的数据访问函数

7.13.1 比较 PLC 软件

要求

- Openness 应用程序连接到 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已通过 Openness 应用程序打开一个项目。
  请参见打开项目 (页 96)

应用

可使用以下方式确定两个设备的软件之间的差异：

- 比较两个已组态 PLC 的软件
- 比较 PLC 软件和项目库
- 比较 PLC 软件和全局库
- 比较 PLC 软件和 PLC 主副本
- 在“在线”状态下比较已组态 PLC 的软件与所连接 PLC 的软件

签名

使用 `CompareTo` 或 `CompareToOnline` 方法进行比较。

```
public CompareResult CompareTo (ISoftwareCompareTarget
compareTarget)
```

```
public CompareResult CompareToOnline ()
```

返回值/参数	功能
CompareResult compareResult	返回比较结果： • FolderContentsDifferent：所比较的文件夹内容不同。 • FolderContentsIdentical：所比较的文件夹内容相同。 • ObjectsDifferent：所比较的对象内容不同。 • ObjectsIdentical：所比较的对象内容相同。 • LeftMissing：该对象不包含于启动比较的对象。 • RightMissing：该对象不包含于正在比较的对象。
ISoftwareCompareTarget compareTarget	类似对象的列表。

程序代码

修改以下程序代码以输出比较结果：

```
private static void WriteResult(CompareResultElement compareResultElement, string indent)
{
    Console.WriteLine("{0}<{1}> <{2}> <{3}> <{4}> <{5}> ",
    indent,
    compareResultElement.LeftName,
    compareResultElement.ComparisonResult,
    compareResultElement.RightName,
    compareResultElement.DetailedInformation);
    WriteResult(compareResultElement.Elements, indent);
}
private static void WriteResult (IEnumerable<CompareResultElement> compareResultElements,
string indent)
{
    indent += " ";
    foreach (CompareResultElement compareResultElement in compareResultElements)
    {
        WriteResult(compareResultElement, indent);
    }
}
```

修改以下程序代码以比较设备的软件：

```
private static void CompareTwoOfflinePlcs(PlcSoftware plcSoftware0, PlcSoftware
plcSoftware1)
{
    if (plcSoftware0 != null && plcSoftware1 != null)
    {
        CompareResult compareResult = plcSoftware0.CompareTo(plcSoftware1);
        WriteResult(compareResult.RootElement, string.Empty);
    }
}
```

修改以下程序代码以比较 PLC 的软件与项目库：

```
private static void ComparePlcToProjectLibrary(Project project, PlcSoftware plcSoftware)
{
    if (project != null && plcSoftware != null)
    {
        CompareResult compareResult = plcSoftware.CompareTo(project.ProjectLibrary);
        WriteResult(compareResult.RootElement, string.Empty);
    }
}
```

修改以下程序代码以比较 PLC 的软件与全局库：

```
private static void ComparePlcToGlobalLibrary(PlcSoftware plcSoftware, GlobalLibrary
globalLibrary)
{
    if (plcSoftware != null && globalLibrary != null)
    {
        CompareResult compareResult = plcSoftware.CompareTo(globalLibrary);
        WriteResult(compareResult.RootElement, String.Empty);
    }
}
```

修改以下程序代码以比较 PLC 的软件与主副本：

```
private static void ComparePlcToMasterCopy(Project project, PlcSoftware plcSoftware)
{
    if (project != null && plcSoftware != null)
    {
        CompareResult compareResult =
plcSoftware.CompareTo(project.ProjectLibrary.MasterCopyFolder.MasterCopies[0]);
        WriteResult(compareResult.RootElement, string.Empty);
    }
}
```

修改以下程序代码以比较 PLC 的软件与所连接 PLC 的软件：

```
private static void ComparePlcToOnlinePlc(PlcSoftware plcSoftware)
{
    if (plcSoftware != null)
    {
        CompareResult compareResult = plcSoftware.CompareToOnline();
        WriteResult(compareResult.RootElement, string.Empty);
    }
}
```

7.13.2 确定 PLC 的状态

要求

- Openness 应用程序连接到 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已通过 Openness 应用程序打开一个项目。
  请参见打开项目 (页 96)

应用

您可确定项目中的一个 PLC 或所有 PLC 的状态。

Openness 分为以下状态：

- 离线
- PLC 已连接（“连接”(Connecting)）
- 在线

- PLC 已断开（ “断开”(Disconnecting)）
- 不兼容
- 不可访问
- 受保护

程序代码

修改以下程序代码以确定某个 PLC 的状态：

```
public static OnlineState GetOnlineState(DeviceItem deviceItem)
{
    OnlineProvider onlineProvider = deviceItem.GetService<OnlineProvider>();
    return onlineProvider.State;
}
```

修改以下程序代码以确定项目中所有 PLC 的状态：

```
public static void DetermineOnlineStateOfAllProjectDevices(Project project)
{
    foreach (Device device in project.Devices)
    {
        foreach (DeviceItem deviceItem in device.DeviceItems)
        {
            OnlineProvider onlineProvider = deviceItem.GetService<OnlineProvider>();
            if (onlineProvider != null)
            {
                OnlineState state = onlineProvider.State;
            }
        }
    }
}
```

7.13.3 访问在线连接的参数

要求

- Openness 应用程序连接至 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)

应用

可使用 Public API 接口确定或设置在线连接的参数：

- 枚举 PLC 的可用连接模式
- 枚举 PLC 的可用接口
- 枚举分配的插槽
- 枚举子网和网关的可用地址
- 设置连接参数。

程序代码：确定连接参数

修改以下程序代码以枚举可用的连接模式、PC 接口和插槽：

```
public static void EnumerateConnectionModesOfPLC(DeviceItem deviceItem)
{
    OnlineProvider onlineProvider = deviceItem.GetService<OnlineProvider>();
    if (onlineProvider == null)
    {
       return; // Only cpu device items can provide OnlineProvider service
    }
    // Accessing connection configuration object
    ConnectionConfiguration configuration = onlineProvider.Configuration;
    // Now access connection configuration members
    foreach (ConfigurationMode mode in configuration.Modes)
    {
        Console.WriteLine("Mode name:{0}", mode.Name);
        foreach (ConfigurationPcInterface pcInterface in mode.PcInterfaces)
        {
            Console.WriteLine("PcInterface name:{0}", pcInterface.Name);
            Console.WriteLine("PcInterface number:{0}", pcInterface.Number);
            foreach (ConfigurationTargetInterface targetInterface in
pcInterface.TargetInterfaces)
            {
                Console.WriteLine("TargetInterface:{0}", targetInterface.Name);
            }
        }
    }
}
```

您也可以按名称访问连接模式和 PC 接口：

```
public static ConfigurationTargetInterface
GetTargetInterfaceForOnlineConnection(OnlineProvider onlineProvider)
{
    ConnectionConfiguration configuration = onlineProvider.Configuration;
    ConfigurationMode mode = configuration.Modes.Find("PN/IE");
    ConfigurationPcInterface pcInterface = mode.PcInterfaces.Find("PLCSIM", 1);
    ConfigurationTargetInterface slot = pcInterface.TargetInterfaces.Find("2 X3");
    return slot;
}
```

修改以下程序代码以枚举 PC 接口上可用的子网和网关地址：

```
public static void EnumeratingPCInterfaceSubnetsAndGateways(ConfigurationPcInterface
pcInterface)
{
    foreach (ConfigurationSubnet subnet in pcInterface.Subnets)
    {
        Console.WriteLine("Subnet name:{0}", subnet.Name);
        foreach (ConfigurationGateway gateway in subnet.Gateways)
        {
            //Get the name of the gateway:
            Console.WriteLine("Gateway name:{0}", gateway.Name);
            //Get the IP address of each gateway:
            foreach (ConfigurationAddress gatewayAddress in gateway.Addresses)
            {
                Console.WriteLine("Gateway Address:{0}", gatewayAddress.Name);
            }
        }
    }
}
```

也可以按名称或 IP 地址访问子网和网关：

```
public static void AccessSubnetAndGatewayOfPCInterface(ConfigurationPcInterface
pcInterface)
{
    ConfigurationSubnet subnet = pcInterface.Subnets.Find("PN/IE_1");
    ConfigurationAddress subnetAddress = subnet.Addresses.Find("192.168.0.1");
    ConfigurationGateway gateway = subnet.Gateways.Find("Gateway 1");
    ConfigurationAddress gatewayAddress = gateway.Addresses.Find("192.168.0.2");
}
```

程序代码：设置连接参数

说明

设置连接参数时，将覆盖之前设置的所有连接参数。如果已直接在 TIA Portal 中设置连接参数，则无需调用 ApplyConfiguration。在调用 ApplyConfiguration 时，如果已存在到 PLC 的在线连接，则会触发异常。

修改以下程序代码以设置插槽参数：

```
public static void SetConnectionWithSlot(OnlineProvider onlineProvider)
{
    ConnectionConfiguration configuration = onlineProvider.Configuration;
    ConfigurationMode mode = configuration.Modes.Find(@"PN/IE");
    ConfigurationPcInterface pcInterface = mode.PcInterfaces.Find("PLCSIM", 1);
    // or network pc interface that is connected to plc
    ConfigurationTargetInterface slot = pcInterface.TargetInterfaces.Find("2 X3");
    configuration.ApplyConfiguration(slot);
    // After applying configuration, you can go online
    onlineProvider.GoOnline();
}
```

修改以下程序代码以设置网关地址参数：

```
public static void SetConnectionWithGatewayAddress(OnlineProvider onlineProvider, string
subnetName, string gatewayAddressName)
{
    ConnectionConfiguration configuration = onlineProvider.Configuration;
    ConfigurationMode mode = configuration.Modes.Find(@"PN/IE");
    ConfigurationPcInterface pcInterface = mode.PcInterfaces.Find("PLCSIM", 1);
    // or network pc interface that is connected to plc
    ConfigurationSubnet subnet = pcInterface.Subnets.Find(subnetName);
    ConfigurationAddress gatewayAddress = subnet.Addresses.Find(gatewayAddressName);
    configuration.ApplyConfiguration(gatewayAddress);
    // After applying configuration, you can go online
    onlineProvider.GoOnline();
}
```

修改以下程序代码以设置子网地址参数：

```
public static void SetConnectionWithSubnetAddress(OnlineProvider onlineProvider, string
subnetName)
{
    ConnectionConfiguration configuration = onlineProvider.Configuration;
    ConfigurationMode mode = configuration.Modes.Find(@"PN/IE");
    ConfigurationPcInterface pcInterface = mode.PcInterfaces.Find("PLCSIM", 1);
    // or network pc interface that is connected to plc
    ConfigurationSubnet subnet = pcInterface.Subnets.Find(subnetName);
    ConfigurationAddressComposition addresses = subnet.Addresses;
    configuration.ApplyConfiguration(addresses[0]);
    // After applying configuration, you can go online
    onlineProvider.GoOnline();
}
```

7.13.4 建立或断开到 PLC 的在线连接

要求

- Openness 应用程序连接到 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)
- 所有设备均已枚举。
  请参见访问设备项 (页 227)。

应用

可以建立到 PLC 的在线连接，或断开现有的在线连接。

程序代码

修改以下程序代码以建立或断开至 PLC 的在线连接：

```
public static void SetOnlineConnection(DeviceItem deviceItem)
{
    OnlineProvider onlineProvider = deviceItem.GetService<OnlineProvider>();
    if (onlineProvider == null) { return; }
    // Go online
    if (onlineProvider.Configuration.IsConfigured)
    {
        onlineProvider.GoOnline();
    }
   // Go offline
   onlineProvider.GoOffline();
}
```

还可以建立或断开到项目中所有可用 PLC 的在线连接。

```
public static void SetOnlineConnectionForAllPLCs(Project project)
{
    foreach (Device device in project.Devices)
    {
        foreach (DeviceItem deviceItem in device.DeviceItems)
        {
            OnlineProvider onlineProvider = deviceItem.GetService<OnlineProvider>();
            if (onlineProvider != null)
            {
                // Establish online connection to PLC:
                onlineProvider.GoOnline();

                // ...

                // Disconnect online connection to PLC:
                onlineProvider.GoOffline();
            }
        }
    }
}
```

7.13.5 块

7.13.5.1 查询“程序块”组

要求

- Openness 应用程序连接至 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)
- 已在项目中确定 PLC。

程序代码

修改以下程序代码以查询“程序块”(Program blocks) 组：

```
private static void GetBlockGroupOfPLC(PlcSoftware plcsoftware)
//Retrieves the system group of a block
{
    PlcBlockSystemGroup blockGroup = plcsoftware.BlockGroup;
}
```

7.13.5.2 枚举用户自定义的块组

要求

- Openness 应用程序连接至 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)
- 已在项目中确定 PLC。

应用

子组将以递归形式考虑在内，以进行枚举。

程序代码：枚举所有组

修改以下程序代码以枚举用户自定义的块组：

```
//Enumerates all block user groups including sub groups
private static void EnumerateAllBlockGroupsAndSubgroups(PlcSoftware plcsoftware)
{
    foreach (PlcBlockUserGroup blockUserGroup in plcsoftware.BlockGroup.Groups)
    {
        EnumerateBlockUserGroups(blockUserGroup);
    }
}

private static void EnumerateBlockUserGroups(PlcBlockUserGroup blockUserGroup)
{
    foreach (PlcBlockUserGroup subBlockUserGroup in blockUserGroup.Groups)
    {
        EnumerateBlockUserGroups(subBlockUserGroup);
        // recursion
    }
}
```

程序代码：访问组

修改以下程序代码以访问所选的用户自定义块组：

```
//Gives individual access to a specific block user group
private static void AccessBlockusergroup(PlcSoftware plcsoftware)
{
    PlcBlockUserGroupComposition userGroupComposition = plcsoftware.BlockGroup.Groups;
    PlcBlockUserGroup plcBlockUserGroup = userGroupComposition.Find("MyUserfolder");
}
```

7.13.5.3 枚举所有块

要求

- Openness 应用程序连接至 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)
- 已在项目中确定 PLC。

应用

如果已知程序块名称，则可对其进行针对性访问。

程序代码：枚举所有块

修改以下程序代码以枚举所有块组的块：

```
private static void EnumerateAllBlocks(PlcSoftware plcsoftware)
//Enumerates all blocks
{
    foreach (PlcBlock block in plcsoftware.BlockGroup.Blocks)
    {
        // Do something...
    }
}
```

程序代码：访问特定块

修改以下程序代码以访问特定块：

```
private static void AccessASingleBlock(PlcSoftware plcsoftware)
//Gives individual access to a block
{
    // The parameter specifies the name of the block
    PlcBlock block = plcsoftware.BlockGroup.Blocks.Find("MyBlock");
}
```

7.13.5.4 查询块/用户数据类型的信息

要求

- Openness 应用程序连接至 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)

应用

Public API 支持针对程序和数据块以及用户数据类型查询以下信息：

- UTC 时间格式的时间戳。
  通过时间戳检查以下内容：
  - 最后一次编译块的时间。
  - 最后一次更改块的时间。
- “一致性”(Consistency) 属性
  在以下情况下，将“一致性”(Consistency) 属性设置为“True”：
  - 块已编译成功。
  - 块在编译后尚未更改。
  - 外部对象未进行任何需要重新编译的更改。
- 使用的编程语言（仅限程序和数据块）
- 块编号
- 块名称
- 块作者
- 块系列
- 块标题
- 块版本

更多详细信息，请参见 Openness 对象模型的块和类型 (页 50)。

程序代码

修改以下程序代码以查询上述信息：

```
private static void GetPlcBlockInformation(PlcSoftware plcSoftware)
{
    PlcBlock plcBlock = plcSoftware.BlockGroup.Blocks.Find("MyBlock");
    // Read information
    DateTime compileDate = plcBlock.CompileDate;
    DateTime modifiedDate = plcBlock.ModifiedDate;
    bool isConsistent = plcBlock.IsConsistent;
    int blockNumber = plcBlock.Number;
    string blockName = plcBlock.Name;
    ProgrammingLanguage programmingLanguage = plcBlock.ProgrammingLanguage;
    string blockAuthor = plcBlock.HeaderAuthor;
    string blockFamily = plcBlock.HeaderFamily;
    string blockTitle = plcBlock.HeaderName;
    System.Version blockVersion = plcBlock.HeaderVersion;
}
```

参见

导入组态数据 (页 345)

7.13.5.5 删除块

要求

- Openness 应用程序连接至 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)
- PLC 未处于在线状态。

程序代码

修改以下程序代码以删除块：

```
//Runs through block group and deletes blocks
private static void DeleteBlocks(PlcSoftware plcsoftware)
{
    PlcBlockSystemGroup group = plcsoftware.BlockGroup;
    //  or BlockUserGroup group = ...;
    for (int i = group.Blocks.Count - 1; i >= 0; i--)
    {
        PlcBlock block = group.Blocks[i];
        if (block != null)
        {
            block.Delete();
        }
    }
}
```

参见

导入组态数据 (页 345)

7.13.5.6 创建块组

要求

- Openness 应用程序连接至 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)

程序代码

修改以下程序代码以创建块组：

```
private static void CreateBlockGroup(PlcSoftware plcsoftware)
//Creates a block group
{
    PlcBlockSystemGroup systemGroup = plcsoftware.BlockGroup;
    PlcBlockUserGroupComposition groupComposition = systemGroup.Groups;
    PlcBlockUserGroup myCreatedGroup = groupComposition.Create("MySubGroupName");
}
```

参见

导入组态数据 (页 345)

7.13.5.7 删除块组

要求

- Openness 应用程序连接至 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)
- PLC 未处于在线状态。

程序代码

修改以下程序代码以删除块组：

```
// Deletes user groups from PlcBlockSystemGroup or PlcBlockUserGroup
private static void DeleteBlockFolder(PlcSoftware plcSoftware)
{
    PlcBlockUserGroup group = plcSoftware.BlockGroup.Groups.Find("myGroup");
    //PlcBlockSystemGroup group = plcSoftware.BlockGroup;
    PlcBlockUserGroupComposition subgroups = group.Groups;
    PlcBlockUserGroup subgroup = subgroups.Find("myUserGroup");
    if (subgroup != null)
    {
        subgroup.Delete();
    }
}
```

参见

导入组态数据 (页 345)

7.13.5.8 查询系统块的系统组

要求

- Openness 应用程序连接至 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)

程序代码：

修改以下程序代码以确定系统为系统块创建的组：

```
PlcSoftware plcSoftware = ...
foreach (PlcSystemBlockGroup systemGroup in plcSoftware.BlockGroup.SystemBlockGroups)
{
    foreach (PlcSystemBlockGroup group in systemGroup.Groups)
    {
        PlcBlockComposition pbComposition = group.Blocks;
        foreach (PlcBlock block in pbComposition)
        {
            //add your code here
        }
    }
}
```

7.13.5.9 枚举系统子组

要求

- Openness 应用程序连接至 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)

程序代码：枚举所有系统子组

修改以下程序代码以枚举所有系统块的系统子组：

```
//Retrieves the system generated group for system blocks
private static void GetSystemgroupForSystemblocks(PlcSoftware plcSoftware)
{
    PlcSystemBlockGroupComposition systemBlockGroups =
plcSoftware.BlockGroup.SystemBlockGroups;
    if (systemBlockGroups.Count != 0)
    {
        PlcSystemBlockGroup sbSystemGroup = systemBlockGroups[0];
        foreach (PlcSystemBlockGroup group in sbSystemGroup.Groups)
        {
            EnumerateSystemBlockGroups(group);
        }
    }
}
private static void EnumerateSystemBlockGroups(PlcSystemBlockGroup systemBlockGroup)
{
    foreach (PlcSystemBlockGroup group in systemBlockGroup.Groups)
    {
        // recursion EnumerateSystemBlockGroups(group);
    }
}
```

程序代码：访问特定子组

修改以下程序代码以访问特定子组：

```
private static void AccessSbGroup(PlcSystemBlockGroup systemBlockGroup)
{
    PlcSystemBlockGroup group1 = systemBlockGroup.Groups.Find("User group XYZ");
    PlcSystemBlockGroup group2 = group1.Groups.Find("User group ZYX");
}
```

参见

添加外部文件 (页 272)

7.13.5.10 添加外部文件

要求

- Openness 应用程序连接到 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已通过 Openness 应用程序打开一个项目：
  请参见打开项目 (页 96)

应用

可以向 PLC 添加外部文件。此外部文件存储在文件系统的指定路径下。

支持以下格式：

- STL
- SCL
- DB
- UDT

说明

不支持访问“外部源文件”(External source files) 文件夹中的组。
如果指定的文件扩展名不是 *.AWL、*.SCL, *.DB 或 *UDT，则会触发异常。

程序代码

修改以下程序代码以在“外部源文件”(External source files) 文件夹中创建块的外部文件。

```
private static void CreateBlockFromFile(PlcSoftware plcSoftware)
// Creates a block from a AWL, SCL, DB or UDT file
{
    PlcExternalSource externalSource =
plcSoftware.ExternalSourceGroup.ExternalSources.CreateFromFile("SomeBlockNameHere","SomePa
thHere");
}
```

7.13.5.11 生成块的源文件

要求

- Openness 应用程序连接至 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)
- PLC 未处于在线状态。

应用

API 接口支持生成 STL 或 SCL 块、数据块和 PLC 类型（用户数据类型）的 UTF-8 格式源文件。要生成块的源文件，调用 `PlcExternalSourceSystemGroup` 实例上的 `GenerateSource` 方法。

生成的源文件的大小取决于此函数的生成选项：

- GenerateOptions.None
  仅生成提供的块的源文件。
- GenerateOptions.WithDependencies
  生成包括所有从属对象的源文件。

接口 `Siemens.Engineering.SW.ExternalSources.IGenerateSource` 指示可以生成源。

块仅支持 STL 和 SCL 编程语言。在以下情况下会触发异常：

- 编程语言不是 STL 或 SCL
- 目标位置已存在同名文件

用户数据类型仅支持“*.udt”文件扩展名。在以下情况下会触发异常：

- 对于 DB，文件扩展名不是“*.db”
- 对于 STL 块，文件扩展名不是“*.awl”
- 对于 SCL 块，文件扩展名不是“*.scl”

程序代码

修改以下程序代码，根据块和类型生成源文件：

```
//method declaration
...
PlcExternalSourceSystemGroup.GenerateSource

(IEnumerable<Siemens.Engineering.SW.ExternalSources.IGenerateSource>
plcBlocks, FileInfo sourceFile, GenerateOptions generateOptions);
...
//examples
...
var blocks = new List<PlcBlock>(){block1};
var fileInfo = new FileInfo(@"C:\temp\SomePathHere.scl");

PlcExternalSourceSystemGroup systemGroup = ...;

systemGroup.GenerateSource(blocks, fileInfo, GenerateOptions.WithDependencies);

// exports all blocks and with all their dependencies(e.g. called blocks, used DBs or UDTs)
// as ASCII text into the provided source file.
...
or
..
var types = new List<PlcType>(){udt1};
var fileInfo = new FileInfo(@"C:\temp\SomePathHere.udt");

PlcExternalSourceSystemGroup systemGroup = ...;

systemGroup.GenerateSource(types, fileInfo, GenerateOptions.WithDependencies );

// exports all data types and their used data types into the provided source file.
...
```

参见

导入组态数据 (页 345)

7.13.5.12 从源生成块

要求

- Openness 应用程序连接到 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)
- PLC 未处于在线状态。

应用

可以从“外部源文件”(External source files) 组的所有外部文件生成块。仅支持格式为 ASCII 的外部文件。

说明

不支持访问“外部源文件”(External source files) 文件夹中的组。

现有块将被覆盖。

如果在调用期间发生错误，则会触发 Exception。每个错误消息的前 256 个字符将包含在 Exception 的通知中。执行 `GenerateBlocksFromSource` 方法前，会将项目复位到处理状态。

程序代码

修改以下程序代码以在“外部源文件”(External source files) 组的所有外部文件中生成块。

```
// Creates a block from an external source file
PlcSoftware plcSoftware = ...;
foreach (PlcExternalSource plcExternalSource in
plcSoftware.ExternalSourceGroup.ExternalSources)
{
    plcExternalSource.GenerateBlocksFromSource();
}
```

7.13.5.13 删除用户数据类型

要求

- Openness 应用程序连接至 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)
- PLC 未处于在线状态。

程序代码

修改以下程序代码以删除用户类型：

```
private static void DeleteUserDataType(PlcSoftware plcSoftware)
{
    PlcTypeSystemGroup typeGroup = plcSoftware.TypeGroup;
    PlcTypeComposition dataTypes = typeGroup.Types;
    PlcType dataType = dataTypes.Find("DataTypeName");
    if (dataType != null)
    {
        dataType.Delete();
    }
}
```

参见

导入组态数据 (页 345)

7.13.5.14 删除外部文件

要求

- Openness 应用程序连接到 TIA Portal
  请参见连接到 TIA Portal (页 69)
- 已通过 Openness 应用程序打开一个项目：
  请参见 打开项目 (页 96)
- PLC 未处于在线状态。

程序代码

修改以下程序代码以从“外部源文件”(External source files) 组中删除外部文件。

说明

不支持访问“外部源文件”(External source files) 组中的组。

```
// Deletes an external source file
private static void DeleteExternalSource(PlcSoftware plcSoftware)
{
    PlcExternalSource externalSource =
plcSoftware.ExternalSourceGroup.ExternalSources.Find("myExternalsource");
    externalSource.Delete();
}
```

7.13.5.15 启动块编辑器

要求

- Openness 应用程序连接至 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)
- 已通过用户界面打开 TIA Portal 的实例。

程序代码

修改以下程序代码以在 TIA Portal 实例中为 `PlcBlock` 类型的对象参考启动关联编辑器：

```
//Opens a block in a block editor
private static void StartBlockEditor(PlcSoftware plcSoftware)
{
    PlcBlock plcBlock = plcSoftware.BlockGroup.Blocks.Find("MyBlock");
    plcBlock.ShowInEditor();
}
```

修改以下程序代码以在 TIA Portal 实例中为 PlcType 类型的对象参考打开关联编辑器：

```
//Opens a udt in udt editor
private static void StartPlcTypEditor(PlcSoftware plcSoftware)
{
    PlcTypeComposition types = plcSoftware.TypeGroup.Types;
    PlcType udt = types.Find("my_udt");
    udt.ShowInEditor();
}
```

参见

导入组态数据 (页 345)

7.13.6 工艺对象

7.13.6.1 工艺对象的功能概述

在 TIA Portal Openness 中，可为已定义的任务选择相应的工艺对象功能。这些任务可在 TIA Portal 之外通过公共 API 进行调用。

这样，每个任务都有特定的代码组件。

功能

工艺对象支持以下功能：

- 查询工艺对象的组成 (页 282)
- 创建工艺对象 (页 283)
- 删除工艺对象 (页 284)
- 编译工艺对象 (页 285)
- 枚举工艺对象 (页 286)
- 查找工艺对象 (页 287)
- 枚举工艺对象的参数 (页 287)
- 查找工艺对象的参数 (页 288)
- 读取工艺对象的参数 (页 289)
- 写入工艺对象的参数 (页 290)

参见

标准库 (页 35)

应用程序 (页 28)

Openness 对象模型 (页 45)

7.13.6.2 工艺对象和版本概述

工艺对象

下表列出了公共 API 中可用的工艺对象。

CPU	固件版本	工艺对象	工艺对象的版本
S7-1200	≥ V4.2	TO_PositioningAxis	V6.0
		TO_CommandTable	
		PID_Compact	V2.3
		PID_3Step	
		PID_Temp	V1.1

CPU	固件版本	工艺对象	工艺对象的版本
S7-1500	< V2.0	High_Speed_Counter	V3.0
		SSI_Absolute_Encoder	V2.0
	≥ V2.0	TO_SpeedAxis	V3.0
		TO_PositioningAxis	
		TO_ExternalEncoder	
		TO_SynchronousAxis	
		TO_OutputCam	
		TO_CamTrack	
		TO_MeasuringInput	
		TO_Cam (S7-1500T)[1)]	
		High_Speed_Counter	
		SSI_Absolute_Encoder	V2.0
		PID_Compact	V2.3
		PID_3Step	
		PID_Temp	V1.1
		CONT_C	
		CONT_S	
		TCONT_CP	
		TCONT_S	
S7-300/400	任意	CONT_C	V1.1
		CONT_S	
		TCONT_CP	
		TCONT_S	
		TUN_EC[2)]	
		TUN_ES[2)]	
		PID_CP[2)]	V2.0
		PID_ES[2)]	
		AXIS_REF	Any

1) 该工艺对象不支持以下 Openness 功能：写入参数。

2) 该工艺对象不支持以下 Openness 功能：枚举参数、查找参数、读取参数、写入参数。

说明

S7-1500 Motion Control

将单独处理 S7-1500 上的工艺对象 TO_OutputCam、TO_CamTrack 和 TO_MeasuringInput。

更多信息，请参见“S7-1500 Motion Control (页 300)”部分。

7.13.6.3 数据类型概述

将 TIA Portal 中的工艺对象参数的数据类型映射到公共 API 中的 C# 数据类型。

数据类型

下表列出了具体的数据类型映射：

格式	TIA Portal 中的数据类型	C# 中的数据类型
二进制数字	Bool	bool
	BBool	bool
	Byte	byte
	Word	ushort
	DWord	uint
	LWord	ulong
整型	SInt	sbyte
	Int	short
	Dint	int
	LInt	long
	USInt	byte
	UInt	ushort
	UDint	uint
	ULInt	ulong
浮点数	Real	float
	LReal	double
	Time	double
字符串	Char	char

格式	TIA Portal 中的数据类型	C# 中的数据类型
	WChar	char
	String	string
	WString	string
硬件数据类型	HW_*	ushort
	Block_*	ushort

* TIA Portal 项目中设备类型扩展的占位符

7.13.6.4 查询工艺对象的组成

要求

- Openness 应用程序将连接至 TIA Portal。
  参见“连接到 TIA Portal (页 69)”
- 已打开一个项目。
  参见“打开项目 (页 96)”
- 在项目中确定 PLC。
  参见“查询 PLC 和 HMI 目标 (页 160)”

程序代码

修改以下程序代码以获取 PLC 的所有工艺对象：

```
private static void GetTechnologicalObjectsOfPLC(PlcSoftware plcSoftware)
// Retrieves all technology objects of a PLC
{
    TechnologicalInstanceDBGroup technologicalObjectGroup =
plcSoftware.TechnologicalObjectGroup;
    TechnologicalInstanceDBComposition technologicalObjects =
technologicalObjectGroup.TechnologicalObjects;
}
```

参见

标准库 (页 35)

7.13.6.5 创建工艺对象

要求

- 将 Openness 应用连接到 TIA Portal。
  请参见 连接到 TIA Portal (页 69)
- 已打开一个项目。
  参见“打开项目 (页 96)”
- 在项目中确定 PLC。
  参见“查询 PLC 和 HMI 目标 (页 160)”

应用

只能创建在“工艺对象和版本概述 (页 279)”部分中列出的工艺对象。如果工艺对象不支持或参数错误，则将发生异常错误。

说明

S7-1500 Motion Control

将单独处理 S7-1500 上的工艺对象 TO_OutputCam、TO_CamTrack 和 TO_MeasuringInput。
更多信息，请参见“S7-1500 Motion Control (页 300)”部分。

程序代码

修改以下程序代码，创建一个工艺对象并将其添加到现有的 PLC 中：

```
private static void CreateTechnologicalObject(PlcSoftware plcSoftware)
// Create a technology object and add to technology object composition
{
    TechnologicalInstanceDBComposition technologicalObjects =
plcSoftware.TechnologicalObjectGroup.TechnologicalObjects;

    string nameOfTO = "PID_Compact_1"; // How the technology object should be named
    string typeOfTO = "PID_Compact"; // How the technology object type is called, e.g. in
    // "Add new technology object"-dialog
    Version versionOfTO = new Version("2.3"); // Version of technology object
    TechnologicalInstanceDB technologicalObject = technologicalObjects.Create(nameOfTO,
typeOfTO, versionOfTO);
}
```

有关工艺对象的可能值和名称组合、类型和版本，请参见“工艺对象和版本概述 (页 279)”部分。

参见

标准库 (页 35)

7.13.6.6 删除工艺对象

要求

- Openness 应用程序将连接至 TIA Portal。
  参见“连接到 TIA Portal (页 69)”
- 已打开一个项目。
  参见“打开项目 (页 96)”
- 在项目中确定 PLC。
  参见“查询 PLC 和 HMI 目标 (页 160)”
- 该工艺对象存在。
  参见“查找工艺对象 (页 287)”

程序代码

修改以下程序代码以删除工艺对象：

```
private static void DeleteTechnologicalObject(TechnologicalInstanceDB technologicalObject)
// Delete a technology object from DB composition and from PLC
{
    technologicalObject.Delete();
}
```

参见

标准库 (页 35)

7.13.6.7 编译工艺对象

要求

- Openness 应用程序将连接至 TIA Portal。
  参见“连接到 TIA Portal (页 69)”
- 项目已打开。
  参见“打开项目 (页 96)”
- 在项目中确定 PLC。
  参见“查询 PLC 和 HMI 目标 (页 160)”
- 该工艺对象存在。
  参见“创建工艺对象 (页 283)”

程序代码：编译工艺对象

修改以下程序代码以编译工艺对象：

```
private static void CompileSingleTechnologicalObject(TechnologicalInstanceDB
technologicalObject)
// Compile a single technology object
{
    ICompilable singleCompile = technologicalObject.GetService<ICompilable>();
    CompilerResult compileResult = singleCompile.Compile();
}
```

程序代码：编译工艺对象组

修改以下程序代码以编译工艺对象组：

```
private static void CompileTechnologicalObjectGroup(PlcSoftware plcSoftware)
// Compile technology object group
{
    TechnologicalInstanceDBGroup technologicalObjectGroup =
plcSoftware.TechnologicalObjectGroup;
    ICompilable groupCompile = technologicalObjectGroup.GetService<ICompilable>();
    CompilerResult compileResult = groupCompile.Compile();
}
```

编译结果

工艺对象的编译结果将进行递归存储。

有关对编译结果进行递归评估的示例，请参见“编译项目 (页 99)”部分。

参见

标准库 (页 35)

7.13.6.8 枚举工艺对象

要求

- Openness 应用程序将连接至 TIA Portal。
  参见“连接到 TIA Portal (页 69)”
- 已打开一个项目。
  参见“打开项目 (页 96)”
- 在项目中确定 PLC。
  参见“查询 PLC 和 HMI 目标 (页 160)”

程序代码

修改以下程序代码以枚举工艺对象：

```
private static void EnumerateTechnologicalObjects(PlcSoftware plcSoftware)
// Enumerate all technology objects
{
    TechnologicalInstanceDBComposition technologicalObjects =
plcSoftware.TechnologicalObjectGroup.TechnologicalObjects;
    foreach (TechnologicalInstanceDB technologicalObject in technologicalObjects)
    {
        // Do something ...
    }
}
```

参见

标准库 (页 35)

7.13.6.9 查找工艺对象

要求

- Openness 应用程序将连接至 TIA Portal。
  参见“连接到 TIA Portal (页 69)”
- 已打开一个项目。
  参见“打开项目 (页 96)”
- 在项目中确定 PLC。
  参见“查询 PLC 和 HMI 目标 (页 160)”

程序代码

修改以下程序代码以查找特定的工艺对象：

```
private static void FindTechnologicalObject(PlcSoftware plcSoftware)
// Find a specific technology object by its name
{
    TechnologicalInstanceDBComposition technologicalObjects =
plcSoftware.TechnologicalObjectGroup.TechnologicalObjects;
    string nameOfTO = "PID_Compact_1";
    TechnologicalInstanceDB technologicalObject = technologicalObjects.Find(nameOfTO);
}
```

参见

标准库 (页 35)

7.13.6.10 枚举工艺对象的参数

要求

- Openness 应用程序将连接至 TIA Portal。
  参见“连接到 TIA Portal (页 69)”
- 已打开一个项目。
  参见“打开项目 (页 96)”
- 在项目中确定 PLC。
  参见“查询 PLC 和 HMI 目标 (页 160)”

- 该工艺对象存在。
  参见“创建工艺对象 (页 283)”或“查找工艺对象的参数 (页 288)”
- 该工艺对象 (页 279)支持此功能。

程序代码

修改以下程序代码以枚举特定工艺对象的参数：

```
private static void EnumerateParameters(PlcSoftware plcSoftware)
// Enumerate parameters of a technology object
{
    string nameOfTO = "PID_Compact_1";
    TechnologicalInstanceDB technologicalObject =
plcSoftware.TechnologicalObjectGroup.TechnologicalObjects.Find(nameOfTO);

    foreach (TechnologicalParameter parameter in technologicalObject.Parameters)
    {
        // Do something ...
    }
}
```

参见

标准库 (页 35)

查找工艺对象 (页 287)

7.13.6.11 查找工艺对象的参数

要求

- Openness 应用程序将连接至 TIA Portal。
  参见“连接到 TIA Portal (页 69)”
- 已打开一个项目。
  参见“打开项目 (页 96)”
- 在项目中确定 PLC。
  参见“查询 PLC 和 HMI 目标 (页 160)”
- 该工艺对象存在。
  参见“创建工艺对象 (页 283)”
- 该工艺对象 (页 279)支持此功能。

程序代码

修改以下程序代码以查找特定工艺对象的参数：

```
private static void FindParameterOfTechnologicalObject(PlcSoftware plcSoftware)
// Find parameters of a technology object
{
    string nameOfTO = "PID_Compact_1";
    TechnologicalInstanceDB technologicalObject =
plcSoftware.TechnologicalObjectGroup.TechnologicalObjects.Find(nameOfTO);

    string nameOfParameter = "Config.InputUpperLimit";
    TechnologicalParameter parameter =
technologicalObject.Parameters.Find(nameOfParameter);
}
```

不同工艺对象的参数

S7-1200 Motion Control 的参数 (页 292)

S7-1500 Motion Control 的参数 (页 300)

PID 控制的参数 (页 316)

计数的参数 (页 317)

Easy Motion Control 的参数 (页 318)

参见

标准库 (页 35)

查找工艺对象 (页 287)

7.13.6.12 读取工艺对象的参数

要求

- Openness 应用程序将连接至 TIA Portal。
  参见“连接到 TIA Portal (页 69)”
- 已打开一个项目。
  参见“打开项目 (页 96)”
- 在项目中确定 PLC。
  参见“查询 PLC 和 HMI 目标 (页 160)”

- 该工艺对象存在。
  参见“创建工艺对象 (页 283)”
- 该工艺对象 (页 279)支持此功能。

程序代码

修改以下程序代码以读取特定工艺对象的参数：

```
private static void ReadParameterOfTechnologicalObject(PlcSoftware plcSoftware)
// Read parameters of a technology object
{
    string nameOfTO = "PID_Compact_1";
    TechnologicalInstanceDB technologicalObject =
plcSoftware.TechnologicalObjectGroup.TechnologicalObjects.Find(nameOfTO);

    string nameOfParameter = "Config.InputUpperLimit";
    TechnologicalParameter parameter =
technologicalObject.Parameters.Find(nameOfParameter);

    // Read from parameter
    string name = parameter.Name;
    object value = parameter.Value;
}
```

参见

标准库 (页 35)

查找工艺对象 (页 287)

7.13.6.13 写入工艺对象的参数

要求

- Openness 应用程序将连接至 TIA Portal。
  参见“连接到 TIA Portal (页 69)”
- 已打开一个项目。
  参见“打开项目 (页 96)”
- 在项目中确定 PLC。
  参见“查询 PLC 和 HMI 目标 (页 160)”

- 该工艺对象存在。
  参见“创建工艺对象 (页 283)”
- 该工艺对象 (页 279)支持此功能。

意外错误

执行以下操作时，将发生 EngineeringException：

- 为一个不支持写入访问的参数设置新值。
- 系统不支持参数的新值类型。

程序代码

修改以下程序代码，写入特定工艺对象的参数：

```
private static void WriteParameterOfTechnologicalObject(PlcSoftware plcSoftware)
// Write parameters of a technology object
{
    string nameOfTO = "PID_Compact_1";
    TechnologicalInstanceDB technologicalObject =
plcSoftware.TechnologicalObjectGroup.TechnologicalObjects.Find(nameOfTO);

    string nameOfParameter = "Config.InputUpperLimit";
    TechnologicalParameter parameter =
technologicalObject.Parameters.Find(nameOfParameter);

    // Write to parameter if the value is writable
    object value = 3.0
    parameter.Value = value;
}
```

不同工艺对象的参数

S7-1200 Motion Control 的参数 (页 292)

S7-1500 Motion Control 的参数 (页 300)

PID 控制的参数 (页 316)

计数的参数 (页 317)

Easy Motion Control 的参数 (页 318)

参见

标准库 (页 35)

查找工艺对象 (页 287)

7.13.6.14 S7-1200 Motion Control

基于硬件地址连接 PROFIdrive

要求

- Openness 应用程序已连接至 TIA Portal。
  请参见连接到 TIA Portal (页 69)。
- 已打开一个项目。
  请参见打开项目 (页 96)。
- 已在项目中确定 S7-1200 PLC。
- 项目中包含 PROFIdrive，且已连接 S7-1200 PLC。
- 工艺对象已存在。
  请参见创建工艺对象 (页 283)。

程序代码

修改以下程序代码，基于硬件地址连接 PROFIdrive 和“TO_PositioningAxis”。

```
//An instance of the technology object axis is already available in the program before
private static void ConnectingDrive(TechnologicalInstanceDB technologicalObject)
{
    //Set axis to PROFIdrive mode
    technologicalObject.Parameters.Find("Actor.Type").Value = 1;

    //Set axis to drive mode
    technologicalObject.Parameters.Find("_Actor.Interface.DataConnection").Value = 0;

    //Set connection to adress of drive. The output will be set automatically.
    technologicalObject.Parameters.Find("_Actor.Interface.ProfiDriveIn").Value = "%I68.0";
    technologicalObject.Parameters.Find("Sensor.Sensor[1].Interface.Number").Value = 1;
    // 1 = Encoder1, 2 = Encoder2;
}
```

基于硬件地址连接 PROFIdrive 的编码器

要求

- Openness 应用程序已连接至 TIA Portal。
  请参见连接到 TIA Portal (页 69)。
- 已打开一个项目。
  请参见打开项目 (页 96)。
- 已在项目中确定 S7-1200 PLC。
- 项目中包含 PROFIdrive，且已连接 S7-1200 PLC。
- 工艺对象已存在。
  请参见创建工艺对象 (页 283)。

程序代码

修改以下程序代码，基于硬件地址连接编码器和“TO_PositioningAxis”：

```
//An instance of the technology object axis is already available in the program before
private static void ConnectingEncoder(TechnologicalInstanceDB technologicalObject)
{
    //Set axis to PROFIdrive mode
    technologicalObject.Parameters.Find("Actor.Type").Value = 1;

    //Set the encoder mode

technologicalObject.Parameters.Find("_Sensor.Sensor[1].Interface.EncoderConnection").Value
 = 7;

    //Set axis to use PROFINET encoder

technologicalObject.Parameters.Find("_Sensor.Sensor[1].Interface.DataConnection").Value =
0;

    //Set connection to adress of drive. The output will be set automatically.
    technologicalObject.Parameters.Find("_Sensor.Sensor[1].Interface.ProfiDriveIn").Value
= "%I68.0";
    technologicalObject.Parameters.Find("Sensor.Sensor[1].Interface.Number").Value = 1;
    // 1 = Encoder1, 2 = Encoder2;
}
```

基于硬件地址连接模拟量驱动装置

要求

- Openness 应用程序已连接至 TIA Portal。
  请参见连接到 TIA Portal (页 69)。
- 已打开一个项目。
  请参见打开项目 (页 96)。
- 已在项目中确定 S7-1200 PLC。
- 项目中包含模拟量驱动装置，且已连接 S7-1200 PLC。
- 工艺对象已存在。
  请参见创建工艺对象 (页 283)。

程序代码

修改以下程序代码，基于硬件地址连接模拟量驱动装置和“TO_PositioningAxis”：

```
//An instance of the technology object axis is already available in the program before
private static void ConnectingEncoder(TechnologicalInstanceDB technologicalObject)
{
    //Set axis to analog drive mode
    technologicalObject.Parameters.Find("Actor.Type").Value = 0;

    //Set axis to drive mode
    technologicalObject.Parameters.Find("_Actor.Interface.DataConnection").Value = 0;

    //Set connection to analog adress of drive
    technologicalObject.Parameters.Find("_Actor.Interface.Analog").Value = "%QW64";
}
```

基于硬件地址连接模拟量驱动装置的编码器

要求

- Openness 应用程序已连接至 TIA Portal。
  请参见连接到 TIA Portal (页 69)。
- 已打开一个项目。
  请参见打开项目 (页 96)。
- 已在项目中确定 S7-1200 PLC。
- 项目中包含模拟量驱动装置，且已连接 S7-1200 PLC。
- 工艺对象已存在。
  请参见创建工艺对象 (页 283)。

程序代码

修改以下程序代码，基于硬件地址连接编码器和“TO_PositioningAxis”：

```
//An instance of the technology object axis is already available in the program before
//Connecting by High Speed Counter mode
private static void ConnectingEncoder(TechnologicalInstanceDB technologicalObject)
{
    //Set axis to analog drive mode
    technologicalObject.Parameters.Find("Actor.Type").Value = 0;

    //Set encoder for high-speed counter mode

technologicalObject.Parameters.Find("_Sensor.Sensor[1].Interface.EncoderConnection").Value
 = 4;
    technologicalObject.Parameters.Find("_Sensor.Sensor[1].Interface.HSC.Name").Value =
"HSC_1"
}
    //An instance of the technology object axis is already available in the program before
    //Connecting by PROFINET/PROFIBUS telegram
    private static void ConnectingEncoder(TechnologicalInstanceDB
    technologicalObject)
{
    //Set axis to analog drive mode
    technologicalObject.Parameters.Find("Actor.Type").Value = 0;
    //Set encoder for PROFINET/PROFIBUS mode

technologicalObject.Parameters.Find("_Sensor.Sensor[1].Interface.EncoderConnection").Value
 = 7;

technologicalObject.Parameters.Find("_Sensor.Sensor[1].Interface.DataConnection").Value =
"Encoder";
    technologicalObject.Parameters.Find("_Sensor.Sensor[1].Interface.ProfiDriveIn").Value
= "%I68.0";
    technologicalObject.Parameters.Find("Sensor.Sensor[1].Interface.Number").Value = 1;
    // 1 = Encoder1, 2 = Encoder2;
}
```

通过数据块连接驱动装置

要求

- Openness 应用程序已连接至 TIA Portal。
  请参见连接到 TIA Portal (页 69)。
- 已打开一个项目。
  请参见打开项目 (页 96)。
- 已在项目中确定 S7-1200 PLC。
- 项目中包含有数据块，且设置为“未优化”(Not optimized)
  如果该数据块为 PROFIdrive 轴类型，则该数据块中包含一个该类型的变量。如，PD_TEL3。
  对于模拟量驱动装置，该数据块中包含一个 WORD 数据类型的变量。
- 工艺对象已存在。
  请参见创建工艺对象 (页 283)。

程序代码

修改以下程序代码，通过数据块连接 PROFIdrive 和“TO_PositioningAxis”。

```
//An instance of the technology object axis is already available in the program before
private static void ConfigureDrivewithDataBlock(TechnologicalInstanceDB
technologicalObject)
{
    //Set axis to PROFIdrive mode
    technologicalObject.Parameters.Find("Actor.Type").Value = 1;

    //Set axis to data block mode
    technologicalObject.Parameters.Find("_Actor.Interface.DataConnection").Value = 1;

    //Set the variable in the data block
    technologicalObject.Parameters.Find("_Actor.Interface.DataBlock").Value =
"Data_block_1.Member_of_type_PD_TEL3";
}
```

程序代码

修改以下程序代码，通过数据块连接模拟量驱动装置和“TO_PositioningAxis”：

```
//An instance of the technology object axis is already available in the program before
//Connecting an analog drive with data block.
private static void ConfigureDrivewithDataBlock(TechnologicalInstanceDB
technologicalObject)
{
    //Set axis to analog mode
    technologicalObject.Parameters.Find("Actor.Type").Value = 0;

    //Set the variable in the data block
    technologicalObject.Parameters.Find("_Actor.Interface.Analog").Value =
"Data_block_1.Static_1";
}
```

通过数据块连接编码器

要求

- Openness 应用程序已连接至 TIA Portal。
  请参见连接到 TIA Portal (页 69)。
- 已打开一个项目。
  请参见打开项目 (页 96)。
- 已在项目中确定 S7-1200 PLC。
- 项目中包含有数据块，且设置为 “未优化”(Not optimized)
  对于 PROFIdrive，则该数据块中包含一个该类型的变量。如，PD_TEL3
- 工艺对象已存在。
  请参见创建工艺对象 (页 283)。

程序代码

修改以下程序代码以根据数据块连接编码器：

```
//An instance of the technology object axis is already available in the program before
private static void ConfigureEncoderwithDataBlock(TechnologicalInstanceDB
technologicalObject)
{
    //Set axis to PROFIdrive mode depending by axis type. 1 = PROFIdrive, 0 = Analog Drive.
    technologicalObject.Parameters.Find("Actor.Type").Value = 1;

    //Set the encoder mode

technologicalObject.Parameters.Find("_Sensor.Sensor[1].Interface.EncoderConnection").Value
 = 7;

    //Set axis to data block mode

technologicalObject.Parameters.Find("_Sensor.Sensor[1].Interface.DataConnection").Value =
1;

    //Set the variable in the data block. For PD_TEL3 and PD_TEL4 "Encoder1" or "Encoder2".
    technologicalObject.Parameters.Find("_Sensor.Sensor[1].Interface.DataBlock").Value =
"Data_block_1.Member_of_Type_PD_TEL3";
}
```

TO_PositioningAxis 和 TO_CommandTable 的参数

有关所有可用的参数列表，敬请访问 internet (https://support.industry.siemens.com/cs/cn/zh/view/109744932) 网页中的产品信息“TIA Portal Openness 中工艺对象的参数”。

每个参数都包含以下属性：

- Openness 中的名称
- Openness 中的数据类型
- 默认访问方式

说明

在 TIA Portal 中，“Openness 中的名称”(Name in Openness) 列位于工艺对象组态的“参数”(Parameter) 视图中。该列同样包含在以下关联的表格中。

更多信息

更多信息，敬请访问 internet (https://support.industry.siemens.com/cs/cn/zh/view/109741731) 中的《SIMATIC STEP 7 S7-1200 Motion Control 功能手册》。

7.13.6.15 S7-1500 Motion Control

创建和查找 TO_OutputCam、TO_CamTrack 和 TO_MeasuringInput

要求

- Openness 应用程序连接到 TIA Portal。
  请参见连接到 TIA Portal (页 69)。
- 已打开一个项目。
  请参见打开项目 (页 96)。
- 已在项目中确定 S7-1500 PLC。
- 已在项目中确定 TO_PositioningAxis、TO_SynchronousAxis 或 TO_ExternalEncoder 类型的工艺对象。

应用

输出凸轮、凸轮轨迹和测量输入工艺对象与定位轴、同步轴或外部编码器工艺对象相关。要访问输出凸轮、凸轮轨迹或测量输入工艺对象，请使用服务 OutputCamMeasuringInputContainer。

程序代码：创建和查找输出凸轮、凸轮轨迹和测量输入工艺对象

修改以下程序代码以创建或查找输出凸轮、凸轮轨迹或测量输入工艺对象：

```
//An instance of the technology object under which the TO_OutputCam, TO_CamTrack or
TO_MeasuringInput should be created is already available in the program before
private static void CreateFind_OutputcamCamtrackMeasuringinput(TechnologicalInstanceDB
technologyObject)
{
    //Retrieve service OutputCamMeasuringInputContainer
    OutputCamMeasuringInputContainer container =
    technologyObject.GetService<OutputCamMeasuringInputContainer>();
    //Get access to TO_OutputCam / TO_CamTrack container
    TechnologicalInstanceDBComposition outputcamCamtrackContainer = container.OutputCams;

    //Find technology object TO_OutputCam or TO_CamTrack
    TechnologicalInstanceDB outputCam = outputcamCamtrackContainer.Find("OutputCamName");
    TechnologicalInstanceDB camTrack = outputcamCamtrackContainer.Find("CamTrackName");

    //Create new technology object TO_OutputCam or TO_CamTrack
    TechnologicalInstanceDB newOutputCam =
    outputcamCamtrackContainer.Create("NewOutputCamName", "TO_OutputCam",
    new Version(3, 0));
    TechnologicalInstanceDB newCamTrack =
    outputcamCamtrackContainer.Create("NewCamTrackName", "TO_CamTrack", new Version(3, 0));

    //Get access to TO_MeasuringInput container
    TechnologicalInstanceDBComposition measuringInputContainer = container.MeasuringInputs;

    //Find technology object TO_MeasuringInput
    TechnologicalInstanceDB measuringInput =
    measuringInputContainer.Find("MeasuringInputName");

    //Create new technology object TO_MeasuringInput
    TechnologicalInstanceDB newMeasuringInput =
    measuringInputContainer.Create("NewMeasuringInput", "TO_MeasuringInput",
    new Version(3, 0));
}
```

S7-1500 Motion Control 的参数

S7-1500 Motion Control 工艺对象的大多数参数可直接映射到数据块变量中，但仍有一些附加参数无法直接到数据块中。在 Openness 中，直接映射的参数与工艺对象参数视图的“数据导航”中的顺序相同。直接映射的参数后跟其它参数，其顺序与表中的顺序相同。

直接映射到工艺对象数据块变量中的参数

如文中所述，通常可访问除以下变量外的所有工艺对象数据块变量：

- 只读变量
- VREF 数据类型的变量
- “InternalToTrace”结构的变量
- “ControlPanel”结构的变量

有关直接映射的参数的更多信息，请参见以下手册的附录：

- 《SIMATIC S7-1500 Motion Control 功能手册》：
  https://support.industry.siemens.com/cs/ww/en/view/109739589 (https://support.industry.siemens.com/cs/cn/zh/view/109739589)
- 《SIMATIC S7-1500T Motion Control 功能手册》：
  https://support.industry.siemens.com/cs/ww/en/view/109481326 (https://support.industry.siemens.com/cs/cn/zh/view/109481326)

对于某些映射到只读数据块变量的工艺参数，需要在 PublicAPI 中为可写状态。允许的值与下面的数据块变量值相同。下表列出了受影响的参数：

Openness 中的名称	数据类型	TO_SpeedAxis	TO_PositioningAxis	TO_SynchronousAxis	TO_ExternalEncoder
Actor.Type	int	X	X	X	
Actor.Interface.EnableDriveOutput	bool	X	X	X	
Actor.Interface.DriveReadyInput	bool	X	X	X	
VirtualAxis.Mode	uint	X	X	X	
Sensor[n].Existent[1)]	bool		X	X	
Sensor[n].Interface.Number[1)]	uint		X	X	
Sensor[n].Type[1)]	int		X	X	
Sensor.Interface.Number	uint				X
Sensor.Type	int				X

Openness 中的名称	数据类型	TO_OutputCam	TO_MeasuringInput
Interface.LogicOperation	int	X	
Parameter.MeasuringInputType	int		X

1) S7-1500 PLC：n=1；S7-1500T PLC：1≤n≤4

未直接映射到工艺对象数据块变量中的参数

对于 S7-1500 Motion Control 工艺对象，未直接映射到数据块变量中的以下附加参数可用：

Openness 中的名称	功能视图中的名称	可能的值	Openness 中的数据类型	TO_SpeedAxis	TO_PositioningAxis TO_SynchronousAxis	TO_ExternalEncoder
_Properties.MotionType	对应于“位置的技术单位”的轴类型	0：线性 1：旋转型	int		X	X
_Units.LengthUnit	位置单位	请参见变量 Units.LengthUnit[2)]	uint		X	X
_Units.VelocityUnit	速度单位	请参见变量 Units.VelocityUnit[2)]	uint	X	X	X
_Units.TorqueUnit	力矩单位	请参见变量 Units.TorqueUnit[2)]	uint	X	X	
_Units.ForceUnit	力单位	请参见变量 Units.ForceUnit[2)]	uint		X	
_Actor.Interface.Telegram	驱动器报文	报文编号 [3)]	uint	X	X	
_Actor.Interface.EnableDriveOutputAddress	驱动器输出地址	PublicAPI 对象	SW.Tags.PlcTag	X	X	

Openness 中的名称	功能视图中的名称	可能的值	Openness 中的数据类型	TO_SpeedAxis	TO_PositioningAxis TO_SynchronousAxis	TO_ExternalEncoder
_Actor.Interface.DriveReadyInputAddress	驱动器就绪输入地址	PublicAPI 对象	SW.Tags.PlcTag	X	X	
_Sensor[n].Interface.Telegram[4)]	编码器报文	报文编号 [3)]	uint		X	
_Sensor[n].ActiveHoming.DigitalInputAddress[4)]	数字量输入	PublicAPI 对象	SW.Tags.PlcTag		X	
_Sensor[n].PassiveHoming.DigitalInputAddress[4)]	数字量输入	PublicAPI 对象	SW.Tags.PlcTag		X	
_PositionLimits_HW.MinSwitchAddress	硬件下限限位开关输入	PublicAPI 对象	SW.Tags.PlcTag		X	
_PositionLimits_HW.MaxSwitchAddress	硬件上限限位开关输入	PublicAPI 对象	SW.Tags.PlcTag		X	
_Sensor.Interface.Telegram	编码器报文	报文编号 [3)]	uint			X
_Sensor.PassiveHoming.DigitalInputAddress	数字量输入	PublicAPI 对象	SW.Tags.PlcTag			X

2) S7-1500 Motion Control 功能手册的单位变量 (TO) 章节给出了可能的值

3) S7-1500 Motion Control 功能手册的 PROFIdrive 报文章节给出了可能的值

4) S7-1500 PLC：n=1；S7-1500T PLC：1≤n≤4

对于输出凸轮、凸轮轨迹和测量输入工艺对象，支持以下附加参数：

Openness 中的名称	功能视图中的名称	可能的值	数据类型
_AssociatedObject	相关对象	PublicAPI 对象	SW.TechnologicalObjects.TechnologicalInstanceDB

程序代码：直接映射的数据块变量

修改以下程序代码以访问直接映射的参数：

```
//An instance of the technology object is already available in the program before
private static void ReadWriteDataBlockTag(TechnologicalInstanceDB technologyObject)
{
    //Read value from data block tag "ReferenceSpeed"
    double value =
    (double)technologyObject.Parameters.Find("Actor.DriveParameter.ReferenceSpeed").Value;

    //Write data block tag "ReferenceSpeed"
    technologyObject.Parameters.Find("Actor.DriveParameter.ReferenceSpeed").Value = 3000.0;
}
```

程序代码：附加参数

修改以下程序代码以访问附加参数：

```
//An instance of the technology object is already available in the program before
private static void ReadWriteAdditionalParameter(TechnologicalInstanceDB technologyObject)
{
    //Read additional parameter "_Properties.MotionType"
    uint value = (uint)technologyObject.Parameters.Find("_Properties.MotionType").Value;

    //Write additional parameter "_Properties.MotionType"
    technologyObject.Parameters.Find("_Properties.MotionType").Value = 1;
}
```

更多信息

更多相关信息，请参见：

- 《SIMATIC S7-1500 Motion Control 功能手册》：
  https://support.industry.siemens.com/cs/ww/en/view/109739589 (https://support.industry.siemens.com/cs/cn/zh/view/109739589)
- 《SIMATIC S7-1500T Motion Control 功能手册》：
  https://support.industry.siemens.com/cs/ww/en/view/109481326 (https://support.industry.siemens.com/cs/cn/zh/view/109481326)

连接驱动器

要求

- Openness 应用程序连接到 TIA Portal。
  请参见连接到 TIA Portal (页 69)。
- 已打开一个项目。
  请参见打开项目 (页 96)。
- 已在项目中确定 S7-1500 PLC。
- 已在项目中确定 TO_SpeedAxis、TO_PositioningAxis 或 TO_SynchronousAxis 类型的工艺对象。
- 已在项目中确定驱动器。

应用

要将轴与驱动器相连，需要在单个调用中同时指定多个值。公共 API 类型 AxisEncoderHardwareConnectionInterface 提供了以下可用来连接和断开执行器或传感器接口的方法：

方法	说明
void Connect(HW.DeviceItem moduleInOut)	连接包含输入和输出地址的模块
void Connect(HW.DeviceItem moduleIn, HW.DeviceItem moduleOut)	连接包含输入或输出地址的模块
void Connect(HW.DeviceItem moduleIn, HW.DeviceItem moduleOut, ConnectOption connectOption)	连接包含输入或输出地址的模块（用于指定附加 ConnectOption）
void Connect(HW.Channel channel)	连接至某个通道
void Connect(int addressIn, int addressOut, ConnectOption connectOption)	连接用于直接指定位地址的过载
void Connect(string pathToDBMember)	连接至某个数据块变量
void Connect(SW.Tags.PlcTag outputTag)	连接至某个 PLC 变量
void Disconnect()	断开现有连接

说明

传感器部件的自动连接

请注意，用户界面中的相关行为在此处仍适用。执行器接口通过其中一种连接方法连接，且报文中包含传感器部件时，系统将自动连接该传感器部件。

可使用以下只读属性确定工艺对象的连接方式：只有在存在特定类型的连接时，才能设置相应的连接值。

属性	数据类型	说明
IsConnected	bool	TRUE：接口已连接 FALSE：接口未连接
InputOutputModule	HW.DeviceItem	已连接包含输入和输出地址的模块
InputModule	HW.DeviceItem	已连接包含输入地址的模块 存在与包含输入和输出地址的模块的连接时，也会设置该值。
OutputModule	HW.DeviceItem	已连接包含输出地址的模块 存在与包含输入和输出地址的模块的连接时，也会设置该值。
InputAddress	int	所连接对象的逻辑输入地址，如 256。
OutputAddress	int	所连接对象的逻辑输出地址，如 256。
ConnectOption	ConnectOption	进行连接时已设置的 ConnectOption 值： • 默认值 只能选择被识别为有效连接伙伴的模块。 • AllowAllModules 相当于在用户界面中选择 “显示所有模块”(Show all modules)。
Channel	HW.Channel	已连接通道
PathToDBMember	string	已连接工艺对象数据块变量
OutputTag	SW.Tags.PlcTag	已连接 PLC 变量（模拟量连接）
SensorIndexInActorTelegram	int	已在执行器报文中连接传感器部件 该属性仅与传感器接口相关。 0：编码器未连接 1：编码器已连接至报文中的第一个传感器接口 2：编码器已连接到报文中的第二个传感器接口 对于执行器接口，该值始终为 0。

说明

访问传感器接口

要访问传感器的接口，可将 SensorInterface[m] 设置为 0≤m≤3。

程序代码：void Connect(HW.DeviceItem moduleInOut)

修改以下程序代码以连接包含输入和输出地址的混合模块：

```
//An instance of technology object and device item is already available in the program
before
private static void UseServiceAxisHardwareConnectionProvider(TechnologicalInstanceDB
technologyObject, DeviceItem devItem)
{
    //Retrieve service AxisHardwareConnectionProvider
    AxisHardwareConnectionProvider connectionProvider =
    technologyObject.GetService<AxisHardwareConnectionProvider>();

    //Connect ActorInterface with DeviceItem
    connectionProvider.ActorInterface.Connect(devItem);

    //Connect first SensorInterface with DeviceItem
    connectionProvider.SensorInterface[0].Connect(devItem);

    //Check ConnectionState of ActorInterface
    bool actorInterfaceConnectionState = connectionProvider.ActorInterface.IsConnected;

    //Check ConnectionState of first SensorInterface
    bool sensorInterfaceConnectionState =
    connectionProvider.SensorInterface[0].IsConnected;
}
```

连接编码器

要求

- Openness 应用程序连接到 TIA Portal。
  请参见连接到 TIA Portal (页 69)。
- 已打开一个项目。
  请参见打开项目 (页 96)。
- 已在项目中确定 S7-1500 PLC。

- 已在项目中确定 TO_ExternalEncoder 类型的工艺对象。
- 已在项目中确定可提供 PROFIdrive 报文 81 或 83 的对象。

应用

要将外部编码器工艺对象与编码器硬件相连，需要在单个调用中同时指定多个值。公共 API 类型 AxisEncoderHardwareConnectionInterface 提供了以下可用来连接和断开传感器接口的方法：

方法	说明
void Connect(HW.DeviceItem moduleInOut)	连接包含输入和输出地址的模块
void Connect(HW.DeviceItem moduleIn, HW.DeviceItem moduleOut)	连接包含输入或输出地址的模块
void Connect(HW.DeviceItem moduleIn, HW.DeviceItem moduleOut, ConnectOption connectOption)	连接包含输入或输出地址的模块（用于指定附加 ConnectOption）
void Connect(HW.Channel channel)	连接至某个通道
void Connect(int addressIn, int addressOut, ConnectOption connectOption)	连接用于直接指定位地址的过载
void Connect(string pathToDBMember)	连接至某个数据块变量
void Connect(SW.Tags.PlcTag outputTag)	与连接编码器无关
void Disconnect()	断开现有连接

可使用以下只读属性确定工艺对象的连接方式：只有在存在特定类型的连接时，才能设置相应的连接值。

属性	数据类型	说明
IsConnected	bool	TRUE：接口已连接 FALSE：接口未连接
InputOutputModule	HW.DeviceItem	已连接包含输入和输出地址的模块
InputModule	HW.DeviceItem	已连接包含输入地址的模块 存在与包含输入和输出地址的模块的连接时，也会设置该值。
OutputModule	HW.DeviceItem	已连接包含输出地址的模块 存在与包含输入和输出地址的模块的连接时，也会设置该值。
InputAddress	int	所连接对象的逻辑输入地址，例如 256。
OutputAddress	int	所连接对象的逻辑输出地址，例如 256。

属性	数据类型	说明
ConnectOption	ConnectOption	进行连接时已设置的 ConnectOption 值： • Default 只能选择被识别为有效连接伙伴的模块。 • AllowAllModules 相当于在用户界面中选择“显示所有模块”(Show all modules)。
Channel	HW.Channel	已连接通道
PathToDBMember	string	已连接数据块变量
OutputTag	SW.Tags.PlcTag	与连接编码器无关
SensorIndexInActorTelegram	int	已连接传感器报文 该属性仅与传感器接口相关。 0：编码器未连接 1：编码器已连接至报文中的第一个传感器接口 2：编码器已连接到报文中的第二个传感器接口 对于执行器接口，该值始终为 0。

程序代码：连接编码器

修改以下程序代码以连接外部编码器工艺对象：

```
//An instance of technology object and device item is already available in the program
before
private static void UseServiceEncoderHardwareConnectionProvider(TechnologicalInstanceDB
technologyObject, DeviceItem devItem)
{
    //Retrieve service EncoderHardwareConnectionProvider
    EncoderHardwareConnectionProvider connectionProvider =
    technologyObject.GetService<EncoderHardwareConnectionProvider>();

    //Connect SensorInterface with DeviceItem
    connectionProvider.SensorInterface.Connect(devItem);

    //Check ConnectionState of SensorInterface
    bool sensorInterfaceConnectionState = connectionProvider.SensorInterface.IsConnected;
}
```

将输出凸轮和凸轮轨迹连接至硬件

要求

- Openness 应用程序连接到 TIA Portal。
  请参见连接到 TIA Portal (页 69)。
- 已打开一个项目。
  请参见打开项目 (页 96)。
- 已在项目中确定 S7-1500 PLC。
- 已在项目中确定 TO_OutputCam 或 TO_CamTrack 类型的工艺对象。
- 已在项目中确定数字量输出模块，例如 TM Timer DIDQ。

应用

要将输出凸轮或凸轮轨迹工艺对象与一个数字量输出相连，需要在单个调用中同时指定多个值。公共 API 类型 OutputCamHardwareConnectionProvider 提供了以下可用来连接和断开执行器或传感器接口的方法：

方法	说明
void Connect(HW.Channel channel)	连接至某个通道
void Connect(SW.Tags.PlcTag outputTag)	连接至某个 PLC 变量
void Connect(int address)	连接用于直接指定位地址的过载
void Disconnect()	断开现有连接

可使用以下只读属性确定工艺对象的连接方式：

属性	数据类型	说明
IsConnected	bool	TRUE：工艺对象已连接 FALSE：工艺对象未连接
Channel	HW.Channel	已连接通道
OutputTag	SW.Tags.PlcTag	已连接 PLC 变量
OutputAddress	int	所连接对象的逻辑输出地址，例如 256。

程序代码：连接输出凸轮或凸轮轨迹工艺对象

修改以下程序代码以连接输出凸轮或凸轮轨迹工艺对象：

```
//An instance of technology object and channel item is already available in the program
before
private static void UseServiceOutputCamHardwareConnectionProvider(TechnologicalInstanceDB
technologyObject, Channel channel)

{
    //Retrieve service OutputCamHardwareConnectionProvider
    OutputCamHardwareConnectionProvider connectionProvider =
    technologyObject.GetService<OutputCamHardwareConnectionProvider>();

    //Connect technology object with Channel
    connectionProvider.Connect(channel);

    //Check ConnectionState of technology object
    bool connectionState = connectionProvider.IsConnected;
}
```

将测量输入与硬件相连

要求

- Openness 应用程序连接到 TIA Portal。
  请参见连接到 TIA Portal (页 69)。
- 已打开一个项目。
  请参见打开项目 (页 96)。
- 已在项目中确定 S7-1500 PLC。
- 已在项目中确定 TO_MeasuringInput 类型的工艺对象。
- 已在驱动器或项目中确定数字量输入模块，例如 TM Timer DIDQ。

应用

要将测量输入工艺对象与一个数字量输入相连，需要在单个调用中同时指定多个值。公共 API 类型 MeasuringInputHardwareConnectionProvider 提供了以下可用来连接和断开执行器或传感器接口的方法：

方法	说明
void Connect(HW.Channel channel)	连接至某个通道
void Connect(HW.DeviceItem moduleIn, int channelIndex)	连接至模块（用于指定附加通道索引）
void Connect(int address)	连接用于直接指定位地址的过载
void Disconnect()	断开现有连接

可使用以下只读属性确定工艺对象的连接方式：

属性	数据类型	说明
IsConnected	bool	TRUE：工艺对象已连接 FALSE：工艺对象未连接
InputModule	HW.DeviceItem	已连接包含输入地址的模块
ChannelIndex	int	与 InputModule 相关的所连接通道的索引
Channel	HW.Channel	已连接通道
InputAddress	int	所连接对象的逻辑输入地址，例如 256。

程序代码：连接测试输入工艺对象

修改以下程序代码以连接测量输入工艺对象：

```
//An instance of technology object and channel item is already available in the program
before
private static void
UseServiceMeasuringInputHardwareConnectionProvider(TechnologicalInstanceDB
technologyObject, Channel channel)
{
    //Retrieve service MeasuringInputHardwareConnectionProvider
    MeasuringInputHardwareConnectionProvider connectionProvider =
    technologyObject.GetService<MeasuringInputHardwareConnectionProvider>();

    //Connect technology object with Channel
    connectionProvider.Connect(channel);

    //Check ConnectionState of technology object
    bool connectionState = connectionProvider.IsConnected;
}
```

将同步轴与主值相连

要求

- Openness 应用程序连接到 TIA Portal。
  请参见连接到 TIA Portal (页 69)。
- 已打开一个项目。
  请参见打开项目 (页 96)。
- 已在项目中确定 S7-1500 PLC。
- 已在项目中确定用作主轴的 TO_PositioningAxis、TO_SynchronousAxis 或 TO_ExternalEncoder 类型的工艺对象。
- 已在项目中确定用作从轴的 TO_SynchronousAxis 类型的工艺对象。

应用

要将同步轴工艺对象与主值相连，需要在单个调用中同时指定多个值。公共 API 类型 SynchronousAxisMasterValues 提供了以下可用来连接和断开主值的方法。主值可作为设定值耦合（S7-1500 PLC、S7-1500T PLC）或实际值耦合 (S7-1500T PLC) 进行连接。所有方法和属性均与这两种耦合类型相关。

方法	说明
int IndexOf (TechnologicalInstanceDB element)	返回主值的相应索引
bool Contains (TechnologicalInstanceDB element)	TRUE：容器包含主值 FALSE：容器不包含主值
IEnumerator GetEnumerator <TechnologicalInstanceDB>()	用于支持各个迭代
void Add (TechnologicalInstanceDB element)	将从轴与主值相连
bool Remove (TechnologicalInstanceDB element)	断开从轴与主值的连接 TRUE：成功断开连接 FALSE：未成功断开连接

可使用以下只读属性：

属性	数据类型	说明
Count	int	主值的计数
IsReadonly	bool	TRUE：容器为只读 FALSE：容器非只读
Parent	IEngineeringObject	返回容器的父级。 在这种情况下的父级指的是服务 SynchronousAxisMasterValues。
this [id] { get; }	TechnologicalInstanceDB	基于索引访问主值

程序代码：连接同步轴和主值

修改以下程序代码以将同步轴与主值相连：

```
//An instance of leading axis and following axis is already available in the program before
private static void UseServiceSynchronousAxisMasterValues(TechnologicalInstanceDB
masterTechnologyObject, TechnologicalInstanceDB synchronousTechnologyObject)
{
    //Retrieve service SynchronousAxisMasterValues
    SynchronousAxisMasterValues masterValues =
    synchronousTechnologyObject.GetService<SynchronousAxisMasterValues>();

    //Connect following axis and leading axis with setpoint coupling
    masterValues.SetPointCoupling.Add(masterTechnologyObject);

    //Get container of connected leading axis with setpoint coupling
    TechnologicalInstanceDBAssociation setPointMasterValues =
    masterValues.SetPointCoupling;

    //Remove connected leading axis with setpoint coupling
    masterValues.SetPointCoupling.Remove(masterTechnologyObject);

    //Connect following axis and leading axis with actual value coupling
    masterValues.ActualValueCoupling.Add(masterTechnologyObject);

    //Get container of connected leading axis with actual value coupling
    TechnologicalInstanceDBAssociation actualValueMasterValues =
    masterValues.ActualValueCoupling;

    //Remove connected leading axis with actual value coupling
    masterValues.ActualValueCoupling.Remove(masterTechnologyObject);
}
```

7.13.6.16 PID 控制

PID_Compact, PID_3Step, PID_Temp, CONT_C, CONT_S, TCONT_CP 和 TCONT_S 的参数

有关所有可用的参数列表，敬请访问 internet (https://support.industry.siemens.com/cs/ww/zh/view/109744932) 网页中的产品信息“TIA Portal Openness 中工艺对象的参数”。

每个参数都包含以下属性：

- 组态中工的名称 (TIA Portal)
- Openness 中的名称
- Openness 中的数据类型

- 默认访问方式
- 值范围

说明

在 TIA Portal 中，“Openness 中的名称”(Name in Openness) 列位于工艺对象组态的“参数”(Parameter) 视图中。

更多信息

更多信息，敬请访问 internet (https://support.industry.siemens.com/cs/ww/zh/view/108210036) 中的《SIMATIC S7-1200/S7-1500 PID control 功能手册》。

7.13.6.17 计数

High_Speed_Counter 和 SSI_Absolute_Encoder 的参数

有关所有可用的参数列表，敬请访问 internet (https://support.industry.siemens.com/cs/ww/zh/view/109744932) 网页中的产品信息“TIA Portal Openness 中工艺对象的参数”。

每个参数都包含以下属性：

- 组态中工的名称 (TIA Portal)
- Openness 中的名称
- Openness 中的数据类型
- 默认访问方式
- 值范围

更多信息

更多信息，敬请访问 internet (http://support.automation.siemens.com/WW/view/zh/59709820) 中的《SIMATIC S7-1500, ET 200MP, ET 200SP 计数、测量和定位输入 功能手册》。

7.13.6.18 Easy Motion Control

AXIS_REF 的参数

有关所有可用的参数列表，敬请访问 internet (https://support.industry.siemens.com/cs/ww/zh/view/109744932) 网页中的产品信息“TIA Portal Openness 中工艺对象的参数”。

每个参数都包含以下属性：

- 组态中工的名称 (TIA Portal)
- Openness 中的名称
- Openness 中的数据类型
- 默认访问方式
- 值范围

说明

在 TIA Portal 中，“Openness 中的名称”(Name in Openness) 列位于工艺对象组态的“参数”(Parameter) 视图中。

更多信息

有关 Easy Motion Control 的更多信息，请参见 STEP 7 (TIA Portal) 信息系统。

7.13.7 变量和变量表

7.13.7.1 设置“PLC 变量”编辑器

要求

- Openness 应用程序连接至 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)
- 已通过用户界面打开 TIA Portal 的实例。

程序代码

修改以下程序代码以在 TIA Portal 实例中启动 PlcTagTable 类型对象参考的相应编辑器：

```
//Opens tagtable in editor "Tags"
private static void OpenTagtableInEditor(PlcSoftware plcSoftware)
{
    PlcTagTable plcTagTable = plcSoftware.TagTableGroup.TagTables.Find("MyTagTable");
plcTagTable.ShowInEditor();
}
```

参见

导入组态数据 (页 345)

7.13.7.2 查询 PLC 变量的系统组

要求

- Openness 应用程序连接到 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)
- 从 PLC 设备项中检索 PlcSoftware 实例。
  请参见 查询 PLC 和 HMI 目标 (页 160)

程序代码

修改以下程序代码以查询 PLC 变量的系统组：

```
//Retrieves the plc tag table group from a plc
private PlcTagTableSystemGroup GetControllerTagfolder(PlcSoftware plcSoftware)
{
    PlcTagTableSystemGroup plcTagTableSystemGroup = plcSoftware.TagTableGroup;
    return plcTagTableSystemGroup;
}
```

7.13.7.3 枚举用户自定义的 PLC 变量组

要求

- Openness 应用程序连接到 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)
- 从 PLC 设备项中检索 PlcSoftware 实例。
  请参见 查询 PLC 和 HMI 目标 (页 160)

应用

子文件夹将被递归考虑以进行枚举。

程序代码：枚举用户自定义的 PLC 变量组

修改以下程序代码以枚举用户自定义的 PLC 变量组：

```
//Enumerates all plc tag table user groups including subgroups
private static void EnumeratePlcTagTableUserGroups(PlcSoftware plcSoftware)
{
    foreach (PlcTagTableUserGroup plcTagTableUsergroup in plcSoftware.TagTableGroup.Groups)
    {
        EnumerateTagTableUserGroups(plcTagTableUsergroup);
    }
}
private static void EnumerateTagTableUserGroups(PlcTagTableUserGroup tagTableUsergroup)
{
    foreach (PlcTagTableUserGroup plcTagTableUsergroup in tagTableUsergroup.Groups)
    {
        EnumerateTagTableUserGroups(plcTagTableUsergroup);
        // recursion
    }
}
```

程序代码：访问用户自定义的组

修改以下程序代码以访问用户自定义的 PLC 变量组：

```
//Gives individual access to a specific plc tag table user folder
private static void AccessPlcTagTableUserGroupWithFind(PlcSoftware plcSoftware, string
folderToFind)
{
    PlcTagTableUserGroupComposition plcTagTableUserGroupComposition =
plcSoftware.TagTableGroup.Groups;
    PlcTagTableUserGroup controllerTagUserFolder =
plcTagTableUserGroupComposition.Find(folderToFind);
    // The parameter specifies the name of the user folder
}
```

7.13.7.4 创建用户自定义的 PLC 变量组

要求

- Openness 应用程序连接到 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)

应用

API 接口支持创建用户自定义的 PLC 变量组。

程序代码

修改以下程序代码以创建用户自定义的 PLC 变量组：

```
//Creates a plc tag table user group
private static void CreatePlcTagTableUserGroup(PlcSoftware plcSoftware)
{
    PlcTagTableSystemGroup systemGroup = plcSoftware.TagTableGroup;
    PlcTagTableUserGroupComposition groupComposition = systemGroup.Groups;
    PlcTagTableUserGroup myCreatedGroup = groupComposition.Create("MySubGroupName");
    // Optional;
    // create a subgroup
    PlcTagTableUserGroup mySubCreatedGroup =
myCreatedGroup.Groups.Create("MySubSubGroupName");
}
```

7.13.7.5 删除用户自定义的 PLC 变量组

要求

- Openness 应用程序连接至 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)

应用

API 接口支持删除特定的用户自定义 PLC 变量表组。

程序代码

修改以下程序代码以删除特定的用户自定义 PLC 变量表组：

```
private static void DeletePlcTagTableUserGroup(PlcSoftware plcSoftware)
{
    PlcTagTableUserGroup group = plcSoftware.TagTableGroup.Groups.Find("MySubGroupName");
    if (group != null)
    {
        group.Delete();
    }
}
```

7.13.7.6 枚举文件夹中的 PLC 变量表

要求

- Openness 应用程序连接到 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)

程序代码：枚举 PLC 变量表

修改以下程序代码以枚举系统组或用户自定义组中的所有 PLC 变量表：

```
//Enumerates all plc tag tables in a specific system group or and user group
private static void EnumerateAllPlcTagTablesInFolder(PlcSoftware plcSoftware)
{
    PlcTagTableComposition tagTables = plcSoftware.TagTableGroup.TagTables;
    // alternatively, PlcTagTableComposition tagTables =
plcSoftware.TagTableGroup.Groups.Find("UserGroup XYZ").TagTables;
    foreach (PlcTagTable tagTable in tagTables)
    {
        // add code here
    }
}
```

程序代码：访问 PLC 变量表

修改以下程序代以码访问 PLC 变量表：

```
//Gives individual access to a specific Plc tag table
private static void AccessToPlcTagTableWithFind(PlcSoftware plcSoftware)
{
    PlcTagTableComposition tagTables = plcSoftware.TagTableGroup.TagTables;
    // alternatively, PlcTagTableComposition tagTables =
plcSoftware.TagTableGroup.Groups.Find("UserGroup XYZ").TagTables;
    PlcTagTable controllerTagTable = tagTables.Find("Tag table XYZ");
    // The parameter specifies the name of the tag table
}
```

7.13.7.7 从 PLC 变量表中查询信息

要求

- Openness 应用程序已连接 TIA Portal。
  参见“连接到 TIA Portal (页 69)”
- 项目已打开。
  参见“打开项目 (页 96)”

应用

通过 PLC 变量表，可访问常量或变量。变量表中的变量组合数等于变量表的变量数。

程序代码

修改以下程序代码以查询 PLC 变量表信息：

```
private static void AccessPlcConstantsUsingFind(PlcTagTable tagTable)
{
    PlcUserConstantComposition plcUserConstants = tagTable.UserConstants;
    PlcUserConstant plcUserConstant = plcUserConstants.Find("Constant XYZ");
    //PlcSystemConstantComposition plcSystemConstants = tagTable.SystemConstants;
    //PlcSystemConstant plcSystemConstant = plcSystemConstants.Find("Constant XYZ");
}
private static void EnumeratePlcTags(PlcTagTable tagTable)
{
    PlcTagComposition plcTags = tagTable.Tags;
    foreach (PlcTag plcTag in plcTags)
    {
        string name = plcTag.Name; string typeName = plcTag.DataTypeName;
        string logicalAddress = plcTag.LogicalAddress;
    }
}
private static void EnumeratePlcTagsUsingFind(PlcTagTable tagTable)
{
    PlcTagComposition plcTags = tagTable.Tags;
    PlcTag plcTag = plcTags.Find("Constant XYZ");
}
```

7.13.7.8 读取上次对 PLC 变量表进行更改的时间

要求

- Openness 应用程序连接至 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)

应用

时间戳的格式为 UTC。

程序代码

修改以下程序代码以读取特定 PLC 变量表的时间戳：

```
//Reads Time-Stamp of a plc Tag Table
private static void GetLastModificationDateOfTagtable(PlcSoftware plcSoftware)
{
    PlcTagTable plcTagTable = plcSoftware.TagTableGroup.TagTables.Find("MyTagTable");
    DateTime modifiedTagTableTimeStamp = plcTagTable.ModifiedTimeStamp;
}
```

7.13.7.9 从组中删除 PLC 变量表

要求

- Openness 应用程序连接到 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)

程序代码

修改以下程序代码以从组中删除特定的变量表：

```
//Deletes a PlcTagTable of a group
private static void DeletePlcTagTableInAGroup(PlcSoftware plcSoftware)
{
    PlcTagTableSystemGroup group = plcSoftware.TagTableGroup;
    PlcTagTable tagtable = group.TagTables.Find("MyTagTable");
    if (tagtable!= null)
    {
        tagtable.Delete();
    }
}
```

7.13.7.10 枚举 PLC 变量

要求

- Openness 应用程序已连接 TIA Portal。
  参见“连接到 TIA Portal (页 69)”
- 项目已打开。
  参见“打开项目 (页 96)”

程序代码：枚举变量表中的 PLC 变量

修改以下程序代码以枚举变量表中的所有 PLC 变量：

```
//Enumerates all plc tags in a specific tag table
private static void EnumerateAllPlcTagsInTagTable(PlcSoftware plcSoftware)
{
    PlcTagTable tagTable = plcSoftware.TagTableGroup.TagTables.Find("Tagtable XYZ");
    foreach (PlcTag tag in tagTable.Tags)
    {
        // add code here
    }
}
```

7.13.7.11 访问 PLC 变量

要求

- Openness 应用程序已连接 TIA Portal。
  参见“连接到 TIA Portal (页 69)”
- 项目已打开。
  参见“打开项目 (页 96)”

应用

使用类型 `PlcTagComposition` 表示一个 PLC 变量集。

程序代码：访问特定的 PLC 变量

修改以下程序代码以访问所需的 PLC 变量。此时，可访问以下属性：

- 名称（只读）
- 数据类型的名称
- 逻辑地址
- 注释
- ExternalAccessible
- ExternalVisible
- ExternalWritable

```
//Gives individual access to a specific plc tag
private static void AccessPlcTag(PlcTagTable tagTable)
{
    PlcTag tag = tagTable.Tags.Find("Tag XYZ");
    // The parameter specifies the name of the tag
}
```

程序代码：创建变量

修改以下程序代码：

```
private static void CreateTagInPLCTagtable(PlcSoftware plcsoftware)
// Create a tag in a tag table with default attributes
{
    string tagName = "MyTag";
    PlcTagTable table = plcsoftware.TagTableGroup.TagTables.Find("myTagTable");
    PlcTagComposition tagComposition = table.Tags;
    PlcTag tag = tagComposition.Create(tagName);
}
```

修改以下程序代码：

```
private static void CreateTagInPLCTagtable(PlcSoftware plcsoftware)
// Create a tag of data type bool and logical address not set
{
    string tagName = "MyTag";
    string dataType = "Bool";
    string logicalAddress ="";
    PlcTagTable table = plcsoftware.TagTableGroup.TagTables.Find("myTagTable");
    PlcTagComposition tagComposition = table.Tags;
    PlcTag tag = tagComposition.Create(tagName, dataType, logicalAddress);
}
```

程序代码：删除变量

修改以下程序代码：

```
private static void DeleteTagFromPLCTagtable(PlcSoftware plcsoftware)
// Deletes a single tag of a tag table
{
    string tagName = "MyTag";
    PlcTagTable table = plcsoftware.TagTableGroup.TagTables.Find("myTagTable");
    PlcTagComposition tagComposition = table.Tags;
    PlcTag tag = tagComposition.Find(tagName);
    if (tag != null)
    {
        tag.Delete();
    }
}
```

7.13.7.12 访问 PLC 常量

要求

- Openness 应用程序已连接 TIA Portal。
  参见“连接到 TIA Portal (页 69)”
- 项目已打开。
  参见“打开项目 (页 96)”

应用

使用类型 PlcUserConstantComposition 表示一个 PLC 用户常量集。此时，可访问以下属性：

- 名称（只读）
- 数据类型的名称
- 值

使用类型 PlcSystemConstantComposition 表示一个 PLC 系统常量集。此时，可访问以下属性：

- 名称（只读）
- 数据类型的名称（只读）
- 值（只读）

程序代码：创建用户常量

修改以下程序代码：

```
private static void CreateUserConstantInPLCTagtable(PlcSoftware plcsoftware)
// Create a user consrant in a tag table
{
    string constantName = "MyConstant";
    PlcTagTable table = plcsoftware.TagTableGroup.TagTables.Find("myTagTable");
    PlcUserConstantComposition userConstantComposition = table.Tags;
    PlcUserComnstant userConstant = userConstantComposition.Create(constantName);
}
```

程序代码：删除用户常量

修改以下程序代码：

```
private static void DeleteUserConstantFromPLCTagtable(PlcSoftware plcsoftware)
// Deletes a single user constant of a tag table
{
    PlcTagTable table = plcsoftware.TagTableGroup.TagTables.Find("myTagTable");
    PlcUserConstantComposition userConstantComposition = table.UserConstants;
    PlcTag userConstant = userConstantComposition.Find("MyConstant");
    if (userConstant != null)
    {
        userConstant.Delete();
    }
}
```

程序代码：访问系统常量

修改以下程序代码：

```
//Gives individual access to a specific system constant
private static void AccessSystemConstant(PlcTagTable tagTable)
{
    PlcTag systemConstant = tagTable.SystemConstants.Find("Constant XYZ");
    // The parameter specifies the name of the tag
}
```

参见

创建用户自定义的 PLC 变量组 (页 321)

删除用户自定义的 PLC 变量组 (页 322)

从组中删除 PLC 变量表 (页 325)

访问 PLC 变量 (页 327)

设置“PLC 变量”编辑器 (页 318)

读取上次对 PLC 变量表进行更改的时间 (页 325)

7.14 基本概念

7.14.1 处理异常

通过公共 API 访问 TIA Portal 时的例外

在通过公共 API 执行 Openness 应用程序的过程中，发生的所有错误都会报告为例外。这些例外可能包含有助于更正所发生错误的信息。

以下两种类型的例外存在明显区别：

- Recoverable (Siemens.Engineering.EngineeringException)
  可以在不受此例外干扰的情况下继续访问 TIA Portal。或者，也可以取消与 TIA Portal 的连接。
  EngineeringExceptions 包括以下类型：
  - 安全相关例外 (EngineeringSecurityException)，例如缺少访问权限时。
  - 访问对象时的例外 (EngineeringObjectDisposedException)，例如访问不再存在的对象时。
  - 访问属性时的例外 (EngineeringNotSupportedException)，例如访问不存在的属性时。
  - 调用时的常规例外 (EngineeringTargetInvocationException)，例如有效调用公共 API 时仍发生错误。
  - 调用时的例外 (EngineeringRuntimeException)，例如无效的转换异常。
  - 调用终止时的例外情况 (EngineeringUserAbortException)，例如用户取消导入动作。

  EngineeringExceptions 具备以下属性：
  - `ExceptionMessageData messageData`: 包含触发例外的原因。
  - `ExceptionMessageData detailMessageData`: 包含有关原因的其它信息。结果将以 <IList> 形式返回.
  - `String message`：从 `MessageData` 和 `DetailMessageData` 返回结果。

  `ExceptionMessageData` 返回以下信息：
  - `String Text`：包含触发例外的原因。
  - `Int ServiceId`：返回触发该例外的服务 ID
  - `Int MessageId`：服务中的唯一 ID。
- NonRecoverable (Siemens.Engineering.NonRecoverableException)
  此例外将关闭 TIA Portal，与 TIA Portal 的连接也将断开。需要使用 Openness 应用程序重新启动 TIA Portal。

程序代码

以下示例显示了可以用来响应例外的选项：

```
try
{
    ...
}

catch(EngineeringSecurityException engineeringSecurityException)
{
    Console.WriteLine(engineeringSecurityException);
}

catch(EngineeringObjectDisposedException engineeringObjectDisposedException)
{
    Console.WriteLine(engineeringObjectDisposedException.Message);
}

catch(EngineeringNotSupportedException engineeringNotSupportedException)
{
    Console.WriteLine(engineeringNotSupportedException.MessageData.Text);
    Console.WriteLine();
    foreach(ExceptionMessageData detailMessageData in
engineeringNotSupportedException.DetailMessageData)
    {
        Console.WriteLine(detailMessageData.Text);
    }
}

catch (EngineeringTargetInvocationException)
{
    throw;
}

catch (EngineeringException)
{
    //Do not catch general exceptions
    throw;
}

catch(NonRecoverableException nonRecoverableException)
{
    Console.WriteLine(nonRecoverableException.Message);
}
```

7.14.2 使用关联

访问关联

关联是指类型级别两个或更多对象之间的关系。

TIA Portal Openness 支持通过索引和“foreach”循环访问关联。不支持通过 string name 进行直接访问。

属性

提供以下属性：

- `int Count`
- `bool IsReadonly`
- `IEngineeringObject Parent`
- `retType this [ int index ] { get; }`

方法

TIA Portal Openness 支持以下方法：

- `int IndexOf ( type )`：返回关联中已传送实例的索引。
- `bool Contains ( type )`：确定已传送实例是否包含在关联中。
- `IEnumerator GetEnumerator <retType>()`：在“foreach”循环中使用以访问对象。
- `void Add ( type )`[1]：将已传送的实例添加至关联。
- `void Remove ( type )`[1]：将已传送的实例从关联中移除。

[1]：并非所有关联均支持。

7.14.3 使用组合

访问组合

组合是关联的特殊情况。组合表达的是两个从属对象之间的语义关系。

属性

提供以下属性：

- `int Count`
- `bool IsReadonly`
- `IEngineeringObject Parent`
- `retType this [int index] {get;}`：对组合对象的索引访问。
  此类型访问仅应采取针对性方式，因为每个索引访问操作都超出了过程限制。

方法

TIA Portal Openness 支持以下方法：

- `retType Create (id, …)`：创建新的实例并将此实例添加到组合中。
  该方法的签名取决于创建实例的方式。并非所有组合都支持此方法。
- `type Find (id, …)`：使用已传送的 ID 扫描实例的组合。
  该搜索不是递归的。该方法的签名取决于搜索实例的方式。并非所有组合都支持此方法。
- `IEnumerator GetEnumerator<retType> ()`：在“foreach”循环中使用以访问对象。
- `Delete (type)`[1]：删除由当前对象参考指定的实例。
- `int IndexOf (type)`：返回组合中已传送实例的索引。
- `bool Contains (type)`：确定已传送实例是否包含在组合中。
- `void Import(string path, ImportOptions importOptions)`[1]：用于每一个包含可导入类型的组合。
  每个导入签名都包含一个“ImportOptions (页 345)”类型的组态参数（“None”、“Overwrite”），用户可以通过该参数控制导入行为。

[1]：并非所有组合均支持。

7.14.4 验证对象的相等性

应用

作为公共 API 的用户，可以检查对象是否与程序代码相同：

- 检查两个对象参考是否与运算符“==”相同。
- 使用 `System.Object.Equals()` 方法可检查这两个对象是否真的与 TIA Portal 相同。

程序代码

修改以下程序代码以检查对象参考类型：

```
...
//Composition
DeviceComposition sameCompA = project.Devices;
DeviceComposition sameCompB = project.Devices;
if (sameCompA.Equals(sameCompB))
{
    Console.WriteLine("sameCompA is equal to sameCompB");
}
if (!(sameCompA == sameCompB))
{
    Console.WriteLine("sameCompA is not reference equal to sameCompB");
}
DeviceComposition sameCompAsA = sameCompA;
if (sameCompAsA.Equals(sameCompA))
{
    Console.WriteLine("sameCompAsA is equal to sameCompA");
}
if (sameCompAsA == sameCompA)
{
    Console.WriteLine("sameCompAsA is reference equal to sameCompA");
}
MultiLingualGraphicComposition notSameComp = project.Graphics;
if (!sameCompA.Equals(notSameComp))
{
    Console.WriteLine("sameCompA is not equal to notSameComp");
}
```

7.14.5 属性的读取操作

属性的组操作和标准读取操作

TIA Portal Openness 支持通过以下类函数访问属性，这些类函数在对象级别均可用：

- 读访问的组操作
- 标准读取操作

组操作的程序代码

```
//Exercise GetAttributes and GetAttributeNames
//get all available attributes for a device,
//then get the names for those attributes, then display the results.
private static void DynamicTest(Project project)
{
    Device device = project.Devices[0];
    IList<string> attributeNames = new List<string>();
    IList<EngineeringAttributeInfo> attributes =
((IEngineeringObject)device).GetAttributeInfos();
    foreach (EngineeringAttributeInfo engineeringAttributeInfo in attributes)
    {
        string name = engineeringAttributeInfo.Name;
        attributeNames.Add(name);
    }
    IList<object> values = ((IEngineeringObject)device).GetAttributes(attributeNames);
    for (int i = 0; i < attributes.Count; i++)
    {
        Console.WriteLine("attribute name: " + attributeNames[i] + " value: " + values[i]);
    }
}
```

读访问的组操作

该类函数适用于任何对象：

```
public abstract IList<object> GetAttributes(IEnumerable<string>
names);
```

标准读取操作

可执行以下操作：

- 检索可用属性的名称：
  在 IEngineeringObject 中，调用类函数 GetAttributeInfos() (页 156)。
- 读取属性的通用类函数
  ```
  public abstract object GetAttribute(string name);
  ```

说明

动态属性不显示在 IntelliSense 中，因为其可用性取决于对象实例的状态。

导出/导入

8

8.1 概述

8.1.1 导入/导出的基本原理

简介

可以导出某些组态数据，然后在编辑之后再将数据重新导入同一项目或不同项目中。

说明

对于使用此处所描述的方法，手动修改和判断源文件，我们不承担任何义务，也不做任何保证。因此，西门子不对使用此描述的全部或部分所导致的任何后果负任何责任。

可导出和可导入的对象

以下组态数据也可通过公共 API 导入或导出：

表格 8-1 项目

对象	导出	导入
项目图形	√	√

表格 8-2 PLC

对象	导出	导入
块	√	√
专有技术保护块	√	–
故障安全块	√	–
系统块	√	–
PLC 变量表	√	√
PLC 变量和常量	√	√
用户数据类型	√	√

表格 8-3 HMI

对象	导出	导入
画面	√	√
画面模板	√	√
全局画面	√	√
弹出画面	√	√
滑入画面	√	√
脚本	√	√
文本列表	√	√
图形列表	√	√
周期	√	√
连接	√	√
变量表	√	√
变量	√	√

完全导出或导出开放式引用

如果上面列出的对象类型属于同一子树，则这些对象类型将与所有对象一起导出或导入。此规则同样适用于相同子树的引用对象。

但是，不能完全导出或导入其它子树中的引用对象。可以导出或导入这些对象的“开放式引用”。

只有属于可导出的对象的组时，相同子树的引用对象才能被导出。在导入/导出期间，对象上的所有动态化将被当作对象，并会被一同导出和导入。

导出内容包括组态期间所更改的所有对象属性。无论将来是否使用更改后的属性，这一点都适用。

示例：已为图形 IO 字段组态了“输入/输出”模式，并为属性“滚动条类型”选择了设置“单击后可见”。在组态过程中已将模式更改为“双状态”。在这种模式下，属性“滚动条类型”不可用。由于“滚动条类型”(Scroll bar type) 属性已更改，即使不使用该属性，它也会包含在导出中。

导入开放式引用

也可导入带开放式引用的对象（参见“导入组态数据 (页 345)”）。

如果引用对象包含在目标项目中，开放式引用将再次自动链接到对象类型。要进行导出，这些对象必须位于相同位置并被分配相同的名称。如果引用对象不包含在目标项目中，将无法解析开放式引用。不会创建其它对象来解析这些开放式引用。

导出和导入文件格式

导出和导入文件格式为 XML。只有 CAx 数据为 AML 格式。所有格式的方案定义在本手册的相关部分进行说明：

- HMI 设备中 XML 格式的数据 (页 354)
- PLC 设备中 XML 格式的数据 (页 410)
- AML 格式的 CAx 数据 (页 439)

导入和导出字体

还可导出和导入在对象上定义的字体。

导入项目中未包含的字体时，导入后会在对象上显示标准字体。不过，导入的字体将存储在数据管理中。

如果未在导入文件中分配字体属性，导入后将为属性分配默认值。

限制条件

导出格式为内部导出且仅对 TIA Portal Openness 的当前版本有效。在未来版本中导出格式可更改。

导入和导出过程中出现的所有错误将作为异常情况进行报告。
有关例外的更多信息，请参见“处理异常 (页 331)”部分。

参见

导入/导出的应用领域 (页 342)

导出组态数据 (页 343)

8.1.2 导入/导出的应用领域

简介

通过导入/导出功能可有针对性地导入特定对象。

可在外部程序中编辑导出的数据，或者在其他 TIA Portal 项目中原封不动的重新使用该数据。

如果导入文件的结构完全正确，则还可以导入外部创建的组态数据，而完全不必首先执行导出操作。

说明

如果导入包含代码错误或错误结构的外部创建的组态数据，则会出现意外错误。

应用领域

导出和导入数据对下列任务有用：

- 用于外部编辑组态数据。
- 用于导入外部创建的组态数据，例如文本列表和变量。
- 用于分配指定的组态数据到不同项目，例如将修改过的过程画面用在不同项目中。
- 用于在 TIA Portal 项目和 ECAD 程序之间复制和调整硬件配置。

参见

导入/导出的基本原理 (页 339)

8.1.3 版本特定的 Simatic ML 导入

应用

在 Openness V14 SP1 及以上版本中，SimaticML 支持可跨版本导入。旧的导出文件可导入到与其版本最接近的两个主版本中。

为此，SimaticML 文件中需包含模型的版本信息，如下所示：

```
<?xml version="1.0" encoding="utf-8"?>
<Document>
    <Engineering version="V14 SP1"/>
    <DocumentInfo>
     ...
    </DocumentInfo>
    <SW.DataBlock ID="0">
     ...
    </SW.DataBlock>
</Document>
```

说明

如果 SimaticML 文件中未提供版本信息，系统将使用当前模型版本。

8.1.4 编辑 XML 文件

简介

使用 XML 编辑器或文本编辑器编辑用于导入组态数据的 XML 文件。

如果您正在进行全面的改动，或者正在创建自定义的对象结构，我们建议您使用带自动完成功能的 XML 编辑器。

说明

更改 XML 内容时，需要全面了解 XML 中的结构和验证规则。 为避免验证错误，只有特殊情况下才可在 XML 结构中手动操作。

8.1.5 导出组态数据

简介

每个起始对象（根）的组态数据都单独导出到一个 XML 文件中。

编辑导出文件需要足够的 XML 知识。使用 XML 编辑器使编辑更加方便。

实例

您有一个过程画面，它包含一个 IO 字段。在此 IO 字段中组态一个外部变量。过程画面的导出包含画面和 IO 字段。不会导出此变量和变量使用的连接。相反，导出中仅包含开放式参考。

导出文件的内容

从起始对象开始，子树的所有对象及其属性都保存到导出文件中。对不同子树的对象的所有引用都只能导出为开放式参考。不同子树中所引用对象的相应属性不会写入导出文件中。

说明

不支持从库中导出对象类型

可以在库中将多个对象创建为一个类型。项目中所使用的对象类型实例可以通过 Openness 应用程序进行编辑，其编辑方式与其它对象一样。导出对象时，导出的实例不会带有类型信息。

将这些对象重新导入项目中时，会覆盖对象类型的实例且实例会与相应对象类型分离。

导出文件并不需要包含对象的所有属性。可以定义要导出什么数据：

- `ExportOptions.None`
  此设置只能导出修改后的数据或与默认情况不同的数据。
  导出文件中还包含对于后续数据导入非常必要的所有值。
- `ExportOptions.WithDefaults`[1]
  也会导出默认值。
- `ExportOptions.WithReadOnly`[1]
  也会导出被写保护的值。

[1]：可将这两个选项与下列语法结合：`Export(path,ExportOptions.WithDefaults | ExportOptions.WithReadOnly);`

导出文件的全部内容都使用英文。与此不相关的是，所包含的所有项目文本都使用所有现有的语言导出和导入。

在导出文件中，所有组态数据都被模型成 XML 对象。

参见

导入/导出的基本原理 (页 339)

导出块 (页 422)

8.1.6 导入组态数据

简介

从先前导出并编辑过的 XML 文件，或者从用户自己创建的 XML 文件中导入组态数据。在导入过程中，将检查此文件中包含的数据。这种方法可以防止 TIA Portal 中的组态数据因为导入而变得不一致。

限制条件

- 导入文件中所有根对象的类型必须相同，如变量表、块等。
- 如果导入文件中包含多个根对象且其中一个无效，则系统不会导入该导入文件中的所有内容。
- 导入文本时，为排除导入故障，必须在目标项目中设置好相应的项目语言。必要时，可通过 Openness 修改语言设置。
- 如果在导入文件中指定的对象属性无效（在 TIA Portal 的图形化用户界面中无法编辑），则导入操作取消。
- 只能导入或导出“分别针对每个连接”(separately for each connection) 字段中列出的区域指针。
- 不能导入库中的对象类型。可以在库中将多个对象创建为一个类型。项目中所使用的对象类型实例可以通过 Openness 应用程序进行编辑，其编辑方式与其它对象一样。导出对象时，导出的实例不会带有类型信息。将这些对象重新导入项目中时，会覆盖对象类型的实例且实例会与相应对象类型分离。
- 不能导入故障安全块。

说明

图形属性的设备相关值范围

如果图形属性的值超出有效的值范围，则这些值将在导入过程中复位为 HMI 设备的最大值。

不同的导入行为

如果要导入的对象已经存在于项目中，则可使用不同的程序代码控制导入行为。否则，导入过程中会在项目中再次创建这些对象。

可对导入行为进行以下设置：

- `ImportOptions.None`
  通过该设置，可导入组态数据，且不会发生覆盖。
  如果正从 XML 文件中导入的对象已经存在于项目中，则导入会被中断并会出现异常。
- `ImportOptions.Override`
  基于该设置，系统在导入组态数据时将自动进行覆盖。
  用户可指定导入时项目中将覆盖的现有对象。导入前，相关对象会被删除并使用默认值重新创建。导入过程中，将使用导入的值覆盖这些默认值。如果现有对象和新对象不在同一组中，则不会进行覆盖。此时，为避免发生命名冲突，系统将取消导入并触发一个异常错误。

导入的操作步骤

如果要导入一个 XML 文件，它包含的数据必须遵循特定规则。导入文件中内容的格式必须正确，不得存在语法错误和数据结构错误。如果进行全面改动，可使用 XML 编辑器在导入前检查这些标准。

将 XML 文件导入到 TIA Portal 时，会首先检查文件包含的数据，确定 XML 代码中是否存在格式错误。如果检查过程中发现错误，将取消导入并在例外中显示这些错误（参见“处理异常 (页 331)”）。

参见

导入/导出的基本原理 (页 339)

导入用户数据类型 (页 436)

8.2 导入/导出项目数据

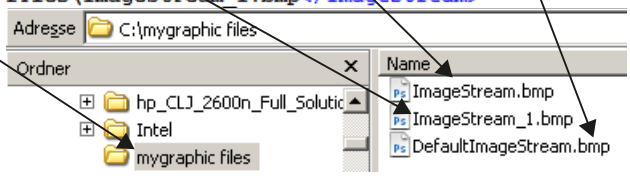
8.2.1 项目图形

8.2.1.1 导出/导入图形

简介

将组态数据从 TIA Portal 导出到 XML 文件时，不包括所选图形或对象引用的图形。在导出过程中，图形单独保存。在 XML 文件中，通过一个相关路径和它们的文件名来引用图形。在 XML 文件中，图象引用被模型成一个对象；其中包含了属性列表和（如果需要的话）链接列表，就像其他对象一样。

```
<Hmi.Globalization.MultiLingualGraphic ID="0">
    <AttributeList>
      <DefaultDithering>False</DefaultDithering>
      <DefaultImageStream external="path">mygraphic files\DefaultImageStream.bmp</DefaultImageStream>
      <DefaultSmoothness>False</DefaultSmoothness>
      <Name>MyGraphic1</Name>
    </AttributeList>
    <ObjectList>
      <Hmi.Globalization.GraphicItem ID="1" CompositionName="Items">
        <AttributeList>
          <Culture>en-US</Culture>
          <Dithering>False</Dithering>
          <ImageStream external="path">mygraphic files\ImageStream.bmp</ImageStream>
          <Smoothness>False</Smoothness>
        </AttributeList>
      </Hmi.Globalization.GraphicItem>
      <Hmi.Globalization.GraphicItem ID="2" CompositionName="Items">
        <AttributeList>
          <Culture>de-DE</Culture>
          <Dithering>False</Dithering>
          <ImageStream external="path">mygraphic files\ImageStream_1.bmp</ImageStream>
          <Smoothness>False</Smoothness>
        </AttributeList>
      </Hmi.Globalization.GraphicItem>
    </ObjectList>
  </Hmi.Globalization.MultiLingualGraphic>
```



导出图形

组态数据的导出仅包含直接选择用于导出的图形。可导出的图形存储在特定语言的 TIA Portal 中。如果使用多语言组态项目，则将导出使用的所有语言版本。

当导出图形时，会在导出文件夹中创建一个新文件夹。通过将 xml 文件名与“文件”相关联来构建文件夹名称。此文件夹包含了导出的图形。如果此文件夹已存在，将创建新的文件夹并使用连续编号进行补充。

使用与项目中使用的文件格式相同的格式保存图形。不改变或转换数据格式，并且分辨率和色深度也保持不变。

ID"default"作为被选为缺省语言的语言的文件扩展名。

如果该文件夹已包含同名文件，将使用一个连续编号对导出图形的文件名进行补充。

导入图形

在导入图形时需要遵守下列要求：

- 图形必须具有 TIA Portal 支持的文件格式。
- 必须在 XML 文件中通过相对路径设置来引用图形。

一旦导出图形，便可以使用图形程序在 TIA Portal 外编辑图形，然后再重新导入该图形。

参见

导入/导出的基本原理 (页 339)

8.2.1.2 导出项目的所有图形

要求

- Openness 应用程序连接到 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)

应用

可导出所有语种项目的图形集合中的单一图形或所有图形。与所有项目图形条目有关的 XML 文件将在导出过程中进行创建，并与导出的图形一起被引用。相关图形和 XML 一起保存到文件系统的相同目录下。

要允许更改导出的图形（“*.jpg”、“*.bmp”、“*.png”、“*.ico”等），这些图形不应进行写保护。

程序代码：导出图形

修改以下程序代码以导出所需图形：

```
//Exports all language variants of a single grafic
Project project = …;
MultiLingualGraphicComposition graphicsComposition = project.Graphics;
MultiLingualGraphic graphic = graphicsComposition.Find("graphicName");
graphic.Export(new FileInfo(@"D:\ExportFolder\graphicName.xml"),
ExportOptions.WithDefaults);
```

程序代码：导出所有图形

修改以下程序代码以导出图形集合中的所有图形：

```
//Exports all graphics of a graphic library
Project project = …;
MultiLingualGraphicComposition graphicsComposition = project.Graphics;
foreach(MultiLingualGraphic graphic in graphicsComposition)
{
    graphic.Export(new FileInfo(string.Format(@"D:\Graphics\{0}.xml", graphic.Name)),
ExportOptions.WithDefaults);
}
```

8.2.1.3 将图形导入到项目

要求

- Openness 应用程序连接到 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)

应用

XML 文件将和图形的语言版本一起被保存到文件系统的目录下。

可在 XML 文件中以相对路径形式引用所有图形。

现在，可将 XML 文件中包含的图形的所有语言版本导入到图形集合中。

还应看到导入组态数据 (页 345)。

程序代码

修改以下程序代码以导入一个或多个图形：

```
//Import all language variants of a single graphic
Project project = …;
MultiLingualGraphicComposition graphicComposition = project.Graphics;
graphicComposition.Import(new FileInfo(@"D:\Graphics\Graphic1.xml"),
ImportOptions.Override);
```

8.2.2 项目文本

8.2.2.1 项目文本的导入

要求

- Openness 应用程序已连接 TIA Portal。
  参见“连接到 TIA Portal (页 69)”
- 项目已打开。
  参见“打开项目 (页 96)”

应用

在 TIA Portal 中，该项目文本位于项目的“语言和资源 (Language & resources)”节点中。这些文本信息将导出到一个“*.xlsx”文件中，用作翻译示例。导出和导入项目文本的限制与 UI 中的限制相同。这些限制包括：

- 导出的文本只能导入到其导出时所处的项目中。
- 只能将文本翻译成项目中可用的语言。必要时，可通过 Openness 添加项目语言。
- 只能重新导入现有文本，如果已删除或者重新创建原始项目中的文本，则该文本的导入会失败。

必须定义以下参数：

名称	示例	说明
pah	new FileInfo ("D:\Test \ProjectText.xlsx")	导出文件的路径
sourceLanguage	new CultureInfo("en-US")	要被翻译的参考语言文本
targetLanguage	new CultureInfo("de-DE")	要翻译成的目标语言文本

说明

多语言文本导出时，将带有该文本所属的父对象。多语言文本不能显式导出。

程序代码：从“语言和资源”节点导出

使用示例参数时会使以下程序代码导出项目文本：

```
project.ExportProjectTexts(new FileInfo(@"D:\Test\ProjectText.xlsx"), new CultureInfo("en-US"), new CultureInfo("de-DE"));
```

导出的多语言文本项的 XML 结构

```
...
  <MultilingualText ID="2" CompositionName="Comment">
    <ObjectList>
      <MultilingualTextItem ID="3" CompositionName="Items">
        <AttributeList>
          <Culture>en-US</Culture>
          <Text>My super tag</Text>
        </AttributeList>
      </MultilingualTextItem>
      <MultilingualTextItem ID="4" CompositionName="Items">
        <AttributeList>
          <Culture>ru-RU</Culture>
          <Text>Мой супер тэг</Text>
        </AttributeList>
      </MultilingualTextItem>
    </ObjectList>
  </MultilingualText>
...
```

8.2.2.2 项目文本的导入

要求

- Openness 应用程序已连接 TIA Portal。
  参见“连接到 TIA Portal (页 69)”
- 项目已打开。
  参见“打开项目 (页 96)”

应用

在 TIA Portal 中，该项目文本位于项目的“语言和资源 (Language & resources)”节点中。可从一个用作翻译示例的“.xlsx”文件中导入项目文本。导出和导入项目文本的限制与 UI 中的限制相同。这些限制包括：

- 导出的文本只能导入到其导出时所处的项目中。
- 只能以文本导出时所处项目支持的语言，导入翻译的文本。
- 只能重新导入现有文本，如果已删除或者重新创建原始项目中的文本，则该文本的导入会失败。

必须定义以下参数：

名称	示例	说明
path	new FileInfo(@"D:\Test \ProjectText.xlsx")	导入文件的路径
updateSourceLang uage	true	如果为 true，则会通过导出文件更新参考语言的文本。 如果为 false，则不会更新参考语言的文本

说明

多语言文本导入时，将带有该文本所属的父对象。多语言文本不能显式导入。

程序代码

使用示例参数时会使以下程序代码导入项目文本：

```
ProjectTextResult result = project.ImportProjectTexts(new FileInfo(@"D:\Test
\ProjectText.xlsx"), true);
```

导入项目文本时，会返回一个对象，指示导入状态以及用于保存导入日志的路径。这些属性可通过以下代码进行访问：

```
ProjectTextResultState resultState = result.State;
FileInfo logFilePath = result.Path;
```

8.3 导入/导出 HMI 设备的数据

8.3.1 XML 文件的结构

简介

来自导入/导出的导出文件中的数据参照基本结构构建。

导出文件的基本结构

导出文件以 XML 格式生成。

XML 文件以文档信息开始。它包含计算机特定安装的数据，项目可通过这些数据导出。

导出文件分为以下两个部分：

- 有关文档的信息
  在该部分中，可使用有效的 XML 语法输入有关导出的自身信息。相应内容将被导入忽略。
  例如，可以插入 `<IntegrityInformation>...</IntegrityInformation>` 块，可在其中放置有关验证的附加信息。XML 文件转发后，接收方可在导入前使用此块，以检查 XML 文件是否已被更改。
- 对象
  本部分包含要导出的元素。

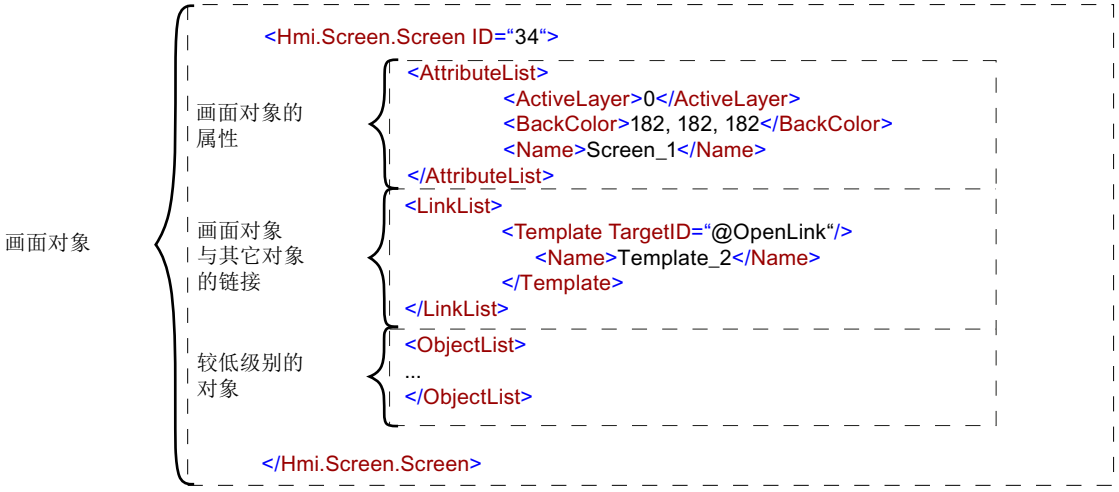
有关文档的信息

画面对象

```
<?xml version="1.0" encoding="UTF-8" ?>
<Document xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <DocumentInfo>
        <UserName>Jane Doe</UserName>
        <Company>Example_Inc</Company>
        <IntegrityInformation>...</IntegrityInformation>
        <Created>2016-04-28T18:05:42.179207Z</Created>
        <ExportSetting>WithDefaults</ExportSetting>
        <InstalledProducts>
            <Product>
                <DisplayName>Totally Integrated Automation Portal</DisplayName>
                <DisplayVersion>V14</DisplayVersion>
            </Product>
            <OptionPackage>
                <DisplayName>WinCC Professional</DisplayName>
                <DisplayVersion>V14</DisplayVersion>
            </OptionPackage>
            <OptionPackage>
                <DisplayName>Siemens TIA Openness</DisplayName>
                <DisplayVersion>V14</DisplayVersion>
            </OptionPackage>
        </InstalledProducts>
    </DocumentInfo>
    <Hmi.Screen.Screen ID ="0">
        <AttributeList>
            <ActiveLayer>0</ActiveLayer>
            <BackColor>189,190,0</BackColor>
            <Height>422</Height>
            <Name>Root screen</Name>
            <Number>1</Number>
            <Visible>True</Visible>
            <Width>640</Width>
        </AttributeList>
        <LinkList>
            <Template TargetID="@OpenLink">
                <Name>Template_1</Name>
            </Template>
        </LinkList>
        <ObjectList>
            <Name>dummy</Name>
        </ObjectList>
    </Hmi.Screen.Screen>
</Document>
```

导出文件的画面对象

XML 文件的附加元素中提供导出元素。



参见

导入/导出的基本原理 (页 339)

8.3.2 导入/导出的数据结构

对象

基本结构对所有对象都相同。

XML 文件中的每个对象都从其类型开始，例如 "Hmi.Screen.Button" 和 ID。ID 将在导出过程中自动创建。

```
<Hmi.Screen.Button CompositionName="ScreenItems" ID="60">
```

除起始对象外，其它对象还包含“CompositionName”XML 属性。此属性的值是预设的。有时需要指定此属性，例如，为了更改按钮按下或释放时的标签。

```
<MultilingualText ID="A" CompositionName="TextOff">
    <ObjectList>
        <MultilingualTextItem ID="B" CompositionName="Items">
            <AttributeList>
                <Culture>en-US</Culture>
                <Text>
                    <body>
                        <p>TextOff</p>
                    </body>
                </Text>
            </AttributeList>
        </MultilingualTextItem>
    </ObjectList>
</MultilingualText>
<MultilingualText ID="C" CompositionName="TextOn">
    <ObjectList>
        <MultilingualTextItem ID="D" CompositionName="Items">
            <AttributeList>
                <Culture>en-US</Culture>
                <Text>
                    <body>
                        <p>TextOn</p>
                    </body>
                </Text>
            </AttributeList>
        </MultilingualTextItem>
    </ObjectList>
</MultilingualText>
```

属性

每一个对象都包含 "AttributeList" 部分中包含的属性。每一个属性都被模型成一个 XML 元素，例如"BackColor"。属性的值被模型化为 XML 内容，例如“204, 204, 204”。

```
<Hmi.Screen.Button ID="2" CompositionName="ScreenItems">
    <AttributeList>
        <BackColor>204, 204, 204</BackColor>
        <ObjectName>Button_1</ObjectName>
    </AttributeList>
</Hmi.Screen.Button>
```

如果需要，为了引用对象，每个对象都包含一个"LinkList"部分。本部分包含到 XML 文件内部和外部的其它对象的链接。每一个链接都被模型成一个 XML 元素。通过模式文件中的目标对象来定义链接的名称。每个链接还包含“TargetID”属性。如果 XML 文件中包含目标对象，则“TargetID”属性的值为“#”加所引用对象的 ID。如果 XML 文件中不包含目标对象，“TargetID”属性的值则为“@OpenLink”。对该对象的实际引用被模型成从属 XML 元素。

```
<Hmi.Tag.Tag ID="17">
    <AttributeList>
        <Name>Tag_1</Name>
    </AttributeList>
    <LinkList>
        <AcquisitionCycle TargetID="@OpenLink">
            <Name>2 s</Name>
        </AcquisitionCycle>
        <Connection TargetID="@OpenLink">
            <Name>HMI_connection</Name>
        </Connection>
    </LinkList>
</Hmi.Tag.Tag>
```

对象与 XML 结构之间的关系

下图显示了导出的 XML 结构与 WinCC 中关联对象之间的关系。

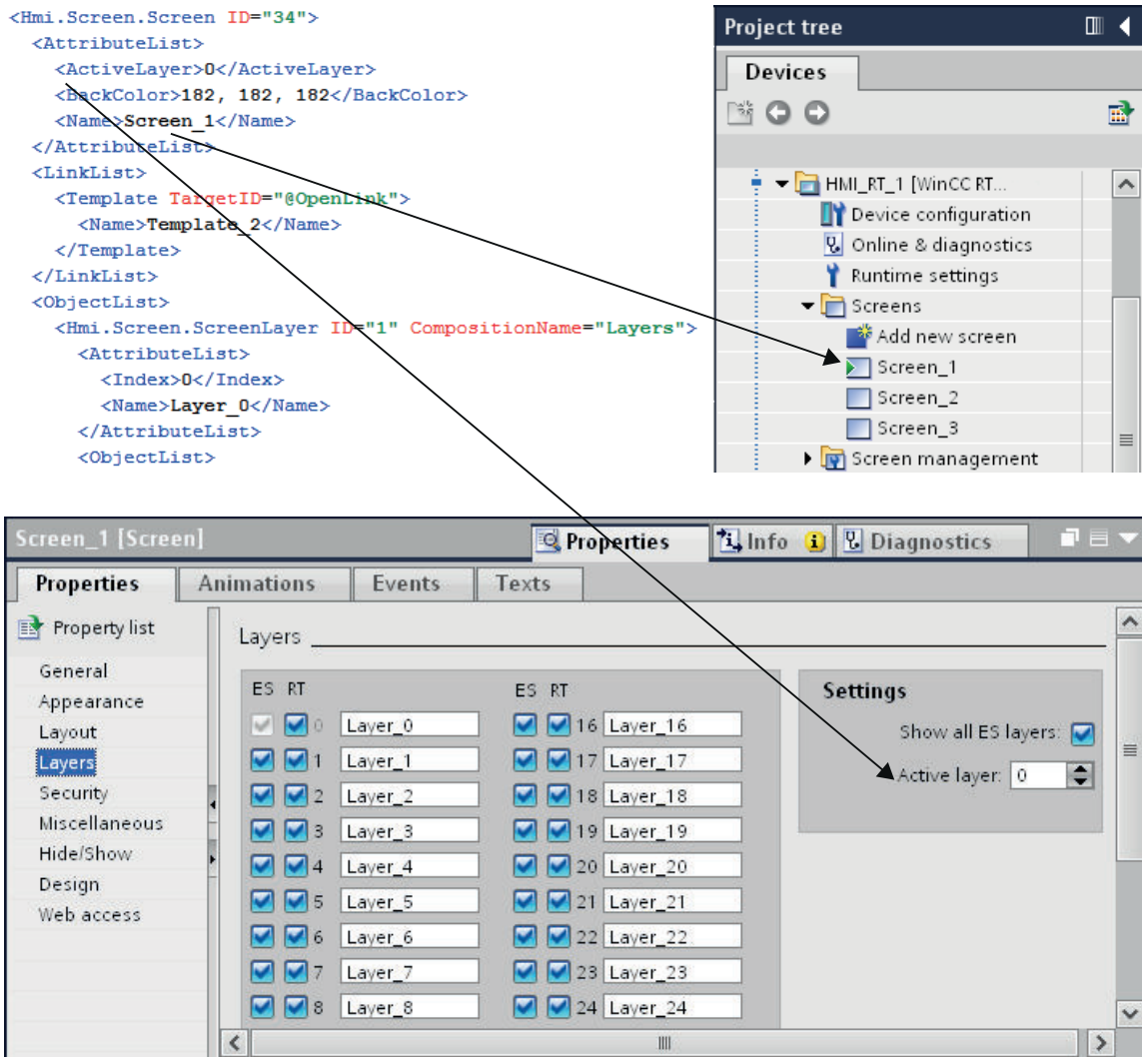


图 8-1　　WinCC 用户界面与 XML 结构之间的关系。

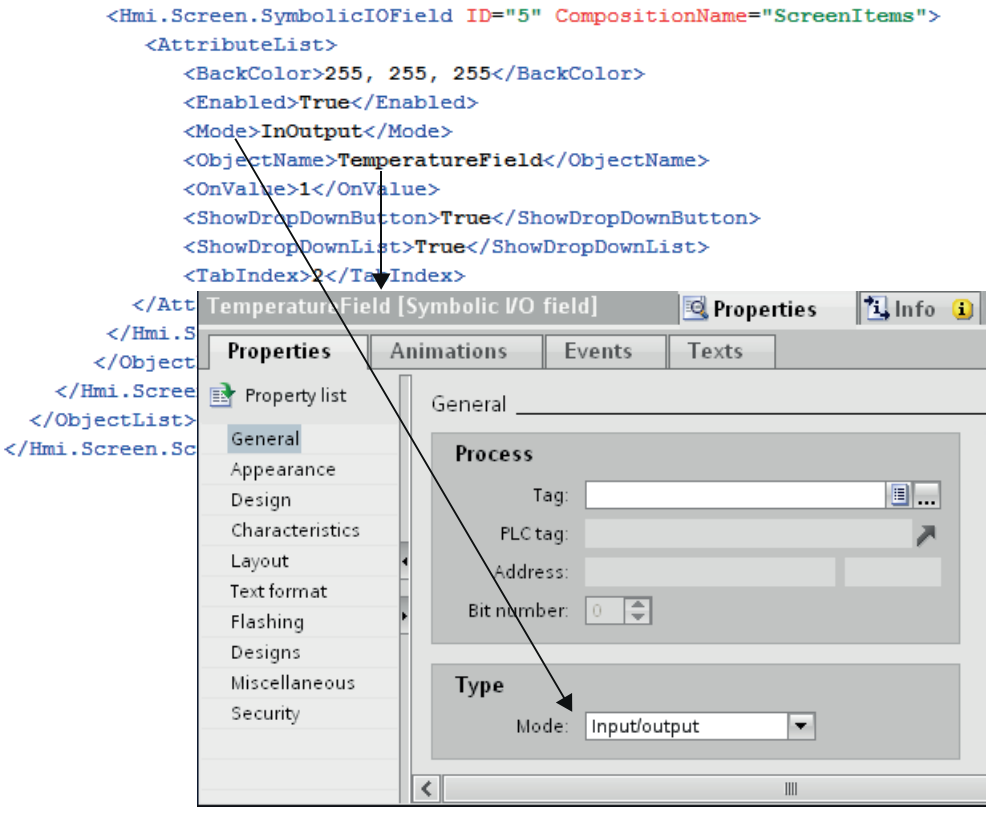


图 8-2　　WinCC 中的设置与 XML 结构之间的关系。

8.3.3　周期

8.3.3.1　导出周期

要求

- Openness 应用程序连接到 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)

应用

API 接口支持将已知 HMI 设备的所有周期导出到 XML 文件中。如果生成相应的导出文件，则表明导出已完成。

程序代码

修改以下程序代码以将 HMI 设备的周期导出至 XML 文件：

```
//Exports cycles from an HMI device
private static void ExportCyclesFromHMITarget(HmiTarget hmitarget)
{
    CycleComposition cycles = hmitarget.Cycles;
    foreach(Cycle cycle in cycles)
       {
           cycle.Export(new FileInfo(string.Format(@"C:\Samples\{0}.xml", cycle.Name)),
ExportOptions.WithDefaults);
       }
}
```

8.3.3.2 导入周期

要求

- Openness 应用程序连接到 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)

应用

使用 `ImportOptions.None` 时，可以根据组合数 (Composition count) 确定实际已导入的周期数。您具有这些导入周期的访问权限。

说明

无法在用户界面中编辑具有属性的标准周期。如果在导入文件中指定应更改这些属性，则导入时会导致 NonRecoverableException 并关闭 TIA Portal。

程序代码

修改以下程序代码以将 XML 文件的一个或多个周期导入 HMI 设备：

```
//Imports cycles to an HMI device
private static void ImportCyclesToHMITarget(HmiTarget hmitarget)
{
    CycleComposition cycles = hmitarget.Cycles;
    string dirPathImport = @"C:\OpennessSamples\Import\";
    string cycleImportFileName = "CycleImport.xml";
    string fullFilePath = Path.Combine(dirPathImport, cycleImportFileName);

    cycles.Import(new FileInfo(fullFilePath), ImportOptions.None);
}
```

参见

导入组态数据 (页 345)

8.3.4 变量表

8.3.4.1 导出 HMI 变量表

要求

- Openness 应用程序连接到 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)

应用

为每个 HMI 变量表导出一个 XMI 文件。API 支持此导出过程。变量表的导出同样适用于子文件夹。

程序代码：从指定文件夹导出所有 HMI 变量表

修改以下程序代码以导出特定文件夹的所有 HMI 变量表：

```
//Exports all tag tables from a tag folder
private static void ExportAllTagTablesFromTagFolder(HmiTarget hmitarget)
{
    TagSystemFolder folder = hmitarget.TagFolder;
    TagTableComposition tables = folder.TagTables;

    foreach (TagTable table in tables)
    {
       FileInfo info = new FileInfo(string.Format(@"C:\OpennessSamples\TagTables\{0}.xml",
table.Name));
        table.Export(info, ExportOptions.WithDefaults);
    }
}
```

程序代码：导出 HMI 变量表

修改以下程序代码以导出单个 HMI 变量表：

```
//Exports a tag table from an HMI device
private static void ExportTagTableFromHMITarget(HmiTarget hmitarget)
{
    string tableName = "Tag table XYZ";
    TagSystemFolder folder = hmitarget.TagFolder;
    TagTableComposition tables = folder.TagTables;
    TagTable table = tables.Find(tableName);

    if (table != null)
    {
       FileInfo info = new FileInfo(string.Format(@"C:\OpennessSamples\TagTables\{0}.xml",
table.Name));
        table.Export(info, ExportOptions.WithDefaults);
    }
}
```

程序代码：导出所有 HMI 变量表

修改以下程序代码以导出所有 HMI 变量表：

```
//Exports all tag tables from an HMI device
private static void ExportAllTagTablesFromHMITarget(HmiTarget hmitarget)
{
    TagSystemFolder sysFolder = hmitarget.TagFolder;

    //First export the tables in underlying user folder
    foreach (TagUserFolder userFolder in sysFolder.Folders)
    {
        ExportUserFolderDeep(userFolder);
    }

    //then, export all tables in the system folder
    ExportTablesInSystemFolder(sysFolder);
}

private static void ExportUserFolderDeep(TagUserFolder rootUserFolder)
{
        foreach (TagUserFolder userFolder in rootUserFolder.Folders)
        {
            ExportUserFolderDeep(userFolder);
        }
        ExportTablesInUserFolder(rootUserFolder);
}

private static void ExportTablesInUserFolder(TagUserFolder folderToExport)
{
     TagTableComposition tables = folderToExport.TagTables;
     foreach (TagTable table in tables)
     {
         string fullFilePath = string.Format(@"C:\OpennessSamples\TagTables\{0}.xml",
table.Name);
         table.Export(new FileInfo(fullFilePath), ExportOptions.WithDefaults);
     }
 }

private static void ExportTablesInSystemFolder(TagSystemFolder folderToExport)
{
     TagTableComposition tables = folderToExport.TagTables;
     foreach (TagTable table in tables)
     {
         string fullFilePath = string.Format(@"C:\OpennessSamples\TagTables\{0}.xml",
table.Name);
         table.Export(new FileInfo(fullFilePath), ExportOptions.WithDefaults);
     }
 }
```

8.3.4.2 导入 HMI 变量表

要求

- Openness 应用程序连接到 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)

程序代码

修改以下程序代码以将 XML 文件的 HMI 变量表导入至用户自定义文件夹或系统文件夹：

```
//Imports a single HMI tag table from a XML file
private static void ImportSingleHMITagTable(HmiTarget hmitarget)
{
    TagSystemFolder folder = hmitarget.TagFolder;
    TagTableComposition tables = folder.TagTables;

    FileInfo info = new FileInfo(@"D:\Samples\Import\myExportedTagTable.xml");
    tables.Import(info, ImportOptions.Override);
}
```

变量导入不正确

如果在变量或被引用变量的名称中使用以下符号，则变量的导入会出错：

- .（句点）
- \（反斜杠）

补救措施 1：

导出之前，请检查以确保要导出的变量或被引用变量的名称不包含句点或反斜杠。

补救措施 2：

在导出文件中，用引号将变量或被引用变量的名称排除在外。

示例

- 带符号的变量名称：
  <name>Siemens.Simatic.Hmi.Utah.Tag.HmiTag:41000_Options_Time_Date\DB_SFX0908_HMI1.Actual_Date_Time.Hour</name>
- 用引号排除的带符号变量名称：
  <name>"Siemens.Simatic.Hmi.Utah.Tag.HmiTag:41000_Options_Time_Date\DB_SFX0908_HMI1.Actual_Date_Time.Hour"</name>

8.3.4.3 从 HMI 变量表导出单个变量

要求

- Openness 应用程序连接到 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)

应用

以下对象模型的对象类型可作为 HMI 变量的子级项存在，并在导出过程中被考虑：

MultilingualText	注释、变量值、显示名称
TagArrayMemberTag	HMI 数组元素
TagStructureMember	HMI 结构元素
Event	已组态事件
MultiplexEntry	已组态的变量多路复用条目

程序代码

修改以下程序代码以将 HMI 变量表中的单个变量导出至 XML 文件：

```
//Exports a selected tag from a tag table
private static void ExportSelectedTagFromTagTable(HmiTarget hmitarget)
{
    TagSystemFolder tagFolder = hmitarget.TagFolder;
    TagTable mytable = tagFolder.TagTables.Find("MyTagTable");

    TagComposition containingTags = mytable.Tags;
    Tag myTag = containingTags.Find("MyTag");

    if (myTag != null)
    {
        FileInfo info = new FileInfo(string.Format(@"C:\OpennessSamples\Tags\{0}.xml",
myTag.Name));
        myTag.Export(info, ExportOptions.WithDefaults);
    }
}
```

8.3.4.4 从 HMI 变量表导入单个变量

要求

- Openness 应用程序连接到 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)

应用

以下对象模型的对象类型可作为 HMI 变量的子级项存在，并在导入过程中被考虑：

MultilingualText	注释、变量值、显示名称
TagArrayMemberTag	HMI 数组元素
TagStructureMember	HMI 结构元素
Event	已组态事件
MultiplexEntry	已组态的变量多路复用条目

程序代码

修改以下程序代码以将 XML 文件的 HMI 变量导入至 HMI 变量表：

```
//Imports a tag into a tag table
private static void ImportTagIntoTagTable(HmiTarget hmitarget)
{
    TagSystemFolder tagFolder = hmitarget.TagFolder;
    TagTable myTable = tagFolder.DefaultTagTable;
    TagComposition tagComposition = myTable.Tags;

    FileInfo info = new FileInfo(@"D:\Samples\Import\myExportedTag.xml");
    tagComposition.Import(info, ImportOptions.Override);
}
```

8.3.4.5 导入/导出 HMI 变量时的特殊考虑事项

简介

特殊注意事项适用于以下 HMI 变量的导出和导入：

- 具有集成连接的外部 HMI 变量
- 具有“UDT”数据类型的 HMI 变量

类似的程序代码

上述 HMI 变量的程序代码与以下程序代码几乎完全相同：

- 程序代码：导出 HMI 变量 (页 366)
- 程序代码：导入 HMI 变量 (页 367)

要求

- Openness 应用程序连接到 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)

导出/导入具有集成连接的外部 HMI 变量时的特殊考虑事项

导出具有集成 HMI 连接的外部 HMI 变量时，导出文件中只保存 HMI 变量到 PLC 的链接，而不是 PLC 变量数据。

导入前，必须确保项目中存在 PLC、相应的 PLC 变量和到相应 PLC 的集成连接。否则，必须在导入前创建这些项。在外部 HMI 变量的后续导入过程中，到 PLC 变量的链接将被再次激活。

在项目的所有变量表中，外部 HMI 变量的名称必须是唯一的。如果在导入过程中没有为 HMI 变量指定合适的变量表，将取消导入。

使用以下 XML 结构可导入具有集成连接的外部 HMI 变量：

```
<Hmi.Tag.Tag ID="1" CompositionName="Tags">
    <AttributeList>
        <Name>MyIntegratedHmiTag_1</Name>
    </AttributeList>
    <LinkList>
        <AcquisitionCycle TargetID="@OpenLink">
            <Name>1 s</Name>
        </AcquisitionCycle>
        <Connection TargetID="@OpenLink">
            <Name>HMI_Connection_MP277_300400</Name>    <- Must exist in the project
        </Connection>
        <ControllerTag TargetID="@OpenLink">
            <Name>Datablock_1.DBElement1</Name>        <- Must exist in the project
        </ControllerTag>
    </LinkList>
</Hmi.Tag.Tag>
```

导出/导入“UDT”数据类型的 HMI 变量时的特殊考虑事项

导出“UDT”数据类型的 HMI 变量时，链接会导出到数据类型中。为便于导出，仅支持版本化的数据类型。

这些数据类型必须保存在项目库中。不支持全局库中的数据类型。

以下规则适用于导入：

- 引用的数据类型必须包含在项目库中。
  如果项目库中不包含该数据类型，导入将被终止。
- 引用的数据类型必须为版本化形式。自 TIA Portal V13 SP1 起支持版本化。
  如果数据类型未版本化，将触发例外。

说明

导入过程中，为解决引用问题，将使用找到的第一个数据类型。

在此，可执行以下操作：首先，搜索项目库的根目录，然后搜索子文件夹。

使用以下 XML 结构可导入“UDT”数据类型的 HMI 变量：

```
<Hmi.Tag.Tag ID="1" CompositionName="Tags">
    ...
    <LinkList>
        <DataType TargetID="@OpenLink">
            <Name>HmiUdt_1 V 1.0.0</Name>    <- Must exist in the project library
        </DataType>
        ...
    </LinkList>
    ...
</Hmi.Tag.Tag>
```

8.3.5 VB 脚本

8.3.5.1 导出 VB 脚本

要求

- Openness 应用程序连接到 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)

应用

将考虑导出所有子级别用户定义文件夹。将为每一个已导出的 VB 脚本创建一个单独的 XML 文件。

程序代码：导出 VB 脚本

修改以下程序代码以将 HMI 设备的所选 VB 脚本导出至 XML 文件：

```
//Exports a single vbscript of an HMI device
private static void ExportSingleVBScriptOfHMITarget(HmiTarget hmitarget)
{
    VBScriptSystemFolder vbScriptFolder = hmitarget.VBScriptFolder;
    VBScriptComposition vbScripts = vbScriptFolder.VBScripts;
    VBScript vbScript = vbScripts.Find("MyVBScript");

    FileInfo info = new FileInfo(string.Format(@"C:\OpennessSamples\Export\Scripts
\{0}.xml", vbScript.Name));
    vbScript.Export(info, ExportOptions.None);
}
```

8.3.5.2 从文件夹导出 VB 脚本

要求

- Openness 应用程序连接到 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)

应用

将为每一个已导出的 VB 脚本创建一个单独的 XML 文件。

程序代码：从用户定义文件夹中导出 VB 脚本

修改以下程序代码以将用户自定义文件夹的 VB 脚本导出至 XML 文件：

```
//Exports vbscripts of a selected vbscript system folder
private static void ExportVBScriptOfSelectedVBScriptSystemFolder(HmiTarget hmitarget)
{
    VBScriptSystemFolder vbScriptFolder = hmitarget.VBScriptFolder;
    VBScriptUserFolderComposition vbUserFolders = vbScriptFolder.Folders;
    VBScriptUserFolder vbUserFolder = vbUserFolders.Find("MyVBUserFolder");
    VBScriptComposition vbScripts = vbUserFolder.VBScripts;

    foreach (VBScript script in vbScripts)
    {
        FileInfo info = new FileInfo(String.Format(@"C:\OpennessSamples\Export\Scripts
\{0}.xml", script.Name));
        script.Export(info, ExportOptions.None);
    }
}
```

程序代码：导出系统文件夹中的所有 VB 脚本

修改以下程序代码以导出系统文件夹中的所有 VB 脚本：

```
//Exports all vbscripts by using a foreach loop
private static void ExportAllVBScripts(HmiTarget hmitarget)
{
    VBScriptSystemFolder vbScriptFolder = hmitarget.VBScriptFolder;
    VBScriptComposition vbScripts = vbScriptFolder.VBScripts;
    if (vbScripts == null) return;

    foreach (VBScript script in vbScripts)
    {
        FileInfo info = new FileInfo(string.Format(@"C:\OpennessSamples\Export\Scripts
\{0}.xml", script.Name));
        script.Export(info, ExportOptions.None);
    }
}
```

8.3.5.3 导入 VB 脚本

要求

- Openness 应用程序连接到 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)

应用

支持批量导入。也可使用具有 Foreach 循环的程序代码 (导出 VB 脚本 (页 370))。

程序代码

修改以下程序代码以将 XML 文件的 VB 脚本导入至 HMI 设备：

```
private static void ImportSingleVBScriptToHMITarget(HmiTarget hmitarget)
{
    VBScriptSystemFolder vbScriptFolder = hmitarget.VBScriptFolder;
    VBScriptComposition vbScripts = vbScriptFolder.VBScripts;
    if (vbScripts 00 null) return;
    {
        FileInfo info = new FileInfo(@"D:\Samples\Import\VBScript.xml");
        vbScripts.Import(info, ImportOptions.None);
    }
}
```

8.3.6 文本列表

8.3.6.1 从 HMI 设备导出文本列表

要求

- Openness 应用程序连接到 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)

应用

导出的文本列表和图形列表包括其所有条目。可以分别导出文本列表和图形列表。

将导出 HMI 设备的文本列表。将为每一个导出的文本列表创建一个单独的 XML 文件。

程序代码

修改以下程序代码以导出 HMI 设备的文本列表：

```
//Export TextLists
private static void ExportTextLists(HmiTarget hmitarget)
{
    TextListComposition text = hmitarget.TextLists;
    foreach (TextList textList in text)
    {
        FileInfo info = new FileInfo(string.Format(@"D:\Samples\Export\{0}.xml",
textList.Name);
        textList.Export(info, ExportOptions.WithDefaults);
    }
}
```

8.3.6.2 将文本列表导入到 HMI 设备

要求

- Openness 应用程序连接到 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)

应用

API 接口支持将文本列表从 XML 文件导入到 HMI 设备。

程序代码

修改以下程序代码以将 XML 文件的文本列表导入至 HMI 设备：

```
//Imports a single TextList
private static void ImportSingleTextList(HmiTarget hmitarget)
{
    TextListComposition textListComposition = hmitarget.TextLists;
    IList<TextList> importedTextLists = textListComposition.Import(new FileInfo(@"D:
\SamplesImport\myTextList.xml"), ImportOptions.Override);
}
```

8.3.6.3 用于文本列表导出/导入的高级 XML 格式

要求

- Openness 应用程序已连接 TIA Portal。
  参见“连接到 TIA Portal (页 69)”
- 项目已打开。
  参见“打开项目 (页 96)”
- 标准导出文本列表
  请参见“从 HMI 设备导出文本列表 (页 373)”
- 标准导入文本列表
  请参见“将文本列表导入到 HMI 设备 (页 374)”

应用

文本列表也可包含格式化文本。这主要涉及以下格式：

- 文本格式化
- 在文本内引用其它对象

如果待导出的文本列表为纯文本格式，则将生成一个高级的 XML 导出格式。其中，对象引用将表示为 Open Links。这同样适用于将要使用格式化文本导入的文本列表。

高级 XML 导出格式也会变得更加复杂。例如，不仅链接了文本列表中的对象名称，还可能通过 Open Link 链接到另一设备的 PLC 变量。此时，要删除 Open Link，必须将所有信息编写到一个字符串中。

```
<?xml version="1.0" encoding="utf-8"?>
<Document>
<!-- ... -->
    <MultilingualText ID="5" CompositionName="Text">
        <ObjectList>
            <MultilingualTextItem ID="6" CompositionName="Items">
                <AttributeList>
                    <Culture>en-US</Culture>
                    <Text>
                        <body>
                            <p>
                                <field ref="0" />
                            </p>
                        </body>
                        <fieldinfos>
                            <fieldinfo name="0" domaintype="HMICommonTextList">
                                <reference TargetID="@OpenLink">
                                    <name>Siemens.Simatic.Hmi.Utah.TextAndGraphicLists.HmiTextList:Empty Text_list_
                                </reference>
                                <subreference TargetID="@OpenLink">
                                    <name>Siemens.Simatic.Hmi.Utah.Tag.HmiTag:t1</name>
                                </subreference>
                                <domaindata>
                                    <format length="9" />
                                </domaindata>
                            </fieldinfo>
                        </fieldinfos>
                    </Text>
                </AttributeList>
            </MultilingualTextItem>
            <MultilingualTextItem ID="7" CompositionName="Items">
                <AttributeList>
                    <Culture>de-CH</Culture>
                    <Text>
                        <body>
                            <p>
                                <field ref="0" />
                            </p>
                        </body>
                        <fieldinfos>
                            <fieldinfo name="0" domaintype="HMICommonTextList">
                                <reference TargetID="@OpenLink">
                                    <name>Siemens.Simatic.Hmi.Utah.TextAndGraphicLists.HmiTextList:Empty Text_list_
                                </reference>
                                <subreference TargetID="@OpenLink">
                                    <name>Siemens.Simatic.Hmi.Utah.Tag.HmiTag:t1</name>
                                </subreference>
                                <domaindata>
                                    <format length="9" />
                                </domaindata>
                            </fieldinfo>
                        </fieldinfos>
                    </Text>
                </AttributeList>
            </MultilingualTextItem>
        </ObjectList>
    </MultilingualText>
```

8.3.7 图形列表

8.3.7.1 导出图形列表

要求

- Openness 应用程序连接到 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)

应用

导出的文本列表和图形列表包括其所有条目。可以分别导出文本列表和图形列表。

为每个图形列表创建一个 XML 文件。图形列表包含中的全局图形对象将被作为 Open Links 导出。

程序代码

修改以下程序代码以导出 HMI 设备的图形列表：

```
//Exports GraphicLists
private static void ExportGraphicLists(HmiTarget hmitarget)
{
    GraphicListComposition graphic = hmitarget.GraphicLists;
    foreach (GraphicList graphicList in graphic)
    {
        FileInfo info = new FileInfo(string.Format(@"D:\Samples\Export\{0}.xml",
graphicList.Name));
        graphicList.Export(info, ExportOptions.WithDefaults);
    }
}
```

8.3.7.2 导入图形列表

要求

- Openness 应用程序连接到 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)

应用

API 接口支持将图形列表从 XML 文件导入到 HMI 设备。

导入中包含图形列表的所有引用图形对象。不包含对全局图形的引用。如果目标项目中存在引用的全局图形，则在导入期间将恢复全局图形的引用。

程序代码

修改以下程序代码以将 XML 文件的图形列表导入至 HMI 设备：

```
//Imports a single GraphicList
private static void ImportSingleGraphicList(HmiTarget hmitarget)
{
    GraphicListComposition graphicListComposition = hmitarget.GraphicLists;
    IList<GraphicList> importedGraphicLists = graphicListComposition.Import(new
FileInfo(@"D:\Samples\Import\myGraphicList.xml"), ImportOptions.Override);
}
```

8.3.8 连接

8.3.8.1 导出连接

要求

- Openness 应用程序连接到 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)

应用

API 接口支持将 HMI 设备的所有连接导出到 XML 文件。

说明

导出集成连接

不支持导出集成连接。

将为每个已导出的连接创建一个单独的 XML 文件。

程序代码

修改以下程序代码以将 HMI 设备的所有连接导出至 XML 文件：

```
//Exports communication connections from an HMI device
private static void ExportConnectionsFromHMITarget(HmiTarget hmitarget)
{
    ConnectionComposition connections = hmitarget.Connections;
    foreach(Connection connection in connections)
    {
        FileInfo info = new FileInfo(string.Format(@"D:\Samples\Export\{0}.xml",
connection.Name));
        connextion.Export(info, ExportOptions.WithDefaults);
    }
}
```

8.3.8.2 导入连接

要求

- Openness 应用程序连接至 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)

应用

API 接口支持将 HMI 设备的所有连接从 XML 文件导入到 HMI 设备。如果要导入多个通信连接，则需为各个通信连接导入相应的 XML 文件。

说明

如果导入连接的项目中已组态一个集成连接，则不会覆盖此连接。此时将取消导入并触发 Exception。

程序代码

修改以下程序代码以将 XML 文件中 HMI 设备的单个连接导入至 HMI 设备：

```
//Imports Communication connections to an HMI device
private static void ImportConnectionsToHMITarget(HmiTarget hmitarget)
{
    ConnectionComposition connections = hmitarget.Connections;
    IList<Connection> importedConnectionLists = connections.Import(new FileInfo(@"D:
\Samples\Import\myConnectionImport.xml"), ImportOptions.Override);
}
```

8.3.9 画面

8.3.9.1 可导出画面对象的概述

应用

可使用公共 API 导出或导入以下画面：

表格 8-4 支持的画面

对象	是否可导出/导入
画面	是
全局画面	是
画面模板	是
永久性区域	是

对象	是否可导出/导入
弹出画面	是
滑入画面	是

可使用公共 API 导出或导入以下画面对象：

表格 8-5　支持的画面对像

范围	对象类型	是否可导出/导入
基本对象	直线	是
	折线	是
	多边形	是
	椭圆	是
	部分椭圆	–
	扇形	–
	椭圆弧	–
	圆弧	–
	圆	是
	矩形	是
	连接器	–
	文本字段	是
	图形视图	是
	管道	–
	双 T 形管	–
	T 形管	–
	弯管	–

范围	对象类型	是否可导出/导入
元素	I/O 字段	是
	图形 I/O 字段	是
	可编辑的文本字段	–
	列表框	–
	组合框	–
	按钮	是
	圆形按钮	–
	指示灯按钮	是
	开关	是
	符号 I/O 字段	是
	日期/时间字段	是
	棒图	是
	符号库	是
	滑块	是
	滚动条	–
	复选框	–
	选项按钮	–
	量表	是
	时钟	是
	存储空间视图	–
	功能键（软键）	是
	组	是
	面板实例	是

范围	对象类型	是否可导出/导入
控件	画面窗口	–
	用户视图	是
	打印作业/脚本诊断	–
	摄像机视图	–
	PDF 视图	–
	配方视图	–
	报警视图	–
	报警指示器	–
	报警窗口	–
	f(x) 趋势视图	–
	f(t) 趋势视图	–
	表格视图	–
	数值表	–
	HTML 浏览器	–
	媒体播放器	–
	通道诊断	–
	WLAN 接收	–
	区域名称	–
	区域信号	–
	有效范围名称	–
	有效范围名称 (RFID)	–
	有效范围信号	–
	充电状况	–
	手轮	–
	帮助指示器	–
	Sm@rtClient 视图	–
	状态/强制	–
	存储空间视图	–
	NC 子程序显示	–
	系统诊断视图	–
	系统诊断窗口	–

参见

导入/导出的基本原理 (页 339)

8.3.9.2 导出 HMI 设备的所有画面

要求

- Openness 应用程序连接到 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)

应用

导出 HMI 设备的所有用户定义画面文件夹的所有汇聚画面时，需要不同的程序代码。

程序代码：导出设备的所有画面

修改以下程序代码以导出 HMI 设备的用户自定义画面文件夹和画面系统文件夹的所有画面：

```
private static void ExportScreensOfDevice(string rootPath, HmiTarget hmitarget)
{
    DirectoryInfo info = new DirectoryInfo(rootPath);
    info.Create();
    //export the ScreenFolder recursive

    string screenPath = Path.Combine(rootPath, "Screens");
    info = new DirectoryInfo(screenPath);
    info.Create();
    ExportScreens(screenPath, hmitarget);
}
```

程序代码：导出用户定义文件夹的所有画面

修改以下程序代码以导出 HMI 设备的用户自定义画面文件夹和画面系统文件夹的所有画面：

```
private static void ExportScreensOfDevice(HmiTarget hmitarget)
{
    ScreenUserFolder folder = hmitarget.ScreenFolder.Folders.Find("MyScreenFolder");
    //or ScreenSystemFolder folder = hmitarget.ScreenFolder;
    ScreenComposition screens = folder.Screens;
    foreach(Screen screen in screens)
    {
        FileInfo info = new FileInfo(string.Format(@"D:\Samples\Screens\{0}\{1}.xml",
folder.Name, screen.Name));
        screen.Export(info, ExportOptions.WithDefaults);
    }
}
```

程序代码：导出独立于用户的设备的所有画面

修改以下程序代码以导出所有画面：

```
public static void ExportScreens(string screenPath, HmiTarget target)
{
    foreach(Screen screen in target.ScreenFolder.Screens)
    {
        screen.Export(new FileInfo(Path.Combine(screenPath, screen.Name + ".xml")),
ExportOptions.WithDefaults);
    }
    foreach(ScreenUserFolder subfolder in target.ScreenFolder.Folders)
    {
        ExportScreenUserFolder(Path.Combine(screenPath, folder.Name), subfolder);
    }
}

private static void ExportScreenUserFolder(string screenPath,ScreenUserFolder folder )
{
    foreach(Screen screen in folder.Screens)
    {
        screen.Export(new FileInfo(Path.Combine(screenPath, screen.Name + ".xml")),
ExportOptions.WithDefaults);
    }
    foreach(ScreenUserFolder subfolder in folder.Folders)
    {
        ExportScreenUserFolder(Path.Combine(screenPath, subfolder.Name), subfolder);
    }
}
```

8.3.9.3 从画面文件夹导出画面

要求

- Openness 应用程序连接到 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)

应用

导出以下画面数据：

画面	数据
属性	ActiveLayer, BackColor, Height, Width, Name, Number, HelpText
打开的链接	Template
组合	• Layers • Animations 导出基于 Runtime Advanced 的所有已组态的动画。 • Events 导出基于 Runtime Advanced 的所有已组态的事件。 • Softkeys 导出所有已组态的软键。

为每个图层导出以下数据：

说明

默认情况下，TIA Portal 中的层名称为空文本。

如果未更改 TIA Portal 中的层名称，则导出的层名称将为空文本。在这种情况下，TIA Portal 中显示的层名称取决于用户界面语言。

如果更改 TIA Portal 中的层名称，修改后的层名称将以所有相关语言显示。

图层	数据
属性	Name, Index, VisibleES
组合	ScreenItems （包括画面项）

导出中不包括：

- SCADA 特定的属性。
- 不包含任何画面项和属性与默认值相同的图层。

程序代码

修改以下程序代码以导出 HMI 设备用户文件夹或系统文件夹中的单个画面：

```
//Exports a single screen from a screen folder
private static void ExportSingleScreenFromScreenFolder(HmiTarget hmitarget)
{
    ScreenUserFolder folder = hmitarget.ScreenFolder.Folders.Find("MyScreenFolder");
    //or ScreenSystemFolder folder = hmitarget.ScreenFolder;
    ScreenComposition screens = folder.Screens;
    Screen screen = screens.Find("Screen_1.xml");
    if (screen == null) return;
    {
        FileInfo info = new FileInfo(string.Format(@"D:\Samples\Screens\{0}\{1}.xml",
folder.Name, screen.Name));
        screen.Export(info, ExportOptions.WithDefaults);
    }
}
```

8.3.9.4 向 HMI 设备导入画面

要求

- Openness 应用程序连接到 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)

应用

只能将画面导入到特定类型的 HMI 设备。该 HMI 设备与导出画面的设备必须具有相同的设备类型。

导入以下画面数据：

画面	数据
属性	ActiveLayer, BackColor, Height, Width, Name, Number, HelpText
打开的链接	Templates
组合	• Layers • Animations 导入所有可组态画面动画。 • Events 导入所有可组态画面事件。 • Softkeys 导入所有可组态画面软键。

为每个图层导入以下数据：

说明

如果在导入前为层名称指定了空文本，则在导入后，TIA Portal 中显示的层名称将取决于用户界面语言。

如果分配了层名称，则在导入后，指定的层名称将以所有相关语言显示。

图层	数据
属性	Name, Index
组合	ScreenItems

限制

- 如果画面的宽度和高度与设备尺寸不匹配，则将取消导入并触发 Exception。不支持调整所含的画面项。因此，某些画面项可能会超出画面边界。这种情况下，将输出编译器警告。
- 设备的所有画面的画面编号必须唯一。如果发现某一画面的画面编号已在设备中创建，则会取消画面导入。如果尚未分配画面编号，则会在导入期间为画面分配一个唯一编号。
- 画面中，各个层的 Z 顺序画面项的布局必须唯一且连续。因此，导入画面后，如果有必要，则会执行一致性检查从而修复布局。此操作可能产生某些画面项的已修改的“选项卡索引”。
  可以在 XML 文件中手动更改画面项的 Z 顺序。第一个画面项排在 Z 顺序中的最后。

说明

如果画面项的属性“根据内容调整大小”(Fit size to content) 已启用，则可在 XML 文件中更改画面项的宽度值和高度值。

说明

不支持从库中导入画面类型

自 WinCC V12 SP1 起，可在库中创建相应类型的画面。项目中所使用的画面类型实例可以通过 Openness 应用程序进行编辑，其编辑方式与其它画面一样。导出画面时，导出的画面类型实例不含类型信息。

将这些画面重新导入项目中时，会覆盖画面类型的实例且该实例会与相应画面类型分离。

程序代码：向 HMI 设备导入画面

修改以下程序代码以使用 For each 循环将画面导入至 HMI 设备：

```
//Imports all screens to an HMI device
private static void ImportScreensToHMITarget(HmiTarget hmitarget)
{
    FileInfo[] exportedScreens = new FileInfo[] {new FileInfo(@"D:\Samples\Import
\Screen_1.xml"), new FileInfo(@"D:\Samples\Import\Screen_2.xml")};
    ScreenUserFolder folder = hmitarget.ScreenFolder.Folders.Find("MyScreenFolder");
    foreach (FileInfo screenFileInfo in exportedScreens)
    {
        folder.Screens.Import(screenFileInfo, ImportOptions.Override);
    }
}
```

程序代码：导入到新创建的用户文件夹

修改以下程序代码以将某以画面导入至新创建的 HMI 设备用户文件夹：

```
//Imports a single screen to a new created user folder of an HMI device
private static void ImportSingleScreenToNewFolderOfHMITarget(HmiTarget hmitarget)
{
    ScreenUserFolder folder = hmitarget.ScreenFolder.Folders.Create("MyFolder");
    folder.Screens.Import(new FileInfo(@"D:\Samples\Import\myScreens.xml"),
ImportOptions.Override);
}
```

8.3.9.5 导出永久性区域

要求

- Openness 应用程序连接到 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)

应用

导出永久性区域的以下数据：

永久性区域	数据
属性	ActiveLayer, BackColor, Height, Width, Name
组合	Layers

为每个图层导出以下数据：

图层	数据
属性	Name, Index
组合	ScreenItems （包括画面项）

程序代码

修改以下程序代码以将 HMI 设备的永久性区域导出至 XML 文件：

```
//Exports a permanent area
private static void ExportScreenoverview(HmiTarget hmitarget)
{
    ScreenOverview overview = hmitarget.ScreenOverview;
    if (overview == null) return;

    FileInfo info = new FileInfo(@"D:\Samples\Screens\ExportedOverview.xml");
    overview.Export(info, ExportOptions.WithDefaults);
}
```

8.3.9.6 导入永久性区域

要求

- Openness 应用程序连接到 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)

应用

导入永久性区域的以下数据：

永久性区域	数据
属性	ActiveLayer, BackColor, Height, Width, Name, Visible, Number
组合	Layers

为每个图层导入以下数据：

图层	数据
属性	Name, Index
组合	ScreenItems （包括画面项）

如果画面的宽度和高度与设备尺寸不一致，则取消导入并触发 Exception。不支持调整所含的设备项（画面项）。因此，某些设备项可能会超出画面边界。这种情况下，将输出编译器警告。

永久性区域内的设备项布局必须是唯一且连续的。因此，导入永久性区域后，会根据需要执行一致性检查以修复布局。此操作可能产生某些设备项的已修改的“选项卡索引”。

程序代码

修改以下程序代码以将 XML 文件的永久性区域导入至 HMI 设备：

```
//Imports a permanent area
private static void ImportScreenOverview(HmiTarget hmiTarget)
{
    FileInfo info = new FileInfo(@"D:\Samples\Screens\ExportedOverview.xml");
    hmiTarget.ImportScreenOverview(info, ImportOptions.Override);
}
```

8.3.9.7 导出 HMI 设备的所有画面模板

要求

- Openness 应用程序连接到 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)

应用

为每个画面模板创建一个 XML 文件。

由于不支持批量导出，因此需要单独枚举和导出所有画面模板。在此操作过程中，确保所使用的画面模板名符合文件系统的文件命名约定。

程序代码：导出设备的所有画面模板

修改以下程序代码以导出特定文件夹中的所有画面模板：

```
public static void ExportScreenTemplatesOfDevice(string rootPath ,
ScreenTemplateUserFolder folder)
{
    string screenPath = Path.Combine(rootPath, "Screens");
    DirectoryInfo info = new DirectoryInfo(screenPath);
    info.Create();

    //export the ScreenTemplateFolder recursive
    ExportScreenTemplates (screenPath, hmitarget);
}
```

程序代码：导出特定文件夹的所有画面模板

修改以下程序代码以导出所有画面模板：

```
//Exports all screen templates of a selected folder
private static void ExportScreenTemplates(string templatePath, HmiTarget hmitarget)
{
    foreach (ScreenTemplate screen in hmitarget.ScreenTemplateFolder.ScreenTemplates)
    {
        screen.Export(new FileInfo(Path.Combine(templatePath, screen.Name + ".xml")),
ExportOptions.WithDefaults);
    }
    foreach (ScreenTemplateUserFolder folder in hmitarget.ScreenTemplateFolder.Folders)
    {
        ExportScreenTemplates(Path.Combine(templatePath, folder.Name), hmitarget);
    }
}
```

8.3.9.8 从文件夹导出画面模版

要求

- Openness 应用程序连接到 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)

应用

导出以下画面模板数据：

画面模板	数据
属性	ActiveLayer, BackColor, Height, Width, Name
组合	• Layers • Animations 导出所有已组态的动画。不导出 SCADA 动画。 • Softkeys 导出所有已组态的软键。

为每个图层导出以下数据：

图层	数据
属性	Name, Index
组合	ScreenItems （包括画面项）

程序代码：导出用户自定义文件夹的一个画面模板

修改以下程序代码以导出系统文件夹或用户自定义文件夹中的单个画面模板：

```
private static void ExportSingleScreenTemplate(string templatePath, HmiTarget hmiTarget)
{
    ScreenTemplateUserFolder folder =
hmiTarget.ScreenTemplateFolder.Folders.Find("MyTemplateFolder");
    //or ScreenTemplateSystemFolder folder = hmiTarget.ScreenTemplateFolder;
    ScreenTemplateComposition templates = folder.ScreenTemplates;
    ScreenTemplate template = templates.Find("templateName");
    if(template == null) return;

    FileInfo info = new FileInfo(string.Format(@"D:\Samples\Templates\{0}\{1}.xml",
folder.Name, template.Name));
    template.Export(info, ExportOptions.WithDefaults);
}
```

程序代码：导出用户自定义文件夹的所有画面模板

修改以下程序代码以导出特定文件夹中的所有画面模板：

```
public static void ExportScreenTemplateUserFolder(string rootPath,
ScreenTemplateUserFolder folder)
{
    DirectoryInfo info = new DirectoryInfo(rootPath);
    info.Create();

    foreach (ScreenTemplate screen in folder.ScreenTemplates)
    {
        screen.Export(new FileInfo(Path.Combine(info.FullName, screen.Name + ".xml")),
ExportOptions.WithDefaults);
    }
    foreach (ScreenTemplateUserFolder subfolder in folder.Folders)
    {
        ExportScreenTemplateUserFolder(Path.Combine(info.FullName, subfolder.Name),
subfolder);
    }
}
```

程序代码：导出特定文件夹的所有画面模板

修改以下程序代码以导出所有画面模板：

```
//Exports all screen templates of a selected folder
private static void ExportScreenTemplates(string templatePath, ScreenTemplateUserFolder
folder)
{
    foreach (ScreenTemplate screen in folder.ScreenTemplates)
    {
        screen.Export(new FileInfo(Path.Combine(templatePath, screen.Name + ".xml")),
ExportOptions.WithDefaults);
    }
    foreach (ScreenTemplateUserFolder subfolder in folders.Folders)
    {
        ExportScreenTemplates(Path.Combine(templatePath, subfolder.Name), subfolder);
    }
}
```

8.3.9.9 导入画面模板

要求

- Openness 应用程序连接到 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)

应用

导入以下画面模板数据：

画面模板	数据
属性	ActiveLayer, BackColor, Height, Width, Name, SetTabOrderInFront
组合	• Layers • Animations 导入所有可组态画面动画。 • Softkeys 导入所有可组态画面软键。

为每个图层导入以下数据：

图层	数据
属性	Name, Index
组合	ScreenItems （包括画面项）

如果画面模板的宽度和高度与设备尺寸不一致，则取消导入且并触发 Exception。不支持调整所含的画面项。因此，某些画面项可能会超出画面边界。这种情况下，将输出编译器警告。

画面模板中画面项的布局必须唯一且连续。因此，导入画面模板后，如果有必要，则会执行一致性检查从而修复布局。此操作可能产生某些画面项的已修改的“选项卡索引”。

程序代码：常规导入

修改以下程序代码以使用 For each 循环将所有画面模板导入至 HMI 设备：

```
//Imports screen templates to an HMI device
private static void ImportScreenTemplatesToHMITarget(HmiTarget hmitarget)
{
    ScreenTemplateUserFolder folder =
hmitarget.ScreenTemplateFolder.Folders.Find("MyTemplateFolder");
    // or ScreenTemplateSystemFolder folder = hmitarget.ScreenTemplateFolder;
    FileInfo[] exportedTemplates = {new FileInfo[] { new FileInfo(@"D:\Samples\Import
\Template_1.xml"), new FileInfo(@"D:\Samples\Import\Template_n.xml") };};
    foreach (FileInfo templateFileName in exportedTemplates)
    {
        folder.ScreenTemplates.Import(templateFileName, ImportOptions.Override);
    }
}
```

程序代码：导入到新创建的用户文件夹

修改以下程序代码以将某个画面模板导入至新创建的 HMI 设备用户文件夹：

```
//Imports screen templates to a user folder of an HMI device
private static void ImportScreenTemplatesToFolderOfHMITarget(HmiTarget hmitarget)
{
    ScreenTemplateUserFolder screenTemplateFolder =
    hmitarget.ScreenTemplateFolder.Folders.Find("MyTemplateFolder");
    ScreenTemplateUserFolder folder = screenTemplateFolder.Folders.Create("MyNewFolder");
    folder.ScreenTemplates.Import(new FileInfo(@"D:\Samples\Import\ScreenTemplate.xml"),
ImportOptions.Override);
}
```

8.3.9.10 导出弹出画面

要求

- Openness 应用程序连接到 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)

应用

导出以下弹出画面数据：

画面模板	数据
属性	ActiveLayer, BackColor, GridColor, Height, Name, ScrollbarBackgroundColor, ScrollbarForegroundColor, Width
组合	• Layers • Events 导出所有已组态的事件。

为每个图层导出以下数据：

图层	数据
属性	Name, Index, VisibleES
组合	ScreenItems 导出所有可导出的画面对象。

程序代码：从文件夹中导出弹出画面

修改以下程序代码以导出系统文件夹或用户自定义文件夹中的单个弹出画面：

```
//Exports a single pop-up screen
private static void ExportSinglePopUpScreen(HmiTarget hmitarget)
{
    ScreenPopupUserFolder folder =
hmitarget.ScreenPopupFolder.Folders.Find("MyPopupFolder");
    //or ScreenPopupSystemFolder folder = hmitarget.ScreenPopupFolder;
    ScreenPopupComposition popups = folder.ScreenPopups;
    ScreenPopup popup = popups.Find("popupName");
    if(popup == null) return;

    FileInfo info = new FileInfo(string.Format(@"D:\Samples\Screens\{0}\{1}.xml",
folder.Name, popup.Name);
    popup.Export(info, ExportOptions.WithDefaults);
}
```

8.3.9.11 导入弹出画面

要求

- Openness 应用程序连接到 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)

应用

导入以下弹出画面数据：

画面模板	数据
属性	ActiveLayer, BackColor, GridColor, Height, Name, ScrollbarBackgroundColor, ScrollbarForegroundColor, Width
组合	• Layers • Events 导出所有已组态的事件。

要进行导入，必须存在以下属性：

- Name
- Height
- Width

为每个图层导入以下数据：

图层	数据
属性	Name, Index, VisibleES
组合	ScreenItems 导入所有可导入的画面对象。

限制

如果设备不支持弹出画面，则导入会被取消，且会触发一个异常。

如果弹出画面的宽度和高度与设备的以下尺寸限制不符，则导入会被取消，且会触发 Exception：

- 最小高度 = 1 个像素
- 最小宽度 = 1 个像素
- 最大高度 = 设备画面高度的六倍
- 最大宽度 = 设备画面宽度的两倍
- 对于运行系统版本为 V13 SP1 的设备，最大高度和最大宽度分别等于设备屏幕的高度和宽度。

程序代码：将弹出画面导入文件夹中

修改以下程序代码以将弹出画面导入弹出画面系统文件夹或用户自定义文件夹：

```
//Imports a pop-up screen to an HMI device
private static void ImportPopupScreenToHMITarget(HmiTarget hmitarget)
{
    FileInfo info = new FileInfo(string.Format(@"D:\Samples\Screens\PopupScreen.xml"));
    hmitarget.ScreenPopupFolder.ScreenPopups.Import(info, ImportOptions.None);
}
```

8.3.9.12 导出滑入画面

要求

- Openness 应用程序已连接 TIA Portal。
  参见“连接到 TIA Portal (页 69)”
- 项目已打开。
  参见“打开项目 (页 96)”

应用

导出滑入画面中的以下数据和值：

画面模板	数据	
属性	Activate	false
	ActiveLayer	0
	BackColor	(182; 182; 182)
	GridColor	(0; 0; 0)
	Dimension	427 属性“Dimension”用于指定滑入画面的宽度或高度，具体取决于所用的滑入画面类型。
	LineColor1	(223; 223; 223)
	LineColor2	(32; 32; 32)
	OperatableAreaColor	(128; 128; 128)
	SlideinType	顶部 (Top)，底部 (Bottom)，左 (Left)，右 (Right) 滑入画面没有名称，但包含有 SlideinType。
	Visibility	FadeOut
组合	Layers	

说明

滑入画面没有名称，但包含有 SlideinType。

每个图层都将导出以下数据：

图层	数据	
属性	Name,	
	Index	
	VisibleES	
组合	ScreenItems	导出所有可导出的画面对象。

程序代码：导出滑入画面

修改以下程序代码以从系统文件夹中导出单个滑入画面：

```
//Exports a single slide-in screen
private static void ExportSingleSlideinScreen(HmiTarget hmitarget)
{
    ScreenSlideinSystemFolder systemFolder = hmitarget.ScreenSlideinFolder;
    var screens = systemFolder.ScreenSlideins;
    ScreenSlidein slidein = screens.Find(SlideinType.Bottom);
    if (slidein == null) return;

    FileInfo info = new FileInfo(string.Format(@"D:\Samples\Screens\{0}\{1}.xml"));
    slidein.Export(info, ExportOptions.WithDefaults);
}
```

8.3.9.13 导入滑入画面

要求

- Openness 应用程序连接到 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)

应用

导入滑入画面的以下数据和值：

画面模板	数据
属性	Activate = false ActiveLayer = 0 Authorization BackColor = (182; 182; 182) Dimension = 427 属性“Dimension”指定滑入画面的宽度或高度，具体取决于这两个属性中的哪个属性可针对特定滑入类型进行修改。 GridColor = (0; 0; 0) LineColor1 = (223; 223; 223) LineColor2 = (32; 32; 32) OperateableAreaColor = (128; 128; 128) SlideinType = Top, Bottom, Left, Right Visibility = FadeOut
组合	Layers

要进行导入，必须存在以下属性：

- SlideinType

为每个图层导入以下数据：

图层	数据
属性	Name, Index, VisibleES
组合	ScreenItems 导入所有可导入的画面对象。

限制

- 如果设备不支持滑入画面，则导入会被取消，且会触发一个异常。
- 如果从其它元素引用滑入画面，则必须通过 openlink 引用滑入画面，而不能通过 SlideinType（例如，在系统函数“ShowSlideinScreen”中）进行引用。
  下表显示了通过相应 openlink 实现的 "SlideinType" 属性映射：

SlideinType	Openlink 名称
Top	GraphX_Slidein_Top
Right	GraphX_Slidein_Right
Bottom	GraphX_Slidein_Bottom
Left	GraphX_Slidein_Left

程序代码：将滑入画面导入文件夹中

修改以下程序代码以将滑入画面导入滑入画面系统文件夹中：

```
//Imports a slide-in screen to an HMI device
private static void ImportSlideinScreenToHMITarget(HmiTarget hmitarget)
{
    FileInfo info = new FileInfo(@"D:\Samples\Screens\SlideInScreen.xml");
    hmitarget.ScreenSlideinFolder.ScreenSlideins.Import(info, ImportOptions.None);
}
```

8.3.9.14 导出带有面板实例的画面

要求

- Openness 应用程序连接到 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)

应用

导出画面中的以下面板实例数据：

画面	数据
属性	Left, Top, Width, Height, ObjectName, Resizing, TabIndex, FaceplateTypeName
接口属性	针对可导出的画面项导出面板实例的所有已组态接口属性。
组合	• 动画 导出所有移动动画。 变量动画与接口属性有关。 • 事件 导出所有已组态的事件。

遵循面板实例的所导出属性的以下规范：

- Resizing
  在任何情况下都会导出属性“Resizing”，这与导出选项无关。
- FaceplateTypeName
  属性 "FaceplateTypeName" 定义相应的面板类型和版本，例如“Faceplate_1 V 0.0.2”。

说明

库文件夹中的面板类型

如果面板类型位于库文件夹中，则需要完整的路径和名称来识别面板类型。关键字“@$@”用于分隔文件夹和/或面板类型名称，例如“Folder_1@$@SubFolder_1@$@Faceplate_1 V 0.0.2”。

导出时不包含面板实例内部画面项的以下数据：

画面项	属性
IO 字段	Flashing on limit violation
图形 IO 字段	Fit embedded graphic object to screen size

程序代码

修改以下程序代码以导出包含面板实例的单个画面：

```
//Exports a single screen including a faceplate instance
private static void ExportSingleScreenWithFaceplateInstance(HmiTarget hmitarget)
{
    ScreenFolder folder = hmitarget.ScreenFolder.Folders.Find("MyScreenFolder");
    ScreenComposition screens = folder.Screens;
    Screen screen = screens.Find("ScreenWithFaceplateName");
    if (screen == null) return;
    {
       FileInfo info = new FileInfo(string.Format(@"D:\Samples\Faceplates\{0}\{1}.xml",
folder.Name, screen.Name));
       screen.Export(info, ExportOptions.WithDefaults);
    }
}
```

8.3.9.15 导入带有面板实例的画面

要求

- Openness 应用程序连接到 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)

应用

导入画面中的以下面板实例数据：

画面	数据
属性	Left, Top, Width, Height, ObjectName, Resizing, TabIndex, FaceplateTypeName
接口属性	针对可导入的画面项导入面板实例的所有已组态接口属性。
组合	• 动画 导入所有移动动画。 变量动画与接口属性有关。 • 事件 导出所有已组态的事件。

要进行导入，必须存在以下属性：

- ObjectName
- FaceplateTypeName

导出和导入时不包含面板实例内部画面项的以下数据：

画面项	属性
IO 字段	Flashing on limit violation
图形 IO 字段	Fit embedded graphic object to screen size

限制

- 未知面板、事件或接口属性
  如果在导入文件中指定项目所不包含的 faceplate type name、event name 或 interface property name，则导入会被中止，并且会触发异常。
- 调整面板实例的行为
  在任何情况下都会导入属性“Resizing”，这与导出选项无关。
  示例：
  如果将 "Resizing" 设为 "KeepRatio"，则会使用 "Height" 属性计算 "Width" 属性值。
  - 面板类型的大小为 100 x 100 像素。如果以 300 x 100 像素大小导入面板实例，并且为 "Resizing" 属性设置了值 "FixedSize"，则导入成功，并会将面板大小设为 100 x 100 像素。
  - 面板类型的大小为 100 x 50 像素。以 100 x 100 像素大小导入面板实例，并且为 "Resizing" 属性设置了值 "KeepRatio"。导入成功，面板大小被设为 200 x 100 像素。

  说明

  所导入面板实例的大小行为

  “Resizing”值和接口属性值会影响所导入面板实例的大小，甚至会影响封闭式画面项的大小。
  为避免在未经请求的情况下更改面板实例的外观，请导入初始大小的面板，或者甚至是导入无 "Width" 和“Height”属性值的面板。

- 异常接口属性值
  - 如果修改导入的属性，则会导入上次使用的接口属性值。
  - 如果属性彼此相关，则在导入期间可能会更改其它属性值。
    示例：面板包括一个 I/O 字段。属性“模式”(Mode) 连接至接口属性。如果先将模式设为“输出”(Output)，然后将属性“隐藏输入”(Hidden input) 设为真，则在导入后不会应用“隐藏输入”(Hidden input) 的值。首次修改会将属性“隐藏输入”(Hidden input) 设为只读型，因此不能应用相应的值。
  - 如果属性值不满足 WinCC 的限制，则会显示面板类型值。
    示例：将量表的显示范围设为 10 - 80。在接口属性中组态属性“最大值” (MaximumValue) 和“最小值”(MinimumValue)。如果设置的最小值超出最大值（例如，100），则在导入后会显示“最小值”(MinimumValue) 的面板类型值。
  - 如果一个接口属性与面板类型中的多个画面项属性相连，则面板实例中的接口属性值将显示首个连接的画面项的相应属性值。
    示例：一个面板包括两个带异常最大值的量表对象。两个量表的最小值连接至一个接口属性。
    如果先设置一个适用于两个量表的最小值，则会设置两个值。
    如果之后设置一个仅适用于第二个量表的值，则只会为第二个量表设置相应值，而第一个量表的值会显示为接口属性。

程序代码：导入包括面板实例的画面

修改以下程序代码以导入包括面板实例的画面：

```
//Imports single screen including a faceplate instance
private static void ImportSingleScreenWithFaceplateInstance(HmiTarget hmitarget)
{
    FileInfo info = new FileInfo(@"D:\Samples\Screens\ScreenFaceplate.xml");
    hmitarget.ScreenFolder.Screens.Import(info, ImportOptions.None);
}
```

8.4 导入/导出 PLC 设备的数据

8.4.1 块

8.4.1.1 块接口部分的 XML 结构

基本原理

导入/导出操作中导出文件的数据为结构化数据，且引用一个基本结构。每个导入文件都必须满足基本结构条件。

导出文件包括导出块的接口部分的所有已编辑的变量和常量。排除具有 "ReadOnly="TRUE"“和 Informative"="TRUE"“的所有属性。

如果信息是冗余的，则它在导入 XML 文件和项目数据中必须完全相同。否则，导入将出现可恢复的异常。

项目数据可以包含比导入 XML 文件更多的数据，例如外部类型可以有附加成员

只有可写入的值可以通过 Openness XML 导入。

根据 Openness 导出设置，导出文件包括一组已定义的属性和元素。

基本结构

导出块的接口部分包含在块的 SimaticML 中的 <Interface> 元素中。根对象是 <Sections> 元素，它表示导出块的接口部分。以下元素描述的顺序表示输入文件中要求的顺序。

```
<Interface>
    <Sections xmlns="http://www.siemens.com/automation/Openness/SW/Interface/v1">
        <Section Name="Input">
            <Member Name="input1" Datatype="Bool" Remanence="Volatile" Accessibility="Public">
              <AttributeList>
                ...
              </AttributeList>
            </Member>
            <Member Name="input2" Datatype="Bool" Remanence="Volatile" Accessibility="Public">
                <AttributeList>
                  ...
                </AttributeList>
            </Member>
        </Section>
        <Section Name="Output">
            <Member Name="output1" Datatype="Bool" Remanence="Volatile" Accessibility="Public">
                <AttributeList>
                    ...
                </AttributeList>
            </Member>
        </Section>
        <Section Name="InOut" />
        <Section Name="Static" />
        <Section Name="Temp" />
        <Section Name="Constant" />
    </Sections>
</Interface>
```

- 部分
  部分表示程序块的单个参数或本地数据
- 成员
  成员表示程序块中使用的变量或常量。根据变量的数据类型，成员可以嵌套或具有其它结构化子元素。
  对于数据类型“ARRAY”，结构元素“Subelement Path”表示数组元素组件的索引。
  只能导出由用户编辑的成员。
- AttributeList
  <AttributeList> 包括成员的所有已定义属性。系统定义或由标准值分配的属性未在 XML 结构中列出。
  成员属性 <ReadOnly> 和 <Informative> 仅在值为 TRUE 时写入 XML 导出文件。

- StartValue
  只有当用户设置了变量或常量的默认值时，才会写入 <StartValue> 元素。

```
...
<Member Name="Static_1" Datatype="Int">
 ...
 <StartValue>1</StartValue>
</Member>
...
```

- 注释
  只有当用户设置了 <Comment> 元素时，才会将其写入。变量或常量的注释将导出为多语言文本：

```
...
 <Member Name="Static_3" Datatype="Struct">
   <AttributeList>
     <BooleanAttribute Name="ExternalAccessible" SystemDefined="true">false</BooleanAttribute>
     <BooleanAttribute Name="ExternalVisible" SystemDefined="true">false</BooleanAttribute>
     <BooleanAttribute Name="ExternalWritable" SystemDefined="true">false</BooleanAttribute>
   </AttributeList>
   <Comment>
     <MultiLanguageText Lang="de-DE">An individual comment</MultiLanguageText>
   </Comment>
 </Member>
...
```

属性

- 主要属性
  主要属性写入 XML 结构的 <Member> 元素中。

```
...
<Member Name="Static_1" Datatype="&amp;quot;User_data_type_1&amp;quot;" Remanence="Retain">
  ...
</Member>
...
```

下表显示了块接口部分的变量或常量的主要属性。

名称	数据类型	默认值	导入条件	注释
名称	STRING	-	必选项	
数据类型	ENUM	-	必选项	
版本	STRING	-	可选	
剩余	ENUM	NonRetain	-	只在非默认值时写入
可访问性	ENUM	Public	-	由系统预定义用户无法更改
信息	BOOL	FALSE	-	

具有标志“Informative”的成员在导入期间将被忽略。如果属性被删除或设置为 FALSE，则发生异常。

说明

剩余设置“Set in IDB”

如果变量或常量的剩余值为“Set in IDB”，则 IDB 中的剩余设置要与所有其它具有剩余值“Set in IDB”的变量和常量相同。

具有“Set in IDB”属性的第一个导入成员在 IDB 中为以下具有剩余值“SetInIDB”的标签和常量定义预期剩余。

- 系统定义的成员属性
  系统定义的成员属性将在 <AttributeList> 元素中列出。系统定义的成员属性具有 <Informative> 标志并在导入时被忽略。

```
...
 <Member Name="Static_3" Datatype="Struct">
   <AttributeList>
     <BooleanAttribute Name="ExternalAccessible" SystemDefined="true">false</BooleanAttribute>
     <BooleanAttribute Name="ExternalVisible" SystemDefined="true">false</BooleanAttribute>
     <BooleanAttribute Name="ExternalWritable" SystemDefined="true">false</BooleanAttribute>
   </AttributeList>
   <Comment>
     <MultiLanguageText Lang="de-DE">An individual comment</MultiLanguageText>
   </Comment>
 </Member>
...
```

Name	类型	默认值	SimaticML 只读（供参考）	注释
At	string	""	FALSE	成员与此结构中的另一成员共享偏移量
SetPoint	bool	FALSE	FALSE	时间可以与工作内存同步
UserReadOnly	bool	FALSE	TRUE	用户不能更改任何成员属性（包括名称）
UserDeletable	bool	TRUE	TRUE	编辑器不允许删除成员
HmiAccessible	bool	TRUE	FALSE	无 HMI 访问，无结构项
HmiVisible	bool	TRUE	FALSE	过滤以减少显示在第一位置的成员数量
Offset	int	-	TRUE	DB，FB，FC (Temp)。适用于经典 PLC 和剩余设置为经典的 Plus PLC。

Name	类型	默认值	SimaticML 只读（供参考）	注释
PaddedSize	int	-	TRUE	DB，FB，FC (Temp)。适用于经典 PLC 和剩余设置为经典的 Plus PLC。仅适用于数组。
HiddenAssignment	bool	FALSE	FALSE	如果与 PredefinedAssignment 匹配，则隐藏调用时的分配
PredefinedAssingment	string	""	FALSE	调用时使用的参数输入
ReadOnlyAssignment	bool	FALSE	FALSE	用户不能在调用时更改预定义分配
UserVisible	bool	TRUE	TRUE	此成员不在 UI 上显示
HmiReadOnly	bool	TRUE	TRUE	此成员对于 HMI 只读
CodeReadOnly	bool	FALSE	TRUE	-

- 用户定义属性
  用户定义属性标记为<ReadOnly>。具有此标记的成员在导入时将被忽略。如果标记被删除或设置为 FALSE，则发生异常。
  未编辑的用户定义属性将从导出中排除。

名称	类型	默认值	SimaticML 只读（供参考）	注释
CFC	IBlockAttribute	---	FALSE	此为有效负载

数据类型“STRUCT”

数据类型“STRUCT”的组成部分在导入/导出文件的 XML 结构中表示为嵌套成员：

```
<Sections xmlns="http://www.siemens.com/automation/Openness/SW/Interface/v2">
  <Section Name="Static">
    <Member Name="Static_1" Datatype="Struct">
      <!-- Basic struct -->
      <Member Name="Static_1" Datatype="Int">
        <!-- First Member of struct -->
        <StartValue>1</StartValue>
      </Member>
      <Member Name="Static_2" Datatype="Int">
        <!-- Second Member of struct -->
      </Member>
      <Member Name="Static_3" Datatype="Struct">
        <!-- A subsequent struct -->
        <Member Name="Static_1" Datatype="Int">
          <!-- First Member of the subsequent struct -->
          <StartValue>3</StartValue>
        </Member>
        <Member Name="Static_2" Datatype="Int">
          <!-- Second Member of the subsequent struct -->
        </Member>
      </Member>
    </Member>
  </Section>
</Sections>
```

数据类型“ARRAY”基本类型

基本数据类型“ARRAY”的组成部分在导入/导出文件的 XML 结构中表示为具有"Path"属性的子元素：

```
<Sections xmlns="http://www.siemens.com/automation/Openness/SW/Interface/v2">
  <Section Name="Static">
    <Member Name="Static_1" Datatype="Array[0..2] of Int" Remanence="Retain">
      <!-- Basic Array -->
      <Subelement Path="0">
        <!-- First Array Component-->
        <StartValue>1</StartValue>
      </Subelement>
      <Subelement Path="1">
        <!-- Second Array Component-->
        <StartValue>2</StartValue>
      </Subelement>
    </Member>
  </Section>
</Sections>
```

UDT 的数据类型“ARRAY”

UDT 的数据类型“ARRAY”的组成部分在导入/导出文件的 XML 结构中表示为 <member> 元素的新 <sections> 元素。UDT 新部分中的成员在 ARRAY 中，分配了具有 "Path" 属性的子元素：

```
<Sections xmlns="http://www.siemens.com/automation/Openness/SW/Interface/v2">
  <Section Name="Static">
    <Member Name="Static_1" Datatype="&quot;User_data_type_1&quot;" Remanence="Retain">
      <Sections> <!-- Sections including the UDT "User_data_type_1" -->
        <Section Name="None">
          <Member Name="Element_2" Datatype="Int">
            <StartValue>47</StartValue>
          </Member>
        </Section>
      </Sections>
    </Member>
    <Member Name="Static_2" Datatype="Array[0..1] of &quot;User_data_type_1&quot;">
      <Sections> <!-- Sections including the UDT "User_data_type_1" -->
        <Section Name="None">
          <Member Name="Element_2" Datatype="Int">
            <Subelement Path="0">  <!-- Component of the array -->
              <StartValue>123</StartValue>
            </Subelement>
          </Member>
        </Section>
      </Sections>
    </Member>
  </Section>
</Sections>
```

“ARRAY”中的数据类型“ARRAY”

在另一个 ARRAY 中，数据类型“ARRAY”的组成部分在导入/导出文件的 XML 结构中表示为具有"Path"属性的子元素。

如果组成部分由用户编辑，则将另一个 ARRAY 中的成员指定为带有 "Path" 属性的子元素：

```
<Sections xmlns="http://www.siemens.com/automation/Openness/SW/Interface/v2">
  <Section Name="Static">
    <Member Name="Static_1" Datatype="Array[0..2] of Struct">
      <Member Name="Static_1" Datatype="Int" />
      <Member Name="Static_2" Datatype="Array[0..1, 0..3, -9..-2] of Struct">
        <Member Name="Static_1" Datatype="Int">
          <Subelement Path="0,0,3,-5">
            <StartValue>1</StartValue>
          </Subelement>
        </Member>
        <Subelement Path="0,0,2,-6">
          <Comment>
            <MultiLanguageText Lang="de-DE">A individual comment</MultiLanguageText>
          </Comment>
        </Subelement>
      </Member>
    </Member>
  </Section>
</Sections>
```

PLC 数据类型 (UDT)

PLC 数据类型的 XML 结构取决于 Openness 导出设置。

- `ExportOptions.None`
  仅当用户将至少设置了一个组合的默认值时，才会写入 PLC 数据类型的元素。对于这些元素，仅当写入两个附加属性 "Name" 和 "Datatype" 时，才能对 <StartValue> 所属的元素成员进行标识。而不会写入其它元素和属性。
- `ExportOptions.WithDefaults`
  始终写入以下属性：
  - `Name`
  - `Datatype`
  - `ExternalAccessible`
  - `ExternalVisible`
  - `ExternalWritable`
  - `SetPoint`
  - `StartValue`
    如果此类型中的默认值是由用户设置的，则仅写入 XML。如果仅在 PLC 数据类型中设置，则不写入。
- `ExportOptions.ReadOnly`
  对于 PLC 数据类型，该设置无意义。与其它设置组合使用时，也不会影响最终结果。

覆盖变量

如果变量使用新的数据类型进行覆盖，则相应元素将以新数据类型的 XML 结构进行表示。以下 XML 结构显示由 BYTE 的 ARRAY 覆盖的数据类型 WORD。

```
<Sections xmlns="http://www.siemens.com/automation/Openness/SW/Interface/v2">
  <Section Name="Input" />
  <Section Name="Output" />
  <Section Name="InOut" />
  <Section Name="Static">
    <Member Name="Static_1" Datatype="Word">
      <!-- Basic Member -->
      <StartValue>16#17</StartValue>
    </Member>
    <Member Name="Static_2" Datatype="Array[0..1] of Byte">
      <AttributeList>
        <StringAttribute Name="At" SystemDefined="true">Static_1</StringAttribute>
      </AttributeList>
        <!-- The AT member -->
      <Subelement Path="0">
        <!-- First overlay byte -->
      </Subelement>
      <Subelement Path="1">
        <!-- Second overlay byte -->
      </Subelement>
    </Member>
  </Section>
  <Section Name="Temp" />
  <Section Name="Constant" />
</Sections>
```

块接口

将执行 ReadOnly="TRUE" 和 Informative="FALS" 的所有属性。块接口的 XML 结构取决于 Openness 的导出设置。

- ExportOptions.None
  此设置只能导出修改后的数据或与默认值不同的数据。
  如果属性定义未指定默认值，则始终写入该属性。
  导出文件中还包含对于后续数据导入必需的所有值。
- ExportOptions.WithDefaults
  通常将写入以下属性
  - Name
  - Datatype
  - HmiAccessible
  - HmiVisible
  - SetPoint（如适用）
  - Offset（如适用）
  - PaddedSize（如适用）

  所有其它属性仅在其值与默认值不同时写入。
  如果已明确设置，则 <StartValue> 元素只写入 XML。
- ExportOptions.ReadOnly
  对于块接口，该设置无意义。与其它设置组合使用时，也不会影响最终结果。

8.4.1.2 对象模型和 XML 文件格式的变化

简介

要通过 Openness 成功将自定义创建的或已编辑的 XML 文件导入到 TIA Portal，文件必须对应于定义的架构。

XML 文件始终包含两个主要部分：

- 接口
- 编译单元

下面介绍了文件所需对应的架构。

接口

接口可包含多个部分（例如，输入、输入输出和静态）：在以下目录中给出了该架构下的所有相关部分：

C:\Program Files\Siemens\Automation\Portal V14\PublicAPI\V14 SP1\Schemas\SW.InterfaceSections_v2.xsd

编译单元

对于 GRAPH、LAD/FBD 和 STL 块的编译单元，存在多个架构。可以在以下目录中找到这些架构：

- GRAPH：C:\Program Files\Siemens\Automation\Portal V ...\PublicAPI\Schemas\SW.PlcBlocks.Graph.xsd
- LAD/FBD：C:\Program Files\Siemens\Automation\Portal V ...\PublicAPI\Schemas\SW.PlcBlocks.LADFBD.xsd
- STL：C:\Program Files\Siemens\Automation\Portal V ...\PublicAPI\Schemas\SW.PlcBlocks.STL.xsd

子架构

所有编译单元还使用以下额外的架构定义：

- Access
- 公共

Access

例如，“访问”节点介绍了以下内容：

- 局部/全局成员以及常数使用情况
- FB、FC 和指令调用
- 用于调用的 DB

可以在以下目录中找到访问架构：

C:\Program Files\Siemens\Automation\Portal V14\PublicAPI\V ...\Schemas\SW.PlcBlocks.Access.xsd

公共

公共部分包含常用的属性和元素，例如，不同类型的注释、文本和令牌。

可以在以下目录中找到公共架构：

C:\Program Files\Siemens\Automation\Portal V14\PublicAPI\V ...\Schemas
\SW.Common.xsd

8.4.1.3 导出块

要求

- Openness 应用程序已连接 TIA Portal。
  参见“连接到 TIA Portal (页 69)”
- 项目已打开。
  参见“打开项目 (页 96)”
- PLC 未在线。

应用

API 接口支持将一致的块和用户数据类型导出到 XML 文件。

XML 文件接收块的名称。支持以下块类型：

- 函数块 (FB)
- 函数 (FC)
- 组织块 (OB)
- 全局数据块 (DB)

支持以下编程语言：

- STL
- FBD
- LAD
- GRAPH
- SCL
  SCL 块导出过程中生成的 xml 文件仅包含块接口。

导出的数据

将导出以下属性，具体取决于块或用户数据类型以及所选 ExportOptions（参见“导出组态数据 (页 343)”）。始终导出以粗体显示的属性。

属性	类型	默认值	只读	有效性
Name	String	下一个可用编号	false	所有块
Number	Int32	-	false	所有块
AutoNumber	Bool	true	false	所有块
HeaderAuthor	String	""	false	所有块
HeaderFamily	String	""	false	所有块
HeaderName	String	""	false	所有块
HeaderVersion	String	“0.1”	false	所有块
MemoryLayout	enum MemoryLayout	-	false	所有块
IsWriteProtectedInAS	Bool	false	false	仅 DB
IsOnlyStoredInLoadMemory	Bool	false	false	仅 DB
IsIECCheckEnabled	Bool	false	false	仅 DB 和 FB
IsRetainMemResEnabled[1]	Bool	false	false	仅 DB 和 FB
MemoryReserve	Unsigned	0	false	仅 DB 和 FB
RetainMemoryReserve[1]	Unsigned	0	false	仅 DB 和 FB
SecondaryType[2]	String	-	false	仅 OB
ProgrammingLanguage	enum ProgrammingLanguage	-	false	所有块
IsKnowHowProtected	Bool	false	true	所有块
InstanceOfName	String	""	false	仅 FB 的背景数据块和 UDT 的背景数据块
InstanceOfNumber	Unsigned Short	-	true	仅 FB 的背景数据块和 UDT 的背景数据块
InstanceOfType	enum BlockType	-	true	仅 FB 的背景数据块和 UDT 的背景数据块

属性	类型	默认值	只读	有效性
OfSystemLibElement	String	""	false	仅 FB 的背景数据块和 UDT 的背景数据块
OfSystemLibVersion	String	""	false	仅 FB 的背景数据块和 UDT 的背景数据块
ParameterPassing	Bool	false	false	仅 FB 和 FC (STL)
CreationDate	DateTime	-	true	所有块
ModifiedDate	DateTime	-	true	所有块
Interface	String	空接口	false	所有块
InterfaceModifiedDate	DateTime	-	true	所有块
ParameterModified	DateTime	-	true	所有块
StructureModified	DateTime	-	true	所有块
CodeModifiedDate	DateTime	-	true	所有块
CompileDate	DateTime	-	true	所有块
IsConsistent	Bool	-	true	所有块
UDAEnableTagReadback	Bool	false	false	仅 FB、FC 和 FB 的背景数据块
UDABlockProperties	String	""	false	仅 FB、FC 和 FB 的背景数据块
HandleErrorsWithinBlock	Bool	false	true	仅 FB、FC 和 OB
PLCSimAdvancedSupport	Bool	false	true	所有块
LibraryConformanceStatus	String	-	false	仅 FB、FC 和 UDT
IsPLCDB	Bool	false	false	仅 DB
SkipSteps	Bool	false	false	仅 GRAPH 块
AcknowledgeErrorsRequired	Bool	true	false	仅 GRAPH 块
PermanentILProcessingInMANMode	Bool	false	false	仅 GRAPH 块
LockOperatingMode	Bool	false	false	仅 GRAPH 块
SetENOAutomatically	Bool	-	false	仅 GRAPH 块
CreateMinimizedDB	Bool	false	false	仅 GRAPH 块
LanguageInNetworks	String	-	false	仅 GRAPH 块

属性	类型	默认值	只读	有效性
ISMultiInstanceCapable	Bool	-	true	仅 FB
ArrayDataType	String	-	true	仅 ArrayDB
ArrayLimitUpperBound	Int32	-	true	仅 DB
GraphVersion	String	-	false	仅 GRAPH 块
WithAlarmHandling	Bool	true	false	仅 GRAPH FB 块
DownloadWithoutReinit	Bool	false	true	如果不是故障安全块，则无 GRAPH
LibraryType	String	-	true	仅 FB 和 FC
LibraryTypeVersionGuid	String	-	true	仅 FB 和 FC

[1] 如果“IsRetainMemResEnabled”属性值为“false”，且“RetainMemoryReserve”属性值不等于“0”，则会触发异常。

[2] 导出 OB 时，将基于 OB 编号额外设置“SecondaryType”。导入期间会检查分配。如果分配错误，则会触发“Recoverable”类型的异常。

更多信息，请参见 TIA Portal 信息系统的“块属性的概述”部分。

程序代码

修改以下程序代码以将不具有专有技术保护的块导出至 XML 文件：

```
//Exports a regular block
private static void ExportRegularBlock(PlcSoftware plcSoftware)
{
    PlcBlock plcBlock = plcSoftware.BlockGroup.Blocks.Find("MyBlock");
    plcBlock.Export(new FileInfo(string.Format(@"D:\Samples\{0}.xml", plcBlock.Name)),
ExportOptions.WithDefaults);
}
```

8.4.1.4 导出具有专有技术保护的块

要求

- Openness 应用程序连接到 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)
- PLC 未处于在线状态。

应用

生成的 XML 文件与没有专有技术保护的块导出文件相似。然而，导出只涉及不使用密码打开块时可见的用户界面数据。

块的属性列表指示相关的块具有专有技术保护。

程序代码

修改以下程序代码以将具有专有技术保护的块的可见数据导出至 XML 文件：

```
private static void ExportBlock(PlcSoftware plcSoftware)
{
    PlcBlock plcBlock = plcSoftware.BlockGroup.Blocks.Find("MyBlock");
    plcBlock.Export(new FileInfo(string.Format(@"D:\Samples\{0}.xml", plcBlock.Name)),
ExportOptions.WithDefaults);
}
```

8.4.1.5 导出故障安全块

导出故障安全块

故障安全块的导出方式与标准块相同。对于故障安全块，属性“ProgrammingLanguage”的值带有前缀“F_”。

说明

如果属性“ProgrammingLanguage”的值带有前缀“F_”，则文件无法导入。

将故障安全块作为标准块导入

如果将所有属性“ProgrammingLanguage”值中的前缀“F_”删除，则可将故障安全块作为标准块导入。

参见

连接到 TIA Portal (页 69)

打开项目 (页 96)

导出块 (页 422)

导出具有专有技术保护的块 (页 426)

8.4.1.6 导出系统块

要求

- Openness 应用程序已连接 TIA Portal。
  参见“连接到 TIA Portal (页 69)”
- 项目已打开。
  参见“打开项目 (页 96)”
- 项目包含系统块。
- PLC 未在线。

应用

仅块组合中可见的系统块可用，如无 SFB 或 SFC。XML 文件的生成过程与块的导出文件过程相似。

程序代码

修改以下程序代码以将块的可见数据导出至 XML 文件：

```
//Exports system blocks
private static void ExportSystemBlocks(PlcSoftware plcsoftware)
{
    PlcSystemBlockGroup sbSystemGroup = plcsoftware.BlockGroup.SystemBlockGroups[0];
    foreach (PlcSystemBlockGroup group in sbSystemGroup.Groups)
    {
        foreach (PlcBlock block in group.Blocks)
        {
            block.Export(new FileInfo(string.Format(@"D:\Samples\{0}.xml", block.Name)),
ExportOptions.WithDefaults);
        }
    }
}
```

8.4.1.7 导入块

要求

- Openness 应用程序已连接 TIA Portal。
  参见“连接到 TIA Portal (页 69)”
- 项目已打开。
  参见“打开项目 (页 96)”
- PLC 未在线。

应用

公共 API 支持从 XML 文件导入采用“STL”、“FBD”、“GRAPH”或“LAD”编程语言的块。支持以下块类型：

- 函数块 (FB)
- 函数 (FC)
- 组织块 (OB)
- 全局数据块 (DB)

说明

导入优化数据块

只有 S7-1200 或更高版本的 CPU 支持优化数据块。如果将已优化的数据块导入到 S7-300 或 S7-400，将触发意外事件并导致导入失败。

对导入操作的响应

导入块时以下规则适用：

- XML 文件包含的数据可以少于项目中的块，例如参数更少。
- 在项目和 XML 文件中，调用信息等冗余信息必须相同。否则，会触发异常。
- XML 文件中的数据就其在 TIA Portal 中被编译的能力而言，可能为“不一致”。
- 不会导入具有“ReadOnly=True”和“Informative=True”属性的特性。
- 缺失的背景数据块不会自动创建。
- 如果 xml 文件中未指定块编号，则系统将自动分配块编号。
- 如果该块在项目中不存在，且未在 xml 文件中指定版本信息，则将分配版本号“0.1”。

程序代码

修改以下程序代码：

```
//Import blocks
private static void ImportBlocks(PlcSoftware plcSoftware)
{
    PlcBlockGroup blockGroup = plcSoftware.BlockGroup;
    IList<PlcBlock> blocks = blockGroup.Blocks.Import(new FileInfo(@"D:\Blocks
\myBlock.xml"), ImportOptions.Override);
}
```

修改以下程序代码：

```
//Import system blocks
private static void ImportSystemBlocks(PlcSoftware plcSoftware)
{
    PlcBlockSystemGroup systemblockGroup = plcSoftware.BlockGroup;
    IList<PlcBlock> blocks = systemblockGroup.Blocks.Import(new FileInfo(@"D:\Blocks
\myBlock.xml"), ImportOptions.Override);
}
```

8.4.2 变量表

8.4.2.1 导出 PLC 变量表

要求

- Openness 应用程序连接到 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)

应用

为每个 PLC 变量表导出一个 XMI 文件。

Public API 支持从系统组及其子组中导出所有 PLC 变量表。

程序代码

修改以下程序代码以从系统组及其子组中导出所有 PLC 变量表：

```
private static void ExportAllTagTables(PlcSoftware plcSoftware)
{
    PlcTagTableSystemGroup plcTagTableSystemGroup = plcSoftware.TagTableGroup;
    // Export all tables in the system group
    ExportTagTables(plcTagTableSystemGroup.TagTables);
    // Export the tables in underlying user groups
    foreach(PlcTagTableUserGroup userGroup in plcTagTableSystemGroup.Groups)
    {
        ExportUserGroupDeep(userGroup);
    }
}

private static void ExportTagTables(PlcTagTableComposition tagTables)
{
    foreach(PlcTagTable table in tagTables)
    {
        table.Export(new FileInfo(string.Format(@"D:\Samples\{0}.xml", table.Name)),
ExportOptions.WithDefaults);
    }
}

private static void ExportUserGroupDeep(PlcTagTableUserGroup group)
{
    ExportTagTables(group.TagTables);
    foreach(PlcTagTableUserGroup userGroup in group.Groups)
    {
        ExportUserGroupDeep(userGroup);
    }
}
```

参见

导出组态数据 (页 343)

8.4.2.2 导入 PLC 变量表

要求

- Openness 应用程序连接到 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)

程序代码

修改以下程序代码以将 PLC 变量表或带有 PLC 变量表的文件夹结构从 XML 文件导入至系统组或用户自定义组：

```
//Imports tag tables to the tag system group
private static void ImportTagTable(PlcSoftware plcSoftware)
{
    PlcTagTableSystemGroup plcTagTableSystemGroup = plcSoftware.TagTableGroup;
    PlcTagTableComposition tagTables = plcTagTableSystemGroup.TagTables;
    tagTables.Import(new FileInfo(@"D:\Samples\myTagTable.xml"), ImportOptions.Override);
    // Or, to import into a subfolder:
    // plcTagTableSystemGroup.Groups.Find("SubGroup").TagTables.Import(new FileInfo(@"D:
\Samples\myTagTable.xml"), ImportOptions.Override);
}
```

参见

关于 TIA Portal Openness 性能的说明 (页 36)

8.4.2.3 导出来自 PLC 变量表的单个变量或常量

要求

- Openness 应用程序连接到 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)

应用

API 接口支持将 PLC 变量表中的单个变量或常量导出到 XML 文件。请确保所使用的变量表名符合文件系统的文件命名约定。

只有为注释至少设定了一种语言时，才能导出变量或常量的注释。如果未针对所有项目语言设置注释，仅会为设定的项目语言导出此注释。

说明

PLC 系统常量

PLC 系统常量从导出和导入中删除。

程序代码

修改以下程序代码以将特定变量或常量从 PLC 变量表导出至 XML 文件：

```
//Exports a single tag or constant of a controller tag table
private static void ExportTag(PlcSoftware plcSoftware, string tagName)
{
    PlcTagTableSystemGroup plcTagTableSystemGroup = plcSoftware.TagTableGroup;
    PlcTag tag = plcTagTableSystemGroup.TagTables[0].Tags.Find(tagName);
    if(tag 0= null) return;

    tag.Export(new FileInfo(string.Format(@"D:\Samples\{0}.xml", tag.Name)),
ExportOptions.WithDefaults);
}

private static void ExportUserConstant(PlcSoftware plcSoftware, string userConstantName)
{
    PlcTagTableSystemGroup plcTagTableSystemGroup = plcSoftware.TagTableGroup;
    PlcUserConstant plcConstant =
plcTagTableSystemGroup.TagTables[0].UserConstants.Find(userConstantName);
    if(plcConstant== null) return;

    plcConstant.Export(new FileInfo(string.Format(@"D:\Samples\{0}.xml",
plcConstant.Name)), ExportOptions.WithDefaults);
}
```

参见

导出组态数据 (页 343)

关于 TIA Portal Openness 性能的说明 (页 36)

8.4.2.4 将单个变量或常数导入 PLC 变量表

要求

- Openness 应用程序连接到 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 已打开一个项目。
  请参见打开项目 (页 96)

应用

可以在单个导入调用中导入各个变量或常量。

说明

常量只能以用户常量进行导入。

程序代码

修改以下程序代码以从 XML 文件导入变量组或逐个导入变量和常量：

```
//Imports tags into a plc tag table
private static void ImportTag(PlcSoftware plcSoftware, string tagtableName)
{
    PlcTagTableSystemGroup plcTagTableSystemgroup = plcSoftware.TagTableGroup;
    PlcTagTable tagTable = plcTagTableSystemgroup.TagTables.Find(tagtableName);
    if(tagTable == null) return;

    tagTable.Tags.Import(new FileInfo(@"D:\Samples\myTags.xml"), ImportOptions.Override);
}

//Imports constants into a plc tag table
private static void ImportConstant(PlcSoftware plcSoftware, string tagtableName)
{
    PlcTagTableSystemGroup plcTagTableSystemgroup = plcSoftware.TagTableGroup;
    PlcTagTable tagTable = plcTagTableSystemgroup.TagTables.Find(tagtableName);
    if(tagTable == null) return;

    tagTable.UserConstants.Import(new FileInfo(@"D:\Samples\myConstants.xml"),
ImportOptions.Override);
}
```

参见

导出组态数据 (页 343)

关于 TIA Portal Openness 性能的说明 (页 36)

8.4.3 导出用户数据类型

要求

- Openness 应用程序连接到 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 项目已经打开。
  请参见打开项目 (页 96)
- PLC 未处于在线状态。

程序代码

修改以下程序代码以将用户数据类型导出至 XML 文件：

```
//Exports a user defined type
private static void ExportUserDefinedType(PlcSoftware plcSoftware)
{
    string udtname = "udt name XYZ";
    PlcTypeComposition types = plcSoftware.TypeGroup.Types;
    PlcType udt = types.Find(udtname);
    udt.Export(new FileInfo(string.Format(@"C:\OpennessSamples\udts\{0}.xml", udt.Name)),
ExportOptions.WithDefaults);
}
```

8.4.4 导入用户数据类型

要求

- Openness 应用程序连接到 TIA Portal。
  连接到 TIA Portal (页 69)
- 已打开一个项目。
  打开项目 (页 96)
- PLC 未处于在线状态。

应用

API 接口支持从 XML 文件导入用户数据类型。

导入文件语法

以下代码示例为用户自定义数据类型的导入文件的一部分：

```
<Section Name="Input">
 <Member Name="Input1" Datatype=quot;myudt1&quot;>
  <Sections>
   <Section Name="None">
    <Member Name="MyUDT1Member1" Datatype="bool"/>
    <Member Name="MyUDT1Member2" Datatype=&quot;myudt1&quot;>
     <Sections...
```

说明

用户自定义数据类型的元素的语法

如果用户自定义数据类型的导入文件中的用户自定义数据类型的元素语法不正确，会触发异常。

确保用户自定义的数据类型标有 `&quot;`。

程序代码

修改以下程序代码以导入用户数据类型：

```
//Imports user data type
private static void ImportUserDataType(PlcSoftware plcSoftware)
{
    FileInfo fullFilePath = new FileInfo(@"C:\OpennessSamples\Import\ExportedPlcType.xml");
    PlcTypeComposition types = plcSoftware.TypeGroup.Types;
    IList<PlcType> importedTypes = types.Import(fullFilePath, ImportOptions.Override);
}
```

参见

导入组态数据 (页 345)

8.4.5 以 OPC UA XML 格式导出数据

要求

- Openness 应用程序连接到 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 项目已经打开。
  请参见打开项目 (页 96)
- PLC 未处于在线状态。

应用

可以通过在 Openness API 上调用操作来以 OPC UA XML 文件的形式导出 PLC 数据。对于操作的输入参数，您需要一个绝对目录路径，用于保存 xml 文件。

程序代码

修改以下程序代码。

```
//Export PLC data as OPC UA XML file
private static void OpcUaExport(Project project, DeviceItem plc)
{
    OpcUaExportProvider opcUaExportProvider =
project.HwUtilities.Find("OPCUAExportProvider") as OpcUaExportProvider;
    if (opcUaExportProvider == null) return;

    opcUaExportProvider.Export(plc, new FileInfo(string.Format(@"D:\OPC UA export files
\{0}.xml", plc.Name)));
}
```

8.5 导入/导出硬件数据

8.5.1 AML 文件格式

简介

AutomationML 基于 XML 且与数据格式无关的开放式标准，用于对工厂的工程组态信息进行存储和交换。AutomationML 旨在将不同专业领域中各种结构各异的现代工程组态工具系统连接在一起，如机械设备工程组态系统、电气设计、HMI、PLC、机器人控制等。

用于导出和导入 CAx 数据的类别模型基于以下 AML 标准：

- 白皮书 AutomationML 的第 1 部分 – AutomationML 架构，2014 年 10 月
- 白皮书 AutomationML 的第 2 部分 – AutomationML 角色库，2014 年 10 月
- 白皮书 AutomationML 的第 4 部分 – AutomationML 逻辑，2010 年 5 月
- 白皮书 AutomationML – AutomationML 通信，2014 年 9 月
- 白皮书 AutomationML – AutomationML 和 eCl@ss 集成，2015 年 11 月
- 应用建议：Automation Project Configuration - AR_001E 版本 V1.0.0，2017 年 5 月

架构

AutomationML 数据交换模型基于 CAEX 模式版本 V2.15。可从 AutomationML e.V (https://www.automationml.org/o.red.c/dateien.html) 主页下载该文件。

8.5.2 Pruned AML

简介

裁剪操作是指通过删除不需要的部分，对文件内容进行优化。使用诸如 EPLAN 之类的外部工具时，硬件配置中自动创建的子模块信息对 EPLAN 无效。因此，这些工具生成 AML 文件时将从硬件配置中删除自动创建的子模块信息。所生成的文件也称为“修剪 AML”。

生成修剪 AML 文件

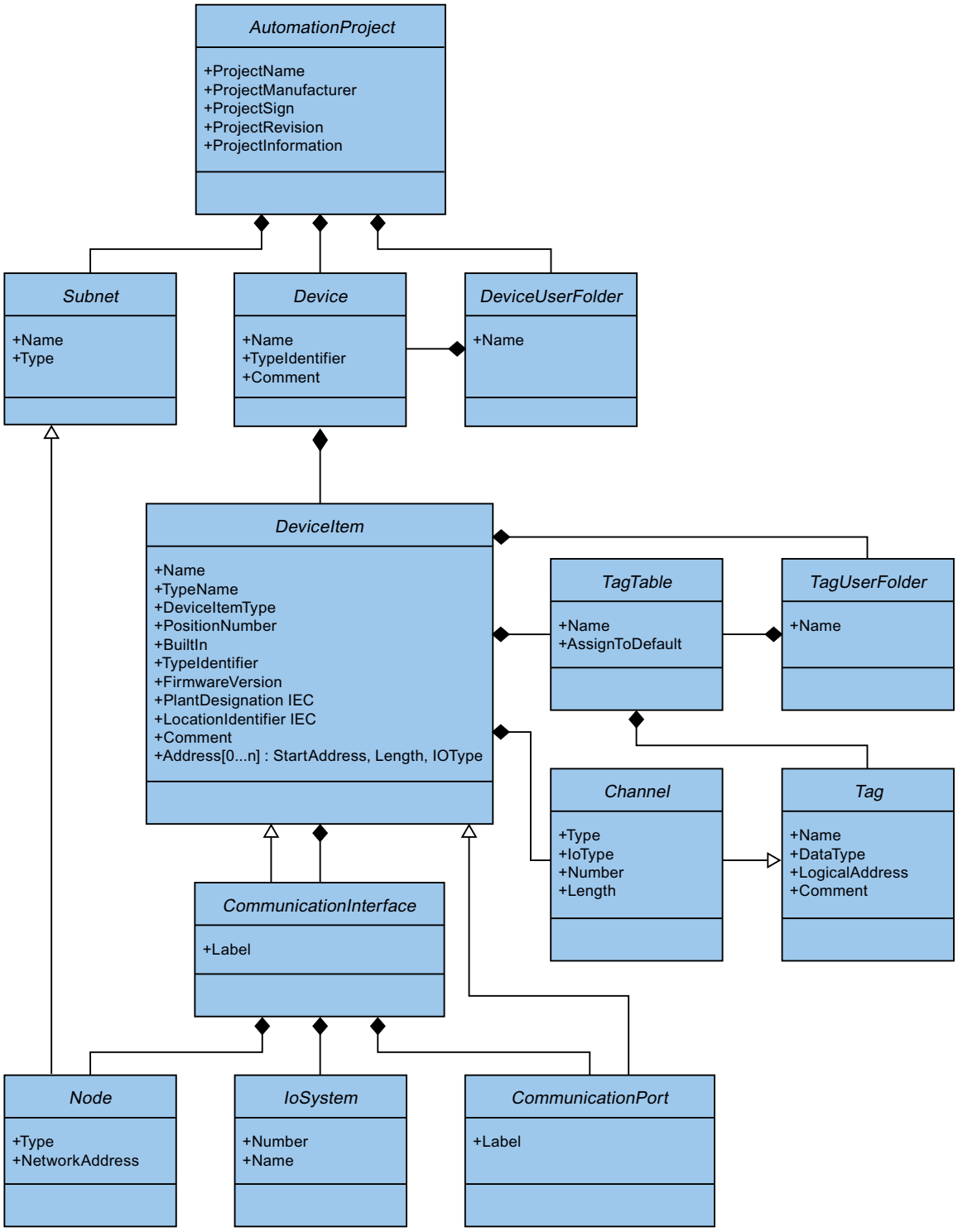
生成修剪 AML 时，需遵循以下规则顺序。

1. 如果设备项可插拔，则不会修剪。
2. 如果设备项的类型为“接口”或“端口”，则不会修剪。
3. 类型为“诊断”的 AddressObjects 与修剪算法无关。
4. 与自动创建子模块相关联的地址对象应位于直接父项（非自动创建的子模块）中。
5. 地址对象应包含在与 Openness 返回的相同顺序中。

8.5.3 CAx 导入/导出的对象和参数概述

导出/导入对象和属性

下图显示了可导出对象及其属性以及 CAx 导入/导出的相关性。


AutomationProject
+ProjectName
+ProjectManufacturer
+ProjectSign
+ProjectRevision
+ProjectInformation
Subnet
+Name
+Type
Device
+Name
+TypeIdentifier
+Comment
DeviceUserFolder
+Name
DeviceItem
+Name
+TypeName
+DeviceItemType
+PositionNumber
+BuiltIn
+TypeIdentifier
+FirmwareVersion
+PlantDesignation IEC
+LocationIdentifier IEC
+Comment
+Address[0...n] : StartAddress, Length, IOType
TagTable
+Name
+AssignToDefault
TagUserFolder
+Name
Channel
+Type
+IoType
+Number
+Length
Tag
+Name
+DataType
+LogicalAddress
+Comment
CommunicationInterface
+Label
Node
+Type
+NetworkAddress
IoSystem
+Number
+Name
CommunicationPort
+Label

8.5.4 用于导入/导出的 CAx 数据的结构

导出文件的基本结构

导入/导出操作中导出文件的数据为结构化数据，且引用一个基本结构。以 AML 格式生成导出文件。

AML 文件以文档信息为开始。该文件包含所导出项目的计算机特定安装的数据。

导出文件由以下两部分组成：

- 更多信息
  <WriterHeader> 包含导出或导入过程的相关信息。导入会忽略 <AdditionalInformation> 部分的内容。
  例如，可以插入 <AdditionalInformation>...</AdditionalInformation> 块，可在其中放置有关验证的附加信息。转发 AML 文件后，接收方可在导入前使用此块来检查 AML 文件是否已被更改。

```
<xml version="1.0" encoding="utf-8">
<CAEXFile xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        FileName="CAx_asterisk_AML_03_V14.aml" SchemaVersion="2.15"
        xsi:noNamespaceSchemaLocation="CAEX_ClassModel_V2.15.xsd">
  <AdditionalInformation>
    <WriterHeader>
      <WriterName>Totally Integrated Automation Portal</WriterName>
      <WriterID>1d4fcebb-1ad6-4881-b01d-bca335d94a46:V1.0</WriterID>
      <WriterVendor>Siemens AG</WriterVendor>
      <WriterVendorURL>www.siemens.com</WriterVendorURL>
      <WriterVersion>1400</WriterVersion>
      <WriterRelease>1400.100.101.16</WriterRelease>
      <LastWritingDateTime>2016-09-29T11:21:34.9551066Z</LastWritingDateTime>
    </WriterHeader>
  </AdditionalInformation>
  <AdditionalInformation AutomationMLVersion="2.0" />
  <AdditionalInformation DocumentVersions="Recommendations">
    <Document DocumentIdentifier="AR APC" Version="1.0.0" />
  </AdditionalInformation>
  ...
```

- 实例层级
  本部分包含所导出内部元素的层级序列。

```
...
  <InstanceHierarchy Name="APC Sample Instance Hierarchy">
  <InternalElement ID="d4dc896a-f4a5-41b6-9c48-8d3a0a5a4343" Name="CAx_asterisk_AML_03_V14">
    <Attribute Name="ProjectManufacturer" AttributeDataType="xs:string" />
    <Attribute Name="ProjectSign" AttributeDataType="xs:string" />
    <Attribute Name="ProjectRevision" AttributeDataType="xs:string" />
    <Attribute Name="ProjectInformation" AttributeDataType="xs:string" />
    <InternalElement ID="544f3a69-5f65-45ba-ac2f-1448db9493fd" Name="PN/IE_1">
      ...
    </InternalElement>

    <InternalElement ID="12116ac0-94b7-49d2-888d-7d39bbc0caf5" Name="S71500/ET200MP station_1">
      ...
    </InternalElement>
    <SupportedRoleClass RefRoleClassPath=
      "AutomationProjectConfigurationRoleClassLib/AutomationProject" />
    <InternalLink Name="Link To Port_1" RefPartnerSideA=
      "726148ce-de5b-4728-8886-4ba273435479:CommunicationPortInterface" RefPartnerSideB=
      "cb9d24f3-8200-4c89-9b40-24ae850e293e:CommunicationPortInterface" />
    <InternalLink Name="Link To Port_2" RefPartnerSideA=
      "cb9d24f3-8200-4c89-9b40-24ae850e293e:CommunicationPortInterface" RefPartnerSideB=
      "726148ce-de5b-4728-8886-4ba273435479:CommunicationPortInterface" />
    .....
    <InternalLink Name="Link To IoSystem_3" RefPartnerSideA=
      "d8f6e006-3778-4a05-aab1-df844fe822fe:LogicalEndPoint_Interface" RefPartnerSideB=
      "2344b7af-329c-4215-92d1-6143b4627b56:LogicalEndPoint_IoSystem" />
  </InternalElement>
 </InstanceHierarchy>
</CAEXFile>
```

内部元素

AML 文件实例层级中的所有对象均为 `InternalElements`。内部元素 `AutomationProject` 包含所有角色类别的所有内部元素。每个内部元素都支持一系列属性。

属性 `<TypeIdentifier>` 识别每个可通过 Openness 创建的硬件对象的对象类型。

说明

自动创建的对象

自动创建的对象只能由其它对象进行创建。这些对象没有属性或类型标识符。这些对象包含在所导出的文件中，但无法触发对特定自动创建对象的导出操作。

在内部元素的 AML 元素末尾，定义以下内容：

- 角色类别

  SupportedRoleClass 元素定义内部元素的对象类型。在用于将标准 AML 映射到 Openness 和 TIA Portal 的对象模型的角色类别库中定义对象类型。

```
...
<InternalElement ID="1d1a37ed-19d9-4a23-bc91-51f5a8e0244b" Name="Ungrouped devices">
  <InternalElement ID="ab193f5d-0375-4a6d-a576-a903e2b77cca" Name="ET 200SP station_1">
    ...
      <InternalElement ID="72d41729-90a7-4de3-9708-a8eeda6b1886" Name="IO device_1">
        ...
        <SupportedRoleClass RefRoleClassPath="
          AutomationProjectConfigurationRoleClassLib/DeviceItem" />
      </InternalElement>
      ...
      <SupportedRoleClass RefRoleClassPath=
          "AutomationProjectConfigurationRoleClassLib/Device" />
    </InternalElement>
    <SupportedRoleClass RefRoleClassPath=
      "AutomationProjectConfigurationRoleClassLib/DeviceUserFolder" />
  </InternalElement>
  ...
```

- 内部连接

  元素 InternalLink 定义连接的通信伙伴。

```
  ...
  <InstanceHierarchy Name="APC Sample Instance Hierarchy">
    <InternalElement ID="d4dc896a-f4a5-41b6-9c48-8d3a0a5a4343" Name="CAx_asterisk_AML_03_V14">
      <Attribute Name="ProjectManufacturer" AttributeDataType="xs:string" />
      <Attribute Name="ProjectSign" AttributeDataType="xs:string" />
      <Attribute Name="ProjectRevision" AttributeDataType="xs:string" />
      <Attribute Name="ProjectInformation" AttributeDataType="xs:string" />
        ...
      <SupportedRoleClass RefRoleClassPath=
        "AutomationProjectConfigurationRoleClassLib/AutomationProject" />
      <InternalLink Name="Link To Port_1" RefPartnerSideA=
        "726148ce-de5b-4728-8886-4ba273435479:CommunicationPortInterface" RefPartnerSideB=
        "cb9d24f3-8200-4c89-9b40-24ae850e293e:CommunicationPortInterface" />
      <InternalLink Name="Link To Port_2" RefPartnerSideA=
        "cb9d24f3-8200-4c89-9b40-24ae850e293e:CommunicationPortInterface" RefPartnerSideB=
        "726148ce-de5b-4728-8886-4ba273435479:CommunicationPortInterface" />
      <InternalLink Name="Link To Port_3" RefPartnerSideA=
        "65307e3e-95fd-41ac-9982-e5e4ffc2fb15:CommunicationPortInterface" RefPartnerSideB=
        "58b1a3f2-f94b-48d1-ab5e-fbc4857cdfbc:CommunicationPortInterface" />
      <InternalLink Name="Link To Port_4" RefPartnerSideA=
        "58b1a3f2-f94b-48d1-ab5e-fbc4857cdfbc:CommunicationPortInterface" RefPartnerSideB=
        "65307e3e-95fd-41ac-9982-e5e4ffc2fb15:CommunicationPortInterface" />
      ...
    </InternalElement>
  </InstanceHierarchy>
</CAEXFile>
```

属性

属性被分配至内部元素，如下所示：

```
...
<InternalElement ID="1d1a37ed-19d9-4a23-bc91-51f5a8e0244b" Name="Ungrouped devices">
  <InternalElement ID="ab193f5d-0375-4a6d-a576-a903e2b77cca" Name="ET 200SP station_1">
    <Attribute Name="TypeIdentifier" AttributeDataType="xs:string">
      <Value>System:Device.ET200SP</Value>
    </Attribute>
    <InternalElement ID="7636c362-a7af-47bb-8a18-e6428a6d61ff" Name="Rack_0">
      <Attribute Name="TypeName" AttributeDataType="xs:string">
        <Value>Rack</Value>
      </Attribute>
      <Attribute Name="PositionNumber" AttributeDataType="xs:int">
        <Value>0</Value>
      </Attribute>
      <Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
        <Value>False</Value>
      </Attribute>
      <Attribute Name="TypeIdentifier" AttributeDataType="xs:string">
        <Value>System:Rack.ET200SP</Value>
      </Attribute>
      ...
      <SupportedRoleClass RefRoleClassPath=
          "AutomationProjectConfigurationRoleClassLib/DeviceItem" />
    </InternalElement>
    <SupportedRoleClass RefRoleClassPath=
      "AutomationProjectConfigurationRoleClassLib/Device" />
  </InternalElement>
  <SupportedRoleClass RefRoleClassPath=
      "AutomationProjectConfigurationRoleClassLib/DeviceUserFolder" />
  <InternalLink Name="Link To Port_1" RefPartnerSideA=
      "5758e2ff-3974-41e9-8bcc-b61a23f1bb58:CommunicationPortInterface"
      RefPartnerSideB="46683602-5129-4504-a9d1-48e6421e2cf0:CommunicationPortInterface" />
  ...
</InternalElement>
```

属性的处理模式

每个属性都将单独定义属性的处理方式，如下所示：

- 忽略
  该属性在导入时会被忽略，且未处于导出文件中。
- 必需
  该属性必须位于导入文件中，且不能在导出文件中删除。
- 可选
  如果该属性在导入文件中丢失，则会指定属性的默认值。如果该属性对某个对象不适用（例如，并非所有模块均配有 FirmwareVersion），则该属性会在导出文件中丢失。

- 仅导出
  该属性值由 TIA Portal 在内部确定（例如，设备项的类型名称）。如果导入文件中存在该属性，则在导入期间会被 TIA Portal 忽略。
- 仅导入
  该属性会影响导入行为。如果该属性在导入文件中丢失，则相关行为将与属性的标准值相关。

参见

AML 类型标识符 (页 448)

8.5.5 AML 类型标识符

内部元素

`TypeIdentifier` 字符串包含多个部分：

- `<TypeIdentifierType>:<Identifier>`

支持以下 `TypeIdentifierType` 值：

- `OrderNumber` 用于指定物理模块
- `GSD` 用于指定基于 GSD/GSDML 的设备
- `System` 用于指定系统和通用设备

Type identifier type：OrderNumber

`OrderNumber` 为硬件目录中所有模块的通用类型标识符（GSD 除外）。AML 类型标识符并非始终与 Openness 类型标识符相同。AML 类型标识符中不含 `FirmwareVersion` 信息。有关固件版本的信息将在一个单独的 AML 属性“`FirmwareVersion`”中进行处理。

TypeIdentifierType 的格式如下所示：

- <OrderNumber>
  示例：OrderNumber:3RK1 200-0CE00-0AA2

说明

订货号中的通配符

硬件目录中存在几个订货号包含“通配符”字符的模块，这类字符用于表示实际硬件的特定集群，例如不同长度的 S7-300 机架。

对于这种情况，可使用特定 OrderNumber 和“通配符”OrderNumber 创建硬件对象的实例。不过，不能将通配符通用于任意位置。示例：可通过以下方式创建 S7-300 机架：

OrderNumber:6ES7 390-1***0-0AA0

或者

OrderNumber:6ES7 390-1AE80-0AA0

请注意，不能对实例使用以下结构：

OrderNumber:6ES7 390-1AE80-0A*0

读取类型标识符时返回的值始终是硬件目录中的订货号。

示例：读取 OrderNumber:6ES7 390-1AE80-0AA0 将返回 OrderNumber:6ES7 390-1***0-0AA0

Type identifier type：GSD

基于 GSD 和 GSDML 的设备类型标识符为 TypeIdentifier = GSD:<Identifier>

该标识符由以下元素组成

- GsdName：GSD 或 GSDML 的名称（大写字母）
- GsdType：可以为：
  - D：设备
  - R：机架
  - DAP：前端模块
  - M：模块
  - SM：子模块
- GsdId：GSD/GSDML 的 ID 编号

CAx 导入/导出支持以下格式的类型标识符：

- GSD.<GsdName>/<GsdType>
  示例：
  GSD：SIEM8139.GSD/DAP
  GSD：GSDML-V2.31-SIEMENS-SINAMICS_DCP-20140313.XML/D
- <GsdName>/<GsdType>/<GsdId>
  示例：
  GSD：SIEM8139.GSD/M/4
  GSD：GSDML-V2.31-SIEMENS-SINAMICS_G110M-20140704.XML/M/IDM_DRIVE_47

Type identifier type：System

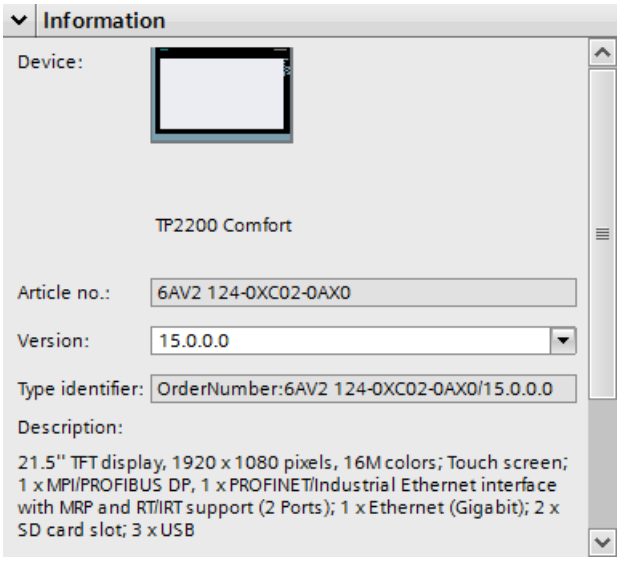
System. 为不能通过其它任何标识符确定的对象的标识符。此 TypeIdentifierType 的格式如下：

- <SystemTypeIdentifier>
  示例：
  System:Device.S7300
  System:Subnet.Ethernet
- <SystemTypeIdentifier>/<AdditionalTypeIdentifier>
  SystemTypeIdentifier 不唯一时，需要 AdditionalTypeIdentifier。
  对于某些对象类型，SystemTypeIdentifier 具有前缀：
  Subnet.
  Device.
  Rack.
  示例：System:Rack.S71600/Large
  通过 OrderNumber 标识符识别包含订货号的机架。

在 TIA Portal 中显示类型标识符

如果要确定一个类型标识符，则可在 TIA Portal 中执行以下查询操作：

1. 在“选项 > 设置 > 硬件配置 > 显示类型标识符”(Options > Settings > Hardware configuration > Display of the type identifier) 中，启用设置“显示设备和模块的类型标识符”(Enable display of the type identifier for devices and modules)。
2. 打开“设备与网络”(Devices & Networks) 编辑器。
3. 在产品目录中选择一个设备。
   “类型标识符”将显示在“信息”(Information) 窗口中。



8.5.6 导出 CAx 数据

要求

- Openness 应用程序已连接 TIA Portal。
  参见“连接到 TIA Portal (页 69)”
- 项目已打开。
  参见“打开项目 (页 96)”

应用

在 TIA Portal 中，可将设备和网络编辑器中的配置导出到 AML 文件中。此功能基于 Openness，用于从项目或设备层级导出硬件数据。

在 Openness 中，可通过以下几种方式导出 CAx 数据：

- 导出函数
  通过 CaxProvider 服务访问导出函数。要获取 CaxProvider 服务，可在 Project 对象处调用 GetService 类函数。
- 命令行接口
  通过传递特定的命令行参数执行“C:\Program Files\Siemens\Automation\Portal V..\Bin\”中的“Siemens.Automation.Cax.AmiHost.exe”。

CAx 的导出和导入限制条件

CAx 不支持导出和导入以下数据：

- 端口间的连接
- 与扩展机架的连接以及扩展机架间的连接
- 多 CPU
- H 设备
- HMI 设备
- 模拟量通道的输出模式和范围
- 封装地址

CAx 不支持导出和导入以下设备和驱动装置：

- 6GK5 124-0BA00-2AC2/V1.0
- 6GK5 116-0BA00-2AC2/V1.0
- 6GK5 108-0BA00-2AC2/V1.0
- 6GK5 106-2BD00-2AC2/V1.0
- 6GK5 106-2BB00-2AC2/V1.0
- 6GK5 124-0BA00-2AA3/V1.0
- 6GK5 116-0BA00-2AA3/V1.0
- 6GK5 112-2BB00-2AA3/V1.0
- 6GK5 108-0PA00-2AA3/V1.0
- 6GK5 108-0BA00-2AA3/V1.0
- 6GK5 106-1BB00-2AA3/V1.0
- 6GK5 104-2BB00-2AA3/V1.0

- 6GK5 008-0GA00-1AB2/V1.0
- 6GK5 008-0BA00-1AB2/V1.0
- 6GK5 005-0GA00-1AB2/V1.0
- 6GK5 005-0BA00-1CA3/V1.0
- 6GK5 005-0BA00-1AB2/V1.0
- 6GK5 005-0BA00-1AA3/V1.0
- 6GK5 004-2BD00-1AB2/V1.0
- 6GK5 004-1GM00-1AB2/V1.0
- 6GK5 004-1GL00-1AB2/V1.0
- 6GK5 004-1BF00-1AB2/V1.0
- 6GK5 004-1BD00-1AB2/V1.0
- 6GK5 101-1BY00-2AA3/V1.6
- 6GK5 101-1BX00-2AA3/V1.6
- 6GK5 101-1BH00-2AA3/V1.6
- 6GK5 101-1BC00-2AA3/V1.6
- 6GK5 101-1BB00-2AA3/V1.6
- 6GK7 377-1AA00-0AA0
- 6GK7 277-1AA10-0AA0
- 6GK7 277-1AA00-0AA0
- 6GK5 602-0BA10-2AA3
- 6GK5 612-0BA10-2AA3
- 6GK5 623-0BA10-2AA3
- 6GK5 627-2BA10-2AA3
- 6AG4141-xxxxx-xxxx，版本高于 V1.0
- 6ES7 431-7QH00-0AB0
- PC 机，配有 1、2、3 或 4 个端口
- 6GK5 991-2AB00-8AA0
- 6GK5 991-2AC00-8AA0
- 6GK5 991-2AD00-8AA0

- 6GK5 991-2AE00-8AA0
- 6GK5 991-2AF00-8AA0
- 6GK5 992-2AL00-8AA0
- 6GK5 992-2AL00-8FA0
- 6GK5 992-2AM00-8AA0
- 6GK5 992-2AN00-8AA0
- 6GK5 992-2AP00-8AA0
- 6GK5 992-2AQ00-8AA0
- 6GK5 992-2GA00-8AA0
- 6GK5 992-2GA00-8AA0
- 6GK5 992-2GA00-8FA0
- 6GK5 992-2HA00-0AA0
- 6GK5 992-2SA00-8AA0
- 6GK5 992-2VA00-8AA0
- 6GK5 992-2AS00-8AA0
- 6GK5 602-0BA00-2AA3
- 6GK5 612-0BA00-2AA3
- 6GK5 613-0BA00-2AA3

程序代码：访问 CaxProvider 服务

修改以下程序代码以访问 CaxProvider 服务：

```
Project project = tiaPortal.Projects.Open(...);
CaxProvider caxProvider = project.GetService<CaxProvider>();

if(caxProvider != null)

{

      // Perform CAx export and import operation

}
```

在项目层级导出 CAx

要在项目层级导出 CAx 数据，则在调用 Export 类函数时需使用以下参数：

名称	示例	说明
ProjectToExport	tiaPortal.Projects[0]	要导出的项目对象
ExportFilePath	new FileInfo(@"D:\Temp\ProjectExport.aml")	AML 文件的完整导出文件路径
LogFilePath	new FileInfo(@"D:\Temp\ProjectExport_Log.log")	日志文件的完整文件路径

修改以下程序代码以在项目级别导出 CAx 数据：

```
caxProvider.Export(project, new FileInfo(@"D:\Temp\ProjectExport.aml"),
new FileInfo(@"D:\Temp\ProjectExport_Log.log"));
```

在设备层级导出 CAx

要在设备层级导出 CAx 数据，则在调用 Export 类函数时需使用以下参数：

名称	示例	说明
DeviceToExport	project.Devices[0]	要导出的设备对象
ExportFilePath	new FileInfo(@"D:\Temp\ProjectExport.aml")	AML 文件的完整导出文件路径
LogFilePath	new FileInfo(@"D:\Temp\ProjectExport_Log.log")	日志文件的完整文件路径

修改以下程序代码以在项目级别导出 CAx 数据：

```
caxProvider.Export(device, new FileInfo(@"D:\Temp\DeviceExport.aml"), new
FileInfo(@"D:\Temp\DeviceExport_Log.log"));
```

通过命令行导出 CAx

要通过命令行导出 CAx 数据，请将“Siemens.Automation.Cax.AmiHost.exe”与以下参数一起使用：

参数	示例	说明
-p	-p "D:\Temp\MyProject.ap14"	指定现有 TIA Portal 项目的完整路径名称。
-d	-d "S7300/ET200M station_1"	可选参数。如果未指定设备，则会以项目级导出。 指定特定 TIA 项目中需要导出的设备或站的名称。
-m	-m "AML"	指定导出/导入模式（导出/导入的格式）： “AML”表示以 AML 格式导出
-e	-e "D:\Import" -e "D:\Import\CAx_Export.aml"	指定要导出的 AML 文件的完整路径。如果仅指定一个路径，则会将项目名用作导出的文件名。

修改以下程序代码以通过命令行在项目级别导出 CAx 数据：

```
Siemens.Automation.Cax.AmiHost.exe -p "D:\Temp\MyProject.ap14" -m "AML" -e
"D:\Import\CAx_Export.aml"
```

修改以下程序代码以通过命令行在设备级别导出 CAx 数据：

```
Siemens.Automation.Cax.AmiHost.exe -p "D:\Temp\MyProject.ap14" -d "S7300/
ET200M station_1" -m "AML" -e "D:\Import\CAx_Export.aml"
```

8.5.7 导入 CAx 数据

要求

- Openness 应用程序连接到 TIA Portal。
  请参见连接到 TIA Portal (页 69)
- 项目已经打开。
  请参见打开项目 (页 96)

应用

在 TIA Portal 中，您可以在设备和网络编辑器中从 AML 文件导入配置。此功能使您能够从项目或设备级别导入硬件数据。

Openness 提供了以下导出 CAx 数据的方法：

- 导入函数
  通过 CaxProvider 服务访问导入函数。要获取 CaxProvider 服务，对 Project 对象调用 GetService 方法。
- 命令行
  通过传递特定的命令行参数执行“C:\Program Files\Siemens\Automation\Portal V..\Bin\”中的“Siemens.Automation.Cax.AmiHost.exe”。

程序代码：访问 CaxProvider 服务

修改以下程序代码：

```
//Access the CaxProvider service
Project project = tiaPortal.Projects.Open(...);
CaxProvider caxProvider = project.GetService<CaxProvider>();

if(caxProvider != null)

{
      // Perform Cax export and import operation
}
```

CAx 导入

要将 CAx 数据导入 TIA Portal 项目中，请将 `Import` 方法与以下参数一起使用：

名称	示例	说明
ImportFilePath	new FileInfo(@"D:\Temp\ProjectExport.aml")	AML 文件的完整导入文件路径
LogFilePath	new FileInfo(@"D:\Temp\ProjectExport_Log.log")	日志文件的完整文件路径
ImportOptions	CaxImportOptions.MoveToParkingLot CaxImportOptions.RetainTiaDevice CaxImportOptions.OverwriteTiaDevice	导入到已经存在的非空项目中时的冲突解决策略。

修改以下程序代码导入 CAx 数据：

```
caxProvider.Import(new FileInfo(@"D:\Temp\ProjectImport.aml"), new
FileInfo(@"D:\Temp\ProjectImport_Log.log"), CaxImportOptions.ParkingLot);
```

导入选项

提供以下 `CaxImportOptions`：

导入选项	说明
MoveToParkingLot	在项目中保留名称冲突的设备，并将这些设备从 CAx 导入到停驻区文件夹
RetainTiaDevice	在项目中保留名称冲突的设备，并不将这些设备从 CAx 导入
OverwriteTiaDevice	用 CAx 中的设备覆盖项目中名称冲突的设备

通过命令行导入 CAx

要通过命令行导入 CAx 数据，请将“Siemens.Automation.Cax.AmiHost.exe”与以下参数一起使用：

参数	示例	说明
-p	-p "D:\Temp\MyProject.ap14"	指定现有 TIA Portal 项目的完整路径名称。
-m	-m "AML"	指定导出/导入模式（导出/导入的格式）： “AML”表示以 AML 格式导出
-i	-i "D:\Import\CAx_Export.aml"	指定要导入的 AML 文件的完整路径。

修改以下程序代码以通过命令行导入 CAx 数据：

```
Siemens.Automation.Cax.AmiHost.exe -p "D:\Temp\MyProject.ap14" -m "AML" -i
"D:\Import\CAx_Export.aml
```

8.5.8 设备和模块的往返行程交换

要求

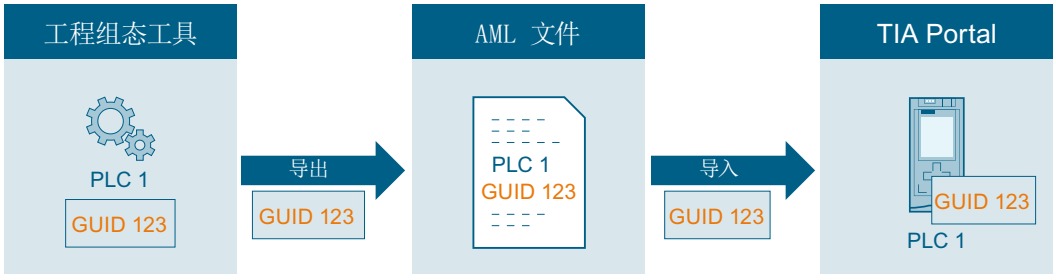
- Openness 应用程序已连接 TIA Portal。
  参见“连接到 TIA Portal (页 69)”
- 项目已打开。
  参见“打开项目 (页 96)”
- PLC 处于离线状态。

应用

可在 TIA Portal 和其它工程组态工具（例如，电气设计工具 EPLAN 或 TIA Selection Tool）之间交换组态数据。要识别导入的设备和导出的设备，请使用全局唯一标识符 AML GUID。

在数据双向交换过程中，设备和非内置设备项（CPU 或模块）等物理设备的 AML GUID 保持不变，但变量、通道之类的虚拟设备除外。

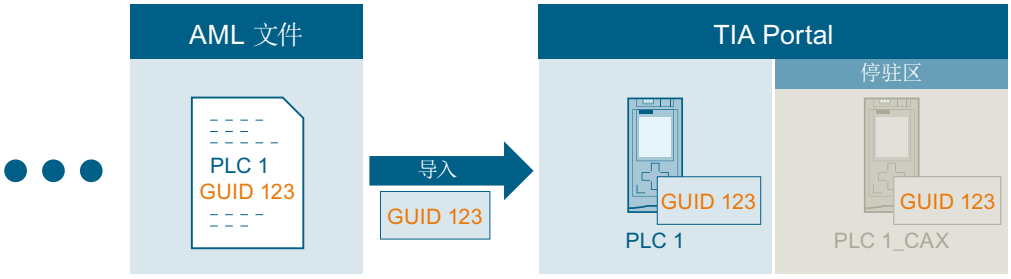
首次从 TIA Portal 导出时，系统将为设备或非内置设备项随机生成一个 AML GUID，之后该 GUID 将保持不变。



如果将设备从工程组态工具导出至空 TIA Portal 项目，则会将 AML GUID 添加到硬件对象的注释中。在 TIA Portal 中，通过菜单“工具 > 设置 > CAx > 导入设置”(Tools > Settings > CAx > Import settings) 启用相应设置时，该 AML GUID 将以当前的编辑语言添加。往返行程过程仅支持通过一种编辑语言存储 AML GUID。导入或导出数据时，通常使用开始数据通信时启动的编辑语言。

对于所有后续导入或导出，AML GUID 对该硬件对象始终保持不变。将恢复对硬件对象的更改。

在 TIA Portal 中，项目对象名称必须唯一。将设备或设备项导入 TIA portal 项目中时，如果项目中已存在相同名称的对象，则将导致命名冲突。在导入过程中，可将发生命名冲突的对象移动到用户自定义的停驻区。所导入对象的名称将使用“_CAX”进行扩展。



说明

复制导入的设备

复制一个带有 AML GUID 的设备或设备项时，则需删除所复制对象注释中的 AML GUID。否则，项目中将存在 AML GUID 相同的设备或设备项，从而导致 AML 文件无效。

导入设置

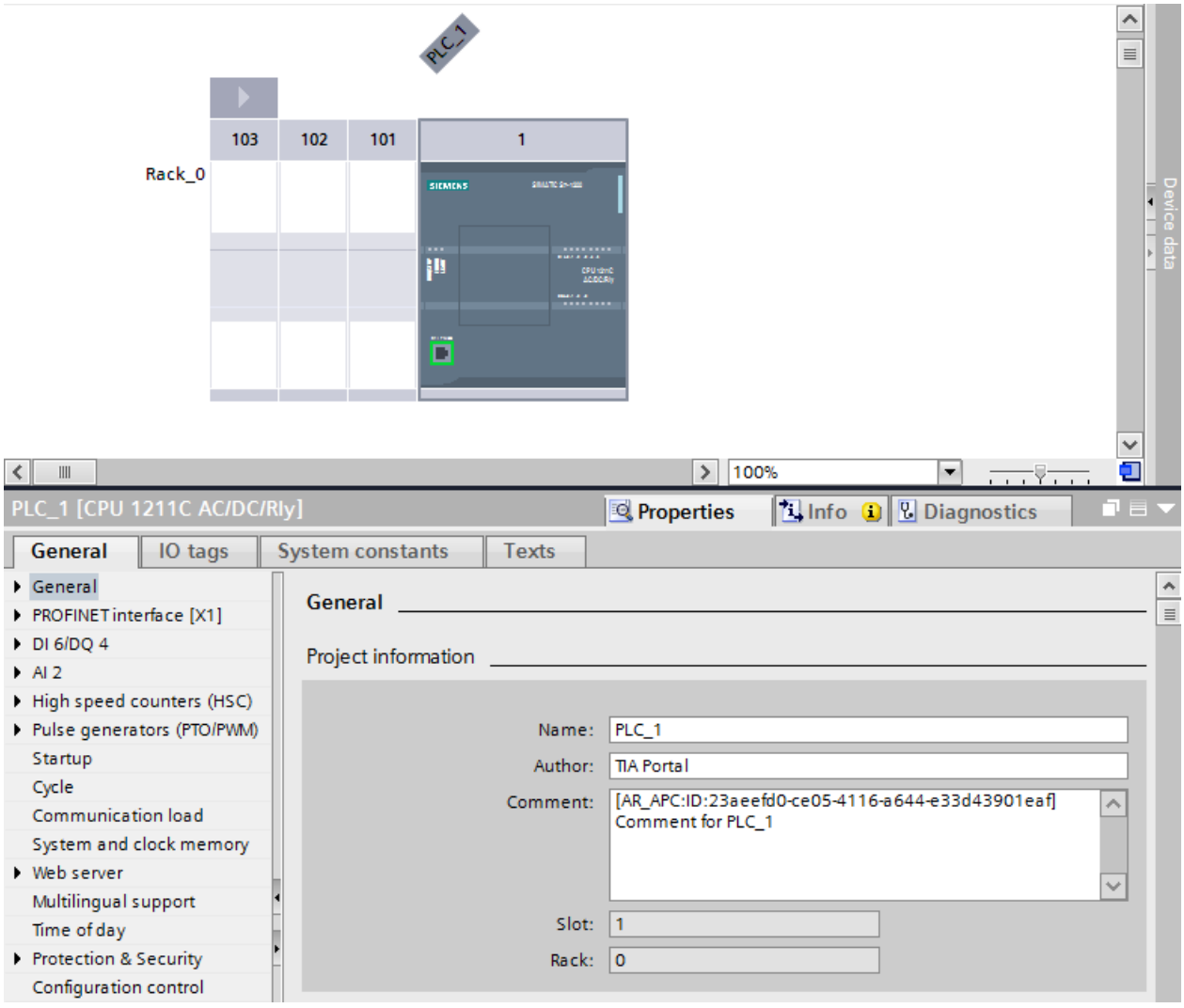
1. 在“选项 > 设置 > CAx > 冲突解决方案设置”(Options > Settings > CAx > Settings for conflict resolution) 下定义停驻区文件夹名称。
   停驻区文件夹可用于存储造成命名冲突的对象。
2. 激活“选项 > 设置 > CAx > 导入设置 > 导入期间保存 GIUD”(Options > Settings > CAx > Import settings > Save GUIDs during import)。

说明

有效 AML GUID

如果在导入前编辑一个 AML GUID，则该 AML GUID 将无效，并且导入会被中止。

导入到 TIA Portal 中之后，AML GUID 会被添加至现有用户注释中，如下所示：



说明

超出注释长度

如果向注释附加 AML GUID 导致超出 500 个字符最大限值，则用户注释值将被减至 500 个字符。将记录相应的信息。

AML 结构

生成的 ID 被导出至 AML 文件，如以下代码片段所示：

```
<InternalElement ID="23aeefd0-ce05-4116-a644-e33d43901eaf"
Name="PLC_1"
```

8.5.9 导出/导入拓扑结构

要求

- Openness 应用程序已连接 TIA Portal。
  参见“连接到 TIA Portal (页 69)”
- 项目已打开。
  参见“打开项目 (页 96)”
- PLC 处于离线状态。

应用

在 TIA Portal 中，可将设备及其拓扑结构信息一同导出到一个 AML 文件。导入到一个空的 TIA Portal 项目中时，所导入的设备项将保留其拓扑结构信息。

<InteralLink> 元素用于指示设备项目间端口互连的详细信息。该信息位于所连接设备的共同父设备下，且变量名称唯一。

"InternalLink" 元素的属性

下表列出了 CAx 导入和导出文件中相关对象的属性：

属性	处理方式	注释
Name	必需	变量名称的格式为 "Link to Port_n"（其中，n 为从 1 开始计数的具体互连端口数量）。
RefPartnerSideA	必需	指示连接的端口.，格式为 UniqueIDOfPort:CommunicationPortInterface
RefPartnerSideB	必需	指示连接的端口.，格式为 UniqueIDOfPort:CommunicationPortInterface

示例：拓扑视图

AML 结构

下图显示了所导出 AML 文件中的部分元素结构。其中，包含两个唯一的 PLC 端口 ID。

```
...
 <InternalElement ID="e1966b52-b8b3-47b4-8866-a754ebb77648" Name="Port_1">
   <Attribute Name="Label" AttributeDataType="xs:string">
```

```
...
 <InternalElement ID="75f31daf-575f-48a2-ab35-8f07a376eb1b" Name="Port_1">
   <Attribute Name="Label" AttributeDataType="xs:string">
```

<InteralLink> 元素包含三个必需属性。

```
<InternalLink Name="Link to Port_1"
  RefPartnerSideA="e1966b52-b8b3-47b4-8866-a754ebb77648:CommunicationPortInterface"
  RefPartnerSideB="75f31daf-575f-48a2-ab35-8f07a376eb1b:CommunicationPortInterface" />
```

8.5.10 设备对象的导出

要求

- Openness 应用程序已连接 TIA Portal。
  参见“连接到 TIA Portal (页 69)”
- 项目已打开。
  参见“打开项目 (页 96)”
- PLC 处于离线状态。

应用

`Device` 对象是一个容器对象，用于进行集中式或分布式组态。该对象为 `DeviceItem` 对象的父对象，在 AML 文件结构中位于 TIA Portal 项目实例层级内部元素的顶部。

CAx 数据导出支持以下由 AML 类型标识符指定的设备类型：

- 物理模块
- 基于 GSD/GSDML 的设备
- 系统

设备可组到一个 `DeviceUserFolder` 对象中。

说明

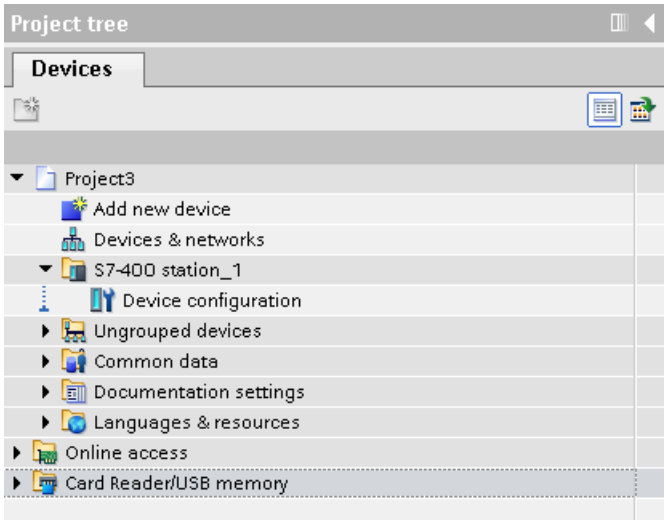
导出一个设备还会导出项目中的所有子网。

属性

下表列出了 CAx 导入和导出文件中相关对象的属性：

属性	处理方式	注释
Name	必需	
TypeIdentifier	必需	
Comment	可选	默认值：“”

示例：导出组态



导出文件的 AML 结构

以下结构示例为不带机架和模块的单个设备“S7-400 station_1”的导出内容：

```
...
<InstanceHierarchy Name="APC Sample Instance Hierarchy">
  <InternalElement ID="288b7850-688e-43b3-941e-d615ba900a02" Name="Project3">
    <Attribute Name="ProjectManufacturer" AttributeDataType="xs:string" />
    <Attribute Name="ProjectSign" AttributeDataType="xs:string" />
    <Attribute Name="ProjectRevision" AttributeDataType="xs:string" />
    <Attribute Name="ProjectInformation" AttributeDataType="xs:string" />
    <InternalElement ID="57611cfd-6da4-444e-ac78-5fbcea20a4e1" Name="S7-400 station_1">
      <Attribute Name="TypeIdentifier" AttributeDataType="xs:string">
        <Value>System:Device.S7400</Value>
      </Attribute>
      <Attribute Name="Comment" AttributeDataType="xs:string">
        <Value>S7400 station</Value>
      </Attribute>
      <SupportedRoleClass RefRoleClassPath=
          "AutomationProjectConfigurationRoleClassLib/Device" />
    </InternalElement>
    <SupportedRoleClass RefRoleClassPath=
      "AutomationProjectConfigurationRoleClassLib/AutomationProject" />
  </InternalElement>
</InstanceHierarchy>
</CAEXFile>
```

参见

用于导入/导出的 CAx 数据的结构 (页 443)

AML 类型标识符 (页 448)

8.5.11 设备对象的导入

要求

- Openness 应用程序已连接 TIA Portal。
  参见“连接到 TIA Portal (页 69)”
- 项目已打开。
  参见“打开项目 (页 96)”
- PLC 处于离线状态。

应用

Device 对象是一个容器对象，用于进行集中式或分布式组态。该对象为 DeviceItem 对象的父对象，在 AML 文件结构中位于 TIA Portal 项目实例层级内部元素的顶部。

CAx 数据导入支持以下由 AML 类型标识符指定的设备类型：

- 物理模块
- 基于 GSD/GSDML 的设备
- 系统
- 通用设备

如果 TIA Portal 中存在一个 DeviceUserFolder 对象，则该设备将归组到指定文件夹中。

如果仅能确定前端模块或 PLC 的标识 (TypeIdentifier)，且这些设备不带机架和设备，则可导入一个通用机架。

示例：TypeIdentifier = System:<Prefix>.Generic

要替换通用设备，以下元素必须处于机架中（如 AML 文件中所述）：

- 中央设备：PLC
- 分布式设备：前端模块

如果设备为通用设备，则属性 BuiltIn 将定义机架或模块的类型：

- 物理：BuiltIn = True
- 通用：BuiltIn = False

示例：导入通用设备

以下结构示例介绍了无机架和模块的通用 "S7-400 station" 设备的导入过程。

```
...
  <InstanceHierarchy Name="APC Sample Instance Hierarchy">
  <InternalElement ID="d4dc896a-f4a5-41b6-9c48-8d3a0a5a4343" Name="MyProject">
    <Attribute Name="ProjectManufacturer" AttributeDataType="xs:string" />
    <Attribute Name="ProjectSign" AttributeDataType="xs:string" />
    <Attribute Name="ProjectRevision" AttributeDataType="xs:string" />
    <Attribute Name="ProjectInformation" AttributeDataType="xs:string" />
    <InternalElement ID="3e6277d1-1b12-4c18-b00e-25e3eac3ac35" Name="S7400 station_1">
      <Attribute Name="TypeIdentifier" AttributeDataType="xs:string">
        <Value>System:Device.Generic</Value>
      </Attribute>
      <Attribute Name="Comment" AttributeDataType="xs:string">
        <Value>S7400 station_1</Value>
      </Attribute>
      <SupportedRoleClass RefRoleClassPath=
          "AutomationProjectConfigurationRoleClassLib/Device" />
    </InternalElement>
     ...
    <SupportedRoleClass RefRoleClassPath=
      "AutomationProjectConfigurationRoleClassLib/AutomationProject" />
  </InternalElement>
 </InstanceHierarchy>
</CAEXFile>
```

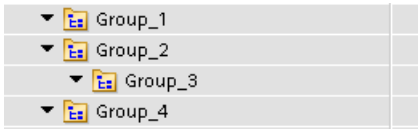
示例：导入设备用户文件夹层次结构

以下结构示例介绍了文件夹层次结构的导入。

```
...
<InternalElement ID="4fe37f4f-2661-4492-95f0-3d8a8160c851" Name="Project1">
      <Attribute Name="ProjectManufacturer" AttributeDataType="xs:string" />
      <Attribute Name="ProjectSign" AttributeDataType="xs:string" />
      <Attribute Name="ProjectRevision" AttributeDataType="xs:string" />
      <Attribute Name="ProjectInformation" AttributeDataType="xs:string" />
      <InternalElement ID="1ee1615f-9c67-432d-a7cc-b795babf67b6" Name="Group_1">
        <SupportedRoleClass RefRoleClassPath=
        "AutomationProjectConfigurationRoleClassLib/DeviceUserFolder" />
      </InternalElement>
      <InternalElement ID="ce14c85a-28de-41aa-ad08-2eb7d0fb755f" Name="Group_2">
        <InternalElement ID="852347e8-3c48-4eb9-8bd8-349d0c7caf34" Name="Group_3">
          <SupportedRoleClass RefRoleClassPath=
          "AutomationProjectConfigurationRoleClassLib/DeviceUserFolder" />
        </InternalElement>
        <SupportedRoleClass RefRoleClassPath=
        "AutomationProjectConfigurationRoleClassLib/DeviceUserFolder" />
      </InternalElement>
      <InternalElement ID="97cf7924-1756-4e32-8716-ac18990e4762" Name="Group_4">
        <SupportedRoleClass RefRoleClassPath=
        "AutomationProjectConfigurationRoleClassLib/DeviceUserFolder" />
      </InternalElement>
      <SupportedRoleClass RefRoleClassPath=
      "AutomationProjectConfigurationRoleClassLib/AutomationProject" />
    </InternalElement>
...
```

导入的用户文件夹层次结构

将以下层次结构导入到项目导航中：



参见

用于导入/导出的 CAx 数据的结构 (页 443)

AML 类型标识符 (页 448)

8.5.12 导出/导入带有设定地址的设备

要求

- Openness 应用程序已连接 TIA Portal。
  参见“连接到 TIA Portal (页 69)”
- 项目已打开。
  参见“打开项目 (页 96)”
- PLC 处于离线状态。

应用

在 TIA Portal 中，可将 IO 设备项的地址对象导出到 AML 文件中。导入到一个空的 TIA Portal 项目中时，所导入的设备项将保留相应 IO 设备项中的地址对象。

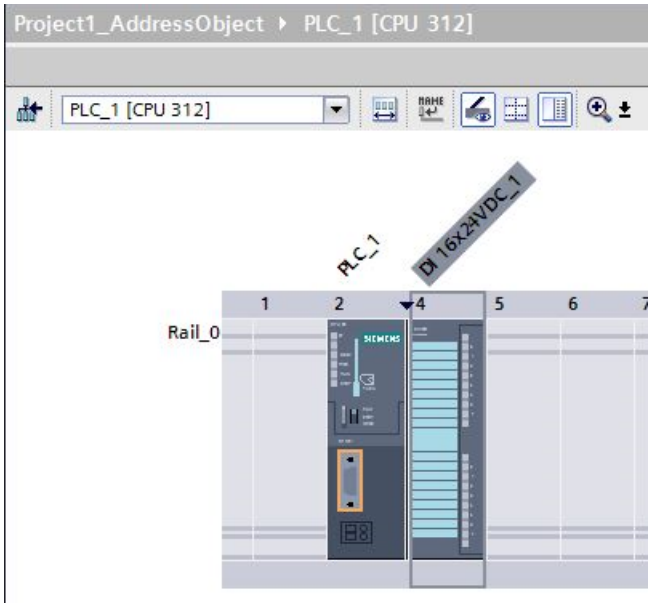
AML 文件中的 Address 属性包含有必需的设置 RefSemantic，用于指定 OrderedListType 的值。

"Address" 元素的属性

下表列出了 CAx 导入和导出文件中相关对象的属性：

属性	处理方式	注释
IoType	必需	输入或输出
Length	可选	通道宽度
StartAddress	必需	IO 设备的起始地址。

示例：带有地址对象的 IO 设备项



AML 结构

下图显示了所导出 AML 文件中的部分元素结构。其中，包含有 Address 元素及其属性。

```
<Attribute Name="Address">
    <RefSemantic CorrespondingAttributePath="OrderedListType" />
    <Attribute Name="1">
      <Attribute Name="StartAddress" AttributeDataType="xs:int">
        <Value>0</Value>
      </Attribute>
      <Attribute Name="Length" AttributeDataType="xs:int">
        <Value>16</Value>
      </Attribute>
      <Attribute Name="IoType" AttributeDataType="xs:string">
        <Value>Input</Value>
      </Attribute>
    </Attribute>
  </Attribute>
```

修剪 XML

裁剪操作是指通过删除 XML 中不需要的部分，对文件内容进行优化。在修剪后的 xml 中，不包含自动创建的子模块信息，且其相应的地址对象位于直接父模块中。

下图显示了修剪前所导出 AML 文件中的部分元素结构。

```
<InternalElement ID="5511a117-42c6-44b7-be5d-0f33cd46e932" Name="AS-i Master_1">
  <Attribute Name="PositionNumber" AttributeDataType="xs:int">
    <Value>4</Value>
  </Attribute>
  <Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
    <Value>True</Value>
  </Attribute>
  <Attribute Name="Address">
    <RefSemantic CorrespondingAttributePath="OrderedListType" />
    <Attribute Name="1">
      <Attribute Name="StartAddress" AttributeDataType="xs:int">
        <Value>20</Value>
      </Attribute>
      <Attribute Name="Length" AttributeDataType="xs:int">
        <Value>256</Value>
      </Attribute>
      <Attribute Name="IoType" AttributeDataType="xs:string">
        <Value>Input</Value>
      </Attribute>
    </Attribute>
```

在修剪后的 AML 文件中，将删除子模块信息 like <InternalElement> 元素，但保留其相应的地址对象。

```
<Attribute Name="Address">
  <RefSemantic CorrespondingAttributePath="OrderedListType" />
  <Attribute Name="1">
    <Attribute Name="StartAddress" AttributeDataType="xs:int">
      <Value>20</Value>
    </Attribute>
    <Attribute Name="Length" AttributeDataType="xs:int">
      <Value>256</Value>
    </Attribute>
    <Attribute Name="IoType" AttributeDataType="xs:string">
      <Value>Input</Value>
    </Attribute>
  </Attribute>
</Attribute>
```

参见

Pruned AML (页 439)

8.5.13 导出/导入带有通道的设备

要求

- Openness 应用程序已连接 TIA Portal。
  参见“连接到 TIA Portal (页 69)”
- 项目已打开。
  参见“打开项目 (页 96)”
- PLC 处于离线状态。

应用

在 TIA Portal 中，可将 IO 设备项的通道对象导出到 AML 文件中。导入到一个空的 TIA Portal 项目中时，所导入的设备项将保留相应 IO 设备项中的通道对象。

节点和子网内元素中的 **<ExternalInterface>** 元素，用于指示节点和子网已连接。

"ExternalInterface" 元素的属性

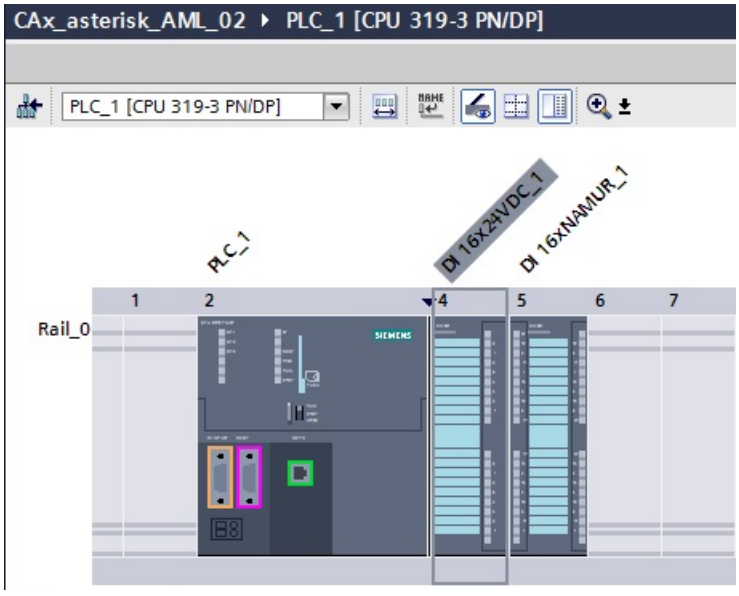
下表列出了 CAx 导入和导出文件中相关对象的属性：

属性	处理方式	注释
IoType	必需	输入或输出
Length	可选	通道宽度（数字量信号为 1，模拟量信号为 16）
Number	必需	通道编号，从 0 开始
Type	必需	模拟量或数字量

通道编号

数字量输入、数字量输出、模拟量输入、模拟量输出以及工艺通道将按照 DI_0, DO_0, AI_0, AO_0,TO_0 规则依次分别编号。首先对设备项通道中的通道进行编号，之后再依次对子设备项中的通道进行编号（深度优先）。每一个附加设备项都有一个从 0 开始的自己通道编号。

示例：带有通道的设备



AML 结构

下图显示了所导出 AML 文件中的部分元素结构。

```
<ExternalInterface ID="31ca16d3-6322-43b6-95bc-e2d7d7bfc7b7" Name="Channel_DI_0"
                RefBaseClassPath="AutomationProjectConfigurationInterfaceClassLib/Channel">
          <Attribute Name="Type" AttributeDataType="xs:string">
            <Value>Digital</Value>
          </Attribute>
          <Attribute Name="IoType" AttributeDataType="xs:string">
            <Value>Input</Value>
          </Attribute>
          <Attribute Name="Number" AttributeDataType="xs:int">
            <Value>0</Value>
          </Attribute>
          <Attribute Name="Length" AttributeDataType="xs:int">
            <Value>1</Value>
          </Attribute>
        </ExternalInterface>
```

8.5.14 设备项对象的导出

要求

- Openness 应用程序已连接 TIA Portal。
  参见“连接到 TIA Portal (页 69)”
- 项目已打开。
  参见“打开项目 (页 96)”
- PLC 处于离线状态。

应用

DeviceItem 对象为 Device 对象的嵌套子项。DeviceItem 类型的对象可以是一个机架或插入的模块。

- 设备第一个子项的类型为“机架”。机架的 PositionNumber 从 0 开始。如果存在多个机架，则会对其进行连续编号（1、2、3...）。
  AML 文件中一个层级内的顺序无限制。
- “机架”类型的所有其它子项均为模块。

CAx 数据导出支持以下由 AML 类型标识符指定的设备项类型：

- 物理模块
- GSD/GSDML 模块

属性

下表列出了 CAx 导入和导出文件中相关对象的属性：

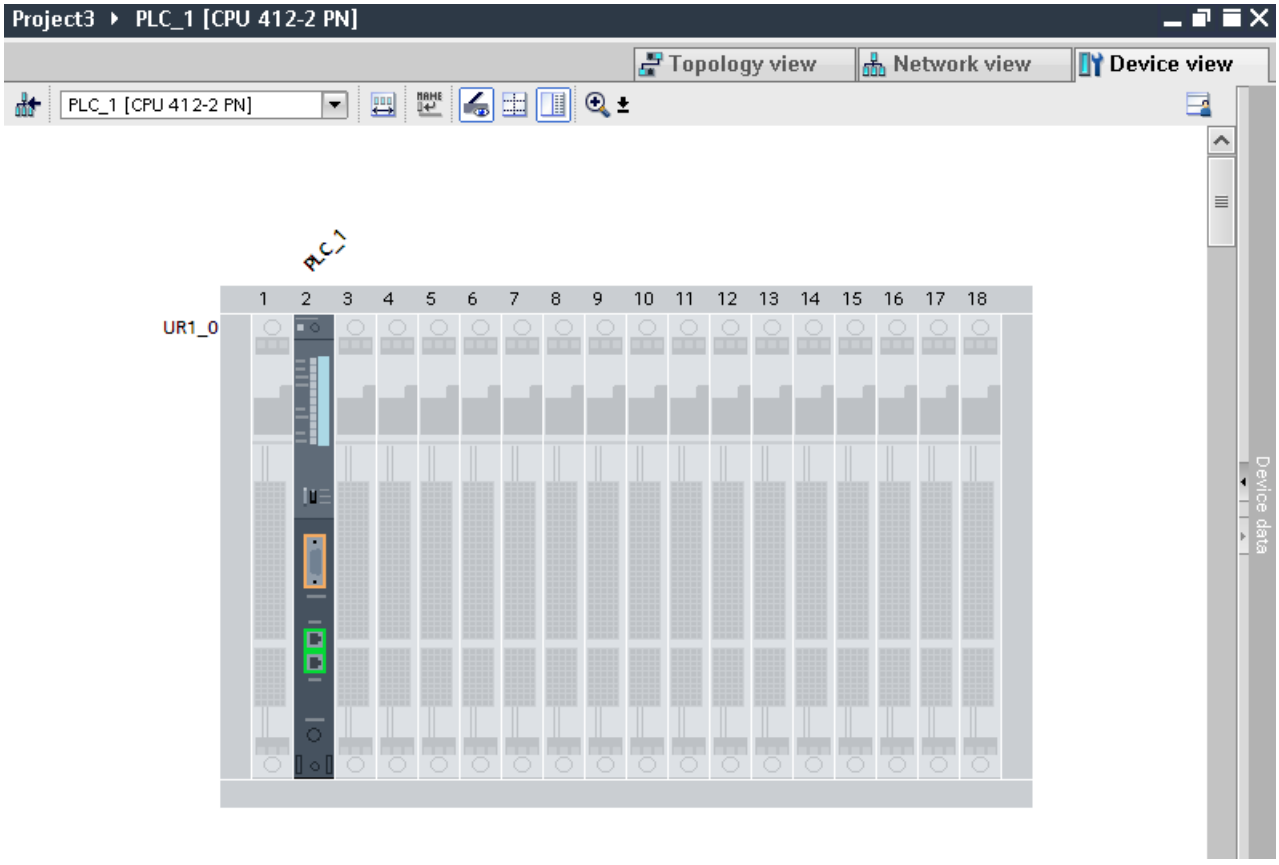
属性	处理方式	注释
Name	必需 “BuiltIn”= TRUE 时仅导出	
TypeName	“BuiltIn”= FALSE 时仅导出	
DeviceItemType	仅导出	仅限 PLC（中央设备）和 HeadModule（分散设备）设备项
PositionNumber	必需	
BuiltIn	可选	默认：FALSE

属性	处理方式	注释
TypeIdentifier	“BuiltIn”= FALSE 时为必须项 “BuiltIn”= TRUE 时被忽略	
FirmwareVersion	可选项， 对象支持固件版本时为必须项	
PlantDesignation IEC	可选	默认值：“”
LocationIdentifier IEC	可选	默认值：“”
Comment	“BuiltIn”= FALSE 时为可选项	默认值：“”
Address	可选	“地址”(Address) 包含嵌套属性

下表显示了对象“DeviceItem”的“Address”属性的嵌套属性：

“Address”的属性	处理方式	注释
StartAddress	必需	
Length	仅导出	不支持导出/导入长度 = 0 的地址。
IoType	必需	输入或输出

示例：导出组态


Project3 ▸ PLC_1 [CPU 412-2 PN]
Topology view
Network view
Device view
PLC_1 [CPU 412-2 PN]
PLC_1
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18
UR1_0
Device data

导出文件的 AML 结构

以下结构示例显示了“UR1_0”和模块“PLC_1”的导出过程。

```
...
<InternalElement ID="6563466e-2de9-42ca-951d-eb8f2545958d" Name="S7-400 station_1">
  <Attribute Name="TypeIdentifier" AttributeDataType="xs:string">
    <Value>System:Device.S7400</Value>
  </Attribute>
  <InternalElement ID="96930368-14ec-43e2-b9b7-c1fefc4b0534" Name="UR1_0">
    <Attribute Name="TypeName" AttributeDataType="xs:string">
      <Value>UR1</Value>
    </Attribute>
    <Attribute Name="PositionNumber" AttributeDataType="xs:int">
      <Value>0</Value>
    </Attribute>
    <Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
      <Value>False</Value>
    </Attribute>
    <Attribute Name="TypeIdentifier" AttributeDataType="xs:string">
      <Value>OrderNumber:6ES7 400-1TA01-0AA0</Value>
    </Attribute>
    <InternalElement ID="a1de449e-4f89-45af-8bbc-f77c28bccd04" Name="PLC_1">
      <Attribute Name="TypeName" AttributeDataType="xs:string">
        <Value>CPU 412-2 PN</Value>
      </Attribute>
      <Attribute Name="DeviceItemType" AttributeDataType="xs:string">
        <Value>CPU</Value>
      </Attribute>
      <Attribute Name="PositionNumber" AttributeDataType="xs:int">
        <Value>2</Value>
      </Attribute>
      <Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
        <Value>False</Value>
      </Attribute>
      <Attribute Name="TypeIdentifier" AttributeDataType="xs:string">
        <Value>OrderNumber:6ES7 412-2EK06-0AB0</Value>
      </Attribute>
      <Attribute Name="FirmwareVersion" AttributeDataType="xs:string">
        <Value>V6.0</Value>
      </Attribute>
     ...
      <SupportedRoleClass RefRoleClassPath=
          "AutomationProjectConfigurationRoleClassLib/DeviceItem" />
    </InternalElement>
    <SupportedRoleClass RefRoleClassPath=
      "AutomationProjectConfigurationRoleClassLib/DeviceItem" />
  </InternalElement>
  <SupportedRoleClass RefRoleClassPath=
      "AutomationProjectConfigurationRoleClassLib/Device" />
</InternalElement>
...
```

参见

基于 GSD/GSDML 的设备和设备项的导出/导入 (页 483)

用于导入/导出的 CAx 数据的结构 (页 443)

AML 类型标识符 (页 448)

8.5.15 设备项对象的导入

要求

- Openness 应用程序已连接 TIA Portal。
  参见“连接到 TIA Portal (页 69)”
- 项目已打开。
  参见“打开项目 (页 96)”
- PLC 处于离线状态。

应用

`DeviceItem` 对象为 `Device` 对象的嵌套子项。`DeviceItem` 类型的对象可以是一个机架或插入的模块。

- 设备第一个子项的类型必须为机架。机架的 `PositionNumber` 从 0 开始。如果存在多个机架，则会对其进行连续编号（1、2、3...）。
  AML 文件中一个层级内的顺序无限制。
- 机架类型的所有其它子项均为模块。

CAx 数据导入支持以下由 AML 类型标识符指定的设备项类型：

- 物理模块
- GSD/GSDML 模块
- 通用模块

如果仅能确定前端模块或 PLC 的标识 (`TypeIdentifier`)，且这些设备不带机架和设备，则可导入一个通用机架。

示例： `TypeIdentifier` = `System:Rack.Generic`

要替换通用机架，以下元素必须处于机架中（如 AML 文件中所述）：

- 中央设备：PLC
- 分布式设备：前端模块

通用机架类型包含在 Device 类型中。因此，待导入 TIA Portal 中的 AML 文件可使用该机架的类型标识符：

此时，TIA Portal 可确定机架的类型标识符。

如果机架和模块为通用型，则属性 `BuiltIn` 将定义该机架或模块的类型：

- 物理：`BuiltIn = True`
- 通用：`BuiltIn = False`

限制条件

导入时，属性 `DeviceItemType` 不相关，因此为可选项。

说明

属性“FirmwareVersion”

如果导入文件中未指定 `FirmwareVersion`，则 CAx 导入将使用 TIA Portal 中有效的最新固件版本。

如果导入文件中存在带有空值的 `FirmwareVersion` 属性，则设备项导入操作失败，并记录一条错误消息。

示例：导入通用设备

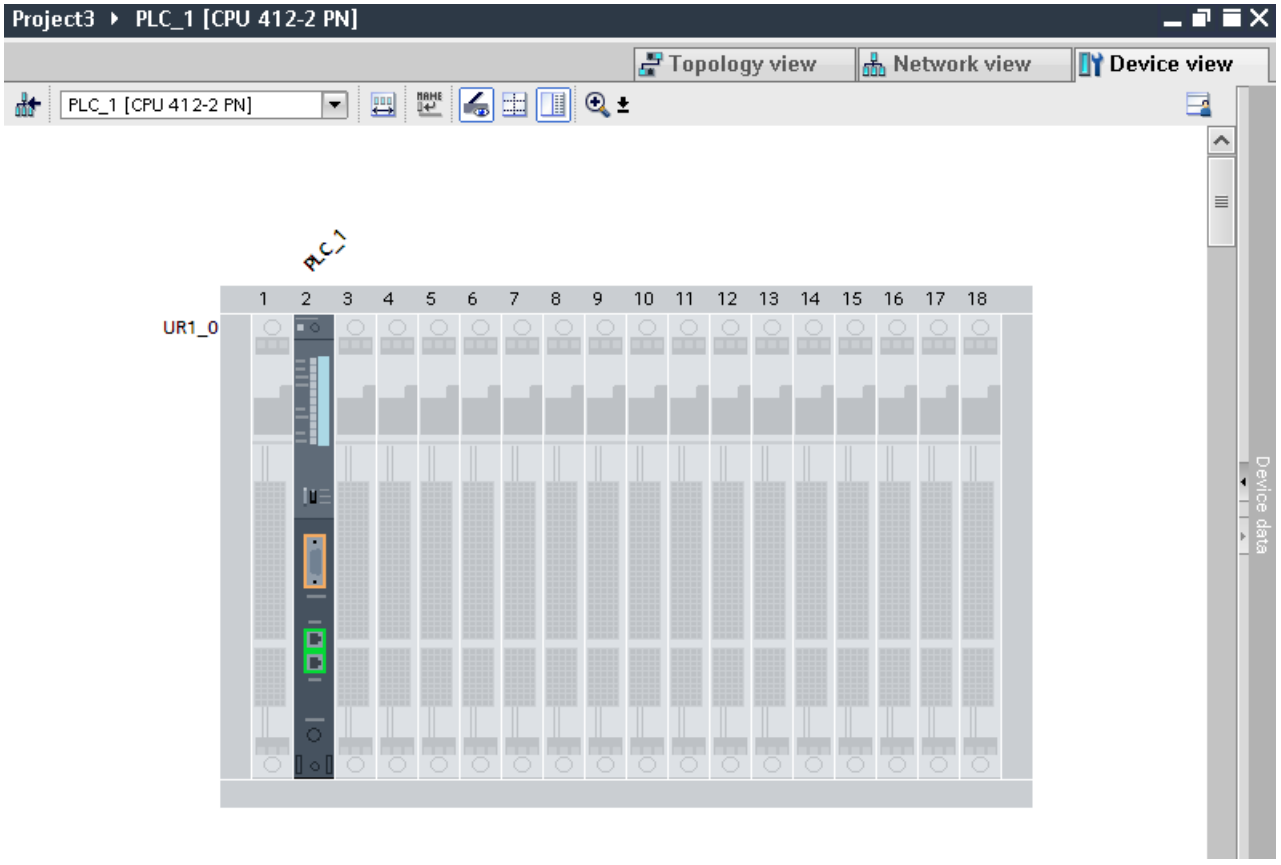
以下结构示例介绍了通用机架“Rack_1”的导入过程。

```
...
<InternalElement ID="6563466e-2de9-42ca-951d-eb8f2545958d" Name="S7-400 station_1">
  <Attribute Name="TypeIdentifier" AttributeDataType="xs:string">
    <Value>System:Device.S7400</Value>
  </Attribute>
  <InternalElement ID="96930368-14ec-43e2-b9b7-c1fefc4b0534" Name="UR1_0">
    <Attribute Name="TypeName" AttributeDataType="xs:string">
      <Value>UR1</Value>
    </Attribute>
    <Attribute Name="PositionNumber" AttributeDataType="xs:int">
      <Value>0</Value>
    </Attribute>
    <Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
      <Value>False</Value>
    </Attribute>
    <Attribute Name="TypeIdentifier" AttributeDataType="xs:string">
      <Value>System:Rack.Generic</Value>
    </Attribute>
    <InternalElement ID="a1de449e-4f89-45af-8bbc-f77c28bccd04" Name="PLC_1">
      <Attribute Name="TypeName" AttributeDataType="xs:string">
        <Value>CPU 412-2 PN</Value>
      </Attribute>
      <Attribute Name="DeviceItemType" AttributeDataType="xs:string">
        <Value>CPU</Value>
      </Attribute>
      <Attribute Name="PositionNumber" AttributeDataType="xs:int">
        <Value>2</Value>
      </Attribute>
      <Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
        <Value>False</Value>
      </Attribute>
      <Attribute Name="TypeIdentifier" AttributeDataType="xs:string">
        <Value>OrderNumber:6ES7 412-2EK06-0AB0</Value>
      </Attribute>
      <Attribute Name="FirmwareVersion" AttributeDataType="xs:string">
        <Value>V6.0</Value>
      </Attribute>

      <SupportedRoleClass RefRoleClassPath=
          "AutomationProjectConfigurationRoleClassLib/DeviceItem" />
    </InternalElement>
    <SupportedRoleClass RefRoleClassPath=
      "AutomationProjectConfigurationRoleClassLib/DeviceItem" />
  </InternalElement>
  <SupportedRoleClass RefRoleClassPath=
      "AutomationProjectConfigurationRoleClassLib/Device" />
</InternalElement>
...
```

导入的组态

下图显示了 TIA Portal 用户界面中的已导入组态：



参见

用于导入/导出的 CAx 数据的结构 (页 443)

AML 类型标识符 (页 448)

8.5.16 基于 GSD/GSDML 的设备和设备项的导出/导入

要求

- Openness 应用程序已连接 TIA Portal。
  参见“连接到 TIA Portal”
- 项目已打开。
  参见“打开项目”
- PLC 处于离线状态。

应用

基于 GSD/GSDML 的设备和设备项的 CAx 导入/导出类似于标准设备的导入/导出。

对于基于 GSD/GSDML 的设备和设备项，可导出的属性不同。如，GSD/GSDML 中包含属性 `Label`。

支持通用设备和机架的导入。导入时，使用标准设备的标识符：

- 导入通用设备：`TypeIdentifier = System:Device.Generic`
- 导入通用机架：`TypeIdentifier = System:Rack.Generic`

如果设备为通用型，则属性 `BuiltIn` 将定义类型：

- 物理：`BuiltIn = True`
- 通用：`BuiltIn = False`

设备的属性

下表列出了 CAx 导入和导出文件中相关的设备属性：

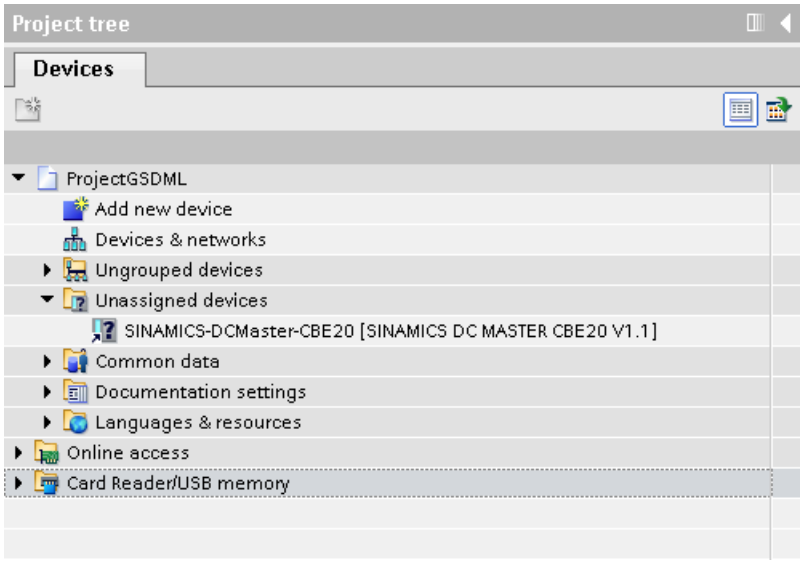
属性	属性的处理方式	注释
Name	导出和导入时必需	
TypeIdentifier	导出和导入时必需	
Comment	导入时可选	

设备项的属性

下表列出了 CAx 导入和导出文件中相关的设备项属性：

属性	属性的处理方式 BuiltIn = FALSE 通用设备项	属性的处理方式 BuiltIn = TRUE 物理设备项	注释
Name	必需	仅导出	
TypeName	仅导出	不适用	
DeviceItemType	仅导出	仅导出	仅限 PLC（中央设备）和 HeadModule（分散设备）设备项
PositionNumber	必需	导出时为必须项 例外： 设备项类型接口：导入时可选 设备项类型端口：如果未指定“Label”属性，则在导入 buldIn 设备时为必须项。如果“PositionNumber”和“Label”均已组态，则导出和导入时“PositionNumber”的优先级较高。	
BuiltIn	可选		默认：FALSE
TypeIdentifier	“BuiltIn”= FALSE 时为必须项	“BuiltIn”= TRUE 时被忽略	
Comment	可选	不适用	
Label	-	- 设备项类型接口：必需 设备项类型端口：如果未指定“PositionNumber”属性，则为必须项。如果“PositionNumber”和“Label”均已组态，则“PositionNumber”具有较高优先级，导入时应考虑到这一点。	

示例：已导出 GSD/GSDML 设备



导出文件的 AML 结构

下图所示为导出的 AML 文件的结构。

```
...
<InternalElement ID="9ae02cde-dfb4-4d43-a649-68b9ede7fc3d" Name="Ungrouped devices">
  <InternalElement ID="12d4ce0f-346d-4bfa-b139-c9d0db64c794" Name="GSD device_1">
    <Attribute Name="TypeIdentifier" AttributeDataType="xs:string">
      <Value>GSD:GSDML-V2.31-SIEMENS-SINAMICS_DCMASTER-20140704.XML/D</Value>
    </Attribute>
    <InternalElement ID="ccb1cb62-67b2-4b8c-951f-10c7ffb4d787" Name="Rack">
      <Attribute Name="TypeName" AttributeDataType="xs:string">
        <Value>Rack</Value>
      </Attribute>
      ...
      <Attribute Name="TypeIdentifier" AttributeDataType="xs:string">
        <Value>GSD:GSDML-V2.31-SIEMENS-SINAMICS_DCMASTER-20140704.XML/R/IDD_14</Value>
      </Attribute>
      <InternalElement ID="74f25b5c-0c09-46d0-9011-f341a3e98a0d" Name="SINAMICS-DCMaster-CBE20">
        <Attribute Name="TypeName" AttributeDataType="xs:string">
          <Value>SINAMICS DC MASTER CBE20 V1.1</Value>
        </Attribute>
        <Attribute Name="DeviceItemType" AttributeDataType="xs:string">
          <Value>HeadModule</Value>
        </Attribute>
        <Attribute Name="PositionNumber" AttributeDataType="xs:int">
          <Value>0</Value>
        </Attribute>
        <Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
          <Value>False</Value>
        </Attribute>
        <Attribute Name="TypeIdentifier" AttributeDataType="xs:string">
          <Value>GSD:GSDML-V2.31-SIEMENS-SINAMICS_DCMASTER-20140704.XML/DAP/IDD_14</Value>
        </Attribute>
        <InternalElement ID="94f34bb9-fe47-4904-b8a1-62e8fb6b1b74"
            Name="SINAMICS DC MASTER CBE20 V1.1">
          <Attribute Name="PositionNumber" AttributeDataType="xs:int">
            <Value>0</Value>
          </Attribute>
          <Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
            <Value>True</Value>
          </Attribute>
          <SupportedRoleClass RefRoleClassPath=
              "AutomationProjectConfigurationRoleClassLib/DeviceItem" />
        </InternalElement>
        ...
        <SupportedRoleClass RefRoleClassPath=
          "AutomationProjectConfigurationRoleClassLib/DeviceItem" />
      </InternalElement>
      <SupportedRoleClass RefRoleClassPath=
          "AutomationProjectConfigurationRoleClassLib/DeviceItem" />
    </InternalElement>
    <SupportedRoleClass RefRoleClassPath=
      "AutomationProjectConfigurationRoleClassLib/Device" />
  </InternalElement>
  <SupportedRoleClass RefRoleClassPath=
      "AutomationProjectConfigurationRoleClassLib/DeviceUserFolder" />
</InternalElement>
...
```

参见

AML 类型标识符 (页 448)

8.5.17 导出/导入子网

要求

- Openness 应用程序连接到 TIA Portal。
  请参见连接到 TIA Portal
- 已打开一个项目。
  请参见打开项目
- PLC 已处于离线状态。

AML 结构

子网对物理网络进行了说明，特别对连接至同一 PROFIBUS、PROFINET、MPI 或 ASI 网络类型的设备进行了说明。

网络和设备项之间的连接被模型化为对网络对象的引用。不存在从网络对象至所连接设备项的引用。网络参数存储在网络对象中。连接至网络的已知设备项的网络接口相关参数存储在该设备项的网络节点对象中。通常使用“通道”、“端口”和“接口”控制通信。

子网在 AML 文件的实例层级中被导出为“子网”角色类别的内部元素。

在 AML 结构中，子网具有以下相关元素：

- “节点”角色类别的内部元素
  定义设备项的接口。
- <InternalLink>
  定义子网的连接伙伴。<InternalLink> 变量名唯一，且始终添加到 AML 文件的项目内部元素下。

```
    ....
    <SupportedRoleClass RefRoleClassPath=
      "AutomationProjectConfigurationRoleClassLib/AutomationProject" />
    <InternalLink Name="Link To Port_1"
      RefPartnerSideA="1e3e4c5b-04c1-4d2c-9aee-cad53cc92dba:CommunicationPortInterface"
      RefPartnerSideB="d45aa36a-a7f2-4862-a266-d6727b9cfd75:CommunicationPortInterface" />
    <InternalLink Name="Link To Subnet_1"
      RefPartnerSideA="beb4eb8e-1a45-45ce-a703-1acfac73e5f3:LogicalEndPoint_Node"
      RefPartnerSideB="1062a384-d3ca-4183-9ac2-0934a5ab7286:LogicalEndPoint_Subnet" />
    <InternalLink Name="Link To Subnet_2"
      RefPartnerSideA="a3e85aed-580a-45c8-943e-da7de8280b7c:LogicalEndPoint_Node"
      RefPartnerSideB="1062a384-d3ca-4183-9ac2-0934a5ab7286:LogicalEndPoint_Subnet" />
  </InternalElement>
 </InstanceHierarchy>
</CAEXFile>
```

- <ExternalInterface>
  在节点和子网内部元素中表示已连接节点和子网。如果节点或子网未连接，则节点和子网的 <ExternalInterface> 元素不存在。

```
...
<InternalElement ID="1062a384-d3ca-4183-9ac2-0934a5ab7286" Name="PN/IE_1">
  <Attribute Name="Type" AttributeDataType="xs:string">
    <Value>Ethernet</Value>
  </Attribute>
  <ExternalInterface ID="55ecf2cb-e72a-4974-8ed0-1d4e1ade3509"
      Name="LogicalEndPoint_Subnet"
      RefBaseClassPath="CommunicationInterfaceClassLib/LogicalEndPoint" />
  <SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/Subnet" />
</InternalElement>
...
```

应用

CAx 导入/导出支持以下类型的子网：

- 以太网
- PROFIBUS
- MPI
- ASi

“子网”元素的属性

下表显示了用于 CAx 导入和导出文件的相关对象属性：

属性	处理	注释
Name	必须项	
Type	必须项	Ethernet、PROFIBUS、MPI 或 ASi

“CommunicationInterface”元素的属性

下表显示了用于 CAx 导入和导出文件的相关对象属性：

属性	处理	注释
Name	必须项	与“固定”设备项无关。
Label	必须项	如果 "BuiltIn" = TRUE，且为相关“DeviceItem”对象指定了“PositionNumber”，则 Label 会丢失。
TypeIdentifier	必须项	
FirmwareVersion	必须项	
TypeName	仅导出	与“BuiltIn”设备项无关。
DeviceItemType	仅导出	仅适用于 CPU 和头模块
PositionNumber	必须项	与“BuiltIn”设备项的导入无关。
BuiltIn	导出时为必须项 导入时为可选项	与“Non-BuiltIn”设备项的导入无关。 导入时默认为 False。
Comment	可选项	不适用于“BuiltIn”设备项。

“CommunicationPort”元素的属性

下表显示了用于 CAx 导入和导出文件的相关对象属性：

属性	处理	注释
Name	必须项	与“BuiltIn”设备项无关。
Label	必须项	如果 "BuiltIn" = TRUE，且为相关“DeviceItem”对象指定了“PositionNumber”，则 Label 会丢失。
TypeIdentifier	必须项	
FirmwareVersion	必须项	
TypeName	仅导出	与“BuiltIn”设备项无关。
DeviceItemType	仅导出	仅适用于 CPU 和头模块。
PositionNumber	必须项	如果未指定“Label”属性，则仅与“BuiltIn”设备项的导入相关。 如果配置了“PositionNumber”和“Label”，则“PositionNumber”获得更高的优先级。
BuiltIn	导出时为必须项 导入时为可选项	与“Non-BuiltIn”设备项的导入无关。 导入时默认为 False。
Comment	可选项	不适用于“BuiltIn”设备项。

“节点”元素的属性

下表显示了用于 CAx 导入和导出文件的相关对象属性：

属性	处理	注释
Name	仅导出	MPI, PROFIBUS, PROFINET
Type	仅导出	Ethernet、PROFIBUS、MPI 或 ASi
NetworkAddress	必须项	
SubnetMask	可选项	PROFINET 对于导入，如果未设置值，则保留默认值。
RouterAddress	可选项	PROFINET 对于导入，如果未设置值，则保留默认值。

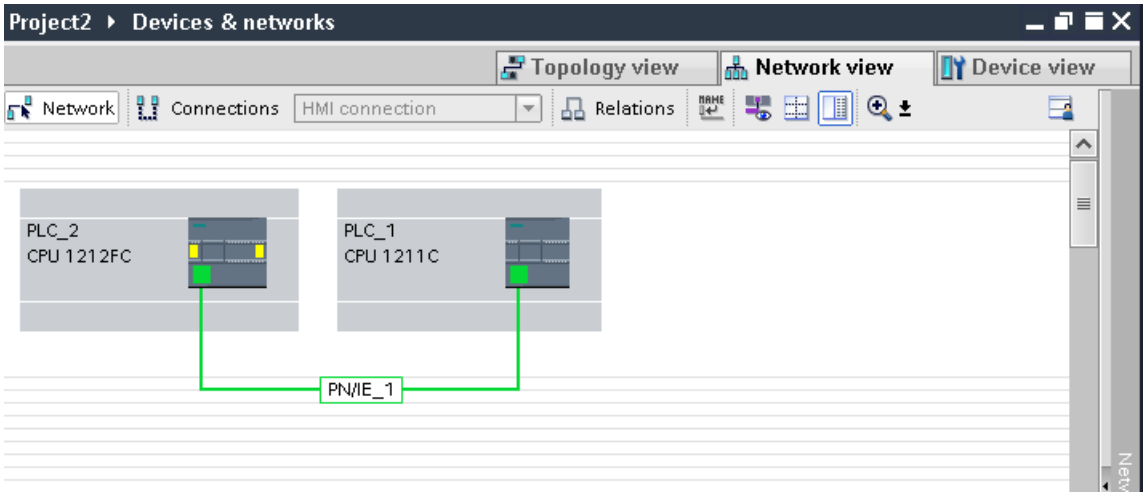
属性	处理	注释
DhcpClientId	可选项	PROFINET 对于导入，如果未设置值，则保留默认值。
IpProtocolSelection	可选项	PROFINET 对于导入，如果未设置值，则保留默认值。 值：项目、Dhcp、UserProgram、OtherPath

“通道”元素的属性

下表显示了用于 CAx 导入和导出文件的相关对象属性：

属性	处理	注释
Type	必须项	数字量或模拟量
IoType	必须项	输入或输出
Number	必须项	
Length	仅导出	

示例：导出的子网



AML 结构

下图所示为导出的 AML 文件的结构：

```
...
<InstanceHierarchy Name="APC Sample Instance Hierarchy">
  <InternalElement ID="e9d2bedb-f8c1-4148-acda-c3c68836c7dd" Name="Project2">
    ...
      <InternalElement ID="1062a384-d3ca-4183-9ac2-0934a5ab7286" Name="PN/IE_1">
      <Attribute Name="Type" AttributeDataType="xs:string">
        <Value>Ethernet</Value>
      </Attribute>
      <ExternalInterface ID="55ecf2cb-e72a-4974-8ed0-1d4e1ade3509"
          Name="LogicalEndPoint_Subnet"
              RefBaseClassPath="CommunicationInterfaceClassLib/LogicalEndPoint" />
      <SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/Subnet" />
    </InternalElement>
    <InternalElement ID="b011dbb1-efa4-46c0-a26f-f9bd047cda4f" Name="S7-1200 station_1">
          ...
          <InternalElement ID="d006e41b-05ff-44ab-baab-fca15f99e86c" Name="PROFINET interface_1">
            ...
            <InternalElement ID="beb4eb8e-1a45-45ce-a703-1acfac73e5f3" Name="E1">
              ...
              <ExternalInterface ID="a365b498-20cc-4e0b-99ca-5c5257632b96"
                  Name="LogicalEndPoint_Node" RefBaseClassPath=
                      "CommunicationInterfaceClassLib/LogicalEndPoint" />
              <SupportedRoleClass RefRoleClassPath=
                  "AutomationProjectConfigurationRoleClassLib/Node" />
            </InternalElement>
            <InternalElement ID="d45aa36a-a7f2-4862-a266-d6727b9cfd75" Name="Port_1">
              ...
              <ExternalInterface ID="32c6ba4a-b01f-4678-b721-ea284779e96c"
                  Name="CommunicationPortInterface"
                  RefBaseClassPath=
                  "AutomationProjectConfigurationInterfaceClassLib/CommunicationPortInterface" />
              <SupportedRoleClass RefRoleClassPath=
              "AutomationProjectConfigurationRoleClassLib/CommunicationPort" />
            </InternalElement>
            <SupportedRoleClass RefRoleClassPath=
              "AutomationProjectConfigurationRoleClassLib/CommunicationInterface" />
          </InternalElement>
          <SupportedRoleClass RefRoleClassPath=
              "AutomationProjectConfigurationRoleClassLib/DeviceItem" />
      </InternalElement>
      <SupportedRoleClass RefRoleClassPath=
        "AutomationProjectConfigurationRoleClassLib/DeviceItem" />
    </InternalElement>
    <SupportedRoleClass RefRoleClassPath=
        "AutomationProjectConfigurationRoleClassLib/Device" />
  </InternalElement>
...
```

```
...
<InternalElement ID="7cf0ea2b-b66f-4ad4-8a03-5a8691cbe04d" Name="PLC_2">
  ...
  <InternalElement ID="b287020d-667b-483d-a8e0-c5466ac2f5c3" Name="PROFINET interface_1">
    <Attribute Name="Label" AttributeDataType="xs:string">
      <Value>X1</...
    <InternalElement ID="a3e85aed-580a-45c8-943e-da7de8280b7c" Name="E1">
      <Attribute Name="Type" AttributeDataType="xs:string">
        <Value>Ethernet</Value>
      </Attribute>
      <Attribute Name="NetworkAddress" AttributeDataType="xs:string">
        <Value>192.168.0.2</Value>
      </Attribute>
      <Attribute Name="SubnetMask" AttributeDataType="xs:string">
        <Value>255.255.255.0</Value>
      </Attribute>
      <Attribute Name="DeviceNumber" AttributeDataType="xs:string">
        <Value>0</Value>
      </Attribute>
      <Attribute Name="IpProtocolSelection" AttributeDataType="xs:string">
        <Value>Project</Value>
      </Attribute>
      <ExternalInterface ID="6ae8eb93-09d3-4f8c-b529-a12148c71bf4"
          Name="LogicalEndPoint_Node"
          RefBaseClassPath="CommunicationInterfaceClassLib/LogicalEndPoint" />
      <SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/Node" />
    </InternalElement>
    <InternalElement ID="1e3e4c5b-04c1-4d2c-9aee-cad53cc92dba" Name="Port_1">
      ...
      <ExternalInterface ID="1f5b2a3d-fcd1-460a-b846-30dadc8726d1"
          Name="CommunicationPortInterface"
          RefBaseClassPath=
          "AutomationProjectConfigurationInterfaceClassLib/CommunicationPortInterface" />
      <SupportedRoleClass RefRoleClassPath=
      "AutomationProjectConfigurationRoleClassLib/CommunicationPort" />
    </InternalElement>
    <SupportedRoleClass RefRoleClassPath=
      "AutomationProjectConfigurationRoleClassLib/CommunicationInterface" />
  </InternalElement>
  <SupportedRoleClass RefRoleClassPath=
      "AutomationProjectConfigurationRoleClassLib/DeviceItem" />
</InternalElement>
...
```

参见

用于导入/导出的 CAx 数据的结构 (页 443)

连接到 TIA Portal (页 69)

打开项目 (页 96)

8.5.18 导出/导入 PLC 变量

要求

- Openness 应用程序连接到 TIA Portal。
  请参见连接到 TIA Portal
- 已打开一个项目。
  请参见打开项目
- PLC 已处于离线状态。

应用

导出和导入的符号及变量被分配给设备项。CAx 导入/导出涉及面向硬件的符号和变量。符号和变量仅基于控制器目标设备项（例如，CPU）进行导出，而不基于其可能引用的其它设备项（例如，I/O 模块）进行导出。与设备类似，变量通常被分组到变量表和层级文件夹结构中。

AML 结构元素

PLC 变量、变量表和变量用户文件夹可通过 CAx 导入/导出功能进行导出和导入。变量对象映射在以下 AML 结构元素中：

- `<InternalElement>`
  变量表和变量用户文件夹被映射为具有相应角色类别的相关 PLC 内部元素。
- `<ExternalInterface>`
  表示一个 PLC 变量，专用于相关变量表或变量用户文件夹的内部元素。

带有 PLC 变量的映射通道通过 `<internal link>` 元素导出为通信伙伴。以下 XML 结构显示了一个示例：

```
...
    <InternalLink Name="Link To Tag_1"
       RefPartnerSideA="b33451f6-d88f-4900-8dbe-41f1be1e3535:Channel_DI_0"
       RefPartnerSideB="b2b937ee-d5db-4826-9340-027b1da22828:Tag_1" />
...
```

PLC 变量用户文件夹

在 CAx 导入和导出文件中，对象“TagUserFolder”仅需“Name”属性。

PLC 变量表的属性

下表显示了用于 CAx 导入和导出文件的相关对象属性：

属性	处理	注释
Name	必须项，"AssignToDefault" = TRUE 时被忽略	
AssignToDefault	仅导入	用于在导入期间识别默认变量表。如果 "AssignToDefault" = TRUE，则会在 TIA Portal 的默认变量表下创建所有变量。

PLC 变量的属性

下表显示了用于 CAx 导入和导出文件的相关对象属性：

属性	处理	注释
Name	必须项	
DataType	必须项	
LogicalAddress	必须项	以国际助记符格式导入和导出
Comment	可选项	

示例：AML 结构

下图显示了以下所导出变量对象的结构：

- 空默认变量表
- 变量用户文件夹“Group_1”
- 包括的变量表“Tag table_1”
- 四个变量

```
...
<InternalElement ID="a310ba93-ba04-49d7-a8e3-004619c7d9d2" Name="Default tag table">
  <SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/TagTable" />
</InternalElement>
<InternalElement ID="0feff703-9c70-4ca9-b3b3-8de8229696dd" Name="Group_1">
  <InternalElement ID="f9269ce4-c015-459f-9f59-8f94bca3b186" Name="Tag table_1">
    <ExternalInterface ID="fc0c8c5a-fd5b-443b-b430-6435b6aa22ff" Name="Tag_1"
    RefBaseClassPath="AutomationProjectConfigurationInterfaceClassLib/Tag">
      ....
    </ExternalInterface>
    <ExternalInterface ID="450d6a1d-81b8-49ae-a104-c0072933d669" Name="Tag_2"
    RefBaseClassPath="AutomationProjectConfigurationInterfaceClassLib/Tag">
      ....
    </ExternalInterface>
    <ExternalInterface ID="3de17a36-b5c5-4fc7-9fc3-47e4a8f95087" Name="Tag_3"
    RefBaseClassPath="AutomationProjectConfigurationInterfaceClassLib/Tag">
      <Attribute Name="DataType" AttributeDataType="xs:string">
        <Value>Word</Value>
      </Attribute>
      <Attribute Name="LogicalAddress" AttributeDataType="xs:string">
        <Value>IW0</Value>
      </Attribute>
    </ExternalInterface>
    <SupportedRoleClass RefRoleClassPath=
       "AutomationProjectConfigurationRoleClassLib/TagTable" />
  </InternalElement>
  <SupportedRoleClass RefRoleClassPath=
    "AutomationProjectConfigurationRoleClassLib/TagUserFolder" />
</InternalElement>
...
```

参见

用于导入/导出的 CAx 数据的结构 (页 443)

连接到 TIA Portal (页 69)

打开项目 (页 96)

8.5.19 IO 系统的导出/导入

要求

- Openness 应用程序连接到 TIA Portal。
  请参见连接到 TIA Portal
- 已打开一个项目。
  请参见打开项目
- PLC 已处于离线状态。

AML 结构

IO 系统在 AML 结构中表示为 <InternalElement>。

作为主机或 IO 控制器的 IO 系统添加到接口设备项的 <CommunicationInterface> 元素下。

```
...
<InternalElement ID="[Communication Interface UniqueID]"
      Name="[Communication Interface Name]">
...
  <!--Node-->
  <InternalElement ID="[Node UniqueID]" Name="[Node Name]">
    ...
    <ExternalInterface ID="[External Interface UniqueID]"
       Name="LogicalEndPoint_Node"
       RefBaseClassPath="CommunicationInterfaceClassLib/LogicalEndPoint" />
    <SupportedRoleClass
       RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/Node" />
  </InternalElement>
  <!--IoSystem-->
  <InternalElement ID="[IoSystem UniqueID]" Name="[IoSystem Name]">
   <Attribute Name="Number" AttributeDataType="xs:integer">
    <Value>[IoSystem Number]</Value>
    </Attribute>
    <ExternalInterface ID="[External Interface UniqueID]"
       Name="LogicalEndPoint_Interface"
       RefBaseClassPath="CommunicationInterfaceClassLib/LogicalEndPoint" />
    <SupportedRoleClass
       RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/IoSystem" />
  </InternalElement>
  <SupportedRoleClass
       RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/CommunicationInterface" />
</InternalElement>
```

作为从设备或 IO 设备的已连接 IO 系统作为 <ExternalInterface> 元素添加到接口设备项的 <CommunicationInterface> 下。

```
<InternalElement ID="[Communication Interface UniqueID]"
    Name="[Communication Interface Name]">
 ...
  <ExternalInterface ID="[External Interface UniqueID]"
    Name="LogicalEndPoint_Interface"
    RefBaseClassPath="CommunicationInterfaceClassLib/LogicalEndPoint" />
  <!--Node-->
  <InternalElement ID="[Node UniqueID]" Name="[Node Name]">
    <ExternalInterface ID="[External Interface UniqueID]"
    Name="LogicalEndPoint_Node"
    RefBaseClassPath="CommunicationInterfaceClassLib/LogicalEndPoint" />
    <SupportedRoleClass
    RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/Node" />
  </InternalElement>
  <SupportedRoleClass
    RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/CommunicationInterface" />
</InternalElement>
```

IO 系统的连接伙伴表示为 <InternalLink> 元素。<InternalLink> 变量添加到 IO 系统和已连接从设备项的共同父项下，例如：项目、DeviceFolder、DeviceItem。

<InternalLink> 变量名称在公共父项中是唯一的。

“IO-system”元素的属性

下表显示了用于 CAx 导入和导出文件的相关对象属性：

属性	处理	注释
Name	必须项	IO 系统名称。如果导入空字符串，则使用默认名称创建 IO 系统。
Number	可选项	如果未指定导入，则应用默认值。

8.5.20 导出/导入多语言注释

要求

- Openness 应用程序连接到 TIA Portal。
  请参见连接到 TIA Portal
- 已打开一个项目。
  请参见打开项目
- PLC 已处于离线状态。

应用

CAx 数据交换可导出和导入以下硬件对象的注释和多语言注释：

- 设备 (Device)
- 模块 (DeviceItems)
- 变量 (Tag)

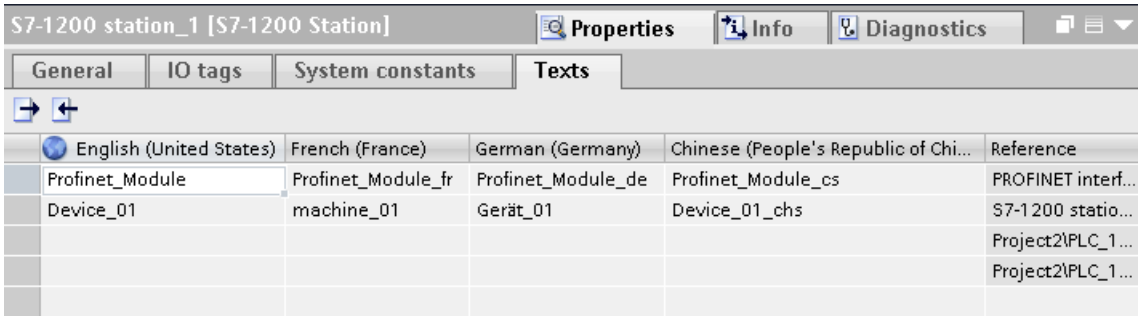
多语言注释的导入/导出包括所有 TIA Portal 语言。

限制

- 导出
  - 只有在存在注释的情况下，才能将“Comment”属性导出至 AML 文件。
- 导入
  - "Comment" 属性可选。
  - 对于虚拟设备项，不能导入注释。

示例：导出带多语言注释的组态

下图显示了 SIMATIC S7 1500 (Device) 及 PLC_1 (DeviceItems) 的组态。对于这两个对象，均以英语、法语、德语和中文设置注释。



AML 结构

导出该组态后，会将多语言注释生成为设备、设备项或变量的嵌套属性。

- 父属性“Comment”应包含采用默认语言的值。
- 对于每个外语注释，都存在一个子属性。

```
...
<Attribute Name="Comment" AttributeDataType="xs:string">
  <Value>English</Value>
  <Attribute Name="aml-lang=en-US" AttributeDataType="xs:string">
    <Value>English</Value>
  </Attribute>
  <Attribute Name="aml-lang=de-DE" AttributeDataType="xs:string">
    <Value>Deutsch</Value>
  </Attribute>
  <Attribute Name="aml-lang=zh-HK" AttributeDataType="xs:string">
    <Value>Chinese</Value>
  </Attribute>
  ...
  ...
</Attribute>
...
```

参见

用于导入/导出的 CAx 数据的结构 (页 443)

连接到 TIA Portal (页 69)

打开项目 (页 96)

8.5.21 AML 属性与 Openness 属性

访问属性和导出/导入属性

通过 Openness，您可以访问硬件对象的属性。用于访问这些（例如，设备项）属性的单个名称与导出/导入 AML 文件中的属性名不同。

属性列表

下表概要说明了两种类型的属性：

表格 8-6　设备和 GSD/GSDML 设备的属性名称

AML 文件	Openness
Name	Name
TypeIdentifier	TypeIdentifier
Comment	Comment

表格 8-7　设备项的属性名称

AML 文件	Openness
Name	Name
TypeIdentifier	映射到 <TypeIdentifier> 子串（即，第一个“/”运算符之前的值）忽略其中的固件版本部分。 映射子串仅适用于以 <OrderNumber:> 前缀开头的 TypeIdentifier，并且其固件版本部分映射到整个 <TypeIdentifier>。
FirmwareVersion	<FirmwareVersion> 映射到 <TypeIdentifier> 的子串（即，第一个“/”运算符之前的值）。映射子串仅适用于以 <<OrderNumber:> 前缀开头的 <TypeIdentifier>，并且具有固件版本部分。
TypeName	TypeName
DeviceItemType（适用于 CPU 和头模块）	**Classification**
PositionNumber	PositionNumber
BuiltIn	**IsBuiltIn**
PlantDesignation IEC	**PlantDesignation**
LocationIndentifier IEC	**LocationIdentifier**
Comment	Comment

表格 8-8　GSD/GSDML 设备项的属性名称

AML 文件	Openness
Name	Name
TypeIdentifier	TypeIdentifier
TypeName	TypeName
DeviceItemType （适用于头模块）	**Classification**
PositionNumber	PositionNumber
BuiltIn	**IsBuiltIn**

AML 文件	Openness
Comment	Comment
Label	Label

表格 8-9　变量的属性名称

AML 文件	Openness
Name	Name
DataType	**DataTypeName**
LogicalAddress	LogicalAddress
Comment	Comment

表格 8-10　变量表的属性名称

AML 文件	Openness
Name	Name
AssignToDefault	**IsDefault**

表格 8-11　地址的属性名称

AML 文件	Openness
StartAddress	StartAddress
Length	Length
IoType	IoType

表格 8-12　端口的属性名称

AML 文件	Openness
Name	Name
TypeIdentifier	TypeIdentifier
FirmwareVersion	FirmwareVersion
TypeName	TypeName
PositionNumber	PositionNumber

AML 文件	Openness
BuiltIn	**IsBuiltIn**
Comment	Comment
Label	Label

表格 8-13　带 IO 接口的设备的属性名称

AML 文件	Openness
Name	Name
TypeIdentifier	TypeIdentifier
FirmwareVersion	FirmwareVersion
TypeName	TypeName
DeviceItemType（适用于 CPU 和头模块）	**Classification**
PositionNumber	PositionNumber
BuiltIn	**IsBuiltIn**
Label	Label
Comment	Comment

表格 8-14　通道的属性名称

AML 文件	Openness
Type	Type
IoType	IoType
Number	未映射到 Openness 中的任何属性。
Length	**ChannelWidth**

主要变化

9

9.1 V14 SP1 中的主要变更

简介

API 对象模型 V14 SP1 中做出了以下更改，可能会影响现有应用程序：

更改	所需程序代码调整
提升主站副本的处理能力	CreateFrom 操作会在库中创建一个基于新对象的主站副本，并将它置于调用该操作的组成中。CreateFrom 操作仅支持包含单个对象的主副本。返回类型与相应组成类型相对应。 以下组成支持 CreateFrom： • Siemens.Engineering.HW.DeviceComposition • Siemens.Engineering.HW.DeviceItemComposition • Siemens.Engineering.SW.Blocks.PlcBlockComposition • Siemens.Engineering.SW.Tags.PlcTagTableComposition • Siemens.Engineering.SW.Tags.PlcTagComposition • Siemens.Engineering.SW.Types.PlcTypeComposition • Siemens.Engineering.SW.TechnologicalObjects.TechnologicalInstanceDBComposition • Siemens.Engineering.SW.Tags.PlcUserConstantComposition • Siemens.Engineering.Hmi.Tag.TagTableComposition • Siemens.Engineering.Hmi.Tag.TagComposition • Siemens.Engineering.Hmi.Screen.ScreenComposition • Siemens.Engineering.Hmi.Screen.ScreenTemplateComposition • Siemens.Engineering.Hmi.RuntimeScripting.VBScriptComposition • Siemens.Engineering.HW.SubnetComposition • Siemens.Engineering.HW.DeviceUserGroupComposition • Siemens.Engineering.SW.Blocks.PlcBlockUserGroupComposition • Siemens.Engineering.SW.ExternalSources.PlcExternalSourceUserGroupComposition • Siemens.Engineering.SW.Tags.PlcTagTableUserGroupComposition • Siemens.Engineering.SW.Types.PlcTypeUserGroupComposition
提升全局库处理能力	全局库内的现有操作现在可以修改操作，例如，从全局库中删除主副本。 UpdateProject 和 UpdateLibrary 不再使用 UpdatePathsMode 和 DeleteUnusedVersionsMode 参数。更新后不会删除未使用的版本

更改	所需程序代码调整
更改 System.String 至 System.IO.FileInfo 更改 System.String 至 System.IO.DirectoryInfo	所有必须指定字符串路径的事件均使用 FileInfo 路径或 DirectoryInfo 路径。例如： • 打开项目 • 打开库 • 创建项目 • 创建全局库 • ...

对象模型中的新项目

名称	类型	命名空间	注释
PlcUserConstant	类别	Siemens.Engineering.SW.Tags	由 PlcConstant 拆分而来。
PlcUserConstantComposition	类别	Siemens.Engineering.SW.Tags	由 PlcConstantComposition 拆分而来。
PlcSystemConstant	类别	Siemens.Engineering.SW.Tags	由 PlcConstant 拆分而来。
PlcSystemConstantComposition	类别	Siemens.Engineering.SW.Tags	由 PlcConstantComposition 拆分而来。
MultilingualTextItem	类别	Siemens.Engineering	访问多语言文本
MultilingualTextItemComposition	类别	Siemens.Engineering	访问多语言文本
TiaPortalTrustAuthority.FeatureTokens	枚举值	Siemens.Engineering	访问 TIA Portal 设置。
TiaPortalSetting	类别	Siemens.Engineering.Settings	访问 TIA Portal 设置。
TiaPortalSettingComposition	类别	Siemens.Engineering.Settings	访问 TIA Portal 设置。
TiaPortalSettingsFolder	类别	Siemens.Engineering.Settings	访问 TIA Portal 设置。
TiaPortalSettingsFolderComposition	类别	Siemens.Engineering.Settings	访问 TIA Portal 设置。
LanguageAssociation	类别	Siemens.Engineering	访问以激活语言。
LanguageComposition.Find	方法	Siemens.Engineering	访问以激活语言。

对象模型中的已修改项

名称	类型	命名空间	注释
PlcConstant	类别	Siemens.Engineering.SW.Tags	发布的 PlcUserConstant 和 PlcSystemConstant 的基本类别。
PlcTag	类别	Siemens.Engineering.SW.Tags	由 PlcConstantComposition 拆分而来。
ITargetComparable	接口	Siemens.Engineering.Compare	字符串属性 DataTypeName 而不是开放连接 DataType。
MultilingualText	类别	Siemens.Engineering	访问多语言文本
ProjectComposition.Create	方法	Siemens.Engineering	参数更改为使用 DirectoryInfo 和字符串。
Project.Subnets	属性	Siemens.Engineering	访问子网
Project.Languages	属性	Siemens.Engineering	移动称为 Siemens.Engineering.LanguageSettings 的一个属性以提供支持语言

在对象模式中移除项

名称	类型	命名空间	注释
PlcConstantComposition	类别	Siemens.Engineering.SW.Tags	PlcSystemConstantComposition 和 PlcUserConstantComposition 中的拆分。
CompareResultElement.PathInformation	属性	Siemens.Engineering.SW.Tags	不再使用
MultilingualText.GetText(CultureInfo cultureInfo)	方法	Siemens.Engineering.Compare	修改访问文本项 MultilingualText 的原理。
TiaPortalTrustAuthority.CustomerIdentification	枚举值	Siemens.Engineering	不再使用
TiaPortalTrustAuthority.ElevatedAccessExtensions	枚举值	Siemens.Engineering	不再使用

行为更改

名称	类型	命名空间	注释
PlcTag.Export(FileInfo path, ExportOptions options)	方法	Siemens.Engineering.SW.Tags	现在属性 LogicalAddress 的值总是导出至内部助记码。在导入时仍使用德国助记码。
PlcTag.LogicalAddress	属性	Siemens.Engineering.SW.Tags	现在属性 LogicalAddress 的值总是返回至内部助记码。写入时使用德国助记码。

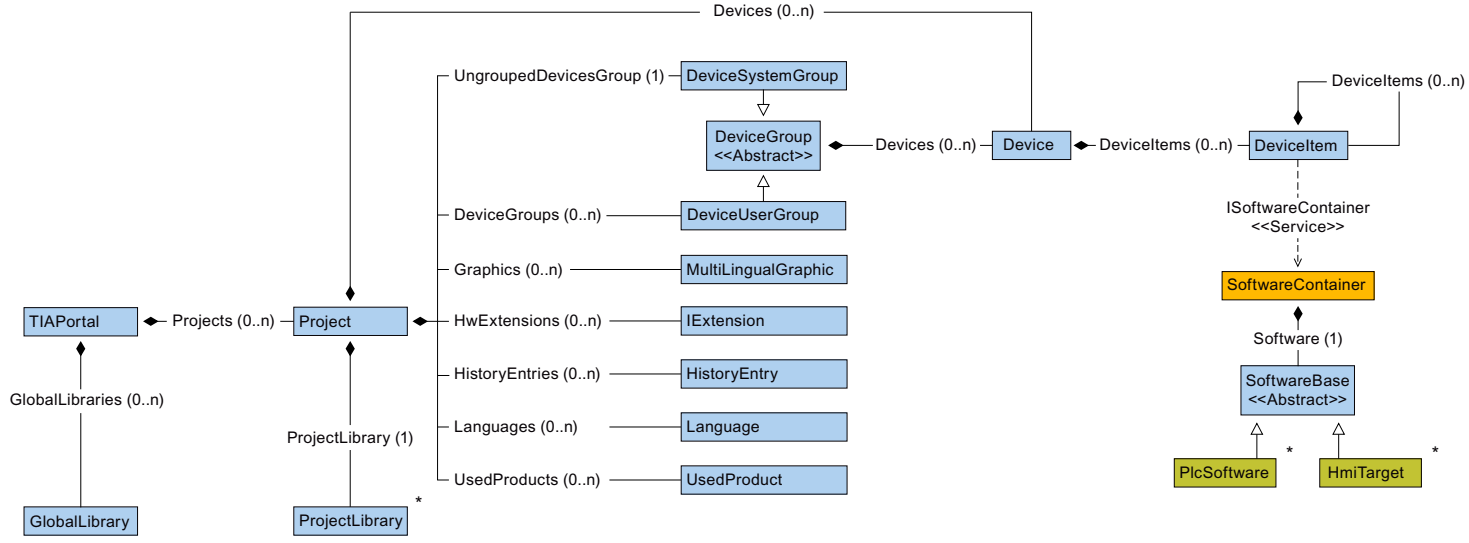
9.2 对象模型中的主要更改

TIA Portal Openness V14 版本的对象模型

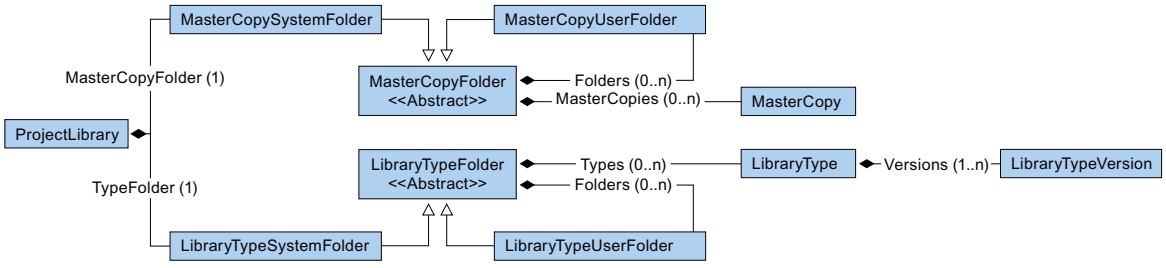
为了便于比较 Openness 的新旧对象模型，下图显示了 TIA Portal V14 的对象模型。

说明

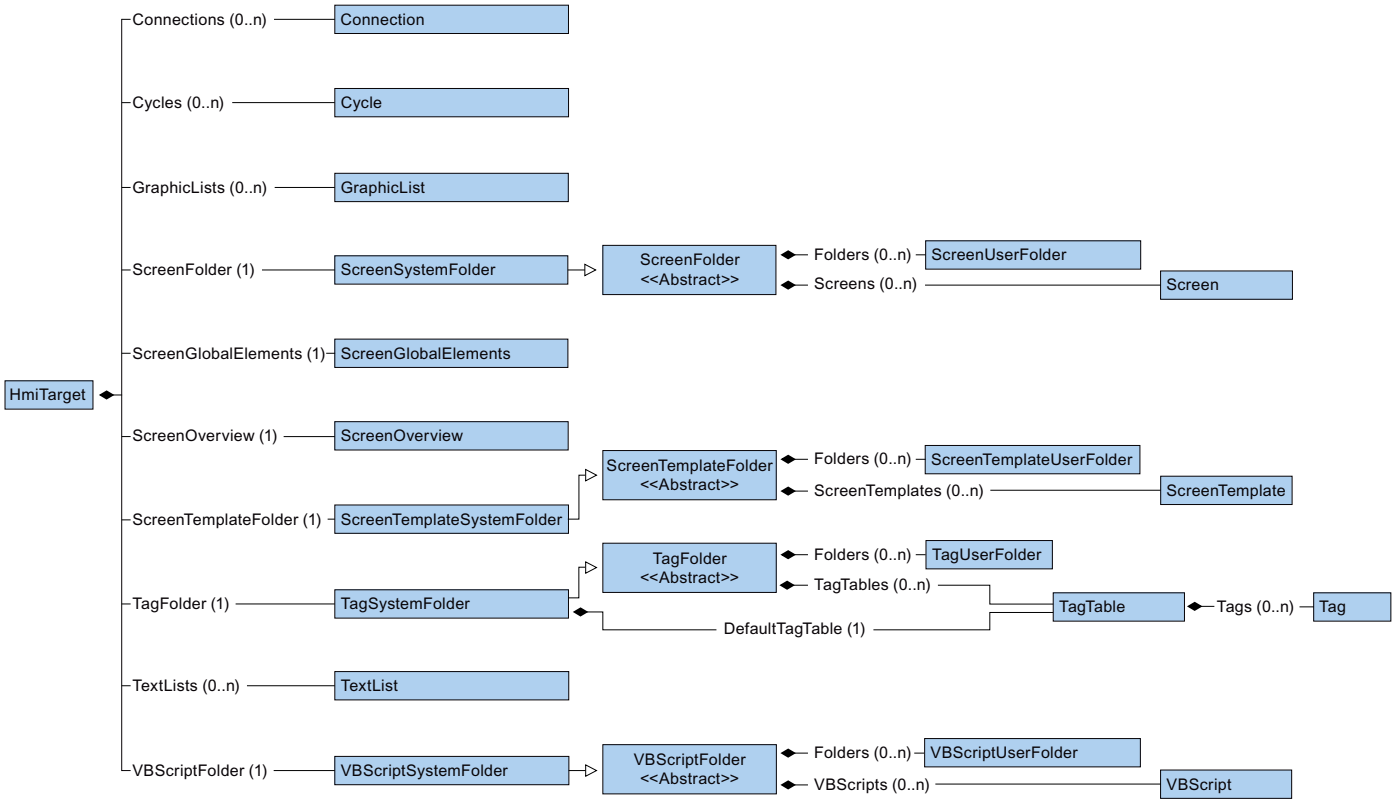
该图中显示的对象模型已过时，有关 Openness V14 SP1 的对象模型信息，请参见 “Openness 对象模型 (页 45)”



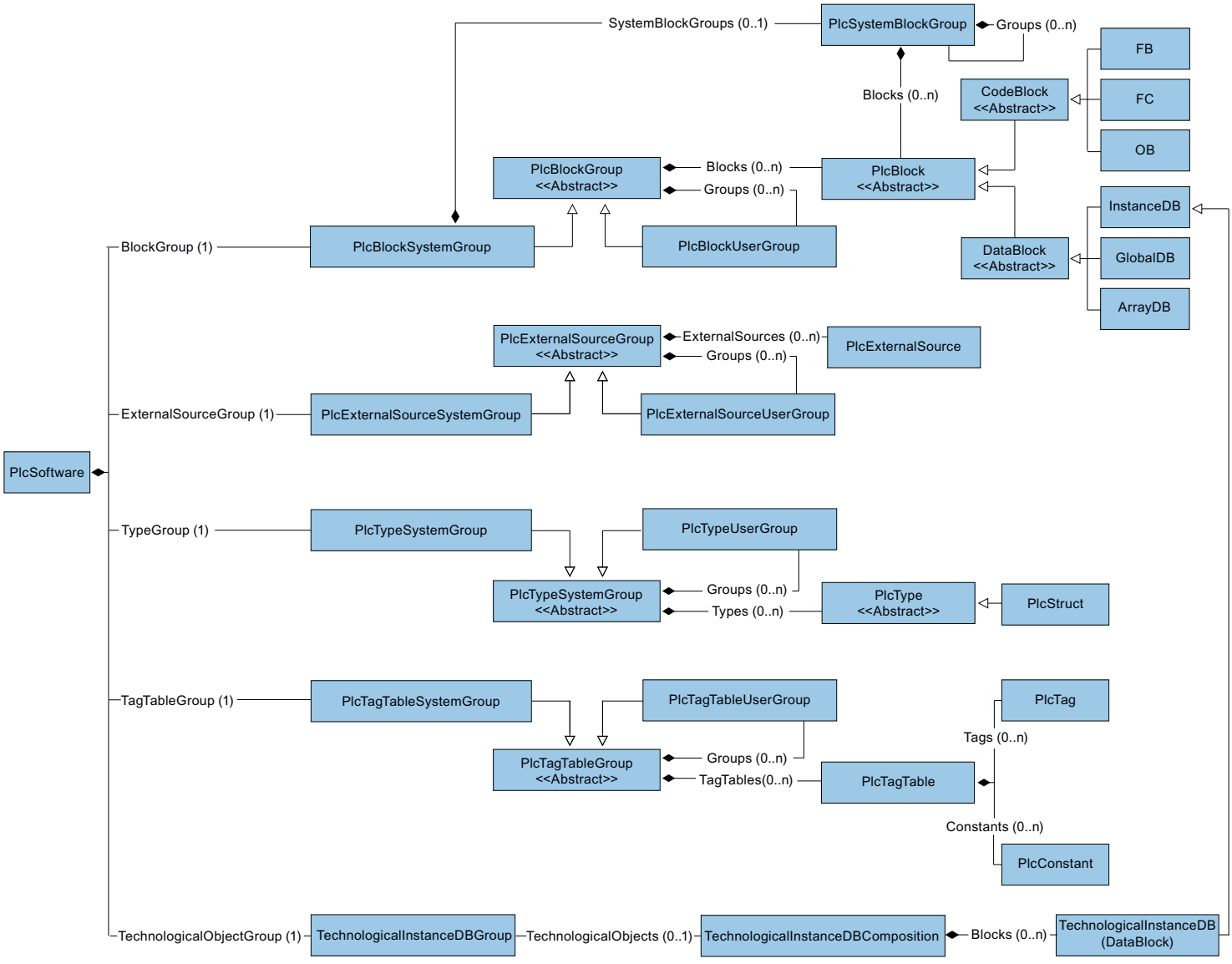
下图所示为 ProjectLibrary 下的对象。



下图所示为 HmiTarget 下的对象。

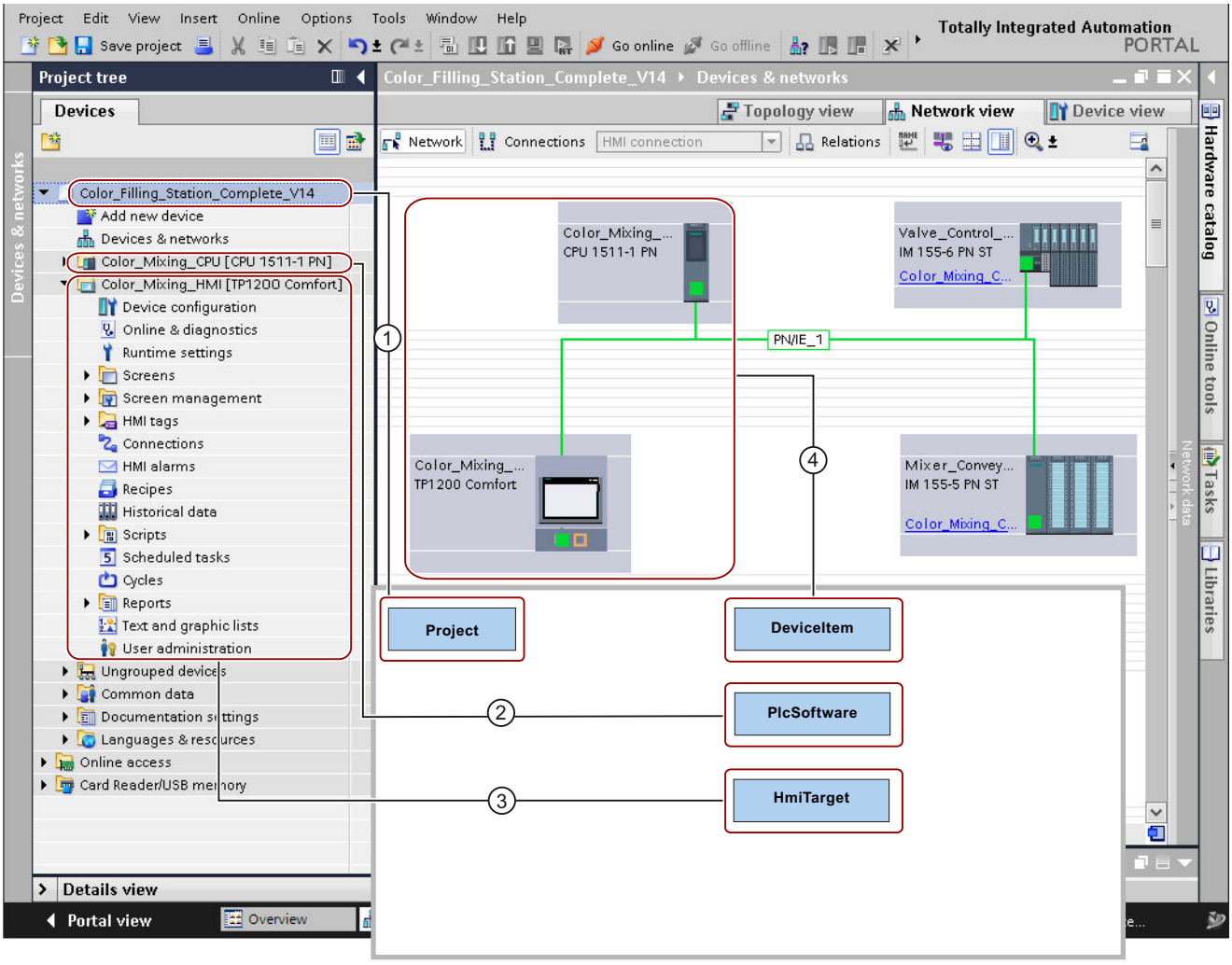


下图所示为 PlcSoftware 下的对象。


SystemBlockGroups (0..1)
PlcSystemBlockGroup
Groups (0..n)
FB
FC
OB
Blocks (0..n)
CodeBlock
<<Abstract>>
PlcBlockGroup
<<Abstract>>
Blocks (0..n)
Groups (0..n)
PlcBlock
<<Abstract>>
InstanceDB
GlobalDB
ArrayDB
BlockGroup (1)
PlcBlockSystemGroup
PlcBlockUserGroup
DataBlock
<<Abstract>>
PlcExternalSourceGroup
<<Abstract>>
ExternalSources (0..n)
Groups (0..n)
PlcExternalSource
ExternalSourceGroup (1)
PlcExternalSourceSystemGroup
PlcExternalSourceUserGroup
PlcSoftware
TypeGroup (1)
PlcTypeSystemGroup
PlcTypeUserGroup
PlcTypeSystemGroup
<<Abstract>>
Groups (0..n)
Types (0..n)
PlcType
<<Abstract>>
PlcStruct
TagTableGroup (1)
PlcTagTableSystemGroup
PlcTagTableUserGroup
PlcTag
Tags (0..n)
PlcTagTableGroup
<<Abstract>>
Groups (0..n)
TagTables(0..n)
PlcTagTable
Constants (0..n)
PlcConstant
TechnologicalObjectGroup (1)
TechnologicalInstanceDBGroup
TechnologicalObjects (0..1)
TechnologicalInstanceDBComposition
Blocks (0..n)
TechnologicalInstanceDB
(DataBlock)

TIA Portal 和 Openness 对象模型之间的关系

下图显示了对象模型和 TIA Portal 中的项目之间的关系：



① “Project”对象对应于 TIA Portal 中已打开的项目。

② “PlcSoftware”对象属于 "SoftwareBase" 类型 ④，对应于 PLC。该对象的内容对应于项目导航中的 PLC 以及对块或 PLC 变量等对象的访问权限。

③ “HmiTarget”对象属于 "SoftwareBase" 类型 ④，对应于 HMI 设备。该对象的内容对应于项目导航中的 HMI 设备以及对画面或 HMI 变量等对象的访问权限。

④ "DeviceItem" 对象对应于“设备和网络”编辑器中的对象。"DeviceItem" 类型的对象可以是一个机架或插入的模块。

9.3 导向功能的变化

简介

API 对象模型 V14 SP1 中的更改仅与已使用 V14 中 HW Config 的导向功能的用户相关。

公共 API 类型的修改

公共 API 类型	新的公共 API 类型
Siemens.Engineering.HW.IAddress	Siemens.Engineering.HW.Address
Siemens.Engineering.HW.IAddressController	Siemens.Engineering.HW.Features.AddressController
Siemens.Engineering.HW.IChannel	Siemens.Engineering.HW.Channel
Siemens.Engineering.HW.IDevice	Siemens.Engineering.HW.Device
Siemens.Engineering.HW.IDeviceItem	Siemens.Engineering.HW.DeviceItem
Siemens.Engineering.HW.IExtension	Siemens.Engineering.HW.Extensions
Siemens.Engineering.HW.IGsd	Siemens.Engineering.HW.Features.GsdObject
Siemens.Engineering.HW.IGsdDevice	Siemens.Engineering.HW.Features.GsdDevice
Siemens.Engineering.HW.IGsdDeviceItem	Siemens.Engineering.HW.Features.GsdDeviceItem
Siemens.Engineering.HW.IHardwareObject	Siemens.Engineering.HW.HardwareObject
Siemens.Engineering.HW.IHwIdentifier	Siemens.Engineering.HW.HwIdentifier
Siemens.Engineering.HW.IHwIdentifierController	Siemens.Engineering.HW.Features.HwIdentifierController
Siemens.Engineering.HW.IIoConnector	Siemens.Engineering.HW.IoConnector
Siemens.Engineering.HW.IIoController	Siemens.Engineering.HW.IoController
Siemens.Engineering.HW.IIoSystem	Siemens.Engineering.HW.IoSystem
Siemens.Engineering.HW.IInterface	Siemens.Engineering.HW.Features.NetworkInterface
Siemens.Engineering.HW.Extensions.ModuleInformationProvider	Siemens.Engineering.HW.Utilities.ModuleInformationProvider
Siemens.Engineering.HW.INode	Siemens.Engineering.HW.Node
Siemens.Engineering.HW.OPCUAExportProvider	Siemens.Engineering.HW.Utilities.OpcUaExportProvider
Siemens.Engineering.HW.IPort	Siemens.Engineering.HW.Features.NetworkPort

公共 API 类型	新的公共 API 类型
Siemens.Engineering.HW.IRole	Siemens.Engineering.HW.Features.HardwareFeature
	Siemens.Engineering.HW.Features.DeviceFeature
	Siemens.Engineering.HW.Utilities.ModuleInformationProvider
Siemens.Engineering.HW.SoftwareBase	Siemens.Engineering.HW.Software
Siemens.Engineering.HW.ISubnet	Siemens.Engineering.HW.Subnet
Siemens.Engineering.HW.ISoftwareContainer	Siemens.Engineering.HW.Features.SoftwareContainer
Siemens.Engineering.HW.ISubnetOwner	Siemens.Engineering.HW.Features.SubnetOwner

Enum 的修改

公共 API 类型	数据类型	新的公共 API 类型	数据类型
Siemens.Engineering.HW.Enums.AddressContext		Siemens.Engineering.HW.AddressContext	
Siemens.Engineering.HW.Enums.AddressIoType		Siemens.Engineering.HW.AddressIoType	
Siemens.Engineering.HW.Enums.AttachmentType		Siemens.Engineering.HW.MediumAttachmentType	
Siemens.Engineering.HW.Enums.BaudRate		Siemens.Engineering.HW.BaudRate	
Siemens.Engineering.HW.Enums.BusLoad		Siemens.Engineering.HW.CommunicationLoad	
Siemens.Engineering.HW.Enums.BusProfile		Siemens.Engineering.HW.BusProfile	
Siemens.Engineering.HW.Enums.CableLength		Siemens.Engineering.HW.CableLength	
Siemens.Engineering.HW.Enums.CableName	ulong	Siemens.Engineering.HW.CableName	long
Siemens.Engineering.HW.Enums.ChannelIoType	byte	Siemens.Engineering.HW.ChannelIoType	int

公共 API 类型	数据类型	新的公共 API 类型	数据类型
Siemens.Engineering.HW.Enums.ChannelType	byte	Siemens.Engineering.HW.ChannelType	int
Siemens.Engineering.HW.Enums.DeviceItemClassifications		Siemens.Engineering.HW.DeviceItemClassifications	
Siemens.Engineering.HW.Enums.InterfaceOperatingModes		Siemens.Engineering.HW.InterfaceOperatingModes	
Siemens.Engineering.HW.Enums.IpProtocolSelection		Siemens.Engineering.HW.IpProtocolSelection	
Siemens.Engineering.HW.Enums.MediaRedundancyRole		Siemens.Engineering.HW.MediaRedundancyRole	
Siemens.Engineering.HW.Enums.NetType		Siemens.Engineering.HW.NetType	
Siemens.Engineering.HW.Enums.ProfinetUpdateTimeMode		已删除	
Siemens.Engineering.HW.Enums.RtClass	byte	Siemens.Engineering.HW.RtClass	int
Siemens.Engineering.HW.Enums.SignalDelaySelection	byte	Siemens.Engineering.HW.SignalDelaySelection	int
Siemens.Engineering.HW.Enums.SyncRole	byte	Siemens.Engineering.HW.SyncRole	int
Siemens.Engineering.HW.Enums.TransmissionRateAndDuplex	uint	Siemens.Engineering.HW.TransmissionRateAndDuplex	int

Siemens.Engineering.HW.IoConnect 的属性值的修改

属性	数据类型	新的名称	数据类型
ProfinetUpdateTimeMode	ProfinetUpdateTimeMode	PnUpdateTimeAutoCalculation	bool
ProfinetUpdateTime		PnUpdateTime	
AdaptUpdateTime		PnUpdateTimeAdaption	
WatchdogFactor		PNWatchdogFactor	
		DeviceNumber	string

Siemens.Engineering.HW.IoController 的属性值的修改

属性	数据类型	新的名称	数据类型
		DeviceNumber	string

Siemens.Engineering.HW.Node 的属性值的修改

属性	数据类型	新的名称	数据类型
HighestAddress		已删除，仅适于子网	
TransmissionSpeed		已删除，仅适于子网	
IsoProtocolUsed		UseIsoProtocol	
IpProtocolUsed		UseIpProtocol	
RouterAddressUsed		UseRouter	
PnDeviceNameAutoGenerated		PnDeviceNameAutoGeneration	
DeviceNumber		已删除，移动到 IoConnector/IoController	

Siemens.Engineering.HW.Subnet 的属性值的修改

属性	数据类型	新的名称	数据类型
HighestAddress	byte	HighestAddress	int
CableConfiguration		PbCableConfiguration	
RepeaterCount		PbRepeaterCount	
CopperCableLength		PbCopperCableLength	
OpticalComponentCount		PbOpticalComponentCount	
OpticalCableLength		PbOpticalCableLength	
OpticalRingEnabled		PbOpticalRing	
OlmP12		PbOlmP12	
OlmG12		PbOlmP12	
OlmG12Eec		PbOlmG12Eec	
OlmG121300		PbOlmG121300	
AdditionalNetworkDevices		PbAdditionalNetworkDevices	

属性	数据类型	新的名称	数据类型
AdditionalDpMaster	byte	PbAdditionalDpMaster	int
TotalDpMaster	byte	PbTotalDpMaster	int
AdditionalPassiveDevice	byte	PbAdditionalPassiveDevice	int
TotalPassiveDevice	byte	PbTotalPassiveDevice	int
AdditionalActiveDevice	byte	PbAdditionalActiveDevice	int
TotalActiveDevice	byte	PbTotalActiveDevice	int
PbCommunicationLoad	BusLoad	PbAdditionalCommunicationLoad	CommunicationLoad
OptimizeDde		PbDirectDateExchange	
MinimizeTslot		PbMinimizeTslotForSlaveFailure	
OptimizeCableConfig		PbOptimizeCableConfiguration	
CyclicDistribution		PbCyclicDistribution	
TslotInit		PbTslotInit	
Tslot		PbTslot	
MinTsdr		PbMinTsdr	
MaxTsdr		PbMaxTsdr	
Tid1		PbTid1	
Tid2		PbTid2	
Trdy		PbTrdy	
Tset		PbTset	
Tqui		PbTqui	
Ttr		PbTtr	
TtrMs		已删除	
TtrTypical		PbTtrTypical	
TtrTypicalMs		已删除	
Watchdog		PbWatchdog	
WatchdogMs		已删除	
Gap	byte	PbGapFactor	int
RetryLimit	byte	PbRetryLimit	int
IsochronMode		IsochronousMode	
AdditionalDevice		PbAdditionalPassivDeviceForIsochronousMode	

属性	数据类型	新的名称	数据类型
TotalDevice		PbTotalPassivDeviceForIsochronousMode	
DpCycleTimeAutoCalc		DpCycleMinTimeAutoCalculation	
TiToAutoCalc		IsochronousTiToAutoCalculation	
Ti		IsochronousTi	
To		IsochronousTo	

Siemens.Engineering.Project 的属性值的修改

属性	数据类型	新的名称	数据类型
.HwExtensions		.HwUtilities	

Siemens.Engineering.HW.Baudrate 的属性值的修改

属性	数据类型	新的名称	数据类型
BaudRate.BAUD_9600		BaudRate.BAUD9600	
BaudRate.BAUD_19200		BaudRate.BAUD19200	
BaudRate.BAUD_45450		BaudRate.BAUD45450	
BaudRate.BAUD_93750		BaudRate.BAUD93750	
BaudRate.BAUD_187500		BaudRate.BAUD187500	
BaudRate.BAUD_500000		BaudRate.BAUD500000	
BaudRate.BAUD_1500000		BaudRate.BAUD1500000	
BaudRate.BAUD_3000000		BaudRate.BAUD3000000	
BaudRate.BAUD_6000000		BaudRate.BAUD6000000	
BaudRate.BAUD_12000000		BaudRate.BAUD12000000	

Siemens.Engineering.HW.CableLength 的属性值的修改

属性	数据类型	新的名称	数据类型
CableLength.Unknown		CableLength.None	
CableLength.Length_20m		CableLength.Length20m	
CableLength.Length_50m		CableLength.Length50m	
CableLength.Length_100m		CableLength.Length100m	
CableLength.Length_1000m		CableLength.Length1000m	
CableLength.Length_3000m		CableLength.Length3000m	

Siemens.Engineering.HW.ChannelIoType 的属性值的修改

属性	数据类型	新的名称	数据类型
ChannelIoType.Unknown		ChannelIoType.Complex	

Siemens.Engineering.HW.IpProtocolSelection 的属性值的修改

属性	数据类型	新的名称	数据类型
IpProtocolSelection.AddressTailoring		IpProtocolSelection.ViaIoController	

Siemens.Engineering.HW.TransmissionRateAndDuplex 的属性值的修改

属性	数据类型	新的名称	数据类型
TransmissionRateAndDuplex.Unknown		TransmissionRateAndDuplex.None	
TransmissionRateAndDuplex.TP10Mbps_HalfDuplex		TransmissionRateAndDuplex.TP10MbpsHalfDuplex	

属性	数据类型	新的名称	数据类型
TransmissionRateAndDuplex.TP10Mbps_FullDuplex		TransmissionRateAndDuplex.TP10MbpsFullDuplex	
TransmissionRateAndDuplex.AsyncFiber10Mbps_HalfDuplex		TransmissionRateAndDuplex.AsyncFiber10MbpsHalfDuplex	
TransmissionRateAndDuplex.AsyncFiber10Mbps_FullDuplex		TransmissionRateAndDuplex.AsyncFiber10MbpsFullDuplex	
TransmissionRateAndDuplex.TP100Mbps_HalfDuplex		TransmissionRateAndDuplex.TP100MbpsHalfDuplex	
TransmissionRateAndDuplex.TP100Mbps_FullDuplex		TransmissionRateAndDuplex.TP100MbpsFullDuplex	
TransmissionRateAndDuplex.FO100Mbps_FullDuplex		TransmissionRateAndDuplex.FO100MbpsFullDuplex	
TransmissionRateAndDuplex.X1000Mbps_FullDuplex		TransmissionRateAndDuplex.X1000MbpsFullDuplex	
TransmissionRateAndDuplex.FO1000Mbps_FullDuplex_LD		TransmissionRateAndDuplex.FO1000MbpsFullDuplexLD	
TransmissionRateAndDuplex.FO1000Mbps_FullDuplex		TransmissionRateAndDuplex.FO1000MbpsFullDuplex	
TransmissionRateAndDuplex.TP1000Mbps_FullDuplex		TransmissionRateAndDuplex.TP1000MbpsFullDuplex	
TransmissionRateAndDuplex.FO10000Mbps_FullDuplex		TransmissionRateAndDuplex.FO10000MbpsFullDuplex	
TransmissionRateAndDuplex.FO100Mbps_FullDuplex_LD		TransmissionRateAndDuplex.FO100MbpsFullDuplexLD	
TransmissionRateAndDuplex.POFPCF100Mbps_FullDuplex_LD		TransmissionRateAndDuplex.POFPCF100MbpsFullDuplexLD	

9.4 导出和导入变更

9.4.1 导出和导入变更

简介

为处理数组元素的注释，在 V14 SP1 中扩展了通过公共 API 导出和导入功能，因而需采用新的架构。从现在起，块接口导入和导出将处理两个架构版本。

- 对于导入：基于命名空间决定所采用的架构版本： <Sections xmlns=http://www.siemens.com/automation/Openness/SW/Interface/v2>
- 对于导出：基于项目版本决定所采用的架构版本。项目 V14 SP1 可采用版本 2，项目 V14 可采用版本 v1

9.4.2 API 的更改

生成源文件

ProgramBlocks 中已删除以下类函数：

- GenerateSourceFromBlocks
- GenerateSourceFromTypes

已添加以下类函数：

- GenerateSource 到 PlcExternalSourceSystemGroup

示例

```
// generate source for V14
var blocks = new List<PlcBlock>(){block1};
var types = new List<PlcBlock>(){udt1};
var fileInfoBlock = new FileInfo("D:\Export\Block.scl");
var fileInfoType = new FileInfo("D:\Export\Type.udt");

PlcBlocksSystemGroup blocksGroup = ...;
blocksGroup.GenerateSourceFromBlocks(blocks, fileInfo);
PlcTypesSystemGroup plcDataTypesGroup = ...;
plcDataTypesGroup.GenerateSourceFromTypes(types, fileInfo);

//generate source as of V14 SP1
var blocks = new List<PlcBlock>(){block1};
var types = new List<PlcBlock>(){udt1};
var fileInfoBlock = new FileInfo("D:\Export\Blocks.scl");
var fileInfoType = new FileInfo("D:\Export\Type.udt");

PlcExternalSourceSystemGroup externalSourceGroup = plc.ExternalSourceGroup;
externalSourceGroup.GenerateSource(blocks, fileInfoBlock);
externalSourceGroup.GenerateSource(types, fileInfoType);
```

9.4.3 架构扩展

注释及起始值的架构扩展

注释及起始值存储于名为“Subelement”的新元素中，该元素可通过“Path"”属性引用数组元素。

Subelement 包含所引用数组元素的起始值和注释。新架构中移除了 StartValue 的“Path”属性。

“Subelement”的架构定义：

```
<xs:element name="Subelement" type="Subelement_T"/>
  <xs:complexType name="Subelement_T">
      <xs:sequence>
          <xs:choice minOccurs="0" maxOccurs="unbounded">
              <xs:element ref="StartValue"/>
              <xs:element ref="Comment"/>
          </xs:choice>
      </xs:sequence>
      <xs:attribute name="Path" type="IndexPath_TP"/>
  </xs:complexType>
```

扩展成员类型：

```
<xs:complexType name="Member_T">
       <xs:sequence>
           <xs:element ref="AttributeList" minOccurs="0" maxOccurs="1"/>
           <xs:choice minOccurs="0" maxOccurs="unbounded">
               <xs:element ref="Member"/>
               <xs:element ref="Sections"/>
               <xs:element ref="StartValue"/>
               <xs:element ref="Comment"/>
               <xs:element ref="Subelement"/>
           </xs:choice>
       </xs:sequence>
</xs:complexType>
```

示例：

将注释及起始值存储在简单数组中：

```
<Member Name="Static_1" Datatype="Array[0..1] of Bool">
    <Comment>
        <MultiLanguageText Lang="de-DE">comment for array</MultiLanguageText>
    </Comment>
    <Subelement Path="0">
        <StartValue>true</StartValue>
        <Comment>
            <MultiLanguageText Lang="de-DE">comment for array element 0</MultiLanguageText>
        </Comment>
    </Subelement>
    <Subelement Path="1">
        <StartValue>true</StartValue>
        <Comment>
            <MultiLanguageText Lang="de-DE">comment for array element 1</MultiLanguageText>
        </Comment>
    </Subelement>
</Member>
```

将注释及起始值存储在 UDT 数组中：

```
<Member Name="Static_1" Datatype="Array[0..1] of &quot;User_data_type_1&quot;">
    <Comment>
        <MultiLanguageText Lang="de-DE">comment for array</MultiLanguageText>
    </Comment>
    <Subelement Path="0">
        <Comment>
            <MultiLanguageText Lang="de-DE">cmt array  0</MultiLanguageText>
        </Comment>
    </Subelement>
    <Sections>
        <Section Name="None">
            <Member Name="Element_1" Datatype="Bool">
                <Subelement Path="0">
                    <StartValue>true</StartValue>
                    <Comment>
                        <MultiLanguageText Lang="de-DE">comment for element 0</MultiLanguageText>
                    </Comment>
                </Subelement>
                <Subelement Path="1">
                    <StartValue>true</StartValue>
                    <Comment>
                        <MultiLanguageText Lang="de-DE">comment for element 1</MultiLanguageText>
                    </Comment>
                </Subelement>
            </Member>
            <Member Name="Element_2" Datatype="Struct">
                <Member Name="Element_1" Datatype="Int">
                    <Subelement Path="0">
                        <StartValue>11</StartValue>
                        <Comment>
                            <MultiLanguageText Lang="de-DE">comment for element 0</MultiLanguageText>
                        </Comment>
                    </Subelement>
                </Member>
            </Member>
        </Section>
    </Sections>
</Member>
```

将注释及起始值存储在结构数组中：

```
<Member Name="Static_1" Datatype="Array[0..1] of Struct">
    <Member Name="Static_1" Datatype="Int">
        <Subelement Path="0">
            <StartValue>11</StartValue>
            <Comment>
                <MultiLanguageText Lang="de-DE">comment for int elem</MultiLanguageText>
            </Comment>
        </Subelement>
    </Member>
    <Member Name="Static_2" Datatype="Bool">
        <Subelement Path="1">
            <StartValue>true</StartValue>
            <Comment>
                <MultiLanguageText Lang="de-DE">comment for bool elem</MultiLanguageText>
            </Comment>
        </Subelement>
    </Member>
</Member>
```

9.4.4 架构更改

SW.PlcBlocks.Access.xsd 中的 Access 节点

Access 节点的 Type 属性已移至以下范围的 Access 子节点

- AbsoluteOffset（必选）
- Address（可选）

```
<StlStatement UId="22">
  <StlToken Text="L" />
  <Access Scope="Address">
    <Address Area="Local" Type="Word" BitOffset="80" />
  </Access>
</StlStatement>
```

Constant 的 Type 属性已为新的 ConstantType 子节点所取代。

```
<Access Scope="LocalConstant">
  <IntegerAttribute Name="NumBLs" Informative="true">5</IntegerAttribute>
  <Constant Name="LocalConstant_A">
    <ConstantType Informative="true">Int</ConstantType>
    <ConstantValue Informative="true">10</ConstantValue>
    <StringAttribute Name="Format" Informative="true">Dec_signed</StringAttribute>
  </Constant>
</Access>
```

Access 中的 Scope 属性值已重命名为 TypedConstant，前提是 ConstantValue 包含类型限定值（例如：int#10）。

Constant 不具有 Type 属性，前提是 ConstantValue 包含类型限定值（例如：int#10）。

Scope 为 LocalVariable，则本地变量不包含 Address 节点。

如果 Access 嵌套于另一个任意级别的 Access 中，则只有外部 Access 必须具有 UId。

SW.PlcBlocks.Access.xsd 中的 Address 节点

Address 节点的 BitOffset 属性变为可选项。

导出绝对访问的声明已更改，如下表所示：

V14 SP1 及以上版本的区域	类型	块编号	位偏移	示例
DB	Block_DB	必须	禁止	OPN %DB12
DB	无序	存在	必须	%DB100.DBX10.3
DB	无序	不存在	必须	%DB100.DBX10.3

V14 SP1 及以上版本的区域	类型	块编号	位偏移	示例
L	无序	禁止	必须	%LW10.0
I Q M	无序	禁止	必须	%I0.0 %Q0.0 %M0.0
T C	无序	禁止	必须	%T0 %C1
Block_FC Block_FB	Block_FC Block_FB	必须	禁止	调用 %FB4，%DB5 Input_1 := %FC10 调用 %FB4，%DB5 Input_2 := %FB11
PeripheryInput	无序	禁止	必须	
Periphery Output	无序	禁止	必须	

SW.PlcBlocks.Access.xsd 中的 Area 节点

Area 节点已获得简化后的枚举列表：

- LocalC 和 LocalN 均为 Local
- DBc、DBv、DBr 已删除。

SW.PlcBlocks.Access.xsd 中的 CallInfo 节点

CallInfo 节点的 Name 属性变为可选项。

CallInfo 节点的 BlockType 属性变为必须项。

+2.2.5 用户块调用

SW.PlcBlocks.Access.xsd 中的 Constant 节点

Constant 节点通过 minOccurs=0 引用 CostantType 节点

Constant 节点不再引用 IntegerAttribute 节点

SW.PlcBlocks.Access.xsd 中的 ConstantValue 节点

ConstantValue 节点获得资料属性

SW.PlcBlocks.Access.xsd 中的 Instruction 节点

Instruction 节点通过 minOccurs=0 引用 Acces 节点

Instruction 节点已删除 Section、Type 和 TemplateReference 属性。

SW.PlcBlocks.Access.xsd 中的 Parameter 节点

Parameter 节点的 SectionName 属性变为可选项。

SW.PlcBlocks.Access.xsd 中 Scope 的值。

Scope 的枚举列表已扩展以下内容：

- TypedConstant
- AddressConstant
- LiteralConstant
- AlarmConstant
- Address
- Statusword
- Expression
- Call
- CallWithType

SW.PlcBlocks.Access.xsd 中的 Statusword 节点

Statusword 的枚举列表已扩展以下内容：

- STW

SW.PlcBlocks.Access.xsd 中的 ConstantType 节点

新节点 ConstantType 已与可选属性 Informative 一起引入。

SW.PlcBlocks.LADFBD.xsd 中的 CallRef 节点

CallRef 节点重命名为 Call 且删除 BooleanAttribute 子节点。

SW.PlcBlocks.LADFBD.xsd 中的 InstructionRef 节点

InstructionRef 节点已为 Part 节点所取代

SW.PlcBlocks.LADFBD.xsd 中的 Part 节点

新节点 ConstantType 已引入并取代 InstructionRef 节点。

- 属性：名称和版本
- 子节点：Instruction 子节点是现有 Equation 的新选择
- 不具有 BooleanAttribute 子节点和 Gate 属性

SW.PlcBlocks.LADFBD.xsd 中的 Wire 节点

Wire 节点的 Name 属性已移除。

SW.PlcBlocks.LADFBD.xsd 中的 TemplateReference 节点

TemplateReference 节点已删除。

SW.PlcBlocks.STL.xsd 中的 StatementList 节点

StatementList (STL_TE) 的枚举列表：

- L_STW 已移除。
- T_STW 已移除。

9.4.5 行为更改

绝对访问

在 V14 中，已针对大多数组合中止了绝对访问的导入。自 V14 SP1 起，绝对访问的导入将适用于以下区域：

- 输入
- 输出
- 内存
- 计时器（若 PLC 支持）

- 计数器（若 PLC 支持）
- DB
- DI

如果同时使用符号访问和绝对访问且未遭到架构或节点类型验证的拒绝，则导入只会在成功解析两种访问信息后才可成功执行。若符号访问所指向的信息与绝对访问的信息不同，则将拒绝导入。

```
<Access Scope="Address">
    <Address Area="Memory" Type="Word" BitOffset="0" />
</Access>


<CallInfo Name="Block_1" BlockType="FC">
    <Parameter Name="Input_1" Section="Input" Type="Int" />
    <Parameter Name="Input_1" Section="Input" Type="Int" />
<!-- Import will be aborted because parameter name 'Input_1' is used more than once -->
    <Parameter Name="Output_1" Section="NotExisting" Type="Int" />
<!-- Import will be aborted because the section 'NotExisting' can not be used. -->
    <Parameter Name="ENO" Section="Output" Type="Int" />
<!-- Import will be aborted because the parameter name 'ENO' is restricted and can not be used. -->
</CallInfo>
```

间接 DB 访问

自 V14 SP1 起，仅在提供“偏移”、“类型”和“符号”后才可导入间接 DB 访问。

```
...
    <Access Scope="LocalVariable" UId="21">
        <Symbol>
            <Component Name="Output_3" />
            <AbsoluteOffset BitOffset="16" Type="Word" />
        </Symbol>
    </Access>
...
```

本地访问的符号和绝对信息

导入“符号访问”时，所提供的所有可能的“绝对访问信息”在未标记为“信息”的情况下均有效。自 V14 SP1 起，若绝对信息不匹配，则将中止导入。

块接口限制

在 V14SP1 中存在多个限制。块接口编辑器的用户十分了解这些限制。块接口编辑器通过添加或增加“_1”来重命名某个参数时，OPNS 导入将中止。

例如，以下为有效限制：

- 复制参数名称
- 段名称错误。包括 FB 块的“返回-段”
- 限制词

导入时排序段

如果导入时所调用的块不存在，则调用侧的接口定义将用于显示所调用的用户块。在 V14 SP1 中，各个段将进行排序，以便在所调用块的块接口中进行显示，前提是所调用块已存在且具有相同参数。

所导入参数的段顺序为：

- 输入
- 输出

以下 STL xml 示例

```
<StlStatement UId="21">
<StlToken Text="CALL" />
    <Access Scope="Call">
        <CallInfo Name="Block_2" BlockType="FC">
            <Parameter Name="Output_1" Section="Output" Type="Int">
                <Access Scope="GlobalVariable">
                    <Symbol>
                        <Component Name="Tag_3" />
                    </Symbol>
                </Access>
            </Parameter>
            <Parameter Name="Input_1" Section="Input" Type="Int">
                <Access Scope="GlobalVariable">
                    <Symbol>
                        <Component Name="Tag_1" />
                    </Symbol>
                </Access>
            </Parameter>
            <Parameter Name="Output_2" Section="Output" Type="Int">
                <Access Scope="GlobalVariable">
                    <Symbol>
                        <Component Name="Tag_4" />
                    </Symbol>
                </Access>
            </Parameter>
            <Parameter Name="Input_2" Section="Input" Type="Int">
                <Access Scope="GlobalVariable">
                    <Symbol>
                        <Component Name="Tag_2" />
                    </Symbol>
                </Access>
            </Parameter>
        </CallInfo>
    </Access>
</StlStatement>
```

将得到结果

```
CALL  "Block_2"
    Input_1  :="Tag_1"
    Input_2  :="Tag_2"
    Output_1 :="Tag_3"
    Output_2 :="Tag_4"
```

唯一的用户块调用名称

在 TIA Portal 中，名称必须是唯一的。例如，变量名称不得与块名称相同。对于公共 API XML 导入，这意味着若 XML 包含一个用户块调用且导入时不存在所调用的块，则这一所调用块的名称必须不同于项目中的所有现有名称。若这一所调用块的名称不唯一，则导入将中止。

在以下示例中，导入将中止，因为所调用块的名称“Tag_1”已用于一个变量表。

```
...
    <SW.Tags.PlcTag ID="1" CompositionName="Tags">
        <AttributeList>
            <DataTypeName>Int</DataTypeName>
            <LogicalAddress>%MW2</LogicalAddress>
            <Name>Tag_1</Name>
        </AttributeList>
    </SW.Tags.PlcTag>
...
...
    <StlStatement UId="21">
        <StlToken Text="CALL" />
        <Access Scope="Call">
            <CallInfo Name="Tag_1" BlockType="FC">
                <Parameter Name="Input_1" Section="Input" Type="Int">
                    <Access Scope="GlobalVariable">
                        <Symbol>
                            <Component Name="Tag_1" />
                        </Symbol>
                    </Access>
                </Parameter>
...
```

在以下示例中，导入将中止，因为两个参数具有相同的名称“Input1”。

```
<StlStatement UId="22">
    <StlToken Text="CALL" />
    <Access Scope="Call">
        <CallInfo Name="Block_1" BlockType="FB">
            <Instance Scope="GlobalVariable">
                <Component Name="Block_1_DB" />
            </Instance>
            <Parameter Name="Input1" Section="Input" Type="Int">
                <Access Scope="GlobalVariable">
                    <Symbol>
                        <Component Name="Tag_9" />
                    </Symbol>
                </Access>
            </Parameter>
            <Parameter Name="Input1" Section="Input" Type="Time">
                <Access Scope="TypedConstant">
                    <Constant>
                        <ConstantValue>T#1s</ConstantValue>
                    </Constant>
                </Access>
            </Parameter>
        </CallInfo>
    </Access>
</StlStatement>
```

库块调用

已导入的 XML 可能包含用户块调用。可通过名称识别这些用户块。

用户块还可调用库元素。这些库元素可作为“库块调用”生成。由于库块所用命名空间与用户块相同，因此，通过名称完成的用户块调用导入可调用库块的执行。

在 V14 SP1 之前，导入试图映射用户块调用和指令块调用之间的参数。导入有时会中止，有时会删除所有未匹配的参数。

自 V14 SP1 起，用户块调用仍将能够找到库块，但调用将不会生效。

块类型不匹配

自 V14 SP1 起，如果 XML 包含“Block_1”的用户块调用且参数多于项目中相应的 FC，则导入将定义一个与 XML 用户块调用匹配的新调用块接口。下一个程序块编译将尝试更新调用。

新的常量范围

在 V14SP1 中，已为常量创建多个新范围。仅在 xml 中的值与常量范围匹配时，导入才会成功。若为某个常量提供的信息与该现有常量未完全匹配，则导入将中止。

```
...
    <Access Scope="LiteralConstant">
        <Constant>
            <ConstantType>Int</ConstantType>
            <ConstantValue>16#0000_0001</ConstantValue>
        </Constant>
    </Access>
...
    <Access Scope="TypedConstant">
        <Constant>
            <ConstantValue>Int#10</ConstantValue>
        </Constant>
    </Access>
...
    <Access Scope="LiteralConstant">
        <Constant>
            <ConstantType>Int</ConstantType>
            <ConstantValue>10</ConstantValue>
        </Constant>
    </Access>
...
    <Access Scope="GlobalConstant">
        <Constant Name="Constant_1" />
    </Access>
...
    <Access Scope="LocalConstant">
        <Constant Name="Constant_1" />
    </Access>
...
    <Access Scope="AddressConstant">
        <Constant Name="Tag_1" />
    </Access>
...
    <Access Scope="AlarmConstant">
        <Constant>
            <ConstantType>C_Alarm</ConstantType>
            <ConstantValue>16#0000_0001</ConstantValue>
        </Constant>
    </Access>
...
```

指令版本标注

自 V14 SP1 起，仅可导入 PLC 上可用于导入的指令版本。若 xml 中无标注的指令版本，则将采用 PLC 中所选的版本。在 LAD 和 FBD 中，一些表示为指令的元素不会采用版本化。仅在不存在版本时才可导入这些元素。

```
...
    <Part Name="MIN" Version="1.0" UId="27" DisabledENO="false">
        <TemplateValue Name="card" Type="Cardinality">2</TemplateValue>
        <TemplateValue Name="value_type" Type="Type">Int</TemplateValue>
    </Part>
    <Part Name="MIN" UId="28" DisabledENO="false">
        <TemplateValue Name="card" Type="Cardinality">2</TemplateValue>
        <TemplateValue Name="value_type" Type="Type">Int</TemplateValue>
    </Part>
...
```

禁用 ENO

1200 和 1500 PLC 使用“禁用 ENO”功能来禁用耗时的 ENO 连接状态计算。

自 V14 SP1 起，DisabledENO 标志仅可在支持该功能的 PLC 上导入。

```
...
    <Part Name="Add" UId="24" DisabledENO="false">
        <TemplateValue Name="Card" Type="Cardinality">2</TemplateValue>
        <TemplateValue Name="SrcType" Type="Type">Int</TemplateValue>
    </Part>
...
```

绝对 L-Stack 访问的类型验证

自 V14 SP1 起，若类型无法使用或映射，则将中止导入。

索引标识验证

在定义“内存符号访问”的情况下无法使用索引访问。例如，本地访问、全局访问、间接访问。

若将文字常量作为索引使用，有符号和无符号整数类型将变为 Dint。自 V14 SP1 起，若提供所涉范围之外的类型，则导入将中止。

所有索引访问均已选中，不论访问类型是否可用作“索引访问”。自 V14 SP1 起，若定义的索引访问无法使用，则将中止导入。

元素排序

自 V14 SP1 起，LAD 和 FBD 中的元素将在导出期间自动按“代码生成顺序”排序。在某些十分罕见的情况下，已导出的 XML 无法再进行导入。在这些情况下，XML 必须进行调整或相应网络必须删除并重新编程。但连接和引用的顺序仍不可靠。

报警常量

在 V14 SP1 中，编译将检查是否已扩展有效的报警常量。由于 V14 中导入的 xml 具有无效报警常量，因此 V14 SP1 中可能会出现项目无法编译的情况。在这种情况下，在 LAD/FDB 编辑器中打开相关网络并删除报警实际操作数。该编辑器将自动重新创建一个有效的报警常量。

```
<FlgNet>
    <Parts>
        <Access Scope="AlarmConstant" UId="21">
            <Constant>
                <ConstantType>C_Alarm</ConstantType>
                <ConstantValue>16#0000_0002</ConstantValue>
            </Constant>
        </Access>
        <Call UId="22">
            <CallInfo Name="Block_1" BlockType="FB">
                <Instance UId="23" Scope="GlobalVariable">
                    <Component Name="Block_1_DB" />
                </Instance>
                <Parameter Name="Input_1" Section="Input" Type="C_Alarm" />
            </CallInfo>
        </Call>
    </Parts>
    <Wires>
        <Wire UId="24">
            <Powerrail />
            <NameCon UId="22" Name="en" />
        </Wire>
        <Wire UId="25">
            <IdentCon UId="21" />
            <NameCon UId="22" Name="Input_1" />
        </Wire>
    </Wires>
</FlgNet>
```

用户块实例限制和指令

在 V14 中，可导入带有实例的用户 FC 块调用，有时甚至可以编译这些调用。

自 V14 SP1 起，仅在支持实例的情况下才可导入实例：FC 用户块调用时，带有实例的现有项目和指令可能无法再编译。在这种情况下，必须删除调用并重新编程。试图进行调用更新或自动修复的任何操作均会失败。

EnEno 可见

在 V14 中，“InstructionRef”的 EN 和 ENO 连接已可用或与 ENENO 标志无关。

自 V14SP1 起，使用了基于元素进行导入期间的 OPNS 及 EN 和 ENO 连接。因此，可自动侦测到不同的 EN 和 ENO 连接使用情况。最为可能的情况是，只有 IEC 计时器和 IEC 计数器框可显示某些问题。

UId 分配

在 V14 SP1 中，部件、访问和连接的 UId 分配发生了更改。一个编译单元内的语句、CallInfo 和操作数的 UId 必须唯一。就 TIA portal 而言，XML 中的 UId 就像密钥一样，除了识别元素外，无任何其它意义。

检查字符串

导入以下内容期间，会针对 Name 执行更为严格的引号、代理字符和控制字符检查

- IntegerAttribute
- StringAttribute
- DateAttribute
- AutomaticTyped
- Component
- Invisible
- Label
- NameCon
- Negated
- TemplateValue
- CallInfo
- Instruction
- Parameter
- Part
- Step

导入以下内容期间，会执行更为严格的代理字符和控制字符检查

- 块和网络的名称
- LineComment 文本
- 字符串常量（String、WString、Char、Wchar 类型）

导入以下内容期间，会执行更为严格的代理字符和控制字符（包括制表符和换行）检查

- 块和网络的注释
- 字符串属性
- 定义多语言文本的节点，例如，Alarmtext、Comments
- Token 文本

不区分大小写的模板运算和参数

自 V14 SP1 起，不区分大小写的指令和调用模板运算或指令参数将会导入并进行自动更正。

将导入以下代码且错误值“Eq”将更正为“EQ”，错误参数“iN1”将更正为“IN1”；

```
<StlStatement UId="22">
    <StlToken Text="CALL" />
    <Access Scope="Call">
        <Instruction Name="CompType">
            <TemplateValue Name="src_type" Type="Type">Variant</TemplateValue>
            <TemplateValue Name="relation" Type="Operation">Eq</TemplateValue>
            <Parameter Name="iN1">
                <Access Scope="GlobalVariable">
                    <Symbol>
                        <Component Name="Tag_12" />
                    </Symbol>
                </Access>
            </Parameter>
            ...
        </Instruction>
    </Access>
</StlStatement>
```

调用中使用的多实例

自 V14 SP1 起，若在不存在的调用中使用多实例，则导入将中止。

以下代码显示了在接口段正确定义多实例的 xml 示例：

```
<SW.Blocks.FB ID="0">
    <AttributeList>
        <Interface>
            <Sections xmlns="http://www.siemens.com/automation/Openness/SW/Interface/v2">
                <Section Name="Input" />
                <Section Name="Output" />
                <Section Name="InOut" />
                <Section Name="Static">
                    <!-- The next line must be present if multiinstance is used in code-->
                    <Member Name="Static_1" Datatype="&quot;Block_2&quot;" />
                </Section>
    ....
    <StlStatement UId="22">
        <StlToken Text="CALL" />
        <Access Scope="Call">
            <CallInfo BlockType="FB">
                <!-- Multiinstace usage-->
                <Instance Scope="LocalVariable">
                    <Component Name="Static_1" />
                </Instance>
                <Parameter Name="Input_1" Section="Input" Type="Int">
                    <Access Scope="GlobalVariable">
                        <Symbol>
                            <Component Name="Tag_9" />
                        </Symbol>
                    </Access>
                </Parameter>
            </CallInfo>
        </Access>
    </StlStatement>
```

STL 中的模板基数

在 STL 中，各个指令的模板基数具有一个固定的默认值，该值为唯一有效值。自 V14 SP1 起，若为基数使用其它值，则导入将中止。

导入间接访问

自 V14 SP1 起，间接访问仅在可编译的情况下才能导入。

```
<StlStatement UId="22">
    <StlToken Text="L" />
    <Access Scope="Address">
        <Indirect Width="Word" Area="Memory">
            <Access Scope="LocalVariable">
                <Symbol>
                    <Component Name="Temp_1" />
                </Symbol>
            </Access>
        </Indirect>
    </Access>
</StlStatement>
```

导入状态字

自 V14 SP1 起，状态字仅在得到语句支持的情况下才能导入。

- L - 支持的状态字：STW
- T - 支持的状态字：STW
- A - 支持的状态字：BR、OV、OS、EQ、NE、GT、Lt、GE、LE、U0、NU
- AN - 支持的状态字：BR、OV、OS、EQ、NE、GT、Lt、GE、LE、U0、NU
- O - 支持的状态字：BR、OV、OS、EQ、NE、GT、Lt、GE、LE、U0、NU
- ON - 支持的状态字：BR、OV、OS、EQ、NE、GT、Lt、GE、LE、U0、NU
- X - 支持的状态字：BR、OV、OS、EQ、NE、GT、Lt、GE、LE、U0、NU
- XN - 支持的状态字：BR、OV、OS、EQ、NE、GT、Lt、GE、LE、U0、NU

说明

大多数状态字仅在 300 和 400 PLC 上有用。

空语句

如果某个语句不具有节点 <StlStatement/>，则导入将中止。如果出现空语句，请添加 <StlToken Text="Empty_Line" /> 节点。

如果某个空语句含有注释，则导入将中止。对于仅含有注释的语句，请使用 <StlToken Text="COMMENT" />。

```
<!-- Declaration of an empty statement -->
<StlStatement UId="23">
    <StlToken Text="EMPTY_LINE" />
</StlStatement>

<!-- Declaration of a statement with only comments-->
<StlStatement UId="22">
    <LineComment>
        <Text>Comment number 1</Text>
    </LineComment>
    <StlToken Text="COMMENT" />
</StlStatement>
```

9.4.6 块属性更改

常规属性的更改

典型 OB 的 AutoNumber 已获得新的默认值（假）

HeaderVersion 已获得新类型 System.Version（而非字符串）

IsKnowHowProtected 也适用于用户自定义数据类型

ILibraryTypeInstance.ConnectedVersion、ILibraryTypeInstance、Dependencies、ILibraryTypeInstance.Dependents 已从常规属性表中删除，因为它们无法在 XML 中导出，也无法通过 API 访问。

MemoryLayout 获得新的默认值：classic PLC 中的 Standard 和 plus PLC 中的 Optimized：

Number 适用于用户自定义数据类型，可在 XML 中表示，还可通过 API 访问

特定属性的更改

如果 IDBofUDT 属于系统库元素，则其 IsOnlyStoredInLoadMemory 和 IsWriteProtectedInAS 均为只读。

OfSystemLibElement 和 OfSystemLibVersion 会从常规属性重置为特定属性。

OfSystemLibVersion 已获得新类型 System.Version（而非字符串）

仅当符合以下条件时，函数和函数块中的 ParameterPassing 才保持为读写模式：

- ProgrammingLanguage 为 STL
- MemoryLayout 为标准模式且
- 接口为空

GraphVersion 已获得新类型 System.Version（而非字符串）

自 Graph 版本 V4 起，针对写入 Graph 的 FB 引入名为 ExtensionBlockName 的新属性

自 Graph 版本 V4 起，针对写入 Graph 的 FB 引入名为 InvalidValuesAcquisition 的新属性

针对代码块引入名为 IsWriteProtected 的新属性

DownloadWithoutReinit 变为只读且也适用于 IDBofFBs

IDBofFBs 的 Supervisions 变为只读。

枚举的更改

ProgrammingLanguage 的枚举值更改如下：

- 引入新的枚举值 F_CALL
- 针对“运动”(Motion) 工艺对象引入新的枚举值 Motion_DB
- GRAPH_SEQUENCE、GRAPH_ACTIONS、GRAPH_ADDINFOS 已从枚举中删除并用 GRAPH 替代。

BlockType 的枚举值更改如下：

- 值 OB、FC、DB、SFC 已删除，因为此枚举仅用于 InstanceOfType 属性

9.5 V14 的更改

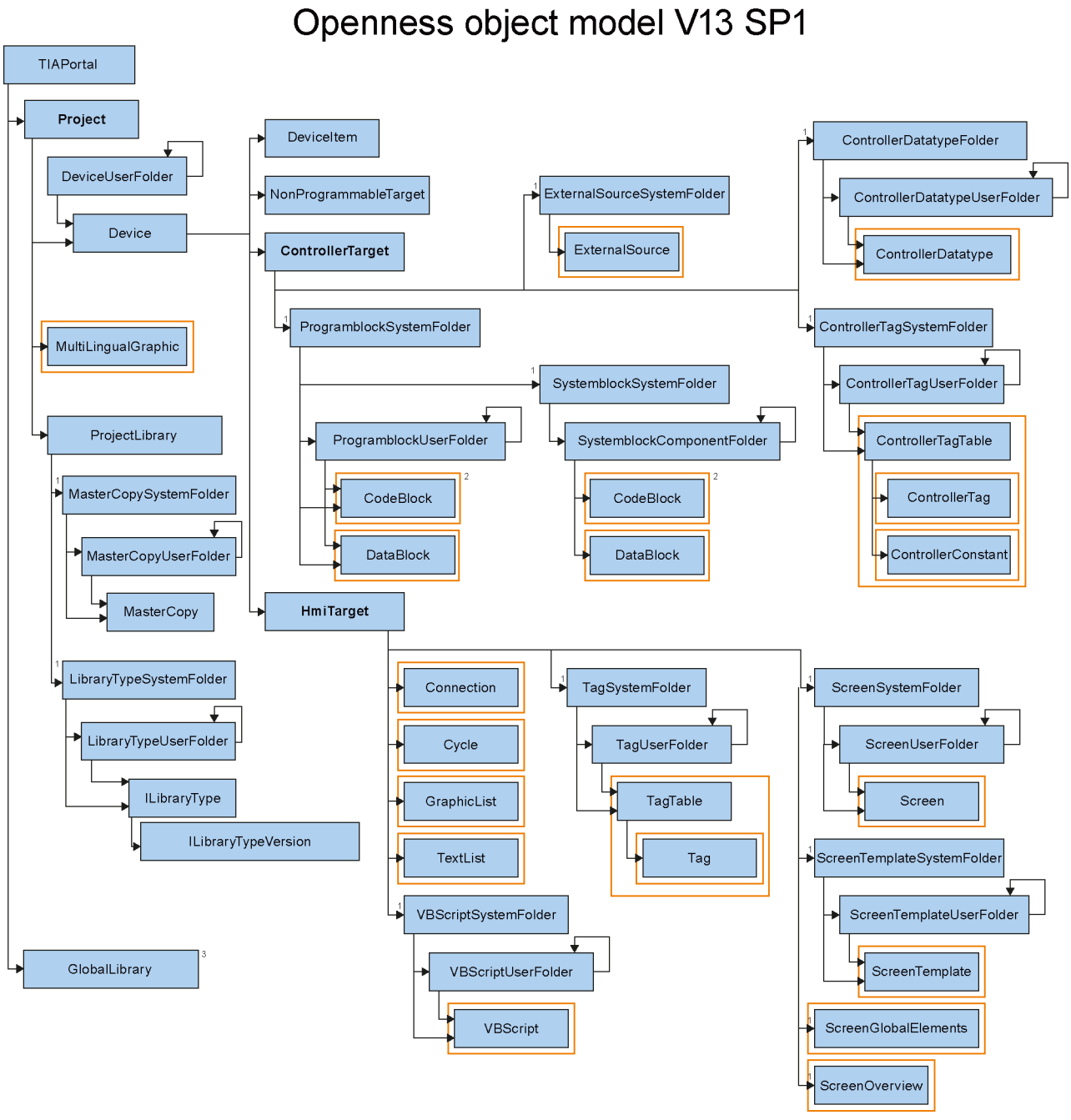
9.5.1 对象模型的主要变化

TIA Portal Openness V13 SP1 及更低版本的对象模型

为了比较 Openness 的旧对象模型和新对象模型，下图给出了 TIA Portal V13 SP1 的对象模型。

说明

该图中显示的对象模型已过时，有关 Openness V14 SP1 的对象模型信息，请参见 “Openness 对象模型 (页 45)”


Openness object model V13 SP1
TIAPortal
Project
DeviceUserFolder
Device
DeviceItem
NonProgrammableTarget
ControllerTarget
ExternalSourceSystemFolder
ExternalSource
ControllerDatatypeFolder
ControllerDatatypeUserFolder
ControllerDatatype
MultiLingualGraphic
ProgramblockSystemFolder
ControllerTagSystemFolder
ControllerTagUserFolder
ControllerTagTable
ControllerTag
ControllerConstant
SystemblockSystemFolder
ProjectLibrary
ProgramblockUserFolder
SystemblockComponentFolder
CodeBlock
DataBlock
MasterCopySystemFolder
MasterCopyUserFolder
MasterCopy
HmiTarget
LibraryTypeSystemFolder
LibraryTypeUserFolder
ILibraryType
ILibraryTypeVersion
Connection
Cycle
GraphicList
TextList
VBScriptSystemFolder
VBScriptUserFolder
VBScript
TagSystemFolder
TagUserFolder
TagTable
Tag
ScreenSystemFolder
ScreenUserFolder
Screen
ScreenTemplateSystemFolder
ScreenTemplateUserFolder
ScreenTemplate
ScreenGlobalElements
ScreenOverview
GlobalLibrary

9.5.2 将应用程序更新到 Openness V14 前

应用程序

将应用程序更新到 Openness V14 前，更改以下设置：

1. 通过添加以下 Openness API，修改对 V14 API 的引用：
   - `Siemens.Engineering`
   - `Siemens.Engfineering.Hmi`
2. 将 Visual Studio 的 .Net framework 更改为版本 4.6.1
3. 通过修改新的 TIA Portal 安装路径，更新程序集解析方法。
   - 如果从注册表进行评估，则可按照下面的示例修改新的关键字：
     `"HKEY_LOCAL_MACHINE\SOFTWARE\Siemens\Automation\_InstalledSW\TIAP14\TIA_Opns\..."`
   - 如果使用应用程序组态文件，则将路径修改为新的安装路径。

9.5.3 主要字符串变更

简介

Openness V14 中做出了以下更改，可能会影响现有应用程序：

更改	所需程序代码调整

编译方法已更改。	可按照下面的示例更改编译方式：
	• Openness V13 SP1（废弃）：
	`controllerTarget.Compile(CompilerOptions.Software, BuildOptions.Rebuild);`
	• Openness V14:
	`plcSoftware.GetService<ICompilable>().Compile();`

添加了新的名称空间。	1. 添加以下名称空间语句：
	`Siemens.Engineering.SW.Blocks;`
	`Siemens.Engineering.SW.ExternalSources;`
	`Siemens.Engineering.SW.Tags;`
	`Siemens.Engineering.SW.Types;`
	2. 删除 `using ControllerTarget = Siemens.Engineering.HW.ControllerTarget` 名称空间语句。
	3. 编译此应用程序。

`ControllerTarget` 已替换为 `PlcSoftware`，在某些情况下，功能已发生更改。	1. 检查文档中属于应用程序功能的代码示例。
	2. 按照下面的示例更新 Openness 应用程序的程序代码。
	– Openness V13 SP1（废弃）：
	`ControllerTarget controllerTarget = deviceItem as ControllerTarget`
	– Openness V14:
	`PlcSoftware plcSoftware = deviceItem.GetService<SoftwareContainer>().Software as PlcSoftware`
	3. 编译此应用程序。

更改	所需程序代码调整

已替换对象。	1. 搜索并替换以下对象：
	`DeviceUserFolderAggregation = DeviceUserGroupComposition`
	`DeviceFolders = DeviceGroups`
	`DeviceUserFolder = DeviceUserGroup`
	`ProgramblockSystemFolder = PlcBlockSystemGroup`
	`ProgramblockUserFolder = PlcBlockUserGroup`
	`IBlock = PlcBlock`
	`ControllerDatatypeSystemFolder = PlcTypeSystemGroup`
	`ControllerDatatypeUserFolder = PlcTypeUserGroup`
	`ControllerDatatype = PlcType`
	`ControllerTagSystemFolder = PlcTagTableSystemGroup`
	`ControllerTagUserFolder = PlcTagTableUserGroup`
	`ControllerTagTable = PlcTagTable`
	`ControllerTag = PlcTag`
	`ControllerConstant = PlcConstant`
	`ExternalSourceSystemFolder = PlcExternalSourceSystemGroup`
	`ExternalSource = PlcExternalSource`
	`IOnline = OnlineProvider`
	`ILibraryType = LibraryType`
	2. 编译此应用程序。

更改	所需程序代码调整

聚合已被替换为组合。	1. 按照以下示例，将代码的各个 Aggregation 都替换为 Composition:
	`ProjectAggregation = ProjectComposition`
	`IDeviceAggregation = IDeviceComposition`
	`TagTableAggregation = TagTableComposition`
	`CycleAggregation = CycleComposition`
	`GraphicListAggregation = GraphicListComposition`
	`TextListAggregation = TextListComposition`
	`ConnectionAggregation = ConnectionComposition`
	`MultiLingualGraphicAggregation = MultiLingualGraphicComposition`
	`UpdateCheckResultMessageAggregation = UpdateCheckResultMessageComposition`
	2. 编译此应用程序。

在每种关系中，文件夹均已被替换为组（HMI 设备除外）。	1. 将程序代码中的各个 Folder 都替换为 Group（与 HMI 设备相关的代码部分除外）。 2. 编译此应用程序。
GetAttributeNames 方法已被替换为 GetAttributeInfos 方法。	1. 使用 `IList<EngineeringAttributeInfo> IEngineeringObject.GetAttributeInfos(AttributeAccessMode attributeAccessMode);` 确定属性。 2. 编译此应用程序。 更多详细信息，请参见确定对象结构和属性 (页 112)。
用于关闭对象的 Close 方法已发生更改。	1. 将 `project.Close(CloseMode.PromptIfModified);` 替换为 `project.Close();`。 2. 编译此应用程序。 更多详细信息，请参见关闭项目 (页 103)。

更改	所需程序代码调整

同时访问已被替换为独占访问和事务。	1. 按照以下示例，将同时访问替换为独占访问和事务。
	– Openness V13 SP1（废弃）：
	`tiaProject.StartTransaction("Reseting project to default");`
	`...`
	`tiaProject.CommitTransaction();`
	– Openness V14:
	`//Use exclusive access to avoid user changes`
	`ExclusiveAccess exclusiveAccess = tiaPortal.ExclusiveAccess();`
	`...`
	`exclusiveAccess.Dispose();`
	`//Use transaction to be able to rollbank changes:`
	`Transaction transaction = exclusiveAccess.Transaction(tiaProject, "Compiling device");`
	`transaction.CommitOnDispose();`
	2. 编译此应用程序。
	更多详细信息，请参见 Exclusive access (页 89)和事务处理 (页 91)。

已更改对 CPU 的在线访问	1. 可按照以下示例更改对 CPU 的在线访问：
	– Openness V13 SP1（废弃）：
	`((IOnline)controllerTarget).GoOffline();`
	– Openness V14:
	`((DeviceItem) plcSoftware.Parent.Parent).GetService<OnlineProvider>().GoOffline();`
	2. 编译此应用程序。

硬件配置已更改	1. 更改硬件配置： `Device.Elements = Device.Items` 2. 删除以下硬件属性： – `Device.InternalDeviceItem` – `Device.SubType` 3. 编译此应用程序。

参见

处理异常 (页 331)

Openness V14 SP1 有哪些新增功能？ (页 17)

连接到 TIA Portal (页 69)

9.5.4 使用 Openness V13 SP1 和早期版本导入生成的文件

应用程序

尝试使用 Openness V13 SP1 或早期版本导入生成的文件时，将因不兼容而产生例外情况。这是由于更改 HMI 变量和 HMI 画面项导致的。下面的表格给出了主要的属性更改，更多详细信息，请参见 TIA Portal 在线帮助中“使用对象和对象组创建画面 > 使用对象 > 组态范围”一章：

更改 HMI 变量

下表列出了 HMI 变量属性的主要更改：

已删除属性	已添加属性
RangeMaximumType RangeMaximum RangeMinimumType RangeMinimum	LimitUpper2Type. LimitUpper2. LimitLower2Type. LimitLower2. LimitUpper1Type LimitUpper1 LimitLower1Type LimitLower1

更改 HMI 画面项

下表列出了滚动条属性的主要更改：

已删除属性	已添加属性
	RangeLower1Color
	RangeLower1Enabled
	RangeLower2Color
	RangeLower2Enabled
	RangeNormalColor
	RangeNormalEnabled
	RangeUpper1Color
	RangeUpper1Enabled
	RangeUpper2Color
	RangeUpper2Enabled
	ScalePosition
	ShowLimitLines
	ShowLimitMarkers
	ShowLimitRanges

下表列出了量表属性的主要更改：

已删除属性	已添加属性
DangerRangeColor	RangeLower1Color
DangerRangeStart	RangeLower1Enabled
DangerRangeVisible	RangeLower2Color
WarningRangeColor	RangeLower2Enabled
WarningRangeStart	RangeNormalColor
WarningRangeVisible	RangeNormalEnabled
	RangeUpper1Color
	RangeUpper1Enabled
	RangeUpper1Start
	RangeUpper2Color
	RangeUpper2Enabled
	RangeUpper2Start

下表列出了棒图属性的主要更改：

已删除属性	已添加属性
AlarmLowerLimitColor	RangeLower1Color
AlarmUpperLimitColor	RangeLower1Enabled
	RangeLower2Color
	RangeLower2Enabled
	RangeNormalColor
	RangeNormalEnabled
	RangeUpper1Color
	RangeUpper1Enabled
	RangeUpper2Color
	RangeUpper2Enabled

索引

参

查

程

创

打

导

读

对

访

复

副

工

公

功

函

集

建

可

库

块

类

例

连

枚

启

全

确

软

删